POLITECNICO DI MILANO

Computer Science and Engineering

Requirements Analysis and Specifications Document

CodeKataBattle

Software Engineering 2 Project

Academic year 2023 - 2024

20 November 2023

Version 1.0

Authors:
Massara Gianluca
Nguyen Ba Chiara Thien Thao
Nicotri Flavia

Professor:
Matteo Giovanni Rossi

Contents

Chapter 1

Introduction

1.1 Purpose

In the last years, more and more students are becoming interested in programming and many educators have realised the need of new innovative methods to improve coding skills. CodeKataBattle aims to assist students in enhancing their programming skills by challenging them with creative tasks in a competitive and stimulating environment.

The following document want to describe the system focusing on the requirements and specification, providing scenarios and use case to specify what the system must do and how it should interact with the stakeholders.

1.1.1 Goals

In the following table we describe the main goals that our system want to achive.

Goal	Description
G.1	Allow students to compete in a tournament
G.2	Allows educators to create challenges for students.
G.3	Allows educators to grade students' projects.
G.4	Allows students to collect badges.

1.2 Scope

The main actors of the system are students and educator. Educator can:

- **Create a tournament:** decide which colleague can create battles within the tournament and defines badges that represent the achievements of individual students;
- **Create a battle:** set configurations and rules for that battle;
- **Evaluate:** manually assign a personal score to the students' works.

Students can:

- **Join a tournament**;
- **Partecipate on a battle:** create a team and complete the project with his code;
- **Collect badges:** based on the rules of the tournament and his performance.

1.2.1 World Phenomena

W.P.	Description
WP.1	Educator wants to create a tournament.
WP.2	Educator wants to create a new battle.
WP.3	A student want to join a tournament.
WP.4	A student wants to join a battle.
WP.5	An educator evaluates the works done by students.

1.2.2 Shared Phenomena

S.P.	Description	Controlled by
SP.1	The system notifies the student about upcoming battles.	Machine
SP.2	The student commits his code.	World
SP.3	The educator configures the tournament rules.	World
SP.4	The student forms a team.	World
SP.5	The educator grants other colleagues the permission to create battles within a tournament.	World
SP.6	The educator configures the battle.	World
SP.7	The student joins a team.	World
SP.8	The system creates a GitHub repository.	Machine
SP.9	The student forks and sets up the GitHub repository.	World
SP.10	GitHub Actions notifies the system about students' commits.	World
SP.11	The system shows the battle score of the team.	Machine
SP.12	The system notifies when the final battle rank becomes available.	Machine
SP.13	The system shows the tournament rank.	Machine
SP.14	The system shows the list of ongoing tournaments.	Machine
SP.15	The system shows the student's badges.	Machine
SP.16	The educator defines the badges of a tournament.	World
SP.17	All users can visualize the profile of a user.	World

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

1.3.2 Acronyms

Acronyms	Meaning
CKB	CodeKataBattle

1.3.3 Abbreviations

Abbreviations	Meaning
WP	World Phenomena
SP	Shared Phenomena
G	Goal
R	Requirement
NFR	Non Functional Requirement
D	Domain Assumption
w.r.t.	with reference to
e.g.	exempli gratia
i.e.	id est
etc.	etcetera

1.4 Revision History

1.5 Reference Documents

- Course slides on WeBeep.
- RASD assignment document.

1.6 Document Structure

The structure of this RASD document is the following:

1. **Introduction**: In this section is presented the purpose of this document highlighting in particular the main goals, the audience which is referred to, the identification of the product and application domain describing world and shared phenomena and, lastly, the terms definitions.
2. **Overall Description**: This chapter describes the possible scenarios of the platform, the shared phenomena presented at the beginning of the document and assumptions on the domain of the application.
3. **Specific Requirements**: Includes all the requirements in a more specific way than the "Overall Description" section. Moreover, it is useful to show functional requirements in terms of use cases diagrams, sequence/activity diagrams.

4. **Formal Analysis Using Alloy**: Includes Alloy models which are used for the description of the application domain and his properties, referring to the operations which the system has to provide.
5. **Effort Spent**: This section shows the effort spent in terms of time for each team member and the whole team.
6. **References**: Includes all documents that were helpful in drafting the RASD.

Chapter 2

Overall Description

2.1 Product perspective

In this section we analyze a list of real scenariosand diagrams illustrating further details about shared phenomenas.

2.1.1 Scenarios

1. **Registration**
   Prof. White reads about CodeKataBattle in a online spot and decides to use the platform to challenge his students. So, he creates an account using his email address and personal information and convinces his colleagues to join the platform.
   Alice and her classmates, after prof. White's lesson, are interested in improving their coding skills and sign up to CKB. Alice provides her email address and personal information; after the email confimation goes well, she can complete the registration process by linking her CKB account to her GitHub account. Her nickname on the platform will be her GitHub one.
   After the registration, both Prof. White and Alice can visualize the profile of all the other students and the information about the ongoing tournaments.

2. **Creation of tournaments**
   Prof. Brown and prof. Smith want to decide whose class is better at programming. So Prof.Brown create a tournament on CKB and invite his colleague to create battles within the tournament. Prof. Brown sets a deadline by which their students can subscribe. A notification is sent to all CKB students.

3. **Creation of battles**
   Prof. Bloom has been granted the permission to create battles in a tournament, so he uploads a code kata battle. First he writes a brief textual description of the exercise, then he includes the software project with build automation scripts and test cases. He also sets some rules: minimum and maximum number of students per group, registration deadline, final submission deadline, automated evaluation parameters.
   Finally, he chooses to include some optional manual evaluation of his students' work.

4. **Students join a battle**
   Bob registered to the CKB platform and received a notification about an interesting tour-

nament and he subscribed to it.
He is notified about a new interesting battle in that tournament is upcoming. Before the registration deadline expires, Bob can form a group, so invites his girlfriend Eva to join him. After the registration deadline ends, they are sent the link of the GitHub repository of the code kata. To be able to start working on the project, Bob and Eva must fork the repository and set up an automated workflow through GitHub Actions to inform the CKB platform about their commits.

5. **Upload of a solution**
   Charlie has joined a battle with his friends and thinks that they have found a good solution to the assigned problem. So he push his code before the final deadline to the main branch of their repository.
   This battle doesn't expect any manual evaluation by the educator, so as soon as push the solution, the platform evaluates it and they can visualize their updated battle score. After the final deadline has expired, Charlie can also see the current tournament ranking.

6. **Evaluation of project**
   When Prof. Cooper created the battle, he decided to include manual evaluation in the battle rules: he wants the system to assign up to 75 points, while he will assign the remaing 25 points. So, scores will always be between 0 and 100.
   After the final deadline, prof. Cooper can see all the groups' work and he evaluates them according to some personal parameters such that design quality, code cleanliness, compliance with specifications.

7. **Creation of gamification badges**
   Prof. Moore has created a tournament and to spice things up he decides to include badges. These are awarded according to the rules specific by him at tournament set up time: in particular, he wants to reward the student with most commits and the student who have attended the most battles.
   The badges are assigned at the end of the tournament and the collected badges can be seen by everyone visiting the student's profile.

2.1.2 Class diagrams

2.1.3 State diagrams

2.2 Product functions

2.2.1 Sign up and login

These functions are available to both students and educators.
The sign up functionality allows users to register on the platform: in particular, each user will provide an email and a password. Then a verification email is sent to the user. After the confirmation, the user must provide their personal information (name, surname, date of birth) and if the user is a student they are required to provide a GitHub account.

Once registered, both students and educators can visualize the profile of all the other users.
The login functionality allows users to access the platform using their email and password.

2.2.2 Creation of a tournament

Educators can create tournaments, programming competitions composed of various code kata battles.
The creator of the tournament can invite other educators and grant them the permission to create battles within the context of the tournament.
When they create a tournament, educators sets a registration deadline and can also define badges, i.e., awards that represent particular achievements of individual students: the educator may use pre-defined variables and rules or create new ones. Badges are awarded automatically at the end of a tournament.
Information about ongoing tournaments is avalaible to all subscribed users.

2.2.3 Creation of a battle

After being granted such permission from the creator of the tournament, educators can create code kata battles within the context of the tournament.
A code kata battle is a programming exercise in a programming language chosen by its creator. It includes a textual description of the exercise and a software project with build automation scripts and test cases.
To create a battle, the educator must upload the code kata, set the rules regarding groups composition (minimum and maximum number of members), set a registration deadline, set a final submission deadline, and set evaluation parameters. They may also specify if they want to include a manual evaluation of the students' work, in addition the mandatory automated one, after the submission deadline expires.
Educators can see the current battle ranking evolve during the battle.

2.2.4 Join a tournament

Students can join any tournament before its registration deadline expires. By signing up to a tournament, they also subscribe to notifications regarding upcoming battles within the context of that tournament.
Students also receive a notification when the tournament ends.
By participating in a tournament, students can earn badges that can be visualized in their profile.

2.2.5 Join a battle

Students subscribed to a tournament can join any upcoming battle before its registration deadline expires.
Students can participate alone or in groups to the battle, according to the rules set for that battle by its creator.

When the registration deadline expires, students are sent an email with the link of the GitHub repository containing the code kata. To start working on the project, they are required to fork the repository and set up an automated workflow, using GitHub Actions, to inform CKB about new commits into the main branch of their repository.
As soon as students push a solution it is automatically evaluated, so they can see their current battle score and their battle rank.
When the submission deadline (and, if required, the manual evaluation stage) ends, students are notified about their final battle score and rank and they can see their updated tournament score.

2.2.6 Evaluation of a project

According to the parameters set by the creator of the battle, the system automatically evaluates the students' work with regard to

1. functional aspects (i.e., the percentage of passed test cases);
2. timeliness (i.e., the time passed between the registration deadline and the last commit);
3. code quality (extracted using static analysis tools).

If the educator has decided to include manual evaluation in the battle rules, they can see all the groups' work and evaluate the students' work according to some personal parameters such that design quality, code cleanliness, compliance with specifications.
The final score is a natural number between 0 and 100 calculated as specified at battle creation time.

2.3 User characteristics

2.4 Assumptions, dependecies and constraints

Chapter 3

Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communication Interfaces

3.2 Functional Requirements

3.3 Performance Requirements

3.4 Design Constraints

3.4.1 Standards compliance

3.4.2 Hardware limitations

3.4.3 Any other constraint

3.5 Software System Atttributes

3.5.1 Reliability

3.5.2 Availability

3.5.3 Security

3.5.4 Maintainability

3.5.5 Portability

Chapter 4

Formal Analysis

Chapter 5

Effort Spent

Chapter 6

References