Software Requirements Specification

for

EasiLendar

Version 1.0 approved

Prepared by Cat Can

TEXAS Group

06/05/2015

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope of Project 1

1.3 Definitions, Acronyms and Abbreviations 2

1.4 Intended Audience 2

1.5 References 3

1.6 Overviews 3

2. Overall Description 5

2.1 Product Perspective 5

2.2 Product Functions 5

2.3 User Classes and Characteristics 5

2.4 Operating Environment 5

2.5 Design and Implementation Constraints 5

2.6 User Documentation 5

2.7 Assumptions and Dependencies 6

3. External Interface Requirements 7

3.1 User Interfaces 7

3.2 Hardware Interfaces 7

3.3 Software Interfaces 7

3.4 Communications Interfaces 7

4. System Features 8

4.1 System Feature 1 8

4.2 System Feature 2 (and so on) 8

5. Other Nonfunctional Requirements 9

5.1 Performance Requirements 9

5.2 Safety Requirements 9

5.3 Security Requirements 9

5.4 Software Quality Attributes 9

5.5 Business Rules 9

6. Other Requirements 10

Appendix A: Glossary 11

Appendix B: Analysis Models 11

Appendix C: To Be Determined List 11

Revision History

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

# Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references, and overview of the SRS. The aim of this document is to gather, analyze, and give an in-depth insight of the complete **EasiLendar software** by defining the problem statement in detail. Nevertheless, it also concentrates on the capabilities required by stakeholders and their needs while defining high-level product features. The detailed requirements of the **EasiLendar software** are provided in this document.

## Purpose

The purpose of this document is to present a detailed description of the Web Publishing System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Regional Historical Society for its approval.

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers. In addition, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops.

In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters, and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team, and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

## Scope of Project

Our meeting planner has following features:

* It draws from Google Calendar and/or other widely used calendaring applications to help choose prospective meeting times.
* It works (although perhaps not as conveniently) for users who do not use the same calendaring application, or who choose not to share that information.
* It respects privacy and security preferences. In particularly, it makes it possible to find a meeting time in common with minimum disclosure of schedule information. An analysis of privacy and security be an integral part of the project.

Many enhancements and extensions are possible, including:

* Draw from multiple calendar and scheduling systems, e.g., Google Calendar and Microsoft Outlook, and include a well-documented protocol or interface for including additional calendar systems.
* Support both web-based and email-based responses from prospective meeting participants, on both desktop/laptop and mobile platforms.
* Support complex attendance requirements, e.g., “persons A, B, and C, are necessary; persons X, Y, and Z are invited but not necessary for the meeting; at least two of M, N, and O be present.”
* Distinguish between preferred meeting times and possible meetings times, flexibly ranking proposed times both by which attendees can attend and by how convenient the time is.
* Handle time zones appropriately and robustly. For example, a teleconference at 3pm for the Seattle participants and midnight for participants in Lisbon and Milano would likely be less desirable than a meeting at 8am for the Seattle participants and 5pm for the Lisbon and Milano participants. Each prospective participant have an opportunity to say which hours of the day are suitable for meetings.

## Definitions, Acronyms and Abbreviations

## Intended Audience

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Author | Person submitting an article to be reviewed. In case of multiple authors, this term refers to the *principal author*, with whom all communication is made. |
| Editor | Person who receives articles, sends articles for review, and makes final judgments for publications. |
| Developer | Member of TEXAS Group, who created this project, have authorities to make change in this project, and take responsibility to develop this project as well. |
| Documentation writer | A writer who engages in technical writing and produces technical documentation that helps people use a product or service. |
| Marketing staff | Marketing staffs consist of a variety of employees, all of whom share a common goal--promoting the company’s brand. |
| Project manager | A professional in the field of project management. Project managers can have the responsibility of the planning, execution and closing of any project, typically relating to construction industry, architecture, aerospace and defense, computer networking, telecommunications or software development. |
| Reader | Anyone visiting the site to read articles. |
| Reviewer | A person that examines an article, and has the ability to recommend approval of the article for publication or to request that changes be made in the article. |
| Stakeholder | Any person with an interest in the project who is not a developer. |
| Tester | A technician who conducts prescribed tests on software programs and applications prior to their implementation to ensure quality, design integrity and proper functionality. |
| User | Reviewer or Author. |

## References

[1] IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

[2] Score-Contest. Project detail at [*http://score-contest.org/2016/projects/meetme.php*](http://score-contest.org/2016/projects/meetme.php)

[3] Sproj3. *Libra: An Economy-Driven Cluster Scheduler Software Requirements Specification* Version <1.0>

[4] Joan, Paul Adams, Bobbie Baker, Charles Charlie. *Software Requirements Specification* Version 1.0 <<Annotated Version>> April 15, 2004 Web Publishing System

[5] E-Store Project. *Software Requirements Specification* Version <4.0>

## Overviews

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, External Interface Requirements section, of this document is written primarily for the developers and describes the Interfaces must be deploy of the product, e.g., User Interfaces, Hardware Interfaces, Software Interfaces, and Communications Interfaces.

The fourth chapter, System Features section, of this document illustrates organizing the functional requirements for the product by system features, the major services provided by the product. It can be organized by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

The next chapter, Other Nonfunctional Requirements, of this document is a comprehensive detail of some nonfunctional requirements. For example performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices; or any requirements about Safety Requirements, Security Requirements, Software Quality Attributes, and Business Rules as well.

The last chapter, Other Requirements, of this section defines any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

# Overall Description

## Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

## Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product respond to anticipated error conditions or invalid inputs. Requirements be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

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