Software Requirements Specification

for

Professional Developer’s Tool

Prepared by

Team Name: Kazan5

Munir Makhmutov munir.makhmutov@gmail.com

Maxim Ibragimov m.nightbird@gmail.com

Farida Makhmutova farida.makhmutova07@gmail.com

University: Innopolis

Country: Russia

16.09.2014

**Contents**

1 Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms and Abbreviations 4

1.4 References 5

1.5 Overview 5

2 Overall Description 5

2.1 Product perspective 5

2.2 Product functions 5

2.3 User characteristics 5

2.4 Constraints 6

2.5 Assumptions and dependencies 6

3 Specific Requirements 6

3.1 Functional Requirements 6

3.2 Non-Functional Requirements 9

3.3 Use Cases 10

4 Supporting Information 15

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author(s)** |
| 7.10.14 | 0.1 | Introduction and overall description were written | Munir Makhmutov, Maxim Ibragimov, Farida Makhmutova |
| 14.10.14 | 0.2 | Introduction and overall description were changed. Requirements and use cases were written | Munir Makhmutov, Maxim Ibragimov, Farida Makhmutova |
| 18.10.14 | 0.3 | Requirements and use cases were changed | Munir Makhmutov, Maxim Ibragimov, Farida Makhmutova |

# Introduction

## Purpose

This document designed for the developers that will implement this product <Product name>. This document describes functional and non-functional requirements of product <Product name>.

## Scope

Document describes a web-based project management system named <Product name> that helps to support distributed software development. This product should provide different permissions for different groups of people (developers, managers and stakeholders). It should provide creating/deleting accounts, communication tools for members, possibility of assigning tasks to developers, controlling of the project execution, tracking progress, statistics of developers’ work and etc.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Notion** | **Definition** |
| Scrum | Scrum is an iterative and incremental agile software development framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project. |
| Backlog | The product backlog is an ordered list of requirements that is maintained for a product. It consists of features, bug fixes, non-functional requirements, etc.—whatever needs to be done in order to successfully deliver a viable product. The product backlog items (PBIs) are ordered by the Product Owner based on considerations like risk, business value, dependencies, date needed, etc. |
| Sprint log | A time period (typically 1–4 weeks) in which development occurs on a set of backlog items that the team has committed to. |
| SRS | A software requirements specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements. (Non-functional requirements impose constraints on the design or implementation such as performance engineering requirements, quality standards, or design constraints.) The specification may include a set of use cases that describe interactions the users will have with the software. |
| DOSE | Distributed and Outsourced Software Engineering is a distributed project in which teams/developers over several places collaborate |
| Authentication | It is the act of confirming the truth of an attribute of a single piece of data (datum) or entity. In contrast with identification which refers to the act of stating or otherwise indicating a claim purportedly attesting to a person or thing's identity, authentication is the process of actually confirming that identity. |

## References

IEEE Recommended Practice for Software Requirements Specifcations (IEEE Std 830-1998). Can be obtained from <https://code.google.com/p/touch-seen/downloads/detail?name=IEEE-STD-830-1998.pdf>

## Overview

SRS organized as follows. General factors that affect the product and its requirements are described in the second part. The third part consists of all requirements and Use-cases.

# Overall Description

## Product perspective

Product is independent and totally self-contained. Product may be used next years for providing DOSE course.

## Product functions

Basic functionality of the product:

* Backlog: a collection of requirements that need to / could be implemented;
* Sprint log: a log that is used as part of a development sprint;
* User management: users need to be able to register with the system; users need to be able to assign users to projects; basic user management (delete account, change personal details);
* Support for different roles (for example developers and ‘master’ – similar to SCRUM master). Functionalities are available depending on user roles;
* User authentication;
* Users can be assigned to requirements; maybe requirements need to be broken down into smaller tasks;
* Requirements can be assigned with ‘points’ according to their complexity. One can also assign point to tasks. Users who implement the requirements/tasks get the ‘points’ of the implemented requirements;
* Developers chart: all developers are ranked according to the ‘points’ received from the implemented tasks/requirements.

Advanced functionalities:

* A dashboard that summarizes ongoing events of (a) project(s);
* Statistics that show how the project performances, compared to the estimates;
* Team communication tools (e.g. a discussion board or group chat);
* Integration with issue trackers (e.g. the tracker from the where the source code is hosted Github, Bitbucket, etc) or other platforms.

## User characteristics

Users of the system are groups of people from around the world.

They are three types of the users in the system:

1. Developers - people who implement a product.
2. Managers – people who track the product creation.
3. Stakeholders – people with an interest in a product.

## Constraints

1. System should be web-based.
2. The product should be implemented in Eiffel programming language.

## Assumptions and dependencies

The system operates only on the condition that all the necessary equipment for it is in working condition. The system operates only on the condition that between its nodes and the user side there is a communication channel.

# Specific Requirements

This section contains all software requirements both functional and non-functional. A requirement has the following properties:

|  |  |
| --- | --- |
| **Requirement ID** | Uniquely identifies requirement |
| **Title** | Gives the requirement a symbolic name |
| **Description** | The definition of the requirement |
| **Priority** | Defines the order in which requirements should be implemented. Priorities are designated (highest to lowest) from 1 to 3. Requirements of priority 1 are mandatory; 2 represents features nice to have, and 3 represents optional features. |
| **Risk** | Specifies the risk of not implementing the requirement. It shows how critical the requirement is to the system as a whole. The following risk levels are defined over the impact of not being implemented correctly.   * Critical (C) It will break the main functionality of the system. The system cannot be used if this requirement is not implemented. * High (H) It will impact the main functionality of the system. Some function of the system could be inaccessible, but the system can be generally used. * Medium (M) It will impact some system features, but not the main functionality. The system can still be used with some limitation. * Low (L) The system can be used without limitation, but with some workarounds. |

.

## Functional Requirements

|  |  |
| --- | --- |
| **Requirement ID** | FR-1 |
| **Title** | The system should allow user to create and manage its own account |
| **Description** | To use the system user have to log in or register. Logged in users can change profile settings and delete account |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-2 |
| **Title** | The system should allow developer to create and manage a project |
| **Description** | To start project developer have to create it. After creation developer becomes the manager of this project. Also manager can change settings of this project or delete it |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-3 |
| **Title** | The system should allow manager to create and manage a backlog |
| **Description** | After project creation manager have to create backlog. Also, if necessary, manager can edit or delete it |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-4 |
| **Title** | The system should allow manager to create and manage a sprint logs |
| **Description** | After backlog creation manager must create a sprint logs. Also, if necessary, manager can edit or delete it. If sprint is started, then manager cannot edit or delete the backlog |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-5 |
| **Title** | The system should allow developers and manager to see project’s backlog |
| **Description** | Developers, manager of the project must have opportunity to see project’s backlog |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-6 |
| **Title** | The system should allow developers and manager to see project’s sprint logs |
| **Description** | Developers and manager of the project must have opportunity to see project’s sprint logs |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-7 |
| **Title** | The system should allow manager to assign tasks with points to the developers |
| **Description** | Manager can check tasks with different points and assign them to different developers of current project |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-8 |
| **Title** | The system should allow manager to divide sprint log into tasks and manage these tasks |
| **Description** | Manager can create, modify and delete tasks, that then will be assigned to developers |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-9 |
| **Title** | The system should allow manager and developer to see his/her tasks |
| **Description** | Manager/developer can see tasks that were assigned to him/her by himself/manager |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-10 |
| **Title** | The system should allow user to see project statistics |
| **Description** | Users can see projects’ statistics that represents percent of their execution with dates when sprint logs were finished |
| **Priority** | Nice to have (2) |
| **Risk** | Medium (M) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-11 |
| **Title** | The system should allow user to chat with other users |
| **Description** | Users can communicate with each other. Also it is possible to create group chats. After project creation group chat with all team members automatically appears. If project manager adds new team member, he/she automatically joins to project chat |
| **Priority** | Nice to have (2) |
| **Risk** | Medium (M) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-12 |
| **Title** | The system should allow forming chart of developers according to their points |
| **Description** | Developers and managers can see developers’ chart. Chart is formed in order to their developers’ points, that they gained by tasks’ performing. Also it can be possible to use other kinds of order to form chart |
| **Priority** | Nice to have (2) |
| **Risk** | Medium (M) |

|  |  |
| --- | --- |
| **Requirement ID** | FR-13 |
| **Title** | The system should allow user to see ongoing events on dashboard |
| **Description** | Users can see dashboard. Dashboard shows ongoing events of the users’ project. If user has more than one project, he sees only one dashboard with ongoing events of every project (it must be clear that every event belongs to its project) |
| **Priority** | Nice to have (2) |
| **Risk** | Medium (M) |

## Non-Functional Requirements

|  |  |
| --- | --- |
| **Requirement ID** | NFR-1 |
| **Title** | The system should be fault tolerant (availability) |
| **Description** | The system must be available at most of time (not less than 99% of time) |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | NFR-2 |
| **Title** | The system should provide data integrity |
| **Description** | All information of this system, including databases, software, the executable code components of the system and chat history and all the data should not be lost, also system should have redundant copies |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | NFR-3 |
| **Title** | The system should provide confidentiality |
| **Description** | Access to certain information should be provided only to corresponding authorized users, which have permissions to it. All passwords should be stored in the system in an encrypted form. Forgotten Password Recovery possible only through the request of the new password |
| **Priority** | Mandatory (1) |
| **Risk** | Critical (C) |

|  |  |
| --- | --- |
| **Requirement ID** | NFR-4 |
| **Title** | The system should be user friendly |
| **Description** | Interface of the system should be intuitively understandable by the user |
| **Priority** | Nice to have (2) |
| **Risk** | Medium (M) |

## Use Cases

|  |  |
| --- | --- |
| **Use case ID:** | UC-1 |
| **Title:** | Account managing |
| **Aim:** | To create and manage user’s account |
| **Covered requirements:** | 1 |
| **Primary actor:** | User |
| **Stakeholder interest:** | Project development, opportunity of communication with other users |
| **Level:** | Account |
| **Precondition:** | User needs account |
| **Main success scenario:** | 1. User registers 2. User logs in 3. User modifies account settings 4. User logs out   **Post condition:**  User logged out |

|  |  |
| --- | --- |
| **Use case ID:** | UC-2 |
| **Title:** | Account deleting |
| **Aim:** | To delete user’s account |
| **Covered requirements:** | 1 |
| **Primary actor:** | User |
| **Stakeholder interest:** | Project development, opportunity of communication with other users |
| **Level:** | Account |
| **Precondition:** | User has account |
| **Main success scenario:** | 1. User logs in 2. User deletes account 3. User fills the form why does he deletes his/her account 4. Information about user’s projects, position in developers chart dissapears   **Post condition:**  User deleted his/her account |

|  |  |
| --- | --- |
| **Use case ID:** | UC-3 |
| **Title:** | Project developing |
| **Aim:** | To create and manage project |
| **Covered requirements:** | 2 |
| **Primary actor:** | Developer that becomes manager |
| **Stakeholder interest:** | Project creation and managing |
| **Level:** | Project |
| **Precondition:** | Developer is registered and logged in |
| **Main success scenario:** | 1. Developer creates new project and becomes its manager 2. Manager adds developers to project 3. Manager changes project‘s configuration if it’s necessary   **Post condition:**  The project is created |

|  |  |
| --- | --- |
| **Use case ID:** | UC-4 |
| **Title:** | Project closing |
| **Aim:** | To close project |
| **Covered requirements:** | 2 |
| **Primary actor:** | Manager |
| **Stakeholder interest:** | Project deleting |
| **Level:** | Project |
| **Precondition:** | Manager created a project |
| **Main success scenario:** | 1. Manager logs in 2. Manager closes the project 3. Project’s developers become free   **Post condition:**  The project is closed |

|  |  |
| --- | --- |
| **Use case ID:** | UC-5 |
| **Title:** | Backlogs, sprint logs and tasks creation and managing |
| **Aim:** | To create and manage backlog and sprint logs |
| **Covered requirements:** | 3, 4, 8 |
| **Primary actor:** | Manager |
| **Stakeholder interest:** | Project realization |
| **Level:** | Project |
| **Precondition:** | Project is created |
| **Main success scenario:** | 1. Manager creates backlog for project 2. Manager divides backlog on sprint logs 3. Manager divides sprint logs into tasks that have to be assigned to developers 4. Manager changes and terminates backlogs, sprint logs and tasks when it’s necessary   **Post condition:**  Backlog, sprint logs and tasks are created |

|  |  |
| --- | --- |
| **Use case ID:** | UC-6 |
| **Title:** | Backlogs, sprint logs and tasks review |
| **Aim:** | To show backlogs, sprint logs and tasks to developers and managers |
| **Covered requirements:** | 5, 6, 9 |
| **Primary actor:** | Developer, Manager |
| **Stakeholder interest:** | Project realization |
| **Level:** | Project |
| **Precondition:** | 1. User is logged in  2. Developer/manager is a member of at least one project  3. Backlog, sprint logs and tasks are created |
| **Main success scenario:** | 1. Developer/manager goes to list of his/her projects 2. Developer/manager chooses one of them 3. Developer/manager sees backlog of chosen project 4. Developer/manager expands backlog’s sprint logs 5. Developer/manager sees his/her tasks   **Post condition:**  Backlog, sprint logs and tasks are shown |
| **Alternative scenario:** | 1. Developer/manager goes to list of his/her projects 2. Developer/manager doesn’t see the backlogs, sprint logs and tasks   **Post condition:**  Backlog, sprint logs and tasks aren’t shown |

|  |  |
| --- | --- |
| **Use case ID:** | UC-7 |
| **Title:** | Assigning tasks |
| **Aim:** | Manager assigns tasks with points to the developers |
| **Covered requirements:** | 7 |
| **Primary actor:** | Manager |
| **Stakeholder interest:** | Manager wants to assign task to the developers to start project implementation |
| **Level:** | Task |
| **Precondition:** | 1. Manager is authorized 2. Project exists 3. Task of the project exists |
| **Main success scenario:** | 1. Manager opens project 2. Manager opens project’s tasks 3. Manager chooses project’s task 4. Manager assigns the task to a developer [or to himself]   **Post condition:**  Task is assigned to developer |

|  |  |
| --- | --- |
| **Use case ID:** | UC-8 |
| **Title:** | Developers chart |
| **Aim:** | The system should allow forming chart of developers |
| **Covered requirements:** | 12 |
| **Primary actor:** | Developer, Manager |
| **Stakeholder interest:** | Ranking developers by their points |
| **Level:** | Project |
| **Precondition:** | Developer is authorized |
| **Main success scenario:** | 1. Developer/manager opens chart 2. Developer/manager chooses criteria of ranking 3. System forms list of developers according chosen criteria (points and etc) |

|  |  |
| --- | --- |
| **Use case ID:** | UC-9 |
| **Title:** | Project’s progress |
| **Aim:** | To show ongoing events on dashboard and project statistics |
| **Covered requirements:** | 10, 13 |
| **Primary actor:** | User |
| **Stakeholder interest:** | User wants to see project’s progress |
| **Level:** | Project |
| **Precondition:** | 1. User is authorized 2. Project exists |
| **Main success scenario:** | 1. User opens project 2. User sees ongoing events on dashboard 3. User chooses parameters that he wants to analyze in project’s statistics 4. User gets project’s statistics |

|  |  |
| --- | --- |
| **Use case ID:** | UC-10 |
| **Title:** | Chat |
| **Aim:** | To organize communication between users |
| **Covered requirements:** | 11 |
| **Primary actor:** | User |
| **Stakeholder interest:** | Users need to discuss projects |
| **Level:** | Chat |
| **Precondition:** | User is authorized |
| **Main success scenario:** | 1. User goes to a chat screen 2. User chooses another user to chat with him/her or creates chat for a group of users 3. User enters a message 4. User sends a message 5. Other members/another member of chat receive/receives this message   **Post condition:**  Message is registered in chat |