Software Requirements Specifications – CSCI311

Dashboard System Project – CSCI311

**Peter Brown**

**Kurt Robinson**

**Joshua Brown**

**Jamie Glennan**

**James Wilson**

**Version 1.01**

CSCI311 Assignment Version: 1.01

Software Requirements Specifications Date: 25/03/2014

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 25/03/2014 | 0.1 | First Draft | Peter Brown |
| 26/03/2014 | 0.2 | Continuation of Draft   * Added first Drafts of Sections 2 & 3 | Peter Brown |
| 27/03/2014 | 0.2.1 | Minor fixing of draft | Peter Brown |
| 03/04/2014 | 1.0 | First draft | Peter Brown |
| 04/04/2014 | 1.01 | Revision of User Management requirements | Peter Brown |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc384400273)

[1. Introduction 4](#_Toc384400274)

[1.1 Purpose 4](#_Toc384400275)

[1.2 Scope 4](#_Toc384400276)

[1.3 Definitions, Acronyms, and Abbreviations 4](#_Toc384400277)

[1.4 References 4](#_Toc384400278)

[1.5 Overview 4](#_Toc384400279)

[2. Overall Description 4](#_Toc384400280)

[2.1 Product perspective 4](#_Toc384400281)

[2.2 Product functions\* 5](#_Toc384400282)

[2.3 User characteristics 6](#_Toc384400283)

[2.4 Constraints 6](#_Toc384400284)

[2.5 Assumptions and dependencies 6](#_Toc384400285)

[3. Specific Requirements 6](#_Toc384400286)

[3.1 Requirements 7](#_Toc384400287)

**Software Requirements Specification**

## 1. Introduction

### 1.1 Purpose

The Purpose of this document is to describe the specifications, requirements and risk evaluation for our Assignment in CSCI311. It contains non-functional requirements, design constraints and other factors necessary to provide a complete and comprehensive understanding of our project. The intended audience of this document includes the Client, Development team and the assignment supervisor/s.

### 1.2 Scope

The aim of this software solution is to develop a tool that can be used as a dashboard for a given software project. It is intended to be able to intake raw data and allow visualization and manipulation of it. It is intended for a client that wants to be able to easily understand a variety of different information types in one easy software solution.

### 1.3 Definitions, Acronyms, and Abbreviations

**Apache Ant** – Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks.

**Python** - Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. The language provides constructs intended to enable clear programs on both a small and large scale

### 1.4 References

### 1.5 Overview

The rest of this document is divided into two main sections:

* The Overall Description (section 2) describes the general factors that affect the system and its requirements.
* The Specific Requirements (section 3) contains all software requirements that the system must meet in order to satisfy customer’s needs.

## 2. Overall Description

### 2.1 Product perspective

Currently there is no existing dashboard system so we are working solely off the requirements and feedback from the user. As there is already a central data store which is provided by the SVN archives we have a localised store of raw information. This information in its raw form is hard to manage and relate unless you are heavily involved with the project and know the design of the data store or have a working knowledge of data repositories. This makes it hard for those not directly involved with the project to track the status of the project and managing the correspondence such as bug reports, update logs and emails sent to the developers.

From this we see the need to centralise the supporting data for the projects and our system aims at extracting the raw data into a dashboard which will be in a more easily interpretable format and represent the project’s information better.

#### 2.1.1 System Interfaces

The Dashboard to be developed is a web-based application that can be integrated to an organization’s

Intranet or deployed on the Internet.

The *Clients* can simultaneously log into the system from any PC that supports Internet services and then view the displayed information in various different representations of their choice.

The *Project Manager* can add and edit the stored data through an administration menu which is available when they are logged into the system. They also can view the information like a client would be able to.

The *Web Server* allows client machines to connect to the Dashboard system and serves their needs via a set of web pages. It acts as a mechanism to accept and process client requests, then retrieve the database and finally return the results to these requests and display them in the correct format.

The *Database* is used to store the system’s user profiles, Dashboards resources and the stored archive of data from the project.

#### 2.1.2 User Interfaces

The user interfaces provided to clients must be user friendly and accessible through the browser (IE, Mozilla, Firefox etc). The user interfaces are different between the Project Manager, Admin and a regular user. Project managers and Admins would be able to manipulate and add data to the system through a separate control panel. Admins would have a higher access which would also allow them to edit the structure of the site and its contents.

All accesses to the Database, which is powered by MySQL server, must be performed indirectly through

GUIs provided by the Dashboard system.

#### 2.1.3 Hardware Interfaces

All components must be able to execute on a personal computer (PC or Mac).

#### 2.1.4 Software Interfaces

##### 2.1.4.1 External System Interface:

* The system is able to import archived data from the SVN.

##### 2.1.4.2 User Interface:\*

* The User must be able to access the dashboard through a web browser
* The System must support the current web browsers of Chrome, IE and Firefox as of March 2014 (\*\*\*\*)

#### 2.1.5 Communication Interfaces\*

* The client machines must communicate with the Web Server over TCP/IP connection
* The Web Server and the Database Server are located on different servers (\*\*\*\*)

#### 2.1.6 Memory Constraints\*

* The client machine must be able to operate within 32MB minimum (including memory for browser)
* The Web Server and the Database Server must be able to operate within 128MB minimum

#### 2.1.7 Operations\*

The Dashboard system must be easy for all users to use, e.g. no specific information or skills (except knowledge on how to access the Internet via Web browser) must be required to use the tool.

The Charting and Data representation must be concise and easy to interpret for people with a decent knowledge of different charting styles.

The Web Server installation and maintenance should be simple enough for a network administrator to perform and should not require any special technical skills from the administrator.

The Database Server should be able to import data from the SVN. Backup and Recovery operations must be specified in case of network failure, database failure, out of power etc.(\*\*\*\*)

### 2.2 Product functions\*

The main functions of the Dashboard system are to retrieve the raw project data, store and display it in a more easily interpretable format such as charts and diagrams.

For Users the Dashboard system helps them to understand the project by (\*\*\*\*)

* Interpreting Raw project Data from the archive and giving it meaning
* Helping them to visualize the project data in a meaningful way which includes:
  + Bug Reports
  + Test Cases
  + Execution traces/logs,
  + Emails
  + Discussions
  + Real world user feedback
  + Maintenance evolution activities
  + Progress

Project Managers must also be able to (in addition to basic users):

* Add and Edit data directly into the system

Admins Must be able to:

* Access all levels of information
* Manage user profiles and accounts on the system

### 2.3 User characteristics

The users of the Dashboard system include System Admins, General Users and Project Managers.

* Administrators have strong knowledge on networks and web applications to be able to install and maintain the Dashboard system.
* Project managers have a good knowledge of the data being worked with and be able to update and change data as well as understand how it is represented.
* General users have a good understanding of the data being represented and its format.

### 2.4 Constraints

The system should strictly obey and satisfy the following constraints:

* Authentication security: the system should enforce user authentication security
* Access control: the system must provide appropriate access right and user interface to each type of user (for instance, administrators and librarians are allowed to access resources profiles and borrower profiles while borrowers are not allowed to)
* Backup and recovery: the backup and recovery of all the system’s database must be easy to perform to prevent databases from corruption and loss risks
* Integrity control: since the system consists of many databases that are correlated with each other, integrity among these databases must be strictly maintained.
* The system must be developed within 13 Weeks and must be released by the end of the session.

### 2.5 Assumptions and dependencies

The following assumptions and dependencies for the system are stated:

* All Users of the system will have an email address
* All Users of the System will understand basic console navigation and be able to read interpreted data

## 3. Specific Requirements

Each requirement (either functional or non-functional one) of the Dashboard system is ranked based on its level of importance.

1. Critical: highest importance level. Critical requirements are those that reflect core functionalities of Dashboard system and must be firstly implemented
2. Essential: second highest importance level. Essential requirements are those that reflect important functionalities of the Dashboard system and must be covered when Critical requirements have successfully implemented
3. Desirable: medium importance level. Desirable requirements are those that reflect necessary but not critical functionalities of the Dashboard system and should be implemented when all Critical and Essential requirements have been fulfilled
4. Optional: lowest importance level. Optional requirements are those that reflect enhanced functionalities of the Dashboard system and should be considered only when all Critical, Essential and Desirable requirements are completed

### 3.1 Requirements

#### 3.1.1 Key requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.1\_01 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The dashboard should be able to import and parse existing raw data | | | | | | |
| **Rationale** | Importing and parsing existing raw data (e.g. bug/issue reports, test cases, execution traces/logs, emails, discussions, real-world user feedback, etc.) From a given large-scale open source project. | | | | | | |
| **Criteria** | Data is stored and translated into relevant data | | | | | | |
| **Source** | SVN Archive, All Users | | | | | | |
| **Created** | Created by Peter Brown 27/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.1\_02 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | Users Must be able to Manipulate Data through the Systems GUI | | | | | | |
| **Rationale** | The user (Project Manager) must be allowed to manipulate the data (e.g. Create new tasks, requests, assigning tasks/requests, generating reports etc.) | | | | | | |
| **Criteria** | Stored data can be manipulated | | | | | | |
| **Source** | Data Store, Project Manager, Admin | | | | | | |
| **Created** | Created by Peter Brown 27/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.1\_03 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The dashboard must translate data into a visual representation | | | | | | |
| **Rationale** | Visualizing those data in a meaningful way to depict the current status of the project in various aspects (e.g. developers’ activities, expertise and networks, user feedback, maintenance and evolution activities and progress, etc.). | | | | | | |
| **Criteria** | Stored data is translated into visual data | | | | | | |
| **Source** | Data Store, Users | | | | | | |
| **Created** | Created by Peter Brown 27/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

#### 3.1.2 User Management Module

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_01 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The dashboard must have a User Management System with differing levels of access | | | | | | |
| **Rationale** | Different Users have access to different levels of information (client) | | | | | | |
| **Criteria** | The dashboard has a user management system that stores a unique username and password for people to log in to the system | | | | | | |
| **Source** | Data Store, Dashboard, User, Admin | | | | | | |
| **Created** | Created by Peter Brown 03/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_02 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The Administrator must be able to add accounts to the User management system | | | | | | |
| **Rationale** | New users must be added to the system (Req# 3.1.2\_07) | | | | | | |
| **Criteria** | The admin is able to add an account to the user management system | | | | | | |
| **Source** | Data Store, Dashboard, Admin, User management System | | | | | | |
| **Created** | Created by Peter Brown 04/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

#### 3.1.3 Data Management Module

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_01 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The data must be pulled from the SVN | | | | | | |
| **Rationale** | The dashboard should be able to import and parse existing raw data (Req# 3.1.1\_01) | | | | | | |
| **Criteria** | Data is retrieved from the SVN | | | | | | |
| **Source** | SVN Archive, Raw Data | | | | | | |
| **Created** | Created by Peter Brown 27/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_02 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The raw data must be translated into information | | | | | | |
| **Rationale** | The dashboard should be able to import and parse existing raw data (Req# 3.1.1\_01) | | | | | | |
| **Criteria** | Data is translated into meaningful information | | | | | | |
| **Source** | Raw Data, Data store, Users | | | | | | |
| **Created** | Created by Peter Brown 27/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

#### 3.1.4 Main Dashboard Module

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_03 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The data must be Accessed from the dashboard | | | | | | |
| **Rationale** | Visualizing those data in a meaningful way to depict the current status of the project in various aspects (e.g. developers’ activities, expertise and networks, user feedback, maintenance and evolution activities and progress, etc.). (Req# 3.1.1\_03) | | | | | | |
| **Criteria** | Data is able to be retrieved from the Data store by the dashboard | | | | | | |
| **Source** | Data Store, Dashboard | | | | | | |
| **Created** | Created by Peter Brown 03/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_04 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The data must be Represented Visually in the dashboard | | | | | | |
| **Rationale** | Visualizing those data in a meaningful way to depict the current status of the project in various aspects (e.g. developers’ activities, expertise and networks, user feedback, maintenance and evolution activities and progress, etc.). (Req# 3.1.1\_03) | | | | | | |
| **Criteria** | Data is able to be shown in a meaningful visual format | | | | | | |
| **Source** | Data Store, Dashboard | | | | | | |
| **Created** | Created by Peter Brown 03/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

#### 3.1.5 Data Manipulation module

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_05 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | New data must be able to be created form the dashboard | | | | | | |
| **Rationale** | The user (Project Manager) must be allowed to manipulate the data (e.g. Create new tasks, requests, assigning tasks/requests, generating reports etc.) (Req#3.1.1\_02) | | | | | | |
| **Criteria** | The user must be able to enter new data | | | | | | |
| **Source** | Data Store, Dashboard, User | | | | | | |
| **Created** | Created by Peter Brown 03/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_06 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | Manipulated data must be saved into the data store | | | | | | |
| **Rationale** | The user (Project Manager) must be allowed to manipulate the data (e.g. Create new tasks, requests, assigning tasks/requests, generating reports etc.) (Req#3.1.1\_02) | | | | | | |
| **Criteria** | The manipulated data is added to the existing data | | | | | | |
| **Source** | Data Store, Dashboard, User | | | | | | |
| **Created** | Created by Peter Brown 03/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

#### 3.1.6 Data Security Module

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | 3.1.2\_09 | **Use Case** | 0 | **Requirement Type** | Functional Requriement | **Rank of Importance** | 1 |
| **Description** | The Data must be secure & backed up | | | | | | |
| **Rationale** | Data must be secure (client) | | | | | | |
| **Criteria** | The data is recoverable in the case of an accident | | | | | | |
| **Source** | Data Store, backup | | | | | | |
| **Created** | Created by Peter Brown 04/03/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

Non-Functional Requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF1 | **Name** | Scalability | **Requirement Type** | Non-Functional | **Rank of Importance** | 6 |
| **Description** | The system should have mechanisms in place to be able to cope with expansion without requiring changes to the internal systems. | | | | | | |
| **Rationale** | By designing the system to cope with expansion it will be able to accommodate an unexpected increase in the size of the system | | | | | | |
| **Source** | Joshua Brown | | | | | | |
| **Created** | 02/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF2 | **Name** | Accessibility | **Requirement Type** | Non-Functional | **Rank of Importance** | 2 |
| **Description** | The system will be able to be accessed by a modern web browser (Chrome, Firefox, IE9+, Safari) | | | | | | |
| **Rationale** | By designing the system to work with modern web browsers, It can prevent issues occurring on the users end. Also for these modern browsers the design is the same | | | | | | |
| **Source** | James Wilson | | | | | | |
| **Created** | 01/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF3 | **Name** | Aesthetics | **Requirement Type** | Non-Functional | **Rank of Importance** | 3 |
| **Description** | The system have a clear and consist user-interface design. The design will be intuitive, consistent. | | | | | | |
| **Rationale** | By having a good design it will allows users to easily navigate and use the site. This will make the users more efficient. | | | | | | |
| **Source** | Peter Brown | | | | | | |
| **Created** | 03/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF4 | **Name** | Performance (Data Store) | **Requirement Type** | Non-Functional | **Rank of Importance** | 5 |
| **Description** | The system should have mechanisms in place to allow the user read and store data in a timely format. | | | | | | |
| **Rationale** | Data not only has to be readable, it must be responsive to the user. If attempting to access data takes too long, the system might think there is something wrong with the media, or the user might be frustrated and less likely to use the system. | | | | | | |
| **Source** | Kurt Robinson | | | | | | |
| **Created** | 02/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF5 | **Name** | Performance(Dashboard) | **Requirement Type** | Non-Functional | **Rank of Importance** | 4 |
| **Description** | The front-end aspect of the system must be responsive to the user. | | | | | | |
| **Rationale** | If the front-end is not responsive or shows no interactivity, it is likely that the user won’t be able to use the system, or won’t want to use the system. | | | | | | |
| **Source** | Peter Brown | | | | | | |
| **Created** | 02/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF6 | **Name** | Performance(Data Manipulation) | **Requirement Type** | Non-Functional | **Rank of Importance** | 7 |
| **Description** | Writing to the data storage module must be consistent with standards for timeliness of data manipulation | | | | | | |
| **Rationale** | Not only is it important to be able to read data, we must also be able to write quickly to the system, including the logging of jobs, bugs and feedback. | | | | | | |
| **Source** | Kurt Robinson | | | | | | |
| **Created** | 02/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF7 | **Name** | Stability | **Requirement Type** | Non-Functional | **Rank of Importance** | 1 |
| **Description** | The system must not perform erratically or show any symptoms of volatility. | | | | | | |
| **Rationale** | Since the system will be interacting with a number of parts, as well as with a number of users, it is important that the system performs exactly as expected. | | | | | | |
| **Source** | Joshua Brown | | | | | | |
| **Created** | 31/03/14 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF8 | **Name** | Sustainablity | **Requirement Type** | Non-Functional | **Rank of Importance** | 8 |
| **Description** | The system will be designed in such a way that it will minimise/remove the amount of need of external interaction with its internal systems | | | | | | |
| **Rationale** | By minimising the exposure to sensitive aspects of the system, we create and insurance against breaking the internal components of the product, as well as create a more secure robust system. | | | | | | |
| **Source** | James Glennan | | | | | | |
| **Created** | 05/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF9 | **Name** | Maintainablity | **Requirement Type** | Non-Functional | **Rank of Importance** | 9 |
| **Description** | The product must be able to be modified and improved as required by the client | | | | | | |
| **Rationale** | By ensuring that the system is well documented, the project can be continued and improved upon as necessary. | | | | | | |
| **Source** | James Wilson | | | | | | |
| **Created** | 01/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requirement#** | NF10 | **Name** | Data Integrity | **Requirement Type** | Non-Functional | **Rank of Importance** | 10 |
| **Description** | Data needs to be correct, without being interfered with by external sources. | | | | | | |
| **Rationale** | This product relies heavily on data being correct and intact for the program to operate effectively. | | | | | | |
| **Source** | James Glennan | | | | | | |
| **Created** | 05/04/2014 | | | | | | |
| **Edited** | n/a | | | | | | |