# Requirements And Specifications Document

Politecnico di Milano
Merlino Lorenzo & Iodice Andrea

November 22, 2023

## Contents

# 1 Introduction

## 1.1 Purpose

The System to Be will allow users to take part in coding battles. Each coding battle will be part of a tournament and consists of a coding project to be completed. Educators will create tournaments and create battles by supplying the project structure. Students will be able to create teams and the teams will be able to submit a solution to the coding battle. Each solution will be scored using different criterias, such as the amount of test cases passed, amount of time required to submit the solution and quality of the code provided. Some criteria will be automatically asserted and the remaining ones will be asserted by the creator of the battle.

### 1.1.1 Goals

The goals the system aims to achieve are:

[G1] Educator is able to create and manage a tournament.
[G2] Educator is able to allow other educators to manage their tournament.
[G3] Educator is able to create and manage code battles for a tournament they are involved in.
[G4] Any user is able to authenticate with their GitHub account.
[G5] Student is able to enroll in tournament and join/create a team.
[G6] Any Team participating in a battle is able to submit a solution.
[G7] Students are able to see their current rank and score in the tournaments they are taking part.
[G8] Educators are able to see Students' current rank and score in the tournaments they are involved in.
[G9] Educators are able to assign a grade to solutions submitted by Teams subscribed to a Battle created by the Educator.

## 1.2 Scope

CodeKataBattle (CKB) is a new platform that helps students improve their software development skills by training with peers on code kata.

Educators use the platform to challenge students by creating code kata battles in which teams of students can compete against each other, thus proving and improving their skills.

Educators create tournaments to which students can enroll. After enrolling in a tournament students can take part in code kata battle by joining or creating a team.

A code kata battle is a programming exercise. The exercise includes a brief textual description and a software project with build automation scripts that contains a set of test cases that the program must pass.

Teams participating in a battle are expected to follow a test-first approach and develop a solution that passes the required tests. Groups deliver their solution to the platform before the deadline. At the end of the battle, the platform and the educators assign scores to groups, in order to create a competition rank.

### 1.2.1 World Phaenomena

Phenomena events that take place in the real world and that the machine cannot observe:

[W1] Educator wants to create a tournament
[W2] Educator wants to allow another educator to create battles in their tournament
[W3] Student wants to participate in a tournament
[W4] A team modifies their implementation of the solution
[W5] Student wants to check his rank

[W6] Student wants to join a team

### 1.2.2 Shared Phaenomena world controlled

Phenomena controlled by the world and observed by the machine:

[SP1] User logs in the system
[SP2] Educator creates a tournament
[SP3] Educator allows another educator to create battles for their tournament
[SP4] Educator creates a battle for their tournament
[SP5] Educator assign grade for a solution
[SP6] Educator closes a tournament
[SP7] Student joins tournament
[SP8] Student creates a team
[SP9] Student joins a team
[SP10] Student invites another student to the team
[SP11] Student checks is rank for a tournament they are participating in
[SP12] A team pushes their solution to GitHub
[SP13] Battle's deadline is reached
[SP14] Student receives invite to a team

### 1.2.3 Shared Phaenomena machine controlled

Phenomena controlled by the machine and observed by the world:

[SP15] System notifies student they have been invited to a team
[SP16] System assign automatic grading
[SP17] System notifies students a new battle is available
[SP18] System notifies students a new tournament is available
[SP19] System notifies a tournament has been closed
[SP20] System notifies a student a battle is ended
[SP21] System notifies a team their solution has been graded
[SP22] System notifies educator that a new submission is ready to be graded
[SP23] System notifies student that the student they have invited to their team has accepted/rejected the invite.

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

- **Coding Battle**: programming exercise proposed by an Educator in the context of a tournament.
- **Tournament**: a series of coding battles, in which Students can take part as teams. It's created and managed by an Educator.
- **Team**: group of students who enroll in a tournament. Students in the same team work together and the solution they submit will be taken into consideration for each member of the team.
- **Rank**: position of a Student in a tournament's leaderboard, which is calculated on each Student's score.
- **Score**: points gained by a Student by submitting a solution to a Coding Battle.
- **Open battle**: a battle that accepts new submissions.

- **Closed battle**: a battle that doesn't accept new submissions but some still need to be evaluated.
- **Completed battle**: a battle that doesn't accept new submissions and where all the submissions have been evaluated.
- **Open tournament**: A tournament where new battles can be created.
- **Closed tournament**: A tournament where new battles cannot be created but some are still not completed.
- **Completed tournament**: A tournament where new battles cannot be created and all battle have been completed
- **Team leader**: Student who created the team.
- **Automated grading**: A series of scores that is automatically assigned to a solution, it is composed of:
  - Number of test passed
  - Time passed after the battle publication
  - Static analysis result
- **User**: either a Student or an Educator
- **Educator**: user of the system, who manages tournaments and battles.
- **Student**: user of the system, who participates in tournaments and battles.

### 1.3.2 Acronyms

- **RASD**: Requirements Analysis and Specification Document

### 1.3.3 Abbreviations

- [**Gn**]: the n-th goal of the system
- [**Wn**]: the n-th world phaenomena
- [**Rn**]: the n-th requirement
- [**SPn**]: the n-th shared phaenomena
- [**UCn**]: the n-th use case
- [**SCn**]: the n-th scenario

## 1.4 Revision History

- Version 1.0

## 1.5 Reference Documents

- The specification of the RASD && DD assignment of the Software Engineering II course, held by professors Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli in Politecnico di Milano, A.Y. 2023/2024.
- Slides of the Software Engineering II course on the WeBeep website.

## 1.6 Document Structure

This document is divided into 6 sections:

1. **Introduction**: it aims to describe the environment and the demands taken into account for this project. In particular it's focused on the reasons and the goals that are going to be achieved with its development
2. **Overall Description** it's a high-level description of the system by focusing on the shared phenomena and the domain model (with its assumption)
3. **Specific Requirements**: it describes in very detail the requirements needed to reach the goals. In addition it contains more details useful for developers (i.e information about HW and SW interfaces)
4. **Formal Analysis**: this section contains a formal description of the main aspect of the World phenomena by using Alloy
5. **Effort Spent**: it shows the time spent to realize this document, divided for each section
6. **References**: it contains the references to any documents and to the Software used in this document

# 2 Overall Description

## 2.1 Product Perspective

### 2.1.1 Scenarios

Note: *The login process will be handled by github. Moreover, these scenarios require that the user has done the login procedure.*

**[SC1]** Educator creates a tournament:
An educator wants to create a new tournament. They log in into the system and access system tools to create a tournament. At this point the educator must define the tournament name and other educators that can create battles. If all the information has been inserted correctly the educator can confirm the creation of the tournament. The system will create the tournament and notify all the students.

**[SC2]** Educator adds another educator in a tournament they manage:
An educator has created a tournament and wants to add another educator to it. The educator will ask the system for a list of all the tournaments they created, they select the one they are interested in, then they will ask the system the list of all the educators, select the ones they are interested in and confirm their choice, the system will signal whether the operation was successful or not. If the operation was successful the added educators will be notified that they can now create battles for this tournament.

**[SC3]** Educator wants to create a battle:
An educator has either created a tournament or has been allowed to create battles for a tournament. The Educator logs into the system and will select a tournament they are managing. After that, the Educator asks the system to create a new battle for said tournament and specifies: the name of the battle, the minimum and maximum size of a team for this battle, the registration deadline, the submission deadline, the scoring configuration and the GitHub repository containing the description of the battle and the software project. The educator confirms the creation of the battle and, if all the information is valid, the battle is created and all the students enrolled in the tournament are notified of the new battle.

**[SC4]** Educator assign grade for a solution:
The educator logs into the system and asks the system for a list of all the submissions ready to be graded. The educator selects the submission they want to grade, review the submission and assign a grading according to the grading configuration. The grading is recorded in the system and the students are notified.

**[SC5]** Educator closes a tournament:
The educator wants to close a tournament, they ask the system for a list of all the open tournaments the Educator is managing. The educator selects the tournament they want to close and asks the system to close it. The system puts the tournament in the closed state, at this point no new battle can be created and once all the battles in the tournament have been completed, the tournament is changed into the status completed.

**[SC6]** Student joins tournament:
A student wants to join a tournament. The student asks the system for a list of all the open tournaments and selects the one they want to join. If the operation is successful the student joins the tournament and they can access all the open battles for that tournament.

**[SC7]** Student creates a team:
A student has joined a tournament and wants to create a team for a battle of that tournament. The student selects the tournament and the battle they are interested in. The student creates a new team, they will have to enter the name of the team, fork the educator's repository and enter the link of their repository. If the name is available, the link to their repository is correct and the student is not in another team for this battle the operation is successful the team is created.

**[SC8]** Student joins a team:
A student has joined a tournament and wants to join a team for a battle of that tournament. The student selects the tournament and the battle and asks the system for a list of all invites for the selected battle. If the student is already in a team, the invite is automatically rejected; otherwise the student can reject the invite and their status will not change or they can accept it. When the student accepts the invite of a team there are two cases: if the team is at maximum capacity the invite will be rejected and the student will be notified; otherwise the system adds the student to the team and invites them to the team repository. Later, whether the invite is rejected or accepted, the team leader who invited the student is notified.

**[SC9]** Student invites another student to the team:
A team leader wants to invite another student to their team. The student selects the tournament and the battle they are interested in. The student asks the system for a list of all students that are enrolled in the tournament but are not in a team for this battle. They select the student they want to invite and ask the system to invite them to the team, if the team is not at maximum capacity the invite is sent.

**[SC10]** Student checks his rank for a tournament they are participating in:
The student wants to check their rank in a tournament. They ask the system for a list of all tournaments they enrolled in and select the correct one. Then they ask the system for the tournament rankings. The system returns all the rankings of the students enrolled in the tournament.

**[SC11]** A team pushes their solution to GitHub:
A member of a team pushes a new version of the project in a repository associated with a team. The system detects the change and, if the team has the minimum number of members, it runs the associated github actions retrieving the output. The previous automatic evaluations are overwritten if there were any or simply stored if this is the first submission of the team.

**[SC12]** Battle's deadline is reached:
When a battle's deadline is reached the battle status is automatically set to closed. The students taking part in the battle are notified. The educator of the battle is noticed. Once the educator finally evaluates all the remaining submissions the status is changed to completed.

#### 2.1.2 Domain Class Diagram

In Figure is represented the Domain class diagram related to CodeKataBattle. It contains all the elements of the domain in which the system operates and the interaction between such elements. In the following are described the most relevant parts of the diagram in order to ease the understanding of the domain.

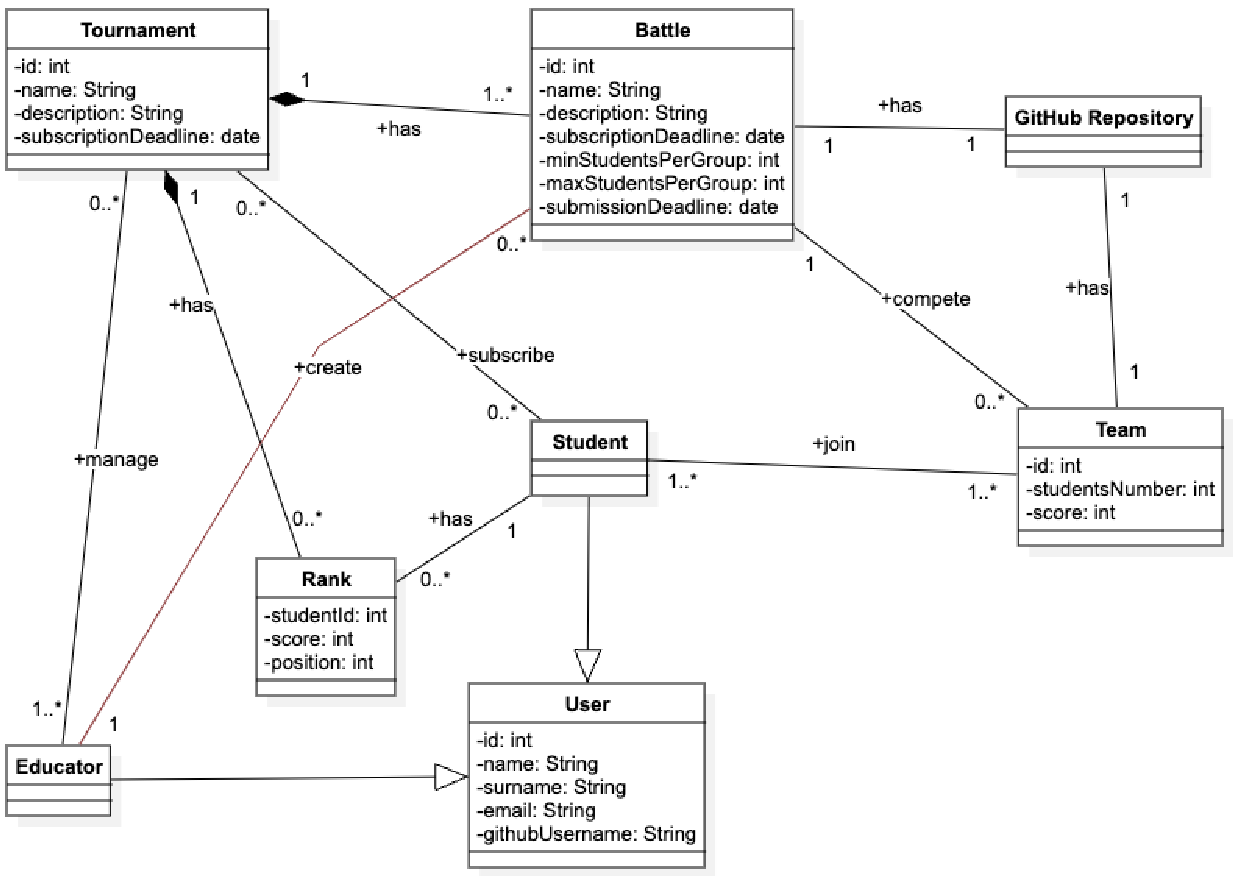


Figure 1: Domain class diagram

- There are two types of users: Students and Educators. They both use the system, so they have some attributes in common, such as name, surname and email.
- Each Educator can manage some Tournaments, which are composed of at least one Battle.
- Each Tournament has a list of ranks: each rank belongs to a Student that is enrolled into the Tournament.
- In order to compete in a Battle, Students have to join a Team, whose minimum and maximum number of Students is determined by the Battle the Team participates in.
- Each Battle is created by an Educator and is associated with a GitHub Repository, which contains information about the battle and the test cases that are going to be used to evaluate Teams' submissions.
- Each Team that competes in a Battle will submit their solution using a GitHub Repository.

### 2.1.3 State Charts

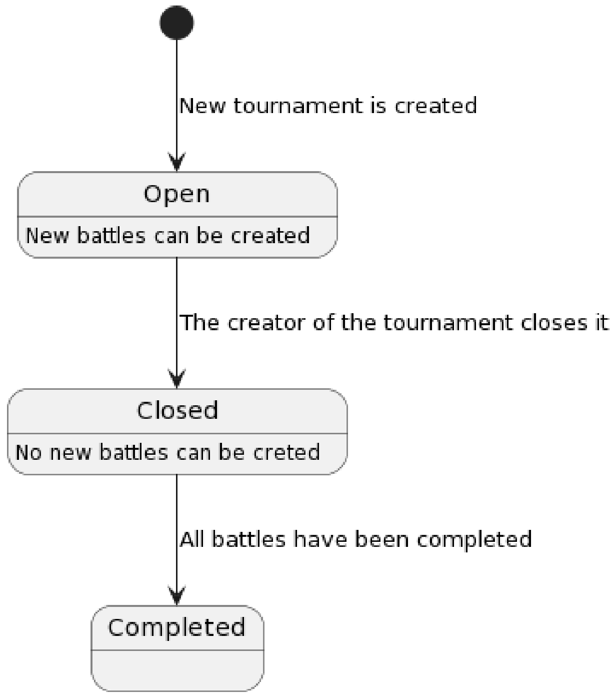

New tournament is created
Open
New battles can be created
The creator of the tournament closes it
Closed
No new battles can be creted
All battles have been completed
Completed

Figure 2: State chart that describes the process of managing a tournament

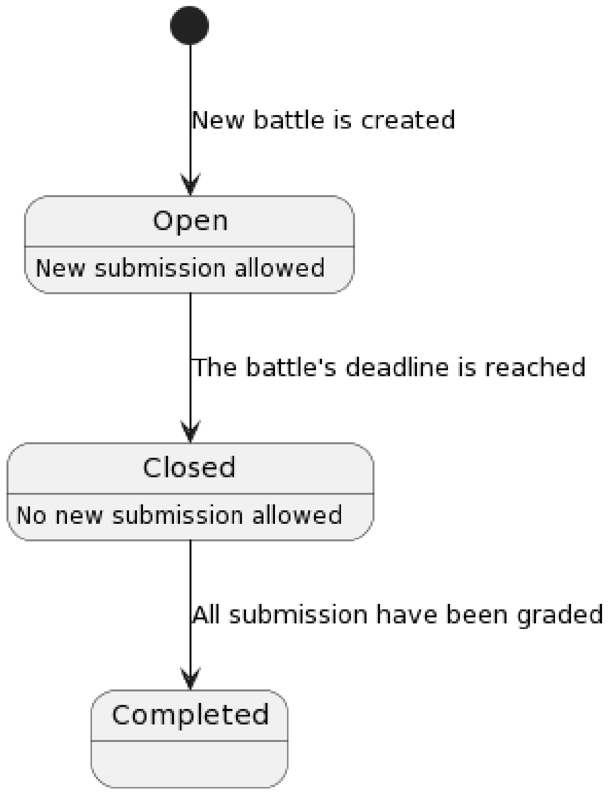
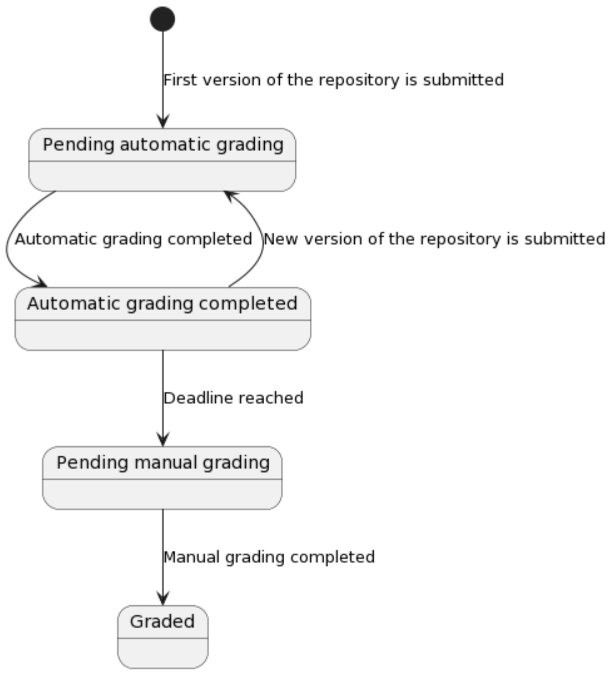
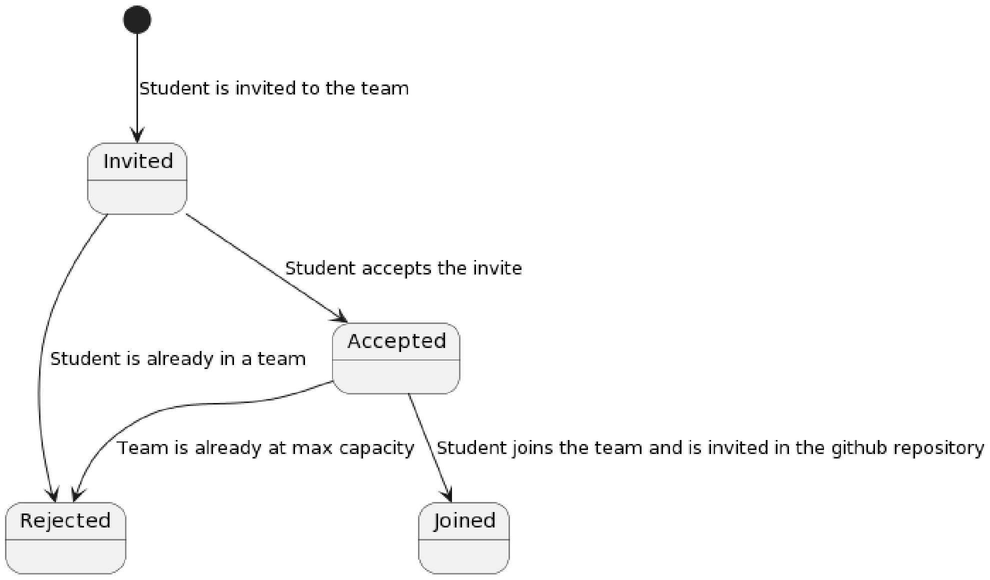

New battle is created
Open
New submission allowed
The battle's deadline is reached
Closed
No new submission allowed
All submission have been graded
Completed

Figure 3: State chart that describes the process of managing a battle



First version of the repository is submitted
Pending automatic grading
Automatic grading completed
New version of the repository is submitted
Automatic grading completed
Deadline reached
Pending manual grading
Manual grading completed
Graded

Figure 4: State chart that describes the process of managing the grading



Student is invited to the team
Invited
Student accepts the invite
Student is already in a team
Accepted
Team is already at max capacity
Student joins the team and is invited in the github repository
Rejected
Joined

Figure 5: State chart that describes the process of managing the invites

## 2.2 Product Functions

### 2.2.1 Sign Up and Login

The sign up and login functions are handled by github, this way the system can manage the user github repositories.

### 2.2.2 Create tournament

This function will be available to educators. Will allow educators to create tournaments

#### 2.2.3 Close tournament

This function will be available to educators. Will allow educators to close a tournament if they have created it.

#### 2.2.4 Add educator to tournament

This function will be available to educators. Will allow educators to abilitate other educators to create battles for the tournament they create.

#### 2.2.5 Create battle

This function will be available to educators. Will allow educators to create battles

#### 2.2.6 Grade solution

This function will be available to educators. Will allow educators to grade a solution for a battle they have created

#### 2.2.7 Join tournament

This function will be available to students. Will allow students to join tournaments

#### 2.2.8 Create team

This function will be available to students. Will allow students to create team for a battle

#### 2.2.9 Invite to team

This function will be available to team leaders. Will allow students to invite other students in their team

#### 2.2.10 Accept invite to team

This function will be available to students. Will allow students to accept an invite to a team.

#### 2.2.11 Reject invite to team

This function will be available to students. Will allow students to reject invites to join a team

#### 2.2.12 Check tournament rankings

This function will be available to students who are participating in the tournament. Will allow students to check the rankings for that tournament.

## 2.3 Users

#### 2.3.1 Educator

A user of the system that creates and manages tournaments and battles.

#### 2.3.2 Student

A user of the system that participates in tournaments and battles

## 2.4 Assumptions, dependencies and constraints

### 2.4.1 Regulatory policies

The system doesn't ask or store any personal data by itself, as github is used for authentication. The system will inherit all restrictions and rules coming from the use of the github api.

### 2.4.2 Domain Assumptions

The following are the assumptions made for the domain. Such assumptions are properties and/or conditions that the system takes for granted, mostly because they are out of the control of the system itself, and therefore need to be verified to assure the correct behavior of the system.

[D1] User has a GitHub account.
[D2] User has access to their GitHub account.
[D3] Educator inserts the correct information when creating a battle.
[D4] Educator provides a correct project structure in the repository for the battle.
[D5] Educator provides correct tests in the repository for the battle.
[D6] Educator provides correct GitHub actions to run test and static analysts in the repository for the battle.
[D7] Student accepts the invitation to the GitHub repository in time.
[D8] The solution pushed in the main branch is the solution to be evaluated.
[D9] GitHub allows for third parties to manage the user account.
[D10] The system is able to connect to GitHub via internet connection.
[D11] The system is reachable via the user's internet connection.
[D12] The Student doesn't alterate the repository in such a way that prevents the GitHub actions created by the creator of the battle to be run.

## 3 Specific Requirements

## 4 Formal Analysis

## 5 Effort Spent

## 6 References