Software Requirements Specification

For

Task Management System

Version 1.0

Prepared by Brent Raymund D. Torreda

Binacore Group

December 26, 2017

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Project Scope 1

1.3 References 2

1.4 Overview of the Document 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Features 3

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 5

2.7 Assumptions and Dependencies 5

3. System Features 5

3.1 System Feature 1 5

3.2 System Feature 2 (and so on) 7

4. External Interface Requirements 7

4.1 User Interfaces 7

4.2 Hardware Interfaces 7

4.3 Software Interfaces 8

4.4 Communications Interfaces 8

5. Other Nonfunctional Requirements 8

5.1 Performance Requirements 8

5.2 Safety Requirements 9

5.3 Security Requirements 9

5.4 Software Quality Attributes 9

6. Other Requirements 10

Appendix A: Glossary 10

Appendix B: Analysis Models 10

Appendix C: Issues List 10

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Brent Torreda | 26/12/17 | Initial | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this document is to present a detailed description of the Task Management 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 presented to Binacore Group for its approval.

## Project Scope

This software system will be a Task Management System for a local accounting firm named BAS and More. This system will be designed to maximize the staff’s and manager’s productivity by providing tools to assist in automating the task review, assignment and time-logging process, which would otherwise have to be performed manually. By maximizing the staff’s work efficiency and production the system will meet the company’s needs while remaining easy to understand and use.

More specifically, this system is designed to allow a staff member such as a bookkeeper or accountant to view and manage open tasks, take responsibility for a task then log his hours spent working on it. The software will facilitate communication between management and staff and with the clients via E-Mail. Preformatted tasks with detailed steps (including videos) on how to perform them are used to facilitate the process. The system also uses a relational database containing a list of Members, the Tasks, Subtasks, Steps to guide how to perform the subtasks and data about client companies.

## References

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

## Overview of the Document

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, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

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

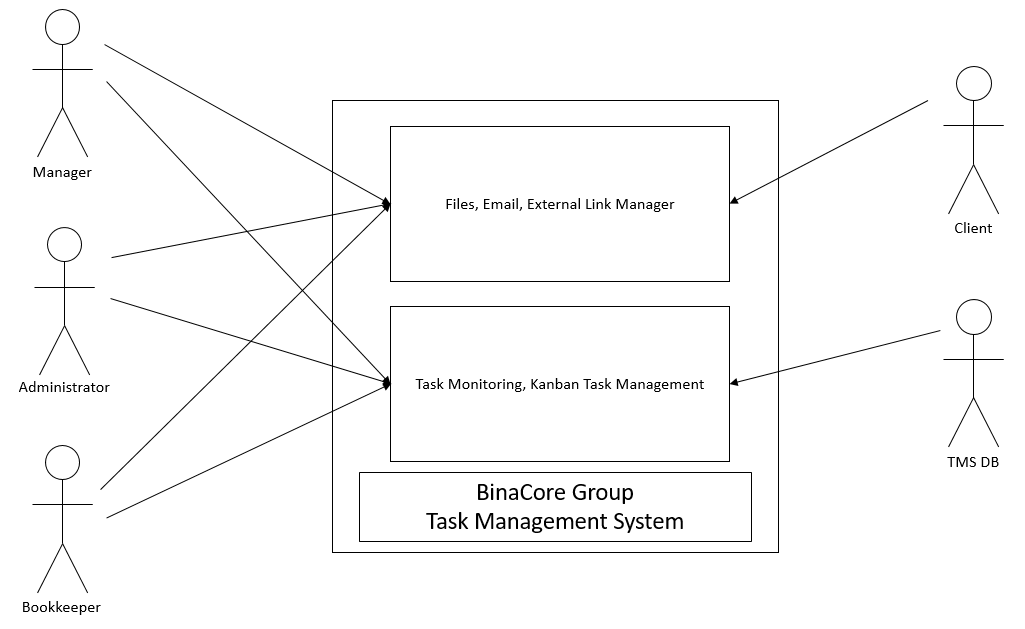
This product is the first of its kind and built custom-made for the target user company. The Task Management System (TMS) is a standalone system but is designed to be expandable to include possible future modules such as automated payroll and possibly interface with other systems.

Figure 1. System Environment

## Product Features

The system will be able to record tasks and each of these tasks can contain as many sub-tasks as necessary. Each sub-task will have a list of steps detailing instructions on how to perform them. The system will allow uploading of videos and images in order to better illustrate the procedures of performing the sub-tasks.

The system will also allow assigning of sub-tasks to team members and provide a means of easily browsing tasks and sub-tasks by status, company or assignee.

Registered team members will be able to log their work by a timer in the sub-task window in the system. The user can also choose to suspend the progress if the task needs a client’s response before it can be continued.

There will also be a role-based system for managing user access. This feature will also define how each user will experience the system. Those with a higher clearance will be able to monitor other users and have an overview of the system while those with regular clearances will only be able to view their current tasks and any available task.

## 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 favored user classes 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).>

# Functional Requirements Specifications

## Bookkeeper Use Cases

### Screen ClippingDiagram

Figure 2. Bookkeeper Use Cases

### Take Responsibility for a Subtask Use Cases

### Brief Description

The Bookkeeper selects

### Initial Step-By-Step Description

REQ-1: REQ-2:

## Manager Use Cases

## Administrator Use Cases

# 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 should 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.>

The server is directly connected to the client systems. Also the client has the access to the database for accessing the account details and storing the login time.

The client access to the database in the server is only read only.

## 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.>

# 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

The system will support a user authentication system with a role-based access control feature. The roles will be based

## 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.>

# 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: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>