Version 1.3

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 22/10/2016 | 1.0 | initial version | Karl Spickermann |
| 11/11/2016 | 1.1 | formatting and content | Daniel Raiser |
| 12/11/2016 | 1.2 | formatting | Daniel Raiser |
| 12/12/2016 | 1.3 | updated OUCD | Daniel Wagner |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 5

1.5 Overview 5

2. Overall Description 5

3. Specific Requirements 8

3.1 Functionality 8

3.1.1 Contact Management 8

3.1.2 Reports and Dashboards 8

3.1.3 Customizable HomeScreen 8

3.1.4 Chat integration 8

3.1.5 API Access 8

3.1.6 Email Integration 8

3.2 Usability 9

3.2.1 Using a browser 9

3.2.2 Contact Sharing 9

3.2.3 Setting up an account 9

3.3 Reliability 9

3.3.1 App availability 9

3.3.2 Mean Time Between Failures 10

3.3.3 Mean Time To Repair 10

3.3.4 Bugs 10

3.4 Performance 10

3.4.1 Capacity 10

3.4.2 Response time 10

3.5 Supportability 10

3.5.1 Language Support 10

3.5.2 Maintenance 11

3.6 Design Constraints 11

3.7 On-line User Documentation and Help System Requirements 11

3.8 Purchased Components 11

3.9 Interfaces 11

3.9.1 User Interfaces 11

3.9.2 Hardware Interfaces 11

3.9.3 Software Interfaces 11

3.9.4 Communications Interfaces 11

3.10 Licensing Requirements 11

3.11 Legal, Copyright, and Other Notices 12

3.12 Applicable Standards 12

4. Supporting Information 12

# Introduction

[The introduction of the **Software Requirements Specification (SRS)** provides an overview of the entire **SRS**. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the **SRS**.]

[Note: The **SRS** document captures the complete software requirements for the system, or a portion of the system. Following is a typical **SRS** outline for a project using only traditional, natural-language style requirements—with **no use-case modeling.** It captures all requirements in a single document, with applicable sections inserted from the Supplementary Specifications (which would no longer be needed). For a template of an **SRS** using use-case modeling, which consists of a package containing Use Cases of the use-case model and applicable Supplementary Specifications and other supporting information, see rup\_srsuc.dot.]

[Many different arrangements of an **SRS** are possible. Refer to [IEEE830-1998] for further elaboration of these explanations, as well as other options for **SRS** organization.]

## Purpose

[Specify the purpose of this **SRS**. The **SRS** fully describes the external behavior of the application or subsystem identified. It also describes nonfunctional requirements, design constraints, and other factors necessary to provide a complete and comprehensive description of the requirements for the software.]

This SRS describes all specifications for the “Friend-Relationship-Management System”. The Friend-Relationship-Management System (FRM) will be an application that helps users who have many friends to manage their contacts and sensible contact information. We aim to invent a system that is the Outlook-Exchange address book for private persons with the maximum amount of control over your information. In this document the usage of the "FRM System"-web-application will be explained. Furthermore reliability, reaction speed and other important characteristics of this project will be specified. This includes design and architectural decisions regarding optimization of these criteria as well.

## Scope

[A brief description of the software application that the **SRS** applies to, the feature or other subsystem grouping, what Use-Case model(s) it is associated with, and anything else that is affected or influenced by this document.]

This software specification applies to the whole "FRM System" web-application. The application provides all the features business people already know from their CRM Systems – for example contact management, reports and dashboards, customizable home screen, chat integration, API access and email Integration. All of this brought down to a user-friendly and uncomplicated level with easy access via internet browsers. The system will be set up in a way, that allows for expansion onto mobile platforms like Android and iOS with use of an applicable framework but it will not integrate these features from the beginning on.

## Definitions, Acronyms, and Abbreviations

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **SRS**. This information may be provided by reference to the project’s Glossary.]

UC Use Case

UCD Use Case Diagram

OUCD Overall Use Case Diagram

SAD Software Architecture Document

RFC Request for Comments

## References

[This subsection provides a complete list of all documents referenced elsewhere in the **SRS**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

|  |  |
| --- | --- |
| **Title** | **Date** |
| [FRM System Website](http://www.frmsystem.de) | 11/23/2016 |
| [FRM System Blog](http://wordpress.frm.bplaced.net/) | 10/24/2016 |

## Overview

[This subsection describes what the rest of the **SRS** contains and explains how the document is organized.]

The following chapters are about our vision and perspective, the requirements for deploying the software on our side and accessing the software on the user side, the demands we have, licensing and the technical realization of this project.

# Overall Description

[This section of the **SRS** describes 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, and makes them easier to understand. Include such items as:

• product perspective

• product functions

• user characteristics

• constraints

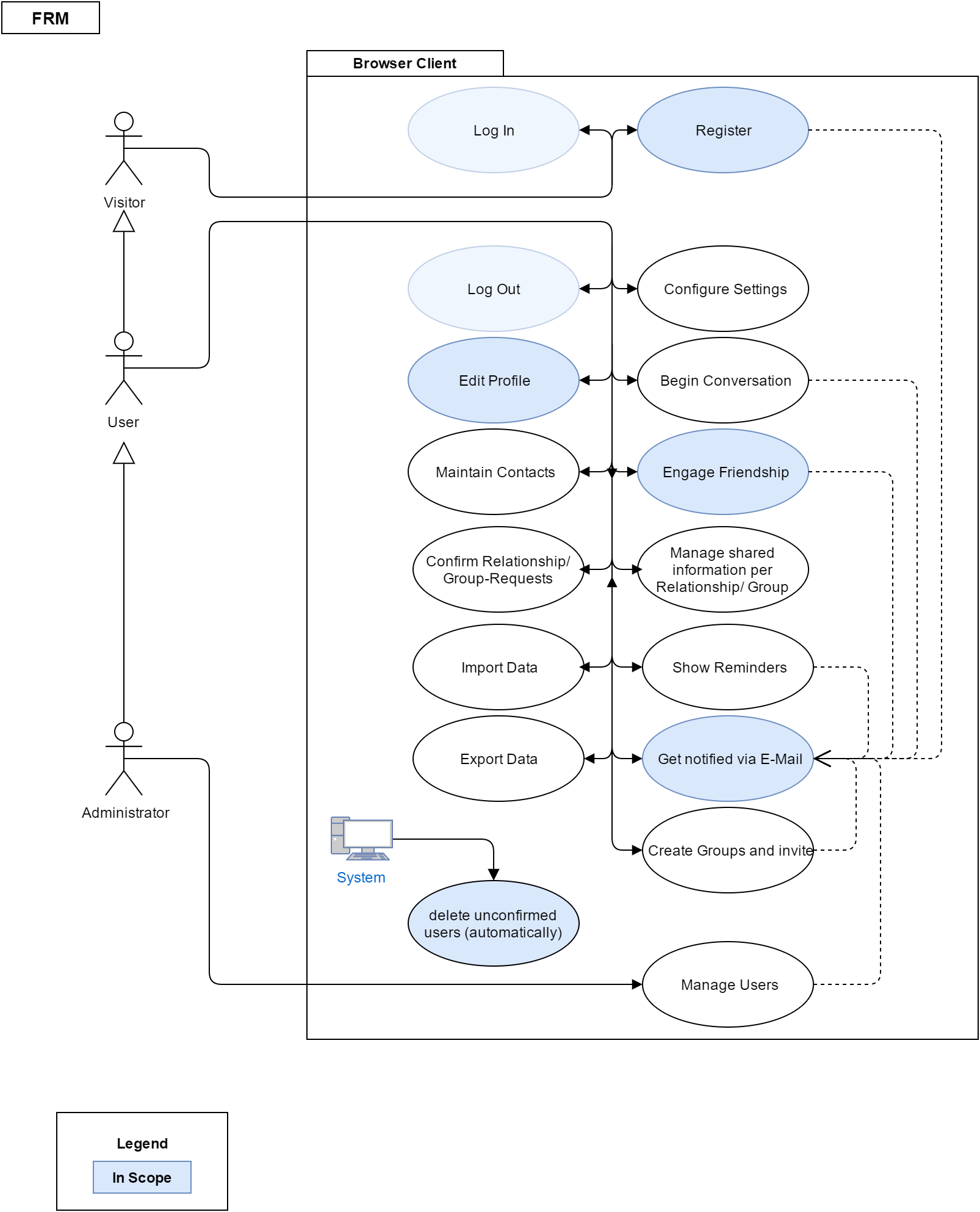
• assumptions and dependencies

• requirements subsets]

Our goal is to make the application as secure and easy to use as possible. In a user-ready state we guarantee that all the sensible information that you store and share with our application is secure. The FRM System will be a place, where you can safely store addresses, bank account information, private telephone numbers and other information that you do not want to put on networks like Facebook but want to share with certain people.

What you can do once you uploaded your information is requesting certain information from other users. You can engage in a “friendship” with a person to e.g. get his address and to share yours. The other person will then receive an email, notifying him about your request. With one click on a link the user can share the requested information. Sharing more or less information than the person requested is of course possible as well, these settings can be handled all individually. Another useful feature are groups: To create a group you get everybody’s user name or email address and add them. When someone accepts the invitation, everybody in this group can access a certain level of information of this person. These individual relationships can also be edited individually on their own to extend the level of information that someone can view, as long as you are in a group relationship. The system will offer notifications for birthdays or custom alarms which remind you that you had no contact with someone for a long time. The system will also offer a basic messaging feature to make communication easier and more central if you wish to. If not you can still use services like Whatsapp and store the preferred way of communication in your profile, so everybody knows how you can be reached the fastest.

The picture on the following page shows the overall use case diagram for our software:



Use Cases:

[Register](https://github.com/d-wagner/frmsystem/blob/docs/docs/published/UC_Registration.pdf)

[Edit Profile](https://github.com/d-wagner/frmsystem/blob/docs/docs/published/UC_EditProfile.pdf)

Engage [Friendship](https://github.com/d-wagner/frmsystem/blob/docs/docs/published/UC_engageFriendship.pdf)

receive e-[mail](https://github.com/d-wagner/frmsystem/blob/docs/docs/published/receiveEmailNotification.pdf) notifications

[search for unconfirmed users](https://github.com/d-wagner/frmsystem/blob/docs/docs/published/UC_Cronjob.pdf)

# Specific Requirements

[This section of the **SRS** contains all 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. When using use-case modeling, these requirements are captured in the Use Cases and the applicable supplementary specifications. If use-case modeling is not used, the outline for supplementary specifications may be inserted directly into this section, as shown below.]

## Functionality

[This section describes the functional requirements of the system for those requirements that are expressed in the natural language style. For many applications, this may constitute the bulk of the **SRS** package and thought should be given to the organization of this section. This section is typically organized by feature, but alternative organization methods may also be appropriate; for example, organization by user or organization by subsystem. Functional requirements may include feature sets, capabilities, and security.

Where application development tools, such as requirements tools, modeling tools, and the like, are employed to capture the functionality, this section of the document would refer to the availability of that data, indicating the location and name of the tool used to capture the data.]

[The requirement description.]

### Contact Management

This feature allows you to organize your contacts into groups and into any context that you see fit. Within this feature you can set up access to information on a contact-to-contact basis and see, who is able to access your information.

### Reports and Dashboards

Users can view statistics in a highly visual, engaging perspective using customized reports and dashboards. This is mainly used to see upcoming birthdays, the time when you last contacted persons and so on, but there will be many possible views to choose from.

### Customizable HomeScreen

The UI of the application is flexible enough to accommodate the visual perspective of everything including the home screen. Layouts can also be changed as well as colors based on preferences.

### Chat integration

You can chat with friends, partners and provide emotional support from the UI directly. We also aim to invent groupchats with several persons. The ability to message a person is also part of the preferences, you can change whether only friends or anyone can message you.

### API Access

Attach your FRM to various services that extend capabilities. In a release-ready state of the app you can integrate your contacts directly into the contacts app of your android phone, query the system for user information and user data etc.

### Email Integration

Pull in all the features and possibilities of your dedicated email client. We aim to integrate

a synchronization feature for outlook exchange active sync address books.

## Usability

[This section includes all those requirements that affect usability. For example,

* specify the required training time for a normal users and a power user to become productive at particular operations
* specify measurable task times for typical tasks or base the new system’s usability requirements on other systems that the users know and like
* specify requirement to conform to common usability standards, such as IBM’s CUA standards Microsoft’s GUI standards]

### Using a browser

The user of our web application has to know how to open and operate a modern browser like Chrome, Firefox or Opera. He is free to use mobile versions as well since we will provide a mobile-friendly interface.

### Contact Sharing

One of the most important features for our user is to share information and add new contacts. The needed time for this needs to be as short as possible, ideally within the click of a link. The maximum needed time to form a request to another user should not exceed the time needed, to ask the other person via a social network to reveal the needed information (excluding the time to answer, since that is not a task for the user in the first place).

### Setting up an account

The time for this should not exceed the time needed on normal social networks, excluding the additional information that our system is able to store. The process should be simple and streamlined, it should be clear what has to be entered and what does not have to be entered.

## Reliability

[Requirements for reliability of the system should be specified here. Some suggestions follow:

* Availability—specify the percentage of time available ( xx.xx%), hours of use, maintenance access, degraded mode operations, and so on.
* Mean Time Between Failures (MTBF) — this is usually specified in hours, but it could also be specified in terms of days, months or years.
* Mean Time To Repair (MTTR)—how long is the system allowed to be out of operation after it has failed?
* Accuracy—specifies precision (resolution) and accuracy (by some known standard) that is required in the system’s output.
* Maximum Bugs or Defect Rate—usually expressed in terms of bugs per thousand lines of code (bugs/KLOC) or bugs per function-point( bugs/function-point).
* Bugs or Defect Rate—categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a “critical” bug; for example, complete loss of data or a complete inability to use certain parts of the system’s functionality.]

### App availability

Our Service should ensure a 98% up-time. This is mainly depending on the ensured up-time by our serverhoster and the time needed to update and maintain the site

### Mean Time Between Failures

3 Months.

### Mean Time To Repair

The user needs the saved data in his everyday life, therefore a fix should not need more than 12 hours in severe cases.

### Bugs

The system should not contain bugs in critical areas where user data is handled or displayed, all data must be safe and bugs in the backend processing are not acceptable. Minor UI bugs are tolerable if they get fixed within a reasonable amount of time and are not mission-critical when it comes to safely handling user data.

## Performance

[The system’s performance characteristics are outlined in this section. Include specific response times. Where applicable, reference related Use Cases by name.

* Response time for a transaction (average, maximum)
* Throughput, for example, transactions per second
* Capacity, for example, the number of customers or transactions the system can accommodate
* Degradation modes (what is the acceptable mode of operation when the system has been degraded in some manner)
* Resource utilization, such as memory, disk, communications, and so forth.

### Capacity

A limit of 1000 manageable contacts per user is planned. We plan to have a server ready, which can deal with at least 500 user at the start of the service without performance issues but plan to upgrade this quickly, depending on how frequented the service is

### Response time

The processing of user data and displaying it should not exceed maximum times of 1 second during normal use on average days and 2 seconds when displaying big amounts of data with many users online at the same time.

## Supportability

[This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including coding standards, naming conventions, class libraries, maintenance access, and maintenance utilities.]

### Language Support

We will use following languages, which will be well supported in the future:

* PHP (Laravel)
* HTML
* CSS
* JavaScript
* MySQL

### Maintenance

Basic Maintenance access for administrators will be possible with a very basic web UI that serves as a utility and should be secure but does not need to be pretty. Access to the backend components and architecture will only be possible via heavily secured SSH access and secure FTP connections.

## Design Constraints

[This section indicates any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

All information about the architectural design of our application stack can be found in our [technology and roles blog](http://frm.bplaced.net/wordpress/?p=32) entry, the main part being the server running the application

## On-line User Documentation and Help System Requirements

[Describes the requirements, if any, for o-line user documentation, help systems, help about notices, and so forth.]

The whole application is programmed for an intuitive and easy use, so the user shouldn’t have any problems with controlling it. However, should there be problems with complicated features in the future, we will surely implement a help page in the app. We are also going to implement small tool tips or information texts on the page itself from the beginning on, to instruct the user what to do with as few words as possible. Furthermore we document all of the features on our own blog, on which users can find information and ask us questions.

## Purchased Components

[This section describes any purchased components to be used with the system, any applicable licensing or usage restrictions, and any associated compatibility and interoperability or interface standards.]

To deploy the system we are renting a Server4you virtual server. Should we see the need in the future we can upgrade it to better specifications with the click of a button.

## Interfaces

[This section defines the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, and the like, so that the software can be developed and verified against the interface requirements.]

### User Interfaces

tbd

### Hardware Interfaces

n/a

### Software Interfaces

tbd

### Communications Interfaces

n/a

## Licensing Requirements

At the time of writing this we don’t use any code which is subject to license requirements except for Open Source such as [Linux Debian](https://www.debian.org/legal/licenses/), [Apache License 2.0](http://www.apache.org/licenses/LICENSE-2.0) and so on.

## Legal, Copyright, and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notices, wordmark, trademark, or logo compliance issues for the software.]

n/a

## Applicable Standards

[This section describes by reference any applicable standard and the specific sections of any such standards which apply to the system being described. For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, and so forth.]

* HTTP standards for communication client 🡨🡪 Server
* HTML and JavaScript / PHP Standards

# Supporting Information

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

* Table of contents
* Index
* Appendices

These may include use-case storyboards or user-interface prototypes. When appendices are included, the **SRS** should explicitly state whether or not the appendices are to be considered part of the requirements.]