Software Requirements Specification

For

[Schedule Assistant]

[01/23/2015]

[2.1]

Prepared by:

Author1, Author2, etc.

**Table of Contents**

[Revision History 3](#_Toc409958281)

[1 Introduction 4](#_Toc409958282)

[1.1 Overview 4](#_Toc409958283)

[1.2 Goals and Objectives 4](#_Toc409958284)

[1.3 Scope 5](#_Toc409958285)

[1.4 Definitions 5](#_Toc409958286)

[1.5 Document Conventions 6](#_Toc409958287)

[1.6 Assumptions 6](#_Toc409958288)

[2 General Design Constraints 6](#_Toc409958289)

[2.1 Product Environment 6](#_Toc409958290)

[2.2 User Characteristics 7](#_Toc409958291)

[2.3 Mandated Constraints 7](#_Toc409958292)

[2.4 Potential System Evolution 8](#_Toc409958293)

[3 Nonfunctional Requirements 8](#_Toc409958294)

[3.1 Usability Requirements 8](#_Toc409958295)

[3.2 Operational Requirements 8](#_Toc409958296)

[3.3 Performance Requirements 8](#_Toc409958297)

[3.4 Security Requirements 9](#_Toc409958298)

[3.5 Safety Requirements 9](#_Toc409958299)

[3.6 Legal Requirements 9](#_Toc409958300)

[3.7 Other Quality Attributes 9](#_Toc409958301)

[3.8 Documentation and Training 9](#_Toc409958302)

[3.9 External Interface 9](#_Toc409958303)

[3.9.1 User Interface 9](#_Toc409958304)

[3.9.2 Software Interface 10](#_Toc409958305)

[4 System Features 10](#_Toc409958306)

[4.1 Feature: Time Away Specification 10](#_Toc409958307)

[4.1.1 Description and Priority 10](#_Toc409958308)

[4.1.2 Use Case: Future Time Away 10](#_Toc409958309)

[4.1.3 Use Case: Past Time Away 11](#_Toc409958310)

[4.1.4 Additional Requirements 11](#_Toc409958311)

[4.2 Feature: Employees List Editing 11](#_Toc409958312)

[4.2.1 Description and Priority 11](#_Toc409958313)

[4.2.2 Use Case: Adding an Employee 11](#_Toc409958314)

[4.2.3 Use Case: Removing an Employee 12](#_Toc409958315)

[4.2.4 Use Case: Update Employee Information 12](#_Toc409958316)

[4.2.5 Additional Requirements 12](#_Toc409958317)

[4.3 Feature: Employees List Editing 12](#_Toc409958312)

[4.3.1 Description and Priority 12](#_Toc409958313)

[4.3.2 Use Case: Generate Schedule 13](#_Toc409958314)

[4.3.3 Additional Requirements 13](#_Toc409958317)

[4.4 Feature: Schedule Editor 13](#_Toc409958312)

[4.4.1 Description and Priority 13](#_Toc409958313)

[4.4.2 Use Case: Generate Updated Calendar 13](#_Toc409958314)

[4.4.3 Additional Requirements 13](#_Toc409958317)

[4.5 Feature: Read-only version of Calendar 13](#_Toc409958312)

[4.5.1 Description and Priority 13](#_Toc409958313)

[4.5.2 Use Case: Generate Calendar 13](#_Toc409958314)

[4.5.3 Additional Requirements 14](#_Toc409958317)

[4.6 Feature: Total Time Away 14](#_Toc409958312)

[4.6.1 Description and Priority 14](#_Toc409958313)

[4.6.2 Use Case: Calculate total time away 14](#_Toc409958314)

[4.6.5 Additional Requirements 14](#_Toc409958317)

[4.7 Feature: Total OnCall Rotations 14](#_Toc409958312)

[4.7.1 Description and Priority 14](#_Toc409958313)

[4.7.2 Use Case: Calculate and view total time on rotation 14](#_Toc409958314)

[4.7.3 Additional Requirements 14](#_Toc409958317)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
|  | **1/23** | **Samaa Gazaa &**  **Adam Guerrero** | **Added initial features and use cases based off of project scope** |
|  | **1/24** | **Samaa Gazaa &**  **Adam Guerrero** | **Collaborated to review features and use cases** |
|  | **1/25** | **Samaa Gazaa &**  **Adam Guerrero** | **Inserted features and use cases relative to system requirements and project scope** |
|  | **1/26** | **Samaa Gazaa &**  **Adam Guerrero** | **Edited Use cases and document index** |
|  | **1/27** | **Samaa Gazaa &**  **Adam Guerrero** | **Formatted and organized project document** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Overview

This section provides an overview of the requirements document and the system specified. After reading this section the reader should understand the purpose of the document and have a general idea what the proposed system will do.

**The Schedule Assistant software is designed for upper management. This software has been created to help upper management control and organize on-call rotations within the organization. This document will provide the user with a thorough outline of the purpose of the software, and what the software has been designed to achieve. This document will also include a high level view of the software to include users, design, and, functional and nonfunctional requirements, features, and constraints.**

The design of this template assumes there will be a separate project document and quality assurance test plan that will address issues such as schedule, cost, development methods, development phases, deliverables and testing procedures. If there are other sources of product requirements such as prototypes, user guides or user interface storyboards they should be mentioned and referenced in this section.

**This document with identify functional and nonfunctional requirements as well as implementation constraints. This document will serve as a handbook to any audience within the organization seeking to achieve a more thorough understanding of the Scheduling Assistant Software. The Software Requirement Specification handbook will include a list of general product features as well as detailed system behavior needed to deliver these features. Detailed system behavior will be expressed as use cases or usage scenarios.**

You may want to identify the intended audience for the document and describe the individual sections of the document.

**The Scheduling Assistant is a web-based software created and designed to assist upper management with organizing, generating, and maintain schedule competency within their company.**

***Example:***

This document defines the requirement for the Innovative Publishing system that is being developed for UMKC Faculty. The purpose of this document is to represent the system requirements in a readable way so that clients and stakeholders can understand them and verify them for correctness but with enough detail that developers can design and implement a software system from them.

This document doesn’t address *project* issues such as schedule, cost, development methods, development phases, deliverables and testing procedures. Those are addressed in a separate project document and quality assurance test plan.

The Innovative Publishing system is a web-based tool for publishing content on the Internet. It provides a way for authors to get direct and specific feedback from readers while imposing minimal additional work for authors and readers.

## Goals and Objectives

Making the goals and objectives of the product explicit guides developers and give direction to the project. During the development of even a small system developers make hundreds of tiny decisions (consciously and unconsciously) not specifically addressed by the formal requirements. Knowing the goals and objectives will help them make the best decisions for the product.

Goals and objectives also help keep the project on track. Without a clear set of goals and objectives the scope of the project can grow or shift as new information arrives. Shifting the focus of the project isn’t bad as long as it is done open and explicitly with everyone in agreement.

***Example:***

The three main goals of the innovative publishing product are:

1. Provide a simple mechanism for readers to offer feedback to authors on content. The rational is that more readers will offer feedback if the process is easy.
2. Use feedback from readers to add value to the content and improve the experience of other readers without intervention from the author. For example, book reviews on Amazon are instant free content that increases the value of the site.
3. The system shouldn’t require any extra effort on the part of the author or other publishing agent.

## Scope

The scope defines the boundaries of the product – what it will and will not do. Clients and other stakeholders need a clear understanding what to expect. It’s at the boundaries of the system where there is the most opportunity for misunderstandings regarding what is and is not going to be implemented.

System features are described below so The system features section below does specify exactly what will be included in the system; however, it is not presented in a way that makes clear functionality at the boundaries of the system.

***Example:***

The innovative publishing system will solicit feedback from readers including written comments. Aggregate feedback from readers may be offered directly to other readers (i.e. articles might be rated), but unedited comments from readers will not automatically be made available to other readers. The reason for this is the quality of unedited comments is hard to control.

## Definitions

This section defines potentially unfamiliar or ambiguous words, acronyms and abbreviations.

**Use case** – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

**Scenario** – one path through a use case

**Actor** – user or other software system that receives value from a use case.

**Role** – category of users that share similar characteristics.

**Product** – what is being described here; the software system specified in this document.

**Project** – activities that will lead to the production of the product described here. Project issues are described in a separate project plan.

**Shall** – adverb used to indicate importance; indicates the requirement is mandatory. “Must” and “will” are synonyms for “shall”.

**Should** – adverb used to indicate importance; indicates the requirement is desired but not mandatory.

**May** – adverb used to indicate an option. For example, “The system may be taken offline for up to one hour every evening for maintenance.” Not used to express a requirement, but rather to specifically allow an option.

**Controls** – the individual elements of a user interface such as buttons and check boxes.

## Document Conventions

This section describes presentation conventions use in the document.

***Example:***

Portions of this document that are incomplete will be marked with TBD. Each TBD item will have an owner and estimated date for resolving the issue.

## Assumptions

In this section list any assumptions on which the requirements, as they are described here, depend. Assumptions are conditions, usually outside the control of the performing organization, that are taken for granted.

Be careful to only document assumptions that are outside the control of the performing organization. For example, the following is not a valid assumption for the requirements document: “We assume that all requirements will be documented.” You might be *assuming* this but there is no point in documenting it as an assumption here because it is something that is primary within the scope and control of the performing organization. The purpose of this section is to document assumptions that are outside the control of the performing organization—conditions that should be noted because someone will be taking responsibility for ensuring they hold.

A distinction can also be made between assumptions that pertain to the requirements and those that pertain to the project as a whole. The software project management plan is the most logical place to document assumptions that pertain to the project as a whole.

***Example:***

It is assumed that the client has an ODBC compliant database installed and this database will be accessible from the machine where the system will run.

It is assumed that the web hosting ISP will allow server-side scripts to access the file system.

# General Design Constraints

## Product Environment

Most software systems don’t exist in isolation. They interact with other systems and are part of a larger system or environment. This section describes the boundaries between the proposed system and its environment. The product context may include other hardware/software systems or a general business context. It may be helpful to specify the product environment with a block diagram that shows the major interfaces between the system under development and its environment.

If the system will use an existing database management system describe the interface here.

Note the user interface, which characterizes an interface between the system and its environment, is described below.

***Example:***

The innovative publishing system will be a component of the existing online course management system used in the computer science department at UMKC. The online course management system has built-in support for authentication. The innovative publishing system will use the existing course management system to identify and authenticate readers. The online course management system also includes a relational database which will be used through the JDBC/ODBC interface.

## User Characteristics

It’s important for developers to have a complete and accurate image of the end users. Even when the requirements of the user interface are described in detail the developer will still make many tiny design decisions during design and implementation. Knowing the general characteristics of the end users will help the developer make better decisions.

If the specific users of the system are know list them here. More likely there will be user roles or categories of uses. For each group of users list their responsibilities, characterize their knowledge of the domain and describe their characteristics including technical sophistication, background and education.

Prioritize users if necessary. This is especially important when user characteristics conflict. For example, if the system must accommodate experience users as well as novices it is important to know which should be given priority in case it’s not possible to accommodate both groups.

## Mandated Constraints

Ideally requirements will be specified in terms of functionality needed and developers will have free rein to design and implement a solution. In practice there are constraints on the eventual design and implementation.

Constraints may be mandated technologies. For example, the client may specify that a specific database management system, programming language, and/or operating system be used.

Constraints limit design and implementation options.

Constraints might be absolute, desirable or optional. If constraints aren’t absolute the motivation for the constraint should also be given.

## Potential System Evolution

The resulting software system should be maintainable and extensible. Knowing the types of anticipated changes aids significantly in establishing an architecture that will accommodate the types of expected changes. This section suggests ways the system is likely to be extended or modified in the future.

# Nonfunctional Requirements

Nonfunctional requirements are properties the system must have. Nonfunctional requirements tend to be orthogonal to functional requirements. For example a system may have the nonfunctional requirement that it be offline no more than 15 minutes at a time and not more than ½ hour each week. The realization of this requirements isn’t limited to one spot in the code. This nonfunctional requirement crosscuts some or all functional requirements.

## Usability Requirements

It’s hard to image a software system that doesn’t have usability as one of its highest nonfunctional quality requirements. It’s not enough to just say that the system should be usable though. Usability requirements must be stated in a quantifiable and testable way.

One method of specifying usability requirements is to specify efficiency, effectiveness and satisfaction goals for specific scenarios of use (section 4) carried out by representative users (section 2.2). A simpler alternative is to design a survey to measure user satisfaction and get consensus on who will take the survey and what will be considered an acceptable aggregate score.

## Operational Requirements

This section describes the general characteristics of the physical environment for the product.

***Example:***

The users’ environment is noisy so the system shouldn’t depend on the user hearing audible output.

## Performance Requirements

The main performance characteristics are speed and capacity (memory). Performance requirements are usually stated as a function of the number of concurrent users. Use this section to state the performance requirements of the system as a whole. If specific transactions have their own performance requirements state these requirements below along with the description of the feature.

***Example:***

System startup time should be less than 3 seconds. With 30 concurrent users no operation should take more than 5 seconds and 95% of the operations should take less than 2 seconds.

## Security Requirements

Access to data and features may be limited to specific users. There may also be a requirement to keep an audit trail of system use. This section describes the security requirements including the levels and what needs to be protected.

## Safety Requirements

The system may affect the safety of the larger environment. For example, there are limits on the intensity of stray electromagnetic radiation from electronic devices used in hospitals. Potential safety concerns should be investigated and documented in this section.

## Legal Requirements

Some security and safety requirements may also be legal requirements. For example, federal law protects confidentiality of medical records.

***Example:***

Student social security numbers will not be visible to other students.

## Other Quality Attributes

There are specific sections above for non-functional quality attributes such as security, performance, etc. In this section describe any other non-functional quality attributes such as portability, availability, etc.

## Documentation and Training

An important part of the total system is the documentation and training that is provided with the system. This section should describe the types and quantity of documentation and training that will be provided with the product.

## External Interface

External interfaces may be user interfaces or software interfaces.

### User Interface

The requirements document shouldn’t contain design or implementation details. The *logical* user interface should be described here. The description here shouldn’t unnecessarily constrain design and implementation options.

The general personality of the interface should be described here. For example, should the interface convey a conservative, professional, authoritative or fun attitude? What is the look and feel? Style? User characteristics were given above in section 2.2 but it may be helpful to characterize the average user here as adult, teenager or child.

User interfaces may contain a mixture of media types. It may be helpful to describe the desired/permissible user interface in terms of media elements.

Should the interface be intuitive or will training be provided. If training is required what types of training will be provided (online help, separate user manual, formal classroom training).

Ease-of-use requirements should be stated in a way that can be verified. For example, “the product should be easy to use” isn’t verifiable because it’s impossible to define “easy” in a measurable way. Must better is “75% of users will be able to use 80% of the features within 20 seconds without prior training”.

### Software Interface

The software interfaces may be locally addressable API’s or remote interfaces using technologies such as web services.

***Example:***

The operations defined by use case 4-7 below will also be available programmatically via XML over HTTP. The exact protocol is described the WSDL document at xyz.

# System Features

In this section we have documented the main features of the web application. Each feature will include a description and priority rating. The description is a brief overview of functional requirement(s) the feature accomplishes. The priority level is based on three factors cost, risk and value. The developer estimates cost and risk, the user estimates the value of the feature. If there are any functional and/or nonfunctional requirements not included within the use case or feature description there will be a section including these needed requirements.

## Feature: Time Away Specification

### Description and Priority

Since the system is expected to assist the user in scheduling time away for each employee, this feature gives the user the ability to include such dates in the system.

Cost: medium

Risk: low

Value: high

### Use Case: Future Time Away

**Actors:** administrator

**Description:** The use-case begins when the administrator acquire time-away information of an employee. In which case the administrator will need the ability to enter the time-away into the schedule.

**Basic Path:**

1. The administrator will go to the schedule page.
2. The administrator will provide the time-away to the time-away form.
3. The system will provide a confirmation and update the schedule.

### Use Case: Past Time Away

**Actors:** administrator

**Description:** The use-case begins as the administrator attempts to enter past time-away information of an employee. The administrator should have the ability to enter time away of past dates.

**Basic Path:**

1. The administrator will go to the schedule page.
2. The administrator will provide the time-away to the time-away form.
3. The system will provide a confirmation and update the schedule.

### Additional Requirements

The user should be presented with a button that easily provide a form to enter the date and name of a time away, alongside with the employees information. In addition, the user should be able to view the changes made instantaneously.

## Feature: Employees List Editing

### Description and Priority

This feature provides the user with the ability to edit the employees list by either: adding, deleting or editing an employee’s record. This feature is highly needed in the case of change in employees’ roles as much as in the hiring process.

Cost: medium

Risk: low

Value: high

### Use Case: Adding an Employee

**Actors:** administrator

**Description:** The use-case begins as the administrator acquires the information of a new employee. The administrator should have the ability to add a new employee record to the employees list.

**Basic Path:**

1. The administrator will go to the employees’ list page.
2. The administrator will provide the needed information about the new employee.
3. The system will include the new record in the employees list.

### Use Case: Removing an Employee

**Actors:** administrator

**Description:** The use-case begins as the administrator desires to remove an employee record from the list. The administrator should have the ability to delete an existing employee record.

**Basic Path:**

1. The administrator will go to the employees’ list page.
2. The administrator will choose the desired employee record to delete.
3. The system will remove the specified record from the employees list.

### Use Case: Update Employee Information

**Actors:** administrator

**Description:** As the administrator desires to update the information included in one or more of the employees records, such as the application each employee is working on, this use case shall occur. The administrator should have the ability to edit the information in an employee’s record easily.

**Basic Path:**

1. The administrator will go to the employees’ list page.
2. The administrator will choose the desired employee record to edit.
3. The administrator will change and fill the form according as needed.
4. The system will update the edited record in the employees list.

### Additional Requirements

The user should be able to easily find the needed buttons on the employees list that will assist him in completing the previously mentioned use case.

***4.3 Feature: On-call schedule***

**Actors:** User/Manager

**4.3.1 Description and Priority**

The on-call schedule feature within the web application is created to automatically generate a schedule which will display the employees who are on call. The schedule that is generated will display and depict a specific time frame the employees are on call.

Cost: medium

Risk: low

Value: high

**4.3.2 Use Case: Generate schedule**

As a user I want to auto generate a schedule that will display the employees who are on call for a certain amount of time; week, weekend, month, etc…

**4.3.3 Additional Requirements**

The on-call schedule feature should be easy to access and user friendly. It should take the user little effort to use, and should be 99.9% accurate. Furthermore the schedule should be readable by employers.

***4.4 Feature: Schedule Editor***

**Actors:** User/Manager

**4.4.1 Description and Priority**

The feature to generate a new schedule once it has been edited will allow the user to create a new version of an already generated schedule. This feature will be helpful if the user has already created a schedule and identifies errors or has last minute changes. The feature will allow them to generate a new updated version of the calendar.

Cost: low

Risk: low

Value: high

**4.4.2 Use Case: Generate updated schedule**

As a user I want to generate a new updated schedule that will allow my employees including the person on call to view their weekly schedule obligations.

**4.4.3 Additional Requirements**

The update feature should provide the ability to re-generate only within a specified time frame with resolution down to one week.

***4.5 Feature: Read-only version of calendar***

**Actors:** All employees

**4.5.1 Description and Priority**

The read-only feature allows the user to create a calendar accessible to all employers. This feature applies restrictions to the calendar so that it is read only. Read-only documents cannot be modified, and are only accessible by specified personnel.

Cost: med

Risk: low

Value: medium

**4.5.2 Use Case: Generate calendar**

As a user I want to create a read only version of the calendar that can be read, but not edited by viewers.

**4.5.3 Additional Requirements**

**N/A**

***4.6 Feature: Total time away***

**Actors:** User/Manager

**4.6.1 Description and Priority**

This feature allows the user to get a summary of an employee’s time away from the office. This will include time away for vacation, sick days, and any other days from office. It will not include basic paid federal holidays such as Christmas, and New Year’s.

Cost: low

Risk: low

Value: medium

**4.6.2 Use Case: Calculate total days off**

As I user I want to view how many days out of the office an employee has had so that I can properly assign on call duties.

**4.6.3 Additional Requirements**

***4.7 Feature: Total On Call Rotations***

**4.7.1 Description and Priority**

This feature will summarize the total of on call rotations that have been conducted by an individual employee.

Cost: med

Risk: low

Value: medium

**4.7.2 Use Case: Calculate and view total time on rotation**

**Actors:** User/Manager

As a user I want to know how many times an individual employee has been on call so that the rotation is fair within the organization.

**4.7.3 Additional Requirements**

**N/A**