Software Requirements Specification

for

DOSE MS

Prepared by

Team Name: Kazan1

|  |  |
| --- | --- |
| Arthur Samigullin | samigullin.art@gmail.com |
| Dina Gimadieva | dina.gimadieva@gmail.com |
| Galiya Zainagtdinova | galia.mirland@gmail.com |

University: Innopolis University

Country: Russia

7.10.2014

**Contents**

1 Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms and Abbreviations 4

1.4 References 4

1.5 Overview 5

2 Overall Description 6

2.1 Product perspective 6

2.2 Product functions 6

2.3 User characteristics 6

2.4 Constraints 6

2.5 Assumptions and dependencies 6

3 Specific Requirements 7

3.1 Functional Requirements 7

3.2 Non-Functional Requirements 7

3.3 Use Cases 10

4 Supporting Information 10

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author(s)** |
| 7.10.2014 | 0.9 | Items 1.1 – 2.3 | Dina Gimadieva  Arthur Samigullin  Galiya Zainagtdinova |
| 14.10.2014 | 0.99 | Items 2.4 – 3.2 | Dina Gimadieva  Arthur Samigullin  Galiya Zainagtdinova |
| 18.10.2014 | 1 alpha | Items 3.3, 4;  correcting mistakes | Dina Gimadieva  Arthur Samigullin  Galiya Zainagtdinova |

# Introduction

## Purpose

The purpose of this document is to present a detailed description of the web-based project management system to support distributed software development. 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 proposed to the DOSE course’s organizers for its approval.

## Scope

This software system will be a management system for group of developers working on one project in one or many locations. This system will be designed to provide ability of record and track progress through successive iterations (sprints) for developing team members, managers and other stakeholders.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Team | Group of developers working on one project in one or many locations accountable for managing, organizing and doing all development work required to add releasable functionality to Product every Sprint. |
| Guest | Unauthorized/unregistered user. |
| User | Person, registered in the system. |
| Developer (D) | Member of Team, involved in the development. By Project Manager’s appointment confirmation, User becomes Developer for that project. |
| Team Leader (TL) | The person responsible for the underlying architecture for the software program, as well as for overseeing the work being done by any other software engineers working on the project. A lead programmer will typically also act as a mentor for new or lower-level software developers or programmers, as well as for all the members on the development team. By Project Manager’s appointment confirmation, User becomes Team Leader for that project. |
| Project Manager (PM) | The person responsible for maintaining the Product Backlog by representing the interests of the stakeholders, and ensuring the value of the work the Development Team does. User becomes Project Manager after creating his own project. |
| Superuser | Administrator of the system, who has rights to add, remove and edit other users. |
| Product | Software need to be designed by Team. |
| Increment | A piece of working software that adds to previously created Increments, where the sum of all Increments - as a whole - forms a Product. |
| Sprint | A time-bounded basic unit of development. Each sprint adds Increment to Project. Sprints are done consecutively, without intermediate gaps. |
| Product Backlog | An ordered list of the work to be done in order to create, maintain and sustain a product. Managed by the Product Owner. |
| Sprint backlog | An overview of the development work to realize a Sprint’s goal, typically a forecast of functionality and the work needed to deliver that functionality. Managed by the Development Team. |
| Requirements | A singular documented physical and functional need that a particular design, Project‘s product must be able to perform. Requirements can be broken down into smaller Tasks. |
| Task | Piece of working software, which one Developer can implement in one Sprint. |
| DOSE | Distributed and Outsourced Software Engineering. |
| Burn-down chart | A graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal. That is, it is a run chart of outstanding work. It is useful for predicting when all of the work will be completed. |
| Owner | Person, who delegates requirement/task to execute (it can be Project Manager or Team Leader). |
| Executor | Person, who accepted with delegation of requirement/task (it can be every role in project). |

## References

You can read about Scrum at Scrum Home website:

1. About terms

https://www.scrum.org/Resources/Scrum-Glossary

1. About Scrum generally

http://www.scrumguides.org/scrum-guide.html

## Overview

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product and describes the general factors that affect the product and its requirements. It 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

The system is being worked on as semester-long assignment for Distributed and Outsourced Software Engineering course.

In addition, it will be used in non-commercial, academic environment, to learn DOSE and project management principles.

Product can be useful for another distributed development teams.

## Product functions

Main functions of Product are to provide:

1. Tasks allocating
2. Labor input counting
3. Progress recording
4. Progress tracking
5. Developers ranking
6. Supporting different roles

## User characteristics

DOSE course organizers and students will use system – distributed development Teams (including product owners).

Each user must have PC or laptop with Internet (Intranet) connection to access the system.

User is supposed to have basic English understanding.

## Constraints

System should be implemented on Eiffel programming language as a web-based system.

## Assumptions and dependencies

System should be compatible and correctly working with Mozilla, Google Chrome, Apple Safari and Internet Explorer (8 and higher versions), have English language as a default.

System should guarantee stable operation for 100 contemporaneous users.

# 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** | 1 |
| **Title** | Project Backlog |
| **Description** | The system must provide creation, modification, deletion list of all projects requirements and tasks (project backlog). Only Project Manager can manipulate with project backlog. |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 2 |
| **Title** | Sprint backlog |
| **Description** | The system must provide ability to choose certain number of tasks from the Project Backlog to perform in the next Iteration. |
| **Priority** | 1 (Mandatory) |
| **Risk** | H (High-Level) |

|  |  |
| --- | --- |
| **Requirement ID** | 3 |
| **Title** | User registration |
| **Description** | The system must provide the ability to create an user account by him/herself or by Superuser. To create an account consists of entering personal information, setting identifier and authenticator. |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 4 |
| **Title** | Appointing user to project |
| **Description** | The system must provide the ability user to create project with adding other users and appointing roles to them.  Added user must get notification about appointing by system or by email with invitation to the project. He/she should confirm appointing to join the project. |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 5 |
| **Title** | Basic user management |
| **Description** | The system must provide the ability user to edit and delete his/her account. User cannot change identifier, but can edit personal information and setting other authenticator. |
| **Priority** | 2 (Nice to have) |
| **Risk** | L (Low) |

|  |  |
| --- | --- |
| **Requirement ID** | 6 |
| **Title** | Support for different roles |
| **Description** | System should has two levels: basic roles, for all system, and roles in some project. The system should provide basic roles: Superuser and User. User, while working on the project, has one of the following roles: Project Manager, Team Leader or Developer. |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 7 |
| **Title** | User authentication |
| **Description** | The system must provide authentication to protect against unauthorized access. |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 8 |
| **Title** | Assign user to requirements |
| **Description** | The system should provide ability to assign user to requirements (or tasks) by Project Manager. Only assigned user (owner and executor) can change status of requirement/task. |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 9 |
| **Title** | Requirements decomposition |
| **Description** | The system should provide ability to break down requirements to tasks. |
| **Priority** | 2 (Nice to have) |
| **Risk** | M (Medium) |

|  |  |
| --- | --- |
| **Requirement ID** | 10 |
| **Title** | Labor input counting |
| **Description** | System should provide assigning “points” according to complexity of requirements (tasks). Users who implement the requirements/tasks get the ‘points’ of the implemented requirements. |
| **Priority** | 1 (Mandatory) |
| **Risk** | H (High) |

|  |  |
| --- | --- |
| **Requirement ID** | 11 |
| **Title** | Developers chart |
| **Description** | System should provide ranking developers according to the ‘points’ received from the implemented tasks/requirements. |
| **Priority** | 1 (Mandatory) |
| **Risk** | H (High) |

|  |  |
| --- | --- |
| **Requirement ID** | 12 |
| **Title** | User’s dashboard |
| **Description** | System may provide dashboard, which summarizes ongoing events of (a) project(s). Dashboard must show information about user's current projects. |
| **Priority** | 3 (Optional) |
| **Risk** | L (Low) |

|  |  |
| --- | --- |
| **Requirement ID** | 13 |
| **Title** | Statistical reports |
| **Description** | System may provide automatically formed statistical reports, which show how the project performances, compared to the estimates (burn-down chart). |
| **Priority** | 2 (Nice to have) |
| **Risk** | L (Low) |

|  |  |
| --- | --- |
| **Requirement ID** | 14 |
| **Title** | Team communication tools |
| **Description** | The system may provide communication tools. User(s) can communicate in private or group chats. |
| **Priority** | 3 (Optional) |
| **Risk** | L (Low) |

|  |  |
| --- | --- |
| **Requirement ID** | 15 |
| **Title** | Integration with issue trackers |
| **Description** | The system may provide integration with tracking systems. |
| **Priority** | 3 (Optional) |
| **Risk** | L (Low) |

## Non-Functional Requirements

|  |  |
| --- | --- |
| **Requirement ID** | 16 |
| **Title** | Maintainability and Extensibility |
| **Description** | System should be easy to extend with new functionality, well documented on English, and system’s code must be readable, maintainable and written in a way that it favors implementation of new functions. |
| **Priority** | 2 (Nice to have) |
| **Risk** | L (Low) |

|  |  |
| --- | --- |
| **Requirement ID** | 17 |
| **Title** | Security |
| **Description** | System must maintain adequate security level with access control lists (described in Appendix 1) and appropriate actions with requirement’s/task’s status for different roles (described in Appendix 2) |
| **Priority** | 1 (Mandatory) |
| **Risk** | C (Critical) |

|  |  |
| --- | --- |
| **Requirement ID** | 18 |
| **Title** | Robustness |
| **Description** | System must inform users about system’s errors (such as losing connection to the Internet or incorrect scenarios) and provide instruments to make a backup of user’s projects. |
| **Priority** | 2 (Nice to have) |
| **Risk** | M (Medium) |

## Use Cases

|  |  |
| --- | --- |
| **Title:** | Create account |
| **Aim:** | To create an account in the system |
| **Primary actor:** | User |
| **Level:** | System Level |
| **Precondition:** | Identifier must be unique |
| **Trigger:** | Click registration unit, enter personal information, set identifier and authenticator |
| **Main success scenario:** | 1. User starts registration 2. User enters personal information. 3. User sets unique identifier 4. User sets authenticator   Post conditions:  User has his own account; user signed in (logged in) the system and has ability to do actions in the system. |
| **Alternative scenarios:** | 1. User entered reserved identifier or another incorrect information   Post conditions:  User sees the message that he/she entered something wrong and he/she should try again. |

|  |  |
| --- | --- |
| **Title:** | Log in |
| **Aim:** | To enter the system |
| **Primary actor:** | User |
| **Level:** | System Level |
| **Precondition:** | User must have an account in the system |
| **Trigger:** | Click to log in, enter the identifier and the authenticator |
| **Main success scenario:** | 1. Click to log in  2. User enter the identifier  3. User enter the authenticator  Post conditions:  User logged in the system and has ability to do actions in the system. |
| **Alternative scenarios:** | 1. User enter identifier or authenticator wrong  Post conditions:  User sees the message that he/she entered identifier or authenticator wrong. |

|  |  |
| --- | --- |
| **Title:** | User management |
| **Aim:** | To edit or delete user account |
| **Primary actor:** | User |
| **Level:** | System Level |
| **Precondition:** | User must be logged in |
| **Trigger:** | Open the user management unit |
| **Main success scenario:** | 1. Click user management  2. User edits personal information or(and) set authenticator  Post conditions:  User’s details have been modified.  1. User deletes his account  Post conditions:  User has been logged out; account has been deleted; user cannot sign in the system using his details (email/username and password) |
| **Alternative scenarios:** | 1. User enters something wrong   Post conditions:  User sees the message that he/she entered something wrong and should try again |

|  |  |
| --- | --- |
| **Title:** | Create Project |
| **Aim:** | To create a new project in the system. |
| **Primary actor:** | User (Project Manager) |
| **Level:** | Project Level |
| **Precondition:** | User must be logged in. |
| **Trigger:** | Click to create a new project |
| **Main success scenario:** | 1. Add a new project  2. Enter project information  3. Add(invite) new team member(s)  Post conditions:  New 'upcoming' project will be added to the board, below any other future projects. |
| **Alternative scenarios:** | 1. User enter something wrong   Post conditions:  User sees the message that he/she entered something wrong and he/she should try again |

|  |  |
| --- | --- |
| **Title:** | Assign complexity |
| **Aim:** | To assess and record complexity ‘points‘ to requirement/task |
| **Primary actor:** | Team Leader |
| **Level:** | Project |
| **Precondition:** | There are existed at least one requirement/task without assessed complexity in sprint backlog |
| **Trigger:** | Project Manager formed sprint backlog |
| **Main success scenario:** | 1. Team Leader picks some requirement/task from sprint backlog 2. Team Leader records complexity ‘points‘ to requirement/task 3. Team Leader saves changes for requirement/task 4. System informs Team Leader about successful saving   Post conditions:  Requirement/task has its own complexity ’points‘; other users can see complexity |
| **Alternative scenarios:** | 1. Team Leader saves changes for requirement/task 2. System displays error message and recommendation to retry   Post conditions: |

|  |  |
| --- | --- |
| **Title:** | Record progress |
| **Aim:** | To update project completeness |
| **Primary actor:** | Team Member’s: Project Manager, Team Leader, Developer |
| **Level:** | Project |
| **Precondition:** | Team Member has assigned requirement/task as Owner or Executor |
| **Trigger:** | Team Member did some progress in requirement/task |
| **Main success scenario:** | 1. Team Member picks requirement/task 2. Team Member assigns new completeness % or accepts 100% completed requirement/task 3. Team Member saves changes for requirement/task 4. The system informs Team Leader about successful saving   Post conditions:  If the task has 100%, it gets ‘done’ status. |
| **Alternative scenarios:** | 1. Team Leader saves changes for requirement/task 2. System shows appropriated error message and advices to retry saving   Post conditions: |

|  |  |
| --- | --- |
| **Title:** | Assign |
| **Aim:** | To assign requirements/tasks from sprint backlog to team members |
| **Primary actor:** | Owner (Project Manager or Team Leader) |
| **Level:** | Project |
| **Precondition:** | Project Manager formed sprint backlog; the task/requirement already exists |
| **Trigger:** | Owner decided to delegate a requirement/task |
| **Main success scenario:** | 1. Owner picks preferred requirement/task 2. Owner selects Executor from Team Members 3. System sends notification about delegation to Executor 4. Call <Assign> 5. Owner receives message about delegation acceptance   Post conditions:  Executor has new requirement/task to do |
| **Alternative scenarios:** | 1. Call <Accept Delegation> 2. Owner receives message about delegation rejection   Post conditions:  Requirement/task hasn’t been assigned to Executor |

|  |  |
| --- | --- |
| **Title:** | Track progress |
| **Aim:** | To see how project is going on time, the remain aware of teamwork |
| **Primary actor:** | Team Member’s: Project Manager, Team Leader, Developer |
| **Level:** | Project |
| **Precondition:** | Project has at least one requirement or task |
| **Trigger:** | Team Member decided to see how project is going on time |
| **Main success scenario:** | 1. Team Member opens project’s/sprint’s backlog 2. Team Member watch for requirements/tasks statuses 3. Team Member builds burnt-down chart or watches for another statistical instrument (if available)   Post conditions: |

|  |  |
| --- | --- |
| **Title:** | Accept delegation |
| **Aim:** | To accept/Decline delegated requirements/tasks from sprint backlog |
| **Primary actor:** | Executor |
| **Level:** | Project |
| **Precondition:** | Owner delegated requirement/task for Executor |
| **Trigger:** | System sent notification about delegation to Executor |
| **Main success scenario:** | 1. Executor read information about delegation 2. Executor decides to Accept or to Decline delegation 3. System sends message about successive delegation to Owner   Post conditions:  If delegation has been accepted, Executor gets new requirement/task to do. |
| **Alternative scenarios:** | 1. System sends message about failed delegation to Owner   Post conditions:  Delegation rejected |

|  |  |
| --- | --- |
| **Title:** | Accept completed |
| **Aim:** | To check and confirm completeness of delegated requirements/tasks |
| **Primary actor:** | Owner (Project Manager or Team Leader) |
| **Level:** | Project |
| **Precondition:** | Executor accepted requirement/task delegation |
| **Trigger:** | Executor record 100% progress (‘completed’ status) to delegated requirement/task |
| **Main success scenario:** | 1. System sent notification to Owner about end of requirement/task 2. Owner checks completed work 3. Owner accept completed requirement/task 4. Executor gets requirement’s/task’s complexity ‘points’   Post conditions:  Task has been completed, Executor got complexity ‘points’ |
| **Alternative scenarios:** | 1. Owner checks completed work 2. Owner decline completed requirement/task and leaves comments 3. Executor gets notification to refine requirement/task   Post conditions:  Task is not completed, Executor got comments to refine requirement/task |

|  |  |
| --- | --- |
| Title: | New task/requirement |
| Aim: | To create new task/requirement |
| Primary actor: | Project Manager |
| Level: | Project |
| Precondition: | User is logged in the system and has permission to create task/requirements (has Project Manager role in Project). |
| Trigger: | Project Manager discovered new system’s requirement or decided to break down existed one. |
| Main success scenario: | 1. Open **Create Task/requirement** 2. Fill the fields 3. Create task.   **Post conditions:** New 'upcoming' task will be added to the project board, below any other future tasks. |

|  |  |
| --- | --- |
| Title: | Edit task/requirement |
| Aim: | To modify task/requirement’s details |
| Primary actor: | Project Manager |
| Level: | Project |
| Precondition: | The task/requirement exists; user is logged in and has permission to edit task/requirements |
| Trigger: | Project Manager discovered new information about system’s requirement/task. |
| Main success scenario: | 1. Locate the task/requirement user wants to edit. 2. Open the Edit Task/requirement info 3. Modify task/requirement's details 4. Save your changes.   **Post conditions:** |

|  |  |
| --- | --- |
| **Title:** | Add member |
| **Aim:** | To add new member to a project group. |
| **Primary actor:** | Project Manager |
| **Level:** | Project |
| **Precondition:** | User (PM) is logged in and has permission to add new users to some existing project |
| **Trigger:** | Project Manager decided to add new team member |
| **Main success scenario:** | 1. Choose project**.** 2. Choose person (by username for signed up Users, and by email for otherwise), which you want to invite. 3. Set role for invited person – Developer or Team Leader (if Team Leader not exists).   Post conditions: New user has been added to project; user gets permissions (according to his role) for actions in the system |

|  |  |
| --- | --- |
| Title: | New sprint |
| Aim: | To create new sprint |
| Primary actor: | Project Manager |
| Level: | Sprint |
| Precondition: | User (PM) is logged in and has permission to create sprints |
| Trigger: | Team needs a new sprint to work with |
| Main success scenario: | 1. Select preferred project 2. Create Sprint   Post conditions:  New 'upcoming' sprint will be added to the board, below any other future sprints. |

|  |  |
| --- | --- |
| Title: | Starting a sprint |
| Aim: | To start a sprint |
| Primary actor: | Project Manager |
| Level: | Sprint |
| Precondition: | User (PM) is logged in and has permission to start sprint. |
| Trigger: | Team needs to start a sprint. |
| Main success scenario: | 1. Select preferred project 2. Select preferred sprint. 3. Drag tasks up from the backlog, as needed. 4. Start Sprint.   Post conditions:  Process activates; tasks assignees get new tasks to do; Project Manager/Team Leader assigns ‘free’ tasks to team members |

|  |  |
| --- | --- |
| Title: | Remove task from a sprint |
| Aim: | To remove task from sprint backlog (before starting a sprint) |
| Primary actor: | Project Manager |
| Level: | Sprint |
| Precondition: | User (PM) is logged in and has permission to remove existing task from a sprint; sprint have not been started; task have not been done. |
| Trigger: | There is an odd task in sprint, which is not needed for work. |
| Main success scenario: | 1. Select preferred project 2. Select preferred sprint. 3. Sprint and its tasks will be displayed. 4. **Remove from Sprint**   Post conditions: |

|  |  |
| --- | --- |
| Title: | Add member |
| Aim: | To add new member(s) to a project group. |
| Primary actor: | Project Manager |
| Level: | Project |
| Precondition: | User (PM) is logged in and has permission to add new users to some existing project; User(s) (new members) have already signed up in the system. |
| Trigger: |  |
| Main success scenario: | 1. Choose project**.** 2. Choose person(s) (by username for signed up Users, and by email for otherwise), whom you want to invite. 3. Set role for invited person(s) – Developer or Team Leader (if Team Leader not exists).   Post conditions: New user(s) are in project team; they automatically got all default permissions in the project. |

|  |  |
| --- | --- |
| Title: | Remove user(s) from project |
| Aim: | To remove member(s) from project team. |
| Primary actor: | Project Manager |
| Level: | Project |
| Precondition: | User (PM) is logged in and has permission to remove users from some existing project; User(s) have already been added to some project team. |
| Trigger: |  |
| Main success scenario: | 1. Choose project**.** 2. Choose person(s) from list of team, who should be removed. 3. Remove user   Post conditions:  User(s) do(es) not have permission to see/edit project information and backlog; user(s) will not see project in their projects list. |

# Supporting Information

## Appendix 1. Access control list

Roles: PM – Project Manager, TL – Team Leader, D – Developer, U – some user (not Team member), SU – Superuser

Actions: R – Read, W – Write (Edit & Create), D – Delete

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role**  **Object** | **PM** | **TL** | **D** | **U** | **SU** |
| Project’s backlog | R, W, D | R, W | R | Deny | R, W, D |
| Sprint’s backlog | R, W, D | R, W | R | Deny | R, W, D |
| Project’s progress (summary, statistics, burn-down chart) | R | R | R | Deny | R |
| Project’s developers chart | R | R | R | Deny | R |
| Public user’s information | R | R | R | R, W, D | R, W, D |
| Private user’s information | Deny | Deny | Deny | R, W, D | R, W, D |
| Task/requirement delegation | R, W, D | R, W, D | R | Deny | R, W, D |

## Appendix 2. Actions with requirements/tasks

Roles: PM – Project Manager, TL – Team Leader, D – Developer.

Actions: ‘+’ – allowed, ‘-’ - deny

|  |  |  |  |
| --- | --- | --- | --- |
| **Role**  **Actions** | **PM** | **TL** | **D** |
| Delegate | + | + | - |
| Accept/Decline | + | + | + |
| Set completeness in % | + | + | + |
| Accept 100% completeness | + | + | - |
| Set complexity ‘points’ | - | + | - |
| Edit/Delete not implemented | + | - | - |
| Edit/Delete implemented | - | - | - |
| Pick to sprint backlog | + | - | - |
| Create (add) | + | - | - |