PROJECT MANAGEMENT SOFTWARE

**Software Requirements Specification**

**Version 1.0  
Status: Draft**

**Prepared by: Quansah Anthony J.K.J  
Developer [backend], System Analyst [Interim]**

THE KNIGHTS  
16th April, 2019

**Table of Contents**

**1.** **Introduction**

1.1 Purpose

1.2 Intended Audience and Reading Suggestions

1.3 Product Scope

1.4 References

**2.** **Overall Description**

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

**3.** **External Interface Requirements**

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

**4.** **Domain Model**

**5.** **System Features (Use Cases)**

5.1 Use Case <Insert the specific use case name>

5.2 …

**6.** **Other Nonfunctional Requirements**

6.1 Performance Requirements

6.2 Safety Requirements

6.3 Security Requirements

6.4 Software Quality Attributes

**7.** **Other Requirements**

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** | **Date of Approval** |
| < Author’s Name> | <Date of Revision> | <Description of Made Changes> | <Version No.> | <Date of Approval> |
|  |  |  |  |  |

# **Introduction**

## **Purpose**

This Software Requirement Documentation is exclusively for the Project Management Software under development by the The Knights. This SRS is intended to provide the scope of the aforementioned software, from customer level to developer level. This SRS focuses on the entire software project as a whole and will thus contain all necessary information associated to it.

## **Intended Audience and Reading Suggestions**

## Below is a table that describes the various stakeholders or audience directly associated to the aforementioned software.

|  |  |  |
| --- | --- | --- |
| **ID** | **Stakeholder** | **Description** |
| S-1 | Customer | Checking correspondence of business goals and functionality requirements to the expectations from implementing the product. |
| S-2 | Development team | Forming the accurate vision of the project, detailed functional and nonfunctional requirements. |
| S-3 | QA team | Making test-plans and test-cases. |
| S-4 | PM/BA | Estimating the quote of the project, planning resources and the timeline of work. |
| S-5 | User | Basing on this document the Terms of Service and the Privacy Policy are created. |

## **Product Scope**

The Project Management Software is a web based software program that:

→ Serves as a project repository for approved projects of varying fields.

→ Allows stakeholders to access uploaded projects or upload projects based on different levels of access rights.

→ Stakeholders may decide to register an account or not; stakeholder (users) with accounts will have more access to some features of the software than free roaming users.

The aforementioned software will go a long way in helping users make their research work more easier and relatively faster due to the availability of different reference materials that will help users to increase their options of approach for projects they may be working on. Targeted users are students, tertiary level ones to be precise.

## **References**

The hyperlink below leads directly to the Version control site for the SRS

[**https://github.com/the-Knights**](https://github.com/the-Knights)

*List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information, so that the reader could access a copy of each reference, including title, author, version number, date, and source or location (if applicable)*

# **Overall Description**

This section:

→ contains general view of the project

→ environment where it will be used

→ description of expected user audience and constraints

→ assumptions and dependencies which can be identified

## **Product Perspective**

The Project Management System is a self-contained product jeered towards helping users improve the quality of their work (projects). Though there may be other similar products in market, this product is not in any way a follow-on member of another product family or a replacement for existing systems. It is new and as specified earlier, self-contained.

**Product Functions**

*Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3.  
A picture of the major groups of related requirements and their interactions, such as a top level data flow diagram or object class diagram, is often effective.*

The PMS has the following functions:

→ Users can Sign up for an account

→ Users can upload projects

→ Users can search for available projects

→ Users can download some parts of projects, ie. project reports but not source code or full content of project scope.

→ Other stakeholders (administrator .ie lecturer) can rate supervised projects.

→ Users without accounts cannot upload projects, however they can search and download some projects.

## **User Classes and Characteristics**

|  |  |  |
| --- | --- | --- |
| **ID** | **User classes** | **Description** |
| U-1 | Administrator | A signed up user who has completed the account activation. He owns expanded rights inside the portal. He performs the content pre moderation. |
| U-2 | Signed up user | A user of the portal who has completed the sign up on the portal and the account activation. |
| U-3 | Not signed up user | A user of the portal who has completed neither the sign up on the portal, nor the account activation. He owns limited rights inside the portal. |

## **Operating Environment**

The operating environment for the Project Management System:

→ The PMS will run on the web

→ Suitable operating systems:

\* Windows

\* IOS

\* Linux OS distros

→ The PMS will require a hardware; personal computer, to be able to run

## **Design and Implementation Constraints**

The bullets below enlists the various implementation constraints needed to implement the PMS:

* SPECIFIC TECHNOLOGIES, TOOLS AND DATABASES TO BE USED
  + - python programming language
    - flask
    - react
    - SQLLite

* + - * HARDWARE LIMITATIONS

Internet-In order

Access to relatively fast internet

IDE

BINDING AGREEMENTS

Work agreement between customer and the\_Knight

STANDARD DATA EXCHANGE FORMAT

JSon

DESIGN CONVENTIONS/ PROGRAMMING STANDARDS

The\_Knights will be in charge of maintainance

LANGUAGE REQUIREMENTS

English

COMMUNICATION PROTOCOLS

HTTP

SMTP

SECURITY CONSIDERATIONS

N/A

## **User Documentation**

*List the user documentation components that will be delivered along with the software. They may include user manuals, online help, and tutorials. Identify any known user documentation delivery formats or standards.*

User documentation formats that will be delivered along with the software:

→ Online help

→ Tutorials

→ User documents

## **Assumptions and Dependencies**

**ASSUMPTIONS**

Information was not given.

The development Team; The\_Knights will base on their thoughts and assumptions while developing the required application, however prioritizing clients interests in the chain of decision making.

**DEPENDENCIES**

**NOTE:**

*In case the customer hasn’t given any information on either of the paragraphs, write your vision in the section Assumptions or mark in the corresponding paragraph that the information wasn’t given, so the development team will base on their thoughts and assumptions while developing the corresponding paragraph, taking into account and carefully following the interests of the client.*

*Dependencies - dependencies of the project on external factors like the date of the new OS version release.*

*If the system is planned to include the components developed in another project, then there is a dependency on its timely delivery.*

# **External Interface Requirements**

*Requirements for the external interface define the tools, software or database elements with which the system or the component should interact.*

*Information from this section will give confidence that the system will properly interact with external components.*

*If different parts of the product have different external interfaces, then there should be a section with their description in the detailed requirements for each part.*

## **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.*

**NOTE:**

*Document the user interface details such as dialogue window configuration in the user interface specification, not in SRS.  
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.*

## **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.*

**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.*

# **Domain Model**

<Optional section>  
  
*Domain model describes the entities structure composing the system, their attributes, operations and interconnection with the other entities.*

# **System Features (Use Cases)**

## **Use Case** <Specify the title of a specific use scenario>

**EXAMPLE:**

|  |  |
| --- | --- |
| **Brief Description:** | *<2 to 3 sentences describing the purpose and goals of use case>* |
| **Business Trigger:** | *< the business event that causes the first interaction>* |
| **Preconditions:** | *<the stable state(s) that the system must be in for the use case to start>* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Basic Flow:** *<the complete flow of events that normally happens when everything goes right >* | | | |
| **Assumptions:** XXX*<the assumptions that are made for the basic flow, e.g. all items are for collection.>* | | | |
| **Line** | **System Actor Action** | | **System Response** |
|  | XXX e.g. The sales assistant asks the system to process a new order. *<write a single sentence describing what the system actor does as an interaction across the system boundary>* | | XXX e.g. The system asks the sales assistant for identification *<write a single sentence describing what the system does in response as an interaction across the system boundary, or, if absolutely essential, what the system does internally>* |
|  | XXX e.g. The sales assistant identifies themselves to the system *<add further lines as necessary until the flow is complete>* | | XXX e.g. The system displays a blank order screen with a blank order line to the sales assistant *<keep the action and the response in pairs. If the system does more than one thing, then add another line for each additional action>* |
|  | Xxxx *<do not put conditions in the flow which refer to the alternate flow: describe in the alternate flow where it inserts itself>* | | Xxxx *<when the flows are stable add hyperlink references to alternate and sub-flows>* (AF[x](https://docs.google.com/document/d/1jPAAf35coxgrh4Wnvsz_hvXFwKDLvERdvEEioed0j_M/edit" \l "AFx)) |
| **Post Condition:** | | XXX *<the state of the system at the end of the basic flow, or things guaranteed to be true at the end of a successful use case>* | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Alternate Flow (AFx): XXXX** *< the number and name of the alternate flow starting with an active verb>* | | | |
| If at line x in xxx[x](https://docs.google.com/document/d/1jPAAf35coxgrh4Wnvsz_hvXFwKDLvERdvEEioed0j_M/edit" \l "BF3) *<insert the name of the flow and the number of the line in which the condition occurs>* xxxx xxxx xxxx, then: *<define the condition under which the alternate flow is executed e.g. ‘the item number entered is found to be invalid’. When the flows are stable, insert a hyperlink to the place where the flow inserts itself>* | | | |
| **Line** | **System Actor Action** | | **System Response** |
|  | Xxxx *<write the lines just as you would in the basic course >* | | Xxxx |
|  | Xxxx *<add further lines as necessary until the flow is complete>* | | Xxxx |
| The use case terminates / The use case restarts at line x in [x](https://docs.google.com/document/d/1jPAAf35coxgrh4Wnvsz_hvXFwKDLvERdvEEioed0j_M/edit" \l "BF3) *<the last line must describe what happens next: the use case terminates; the use case restarts where it left off; the use case jumps back and restarts at an earlier step; the use case jumps forward and restarts at later step. When the flows are stable, insert a hyperlink to where the use case restarts>* | | | |
| **Post Condition:** | | XXXX *<you might wish to describe post-conditions for an alternate flow where the use case terminates and the post-conditions are different from those of the basic flow>* | |

|  |
| --- |
| **Alternate Flow (AFx): XXXX** *<add the names of further alternate flows as you think of them. Add the detail of the alternate flow after the basic flow has been detailed. It is possible to have extensions on the extensions. Write them exactly the same way as other extensions>* |

|  |  |  |
| --- | --- | --- |
| **Sub-Flow: XXX** *<where there is procedure that is common to more than one flow in the use case, or where there are different flows following a case statement or selection statement, create sub-flows and ‘call’ them from the using flow. Name the sub-flows starting with an active verb>* | | |
| **Line** | **System Actor Action** | **System Response** |
|  | xxxx xxxx xxxx *<write the lines of the sub-flow just as you would in the basic course, describing what happens largely as interactions across the boundary >* | Xxxx |
|  | xxxx xxxx xxxx *<Add further lines as necessary>* | Xxxx *<When the flows are stable, insert a hyperlinks to where the flows restart>* Return[1](https://docs.google.com/document/d/1jPAAf35coxgrh4Wnvsz_hvXFwKDLvERdvEEioed0j_M/edit" \l "BF3), Return [2](https://docs.google.com/document/d/1jPAAf35coxgrh4Wnvsz_hvXFwKDLvERdvEEioed0j_M/edit" \l "BF1) |

|  |
| --- |
| **Business Rules:** |
| 1. xxxx *<include here a description of any business rules, detailed internal algorithms or procedures that are not part of the externally visible behaviour, but are vital to the functional definition of the system>* 2. xxxx *<if they have already been defined in a business rules document, then include the reference here instead of the description>* |

|  |
| --- |
| **Non Functional Requirements:** |
| 1. xxxx *<include here any non-functional requirements that are specific to this use case.* *Those which apply generally should go in the Non-Functional Requirements Document>* 2. xxxx *<include here any non-functional requirements that are specific to this use case.* *Those which apply generally should go in the Non-Functional Requirements Document>* |

|  |
| --- |
| **Data Requirements:** |
| 1. xxxx *<include here any specific data requirements. If these have already been specified in the Data Dictionary, then include here a reference to the entry instead>* |

|  |
| --- |
| **Activity Diagram:** |
| *<if there is complex iteration and selection, include an activity diagram, or a reference/hyperlink to one here. Activity diagrams should not duplicate or replace the text of the flows but augment it where prose is difficult to use to describe complex conditionality. There is no need to include every line in the use case as an activity.>* |

|  |
| --- |
| **Prototype Screen:** |
| *<if you have a prototype screen for this use case, include it or a reference/hyperlink to it here. If might be at proof of concept or detailed level depending on the importance of the screen design to the user. Make sure the text of the use case is consistent with the prototype >* |

|  |  |  |
| --- | --- | --- |
| **Screen Entry Exception Table:** *<an optional table for simple exception handling of entries into fields on the prototype screen where adding an alternate flow would be overkill and the flow of the use case remains unchanged. This table could be used in place of the Data Dictionary>* | | |
| **Field** | **Constraint** | **Response** |
| 1. e.g. Customer Surname | 30 chars max | Message: “Exceeds 30 chars max – please re-enter” |
|  |  |  |

## **…**

# **Other Nonfunctional Requirements**

## **Performance Requirements**

*State performance requirements*

## **Safety Requirements**

*Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Refer to any external policies or regulations that state safety issues that affect the product’s design or use.*

## **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.*

## **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**

**EXAMPLE:**

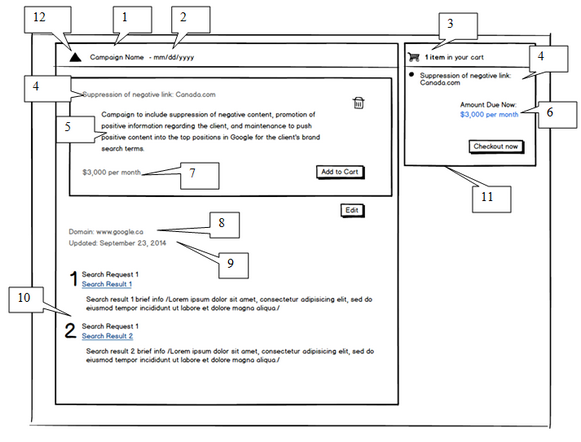
|  |  |
| --- | --- |
| **Term** | **Description** |
| Project | Modification of the existent Reputation.ca website |
| SERP | Search Engine Result Page. Page with the results available to the User after search (clicking on Get Quote button Free Analysis). This page shall contain 1-20 results. |
| TDB | To be detailed |
| Reputation Management | Management of results which were obtained on SERP (view, edit, delete, pay). Available only for the Authorized User. |
| AJAX | A group of interrelated Web development techniques used on the client-side to create asynchronous Web applications (see <http://en.wikipedia.org/wiki/Ajax_(programming)>) |

**Appendix B: Analysis Models**

*An additional optional section which includes models referring to the product described in the SRS, e.g. diagrams of data flows, class diagrams, state machine diagrams, entity-relationship models*.

**Appendix C: Interface Examples**

**EXAMPLE1:**



**Figure 4. Campaigns**

Campaign Name (1) – The keywords entered for search.

Date (2) – The date of the first search of the keywords (1)

Search Result (10) – The results of the latest search for the specified key words (1)

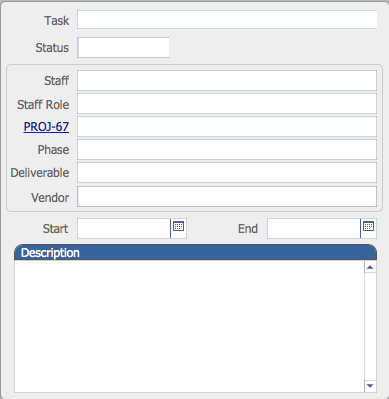
Update date (9) – The date of the last search for the specified key words (1)

Due Amount (6) – The sum payable for all selected items from all Campaigns.

Cart (11) – For all items selected from all Campaigns.

Arrow button (12) – By clicking on this button, the opened Campaign will be folded up. If there are several Campaigns, they shall be displayed as an accordion, each may be unrolled.

**EXAMPLE2:**



|  |  |
| --- | --- |
| **Field** | **Description** |
| Task Name | Editable text field. Max. number of symbols: 100  Symbols type: letters and digits only, both capital and lowercase |
| Status | Not editable text field.  Auto filled. |
| Staff | Not editable text field.  Auto filled. |
| Staff Role | Not editable text field - Link.  Auto filled. |
| PROJ-67 (Project Number) | Not editable dynamic field label from JSON.  Auto filled.  onClick handler shall be put on it. |
| Project Name | Not editable text field.  Auto filled. |
| Phase | Not editable text field.  Auto filled. |
| Deliverable Type | Not editable text field.  Auto filled. |
| Vendor | Not editable text field.  Auto filled. |
| Start Date | Editable.  Field mask:  mm/dd/yyyy  Auto-changeable after the task position change. |
| End Date | Editable.  Field mask:  mm/dd/yyyy  Auto-changeable after the task position change. |
| Description | Editable text field.  Max. number of symbols: 10,000  Symbols type: letters (capital and lowercase), digits and punctuation marks. |

**Appendix D: To Be Determined List**

*This section should include all questions and issues which need clarification.*