19-6-2015

Mike Rooijackers, Noor van Oekel, Jordi Knol, Maaike Jansen, Tim Hermens

Group E

Requirements checklist

PTS6

Table of Contents

[Introduction 2](#_Toc422909569)

[General requirements 3](#_Toc422909570)

[List of requirements 3](#_Toc422909571)

[To-do 3](#_Toc422909572)

[Module A: maven archetype for assignments 4](#_Toc422909573)

[List of requirements 4](#_Toc422909574)

[To-do 4](#_Toc422909575)

[Module B: workspace management module 5](#_Toc422909576)

[List of requirements 5](#_Toc422909577)

[To-do 5](#_Toc422909578)

[Module C: services module 6](#_Toc422909579)

[List of requirements 6](#_Toc422909580)

[To-do 6](#_Toc422909581)

[Module D: HTML client for participants 7](#_Toc422909582)

[List of requirements 7](#_Toc422909583)

[To-do 7](#_Toc422909584)

[Module E: HTML admin console 9](#_Toc422909585)

[List of requirements 9](#_Toc422909586)

[To-do 9](#_Toc422909587)

[Module F: HTML client for spectators 10](#_Toc422909588)

[List of requirements 10](#_Toc422909589)

[To-do 10](#_Toc422909590)

[Module G: JavaFX client for participants 11](#_Toc422909591)

# Introduction

This document provides an overview of the functionality that has been implemented per module. Since the project team was unable to fully implement all of the modules, there are a couple of small to-do lists for the modules that do not yet meet the requirements.

For more information about the architecture of the Master of Code software, read the software architecture document.

# General requirements

The following requirements are not bound to a specific module, but apply to all of the Master of Code software.

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 0 | The system has been developed in Java SE 8 or Java EE 7. | Done |
| 1 | Visual Paradigm is used to create UML diagrams. | Done |
| 2 | Javadoc has been added to public methods so that other Java developers can work with the code. | In-progress |
| 3 | Libraries other than the default Java SE 8/EE 7 libraries are only used after approval by the client. | Done |
| 4 | Either Git or SVN is used as a revision control system, but not both. | Done |
| 5 | The HTML and JavaScript is HTML5 compliant. | Done |
| 6 | Whenever different alternative techniques are available, the latest technique will be used. | Done |
| 7 | The system can be fully internationalized by adjusting some text files. | To-do |
| 8 | The system supports Dutch and English. | To-do |
| 9 | The documentation is written in English. | Done |

## To-do

* The Javadoc needs to be added to all of the public methods for the assignments.
* The system only supports English so far. It is not possible to internationalize the system yet.

# Module A: maven archetype for assignments

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 10 | The archetype requires configuration by the final user for the following elements:   * artifact ID * group ID * default package | Done |
| 11 | The archetype generates a default Java-Maven project that is able to compile. No changes are needed. | Done |
| 12 | The generated Java-Maven project executes all of the available tests in the project with “mvn test”. | Done |
| 13 | The generated Java-Maven project can execute a specific test with “mvn –Dtest [name of the test]”. | Done |
| 14 | The generated Java-Maven project is configured for Java 8. | Done |
| 15 | The generated Java-Maven project can produce an artifact through a preconfigured target. This artifact is a zipped maven project structure without compiled classes. | Done |
| 16 | The generated Java-Maven project generates a skeleton for each of the following:   * read-only files * source code files * editable source code files * hints * user test * system test * [name, organization name, web URL, URL to logo] of the author * [name, description for spectators (HTML5), description for participants (HTML5)] of the assignment * pre-meta-data which makes all of the above readable for the machine | Done |
| 17 | The executable tests conform to TestNG. | Done |
| 18 | The configuration mechanism allows the administrator to configure the order of the hints. | Done |
| 19 | The meta-data of an assignment is configured as annotations. | Done |

## To-do

All of the requirements and functionalities for module A have been implemented.

# Module B: workspace management module

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 20 | The WMM (workspace management module) can create a workspace for the given team. | Done |
| 21 | The folder where the WMM creates the workspaces is configurable. | Done |
| 22 | The WMM can extract the assignment-artifact in a folder when the assignment artifact is provided. The result will be the default maven structure of the assignment artifact. | Done |
| 23 | The WMM can make all of the meta-data available. | Done |
| 24 | The WMM is compatible with both Java SE and Java EE. | Done\* |
| 25 | When given a team, assignment and editable file (including its new content), the WMM can replace the current content with the new content for the given file in the workspace of the team. | Done |
| 26 | When given a team and assignment, the WMM can compile the assignment and make the results available for the team. | Done |
| 27 | When given a team, assignment and user test, the WMM can execute the test and make the results available for the team. | Done |
| 28 | When integrated into a JEE7 container, compilations and tests for a team may not affect the compile or test tasks for another team negatively. | Done |
| 29 | When integrated into a JEE7 container, the WMM should be able to scale without the use of a NFS (Network File System). | Done |

\*Reading meta-data via JEE doesn’t yield any results. As a workaround, the meta-data is stored in a separate object file.

## To-do

The problem with reading meta-data should be resolved for JEE. Other than that, all of the functionalities have been implemented.

# Module C: services module

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 30 | The SM (Services Module) supports creating a competition based on a name, date, start time, end time and location. | In-progress |
| 31 | The SM supports linking and unlinking assignments to/from a competition. | In-progress |
| 32 | The SM supports changing the order of assignments in a competition. | In-progress |
| 33 | The SM supports starting and stopping a competition. | Done |
| 34 | The SM supports starting, stopping, pausing and freezing a round. | Done |
| 35 | The SM supports configuration of the maximum score of a round based on the maximum number of seconds and the difficulty. | Done |
| 36 | The number of points that a team earns within a round equals [number of seconds left] \* [difficulty of the assignment] | Done |
| 37 | The SM supports setting the minimum and maximum size of a team for a competition. | In-progress |
| 38 | The SM allows the initiator to create a team for a competition. | Done |
| 39 | The SM allows the initiator to invite other (registered and unregistered) users to join their team. The SM sends an email to the given mail address with a link to accept or decline the invitation. The link and text in the mail are both configurable. | Done\* |
| 40 | The SM prevents unauthorized access to a team profile or user profile, competition, round, assignment or workspace through the JEE container mechanisms. | Done |
| 41 | The SM keeps track of the current scores of the teams at all times. | Done\*\* |
| 42 | The SM provides an API for creating a user, where the user has to fill in a mail address, password, name, phone number and organization. | Done |
| 43 | The SM verifies the authenticity of the mail address of a user after registering by sending a mail to this address containing an activation link. The text and link are both configurable. | To-do |
| 44 | The SM keeps track of the scores for each team in the competition. | Done\*\* |
| 45 | The ranking of a team within a competition is determined by adding up their scores for each round. | To-do |
| 46 | A user can only be a member of one team in a competition. | Done |

\*The user is added to the team automatically if the user exists and isn’t part of another team yet.

\*\*Each round the score of a team for that round is added to the total score in the database.

## To-do

* For the requirements with status “in-progress” or “to-do”, data can be added to the database. This is all done manually and should be fixed so that the system automatically adds this data instead.
* The calculation for the final rankings and scores isn’t supported yet.

# Module D: HTML client for participants

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 47 | The client supports assembling teams so that the selected users can participate in the competition together. | Done |
| 48 | The client supports configuring audio-profiles for music and sound effects so that different profiles can be used before, during and after the competition. | To-do |
| 49 | An initiator can create a new team to enter the competition with. | Done |
| 50 | An initiator can invite other people to their team. | In-progress\* |
| 51 | The client supports reading the assignment description for participants. | Done |
| 52 | The client supports registering users. | Done |
| 53 | The client supports registering teams that can participate in the competition and get their own workspace. | Done\*\* |
| 54 | A participant can remove himself/herself from the team that they have joined. | To-do |
| 55 | An initiator can remove other members from their own team. | To-do |
| 56 | After being removed from a team, a user no longer has access to the team’s workspace. | Done |
| 57 | A user can respond to an invitation by accepting or declining. Upon accepting the invitation, they automatically join the team. | To-do |
| 58 | The client provides an overview of actions from all of the teams during the competition. This overview should always be up-to-date. | Done |
| 59 | At the end of a competition, the client shows a list of the final rankings and scores. | To-do |
| 60 | The client displays information about the upcoming competition if it hasn’t started yet. This includes the goal of the competition, date, start time and number of rounds. | In-progress |
| 61 | The client supports viewing the Javadoc of the source code files. | Done |
| 62 | The client supports viewing published hints. | Done |
| 63 | The client supports viewing read-only source code files. | Done |
| 64 | Read-only source code files cannot be edited by participants. | Done |
| 65 | The client supports viewing every detail of the current round, including its duration. | Done |
| 66 | The client supports viewing the scores of all participating teams. | Done |
| 67 | When a team is working on an assignment, they can see what their score for the round would be if they turned in the assignment at that exact moment without errors. | Done |

\*It’s only possible to invite people when creating the team. After creation, new members cannot be added to the team.

\*\*Not created automatically, but can be linked through their teamId.

## To-do

* It should be possible to configure audio-profiles. As for now, no sound effects or music files are used for the competitions and it is not possible to link sounds or music to a certain event or action.
* An initiator should be able to add new members to their team after the team has been created. They need to be able to configure the team so they can add and remove members.
* A user has to accept or decline an invitation. In the current situation, a user is automatically added to a team once they have been invited, meaning that the invitation is always accepted.
* A team member should be able to remove themselves from a team. This featured is not supported yet as the team cannot be edited after creation.
* The GUI screens for before and after the competition have not been created and implemented yet. However, there are screen designs for these web pages, which can be found in the document “Global GUI designs module D”.

# Module E: HTML admin console

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 68 | The client supports setting the difficulty of an assignment. | To-do |
| 69 | The client supports configuring information about the competition. | To-do |
| 70 | The client supports extending the time of a round. | To-do |
| 71 | The client supports starting a round. | Done |
| 72 | The client supports freezing the round that is currently playing. The client will not be available for participants when this happens. | Done |
| 73 | The client supports pausing the round that is currently playing. | Done |
| 74 | The client support resuming a paused or frozen round. | Done |
| 75 | The client supports stopping a round. | Done |
| 76 | The client supports starting a competition. | Done |
| 77 | The client supports stopping a competition. | Done |
| 78 | The client supports overwriting the meta-data of an assignment. | To-do |
| 79 | The client supports revealing a hint sooner than the time that is set for that specific hint. | To-do |
| 80 | The client supports replacing an existing hint with a new one. | To-do |
| 81 | The client supports configuring the duration of a round. This includes the moments at which hints will be revealed to the participants. | To-do |
| 82 | The client supports selecting assignments for the competition. | To-do |
| 83 | The client supports selecting teams that are allowed to participate in the competition. | To-do |
| 84 | The client supports monitoring the performance of the servers. | To-do |
| 85 | The client supports viewing competitor information. | In-progress |
| 86 | The client supports registering competitors. | In-progress |
| 87 | The client supports removing members from all of the registered teams. | To-do |
| 88 | The client supports editing registered teams by management members. | To-do |

## To-do

* For each of the requirements with status “to-do”, the GUI has already been fully created and implemented. The only thing that’s left is retrieving data from the web sockets and displaying it on the web page.
* Displaying a timer on every web page of the HTML client. This timer shows the time left until the end of the current round (when the competition has already started).

# Module F: HTML client for spectators

## List of requirements

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Status |
| 89 | If the competition hasn’t started yet, the client displays the information about the competition. Other data, such as the rounds and participating teams, are not visible to spectators. | To-do |
| 90 | If the competition has ended, the client displays the final scores and scores per round for all of the teams. Other data, such as the rounds and newsfeed, are not visible to spectators. | To-do |
| 91 | Once the competition has started, the client needs to display the following data:   * Information about the competition * Information about the current assignment * Information about the teams that are participating, such as members and scores * The scores and rankings * Information about the rounds in the competition * News about actions of the participating teams | Done |
| 92 | The client supports viewing the scores for different rounds. | Done |
| 93 | The information should be kept up-to-date at all times. | In-progress |

## To-do

* The GUI screens for before and after the competition have not been created and implemented yet. However, there are screen designs for these web pages, which can be found in the document “Global GUI designs module F”.
* Displaying the data that can be retrieved via the web sockets on the right web pages. This should replace the mock data that is currently being used. The HTML files use AngularJS to display the data on the web pages.
* Displaying a timer on every web page of the HTML client. This timer shows the time left until the end of the current round (when the competition has already started).

# Module G: JavaFX client for participants

The team hasn’t started developing module G yet.