Requirements And Specifications Document

Politecnico di Milano
Merlino Lorenzo & Iodice Andrea

December 7, 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.

A Educator cannot be a student and vice versa.

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. If the name is available and the student is not in another team for this battle the operation is successful, the team is created and a fork of the original repository is created on the student account.

[**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.

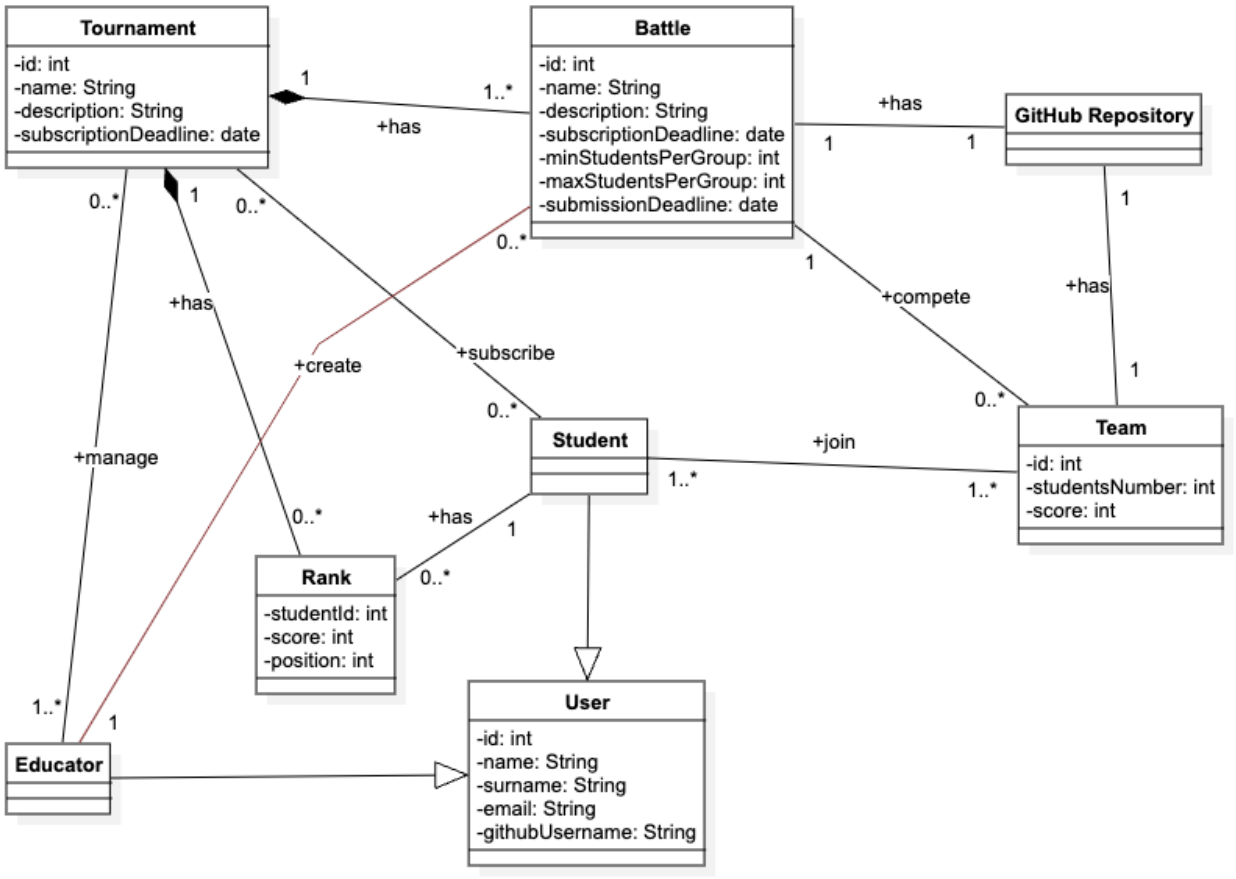

Figure 1: Domain class diagram

- 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

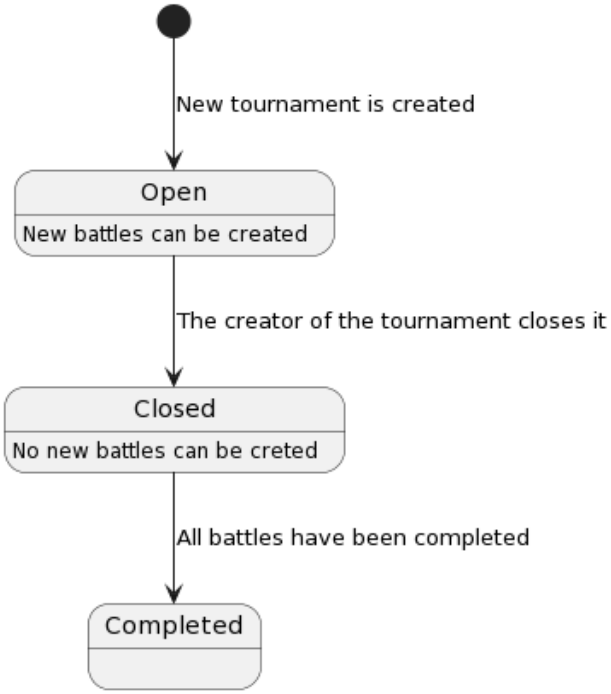

Figure 2: State chart that describes the process of managing a tournament

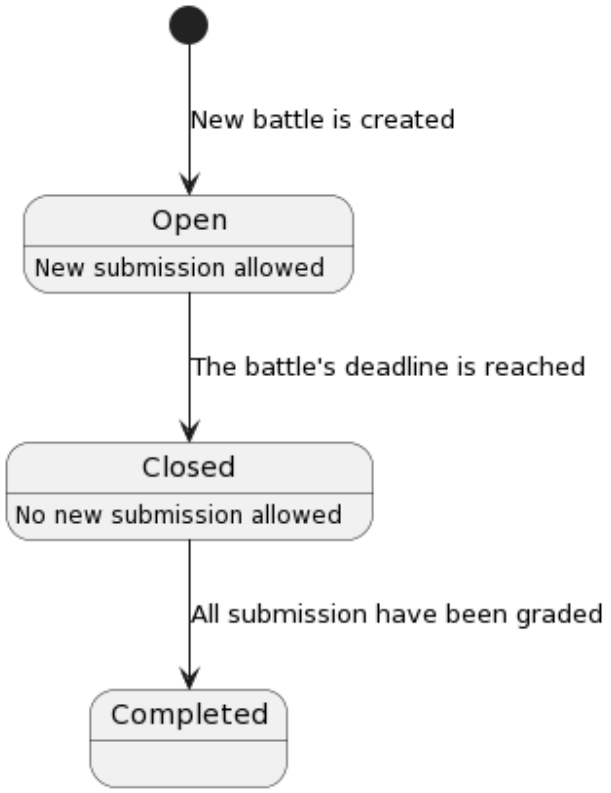

Figure 3: State chart that describes the process of managing a battle

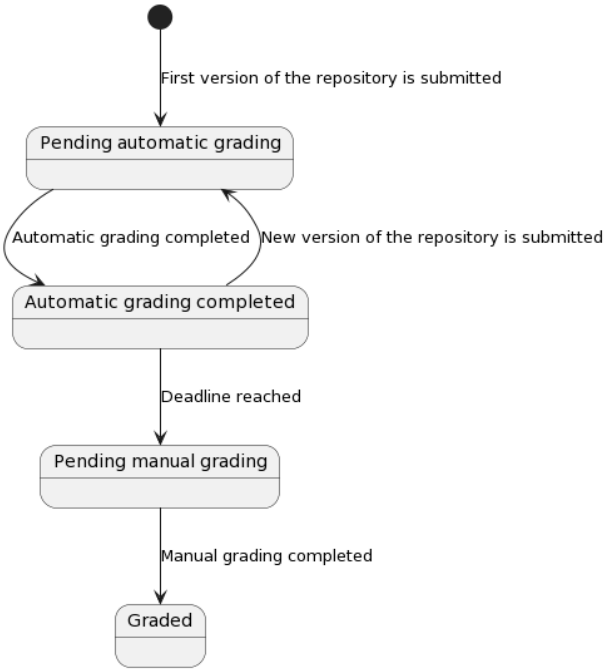

Figure 4: State chart that describes the process of managing the grading

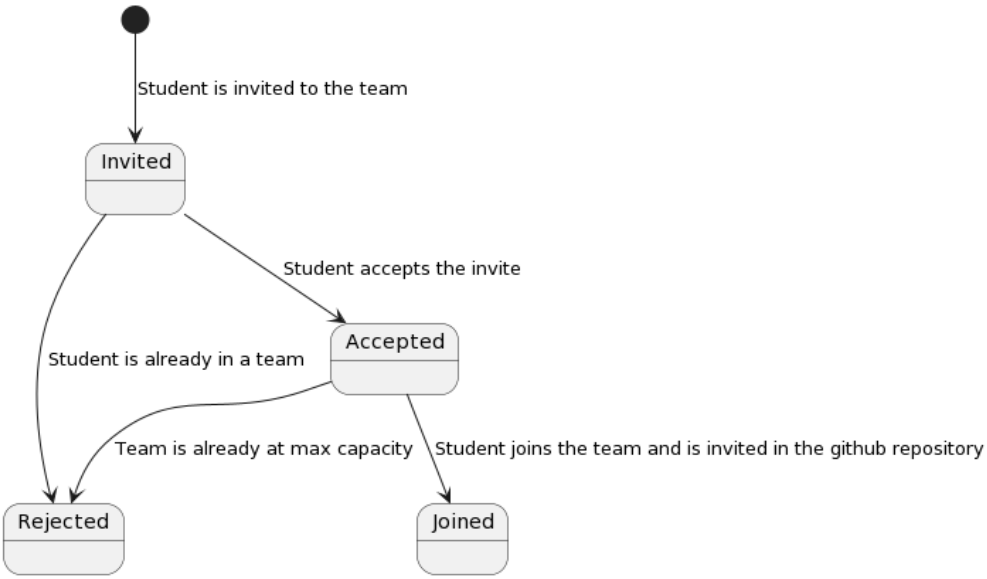

Figure 5: State chart that describes the process of managing the invites

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.
[**D13**] The Educator's repository is public.
[**D14**] The Student's repository is public.

3 Specific Requirements

3.1 External interfaces

The system will have to interface with:

- Educators
- Students
- Github api

3.1.1 Hardware interfaces

The system will not have to interface with any particular hardware outside of the users' devices.

3.1.2 Software interfaces

The system will have to interface with:

- Github authentication system to log in users
- Github api to manage users repositories
- Github api to run github the required actions

3.1.3 Communication interfaces

The system will have to communicate through internet with:

- Github
- Users

3.2 Functional requirements

In the following are specified all the requirements that the system has to fulfill in order to work properly.

3.2.1 Requirements index

[**R1**] The system allows Students to log in via GitHub.
[**R2**] The system allows Educators to log in via GitHub.
[**R3**] The system allows Educators to create a tournament.
[**R4**] The system allows Educators to view a list of only tournaments they manage.
[**R5**] The system allows Educators to add other Educators to a tournament they manage.
[**R6**] The system allows Educators to view a list of all Educators that manage the tournament they are managing.
[**R7**] The system allows Educators to end a tournament they manage.
[**R8**] The system allows Educators to view a list of Students that are enrolled in a tournament they manage.
[**R9**] The system allows Educators to create a battle in a tournament they manage.
[**R10**] The system allows Educators to view a list of teams that are participating to a battle they have created.
[**R11**] The system allows Educators to view a list of submitted solutions for a battle they have created.
[**R12**] The system allows Educators to view the repository of each solution submitted to a battle they have created.
[**R13**] The system allows Educators to grade a solution for a battle they have created, after the deadline of a battle.

[**R14**] The system allows Educators to see the grade given by the automated tests to a solution of a battle.
[**R15**] The system allows Educators to view the rank scoreboard of a tournament they manage.
[**R16**] The system allows Students to view a list of all available tournaments.
[**R17**] The system allows Students to join a tournament.
[**R18**] The system allows Students to view a list of tournaments they are enrolled in.
[**R19**] The system allows Students to view a list of all available battles of a given tournament.
[**R20**] The system allows Students to create a team for a battle.
[**R21**] The system allows Team Leaders to invite other students to their team.
[**R22**] The system allows Students to view a list of invitations received.
[**R23**] The system allows Students to accept an invite.
[**R24**] The system allows Students to decline an invite.
[**R25**] The system allows Students to view a list of battles they have joined, given a tournament.
[**R26**] The system allows Students to view the description of a battle they are enrolled into.
[**R27**] The system allows Students to view the repository of a battle they are enrolled into.
[**R28**] The system allows Students to view the score of the solution submitted, given a battle, and after the solution has been graded.
[**R29**] The system allows Students to view the rank scoreboard of a tournament they are enrolled in.
[**R30**] The system runs GitHub Actions on battles' repository.
[**R31**] The system detects when a new version of the main branch is available.
[**R32**] System notifies student they have been invited to a team.
[**R33**] System assign automatic grading.
[**R34**] System notifies students a new battle is available.
[**R35**] System notifies students a new tournament is available.
[**R36**] System notifies a tournament has been closed.
[**R37**] System notifies a student that a battle has ended.
[**R38**] System notifies a team their solution has been graded.
[**R39**] System notifies educators that a new submission is ready to be graded.
[**R40**] System notifies students that the student they have invited to their team has accepted/rejected the invite.
[**R41**] System is able to fork the educator's repository.

3.2.2 Use Case Diagrams

In the following are provided the use case diagrams deduced from the scenarios reported in **??**. They help identify the actors interacting with the system and their role within each use case.

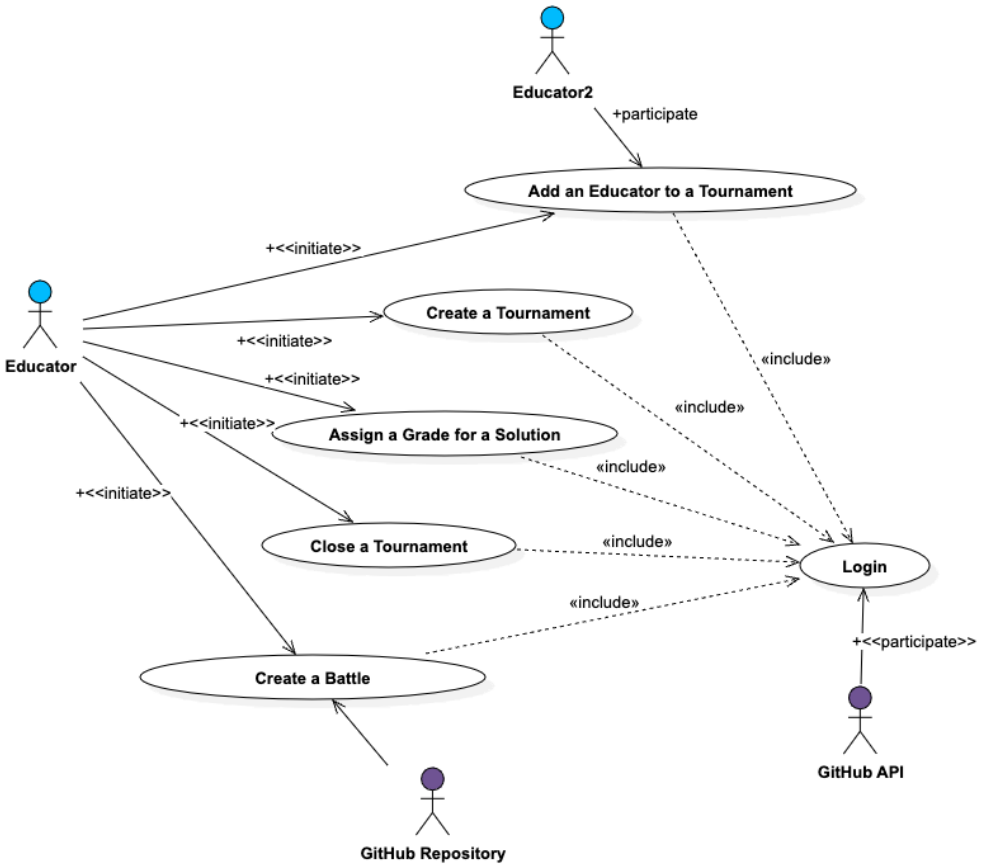

Figure 6: Educator Use Case Diagram

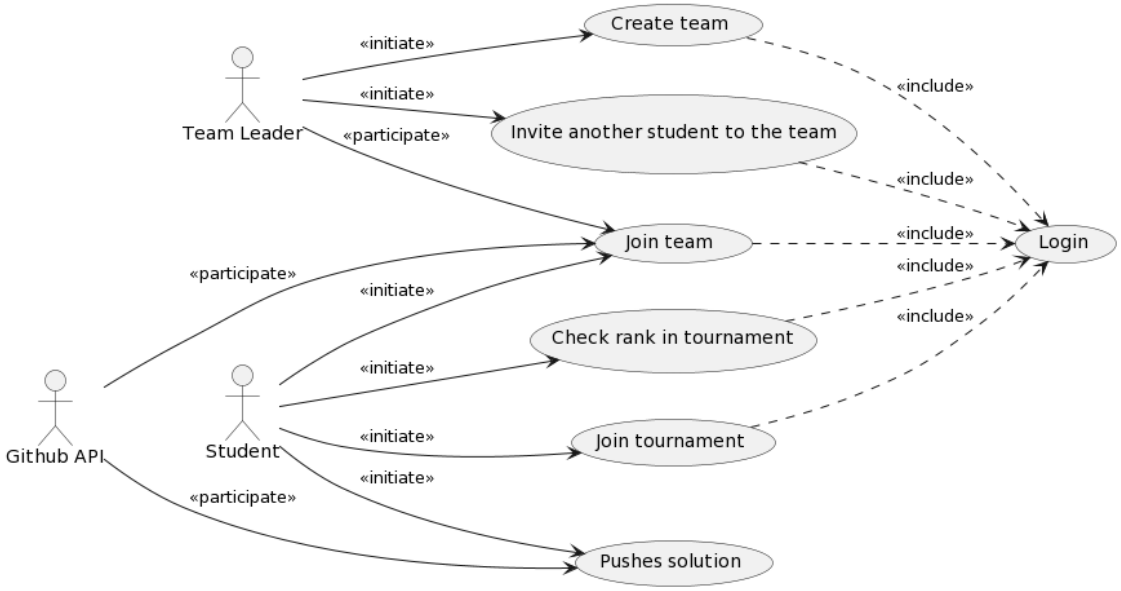

Figure 7: Student Use Case Diagram

3.2.3 Use Cases

[UC1] Educator creates a tournament

Name	Educator creates a tournament
Actors	Educator Students
Entry Condition	The Educator is logged in the system
Event Flow	1- The Educator clicks the "Create Tournament" button 2- The system shows a form, asking for details about the tournament 3- The Educator inserts the requested data 4- The system checks if the data is correct 5- The Educator clicks the "Confirm tournament creation" button 6- The system shows a success screen, along with info about the tournament 7- The system notifies all Students about the new tournament
Exit Condition	The Educator successfully creates a tournament
Exception	3- The data inserted is not valid. The system returns to Event 2

[UC2] Educator adds another educator in a tournament they manage

Name	Educator adds another educator in a tournament they manage
Actors	Educator Educator2
Entry Condition	The Educator is logged in the system and is managing a tournament
Event Flow	1- The Educator selects a tournament they manage 2- The system shows a screen with details about the selected tournament 3- The Educator clicks the "Add an Educator to tournament" button 4- The system shows a list of educators that are not currently managing the selected tournament 5- The Educator selects an Educator to add and clicks the "Add" button 6- The system notifies Educator2 7- The website shows a success screen, along with info about the tournament
Exit Condition	The Educator successfully added Educator2 to the tournament
Exception	5- Educator2 was already added previously

[UC3] Educator creates a battle

Name	Educator creates a battle
Actors	Educator Students
Entry Condition	The Educator is logged in the system and is managing a tournament
Event Flow	1- The Educator selects a tournament they manage 2- The system shows a screen with details about the selected tournament 3- The Educator clicks the "Create battle" button 4- The system shows a form, asking for details about the battle 5- The Educator inserts the requested data 6- The system checks if the data is correct 7- The Educator clicks the "Confirm battle creation" button 8- The system shows a success screen, along with info about the battle 9- The system notifies all Students about the new battle
Exit Condition	The Educator successfully creates a battle
Exception	5- The data inserted is not valid. The system returns to Event 4

[UC4] Educator assigns a grade for a solution

Name	Educator assigns a grade for a solution
Actors	Educator Students
Entry Condition	The Educator is logged in the system and is in a battle's management screen. The battle has ended.
Event Flow	1- The Educator clicks on the "Evaluate submissions" button 2- The system shows a list of all the submissions ready to be graded 3- The Educator selects the submission they want to evaluate 4- The system shows submission's code and the score of automated tests 5- The Educator selects the grade to assign 6- The system records the grade 7- The system notifies the Students
Exit Condition	The Educator successfully grades a submission
Exception	5- Invalid grade. The system returns at Event 4

[UC5] Educator closes a tournament

Name	Educator closes a tournament
Actors	Educator
Entry Condition	The Educator is logged in the system and is managing a tournament.
Event Flow	1- The Educator selects a tournament they manage 2- The system shows a screen with details about the selected tournament 3- The Educator clicks the "Close tournament" button 4- The system puts the tournament in the "Closed" state 5- On Exit Condition, the system puts the tournament in the "Completed" status.
Exit Condition	All battles are over

[UC6] Student joins a tournament

Name	Student joins a tournament
Actors	Student Github API
Entry Condition	The Student is logged in the system
Event Flow	1- The Student selects an open tournament they are not enrolled in 2- The system shows a screen with details about the selected tournament 3- The Student clicks the "Enroll into tournament" button 4- The system shows a success screen, along with info about the tournament
Exit Condition	The Student successfully enrolled in the tournament
Exception	1- The student selects an open tournament they are already enrolled in. The system does not show the "Enroll into tournament" button

[UC7] Student creates a team

Name	Student creates a team
Actors	Student
Entry Condition	
Event Flow	1- The Student selects the tournament 2- The System returns the view of the tournament 3- The Student select the battle 4- The System return the view of the battle 5- The Student ask the system to create a team 6- The System checks team name is available 7- The System checks the student is not in another team for this battle 8- The System creates a fork of the battle's repository on the student's account 9- The Team is created
Exit Condition	The Team is created
Exception	3- The student is already in a team, the team name is not available or the fork cannot be created

[UC8] Student joins a team

Name	Student joins a team
Actors	Student
Entry Condition	The Student is logged in the system, has enrolled in a tournament and has been invited in a team
Event Flow	1- The Student selects the tournament 2- The System returns the view of the tournament 3- The Student select the battle 4- The System return the view of the battle 5- The Student ask the system for a list of all the invites for the current battle 6- The System return the invites 7- The Student accept an invite 8- The System checks that the student is not in a team for this battle and the team is not at maximum capacity 9- The Student is invited to the github repository 10- Team leader is notified 11- The Student joins the team
Exit Condition	The Student joins the team
Exception	1- The Student is already in a team 2- The Team is at maximum capacity

[UC9] Student invites another student to the team

Name	Student invites another student to the team
Actors	Student Student2
Entry Condition	The Student is logged in the system, has enrolled in a tournament and is part of a team
Event Flow	1- The Student selects the tournament 2- The System returns the view of the tournament 3- The Student select the battle 4- The System return the view of the battle 5- The Student ask the system for a list of all available students for this battle 7- The System return the list 7- The Student select a student 8- The System checks that the student is the team leader 9- The System checks that the invited student is not in a team for this battle and the team is not at maximum capacity 10- The System sends the invite
Exit Condition	The Invite is sent
Exception	1- The invited student is already in a team 2- The Team is at maximum capacity 3- The Student is not team leader

[UC10] Student checks his rank for a tournament they are participating in

Name	Student checks his rank for a tournament they are participating in
Actors	Student
Entry Condition	The Student is logged in the system, has enrolled in a tournament
Event Flow	1- The Student selects the tournament 2- The System returns the view of the tournament 3- The Student ask the system for the rankings 4- The System return the rankings
Exit Condition	The System return the rankings

[UC11] A team pushes their solution to GitHub

Name	A team pushes their solution to GitHub
Actors	Student Github API
Entry Condition	The Student pushes a new version on the main branch in a repository associated with a team
Event Flow	1- The Student pushes a new version in the main branch 2- The Github API signals to the system that a new submission is available 3- The System checks if the team has the minimum number of members 4- The System runs the associated github actions retrieving the output 5- The System completes the automatic grading of the solution
Exit Condition	The System completes the automatic grading of the solution
Exception	1- The team has not the minimum number of members

[UC12] Battle's deadline is reached

Name	Battle's deadline is reached
Actors	Educator
Entry Condition	The System detects that a battle's deadline has been reached
Event Flow	1- The System changes the status of the battle to closed 2- The System notifies the students that the battle is closed 3- The System notifies the educator that they need to grade the solutions of this battle
Exit Condition	The System notifies the educator that they need to grade the solutions of this battle

[UC13] Login of a User

Name	Login of a User
Actors	User Github API
Entry Condition	The user logs in the system
Event Flow	1- The User ask the system to log in 2- The System redirects the user to the Github authentication 3- The User completes the Github authentication 4- The Github api returns an access token to the user 5- The User ask the system to log in with the access token 6- The System ask the Github api to verify the access token 7- The Github api returns that the token is valid 8- The System confirms to the user that they are logged in
Exit Condition	The user is logged in the system
Exception	3- The access token is not valid

3.2.4 Sequence diagrams

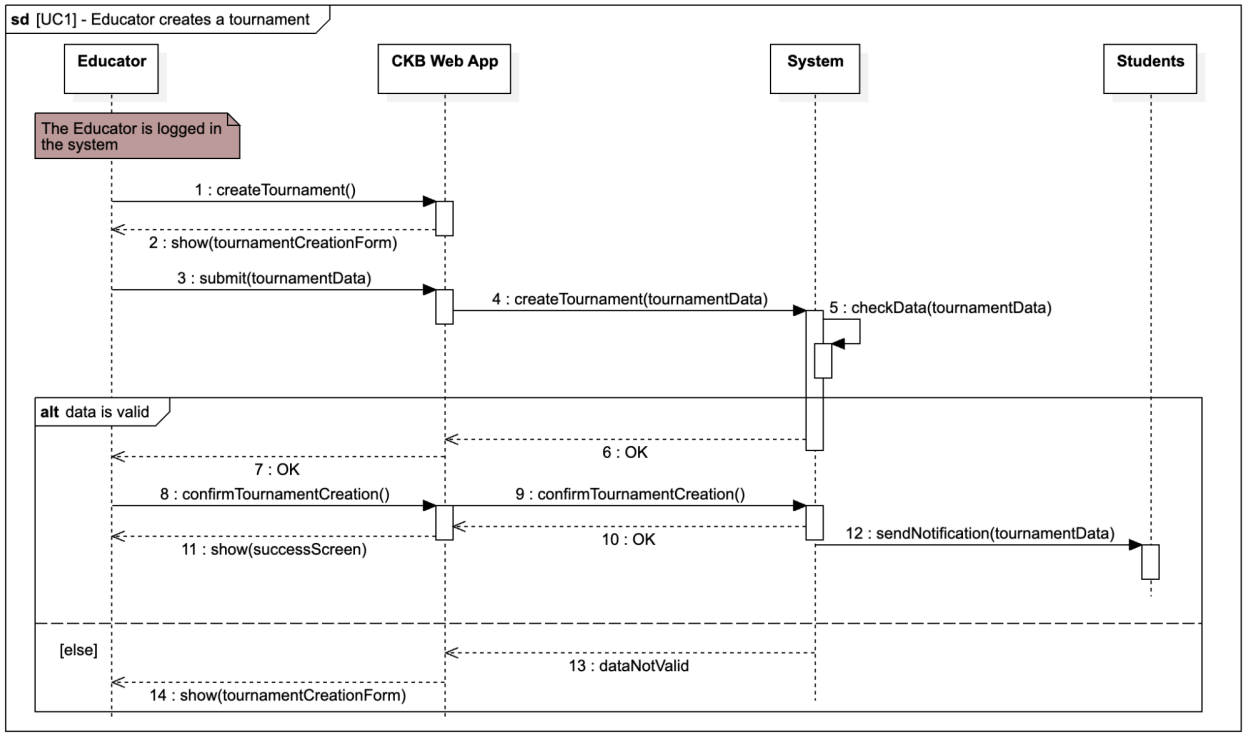

Figure 8: [UC1] Educator creates a tournament

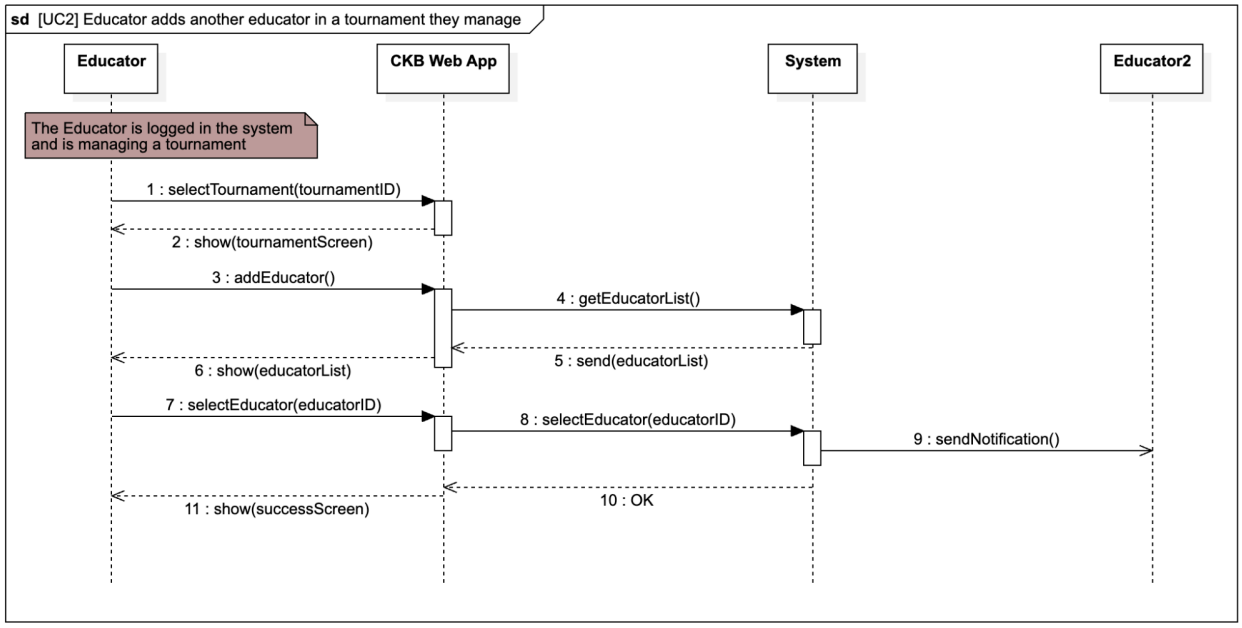

Figure 9: [UC2] Educator adds another educator in a tournament they manage

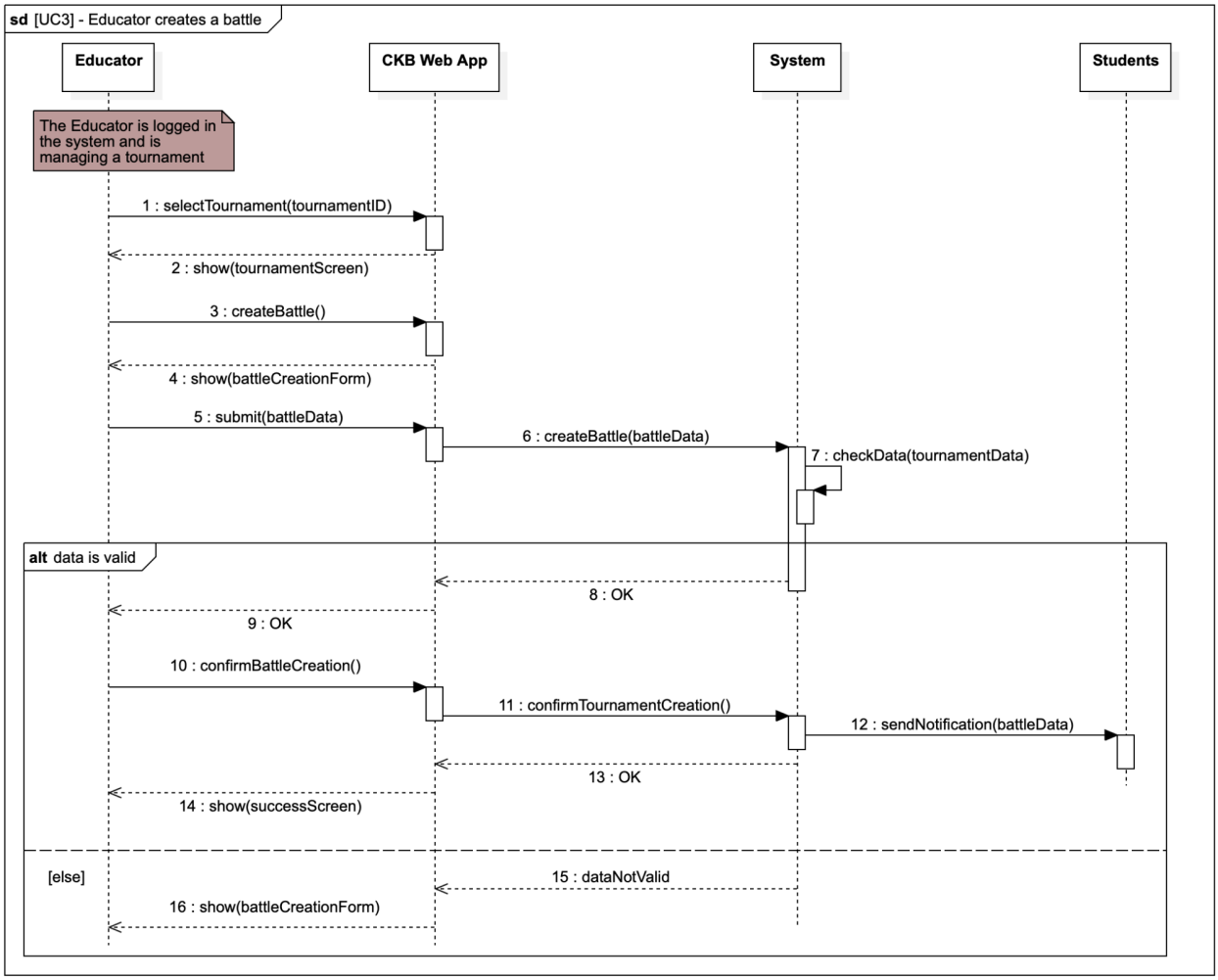

Figure 10: [UC3] Educator creates a battle

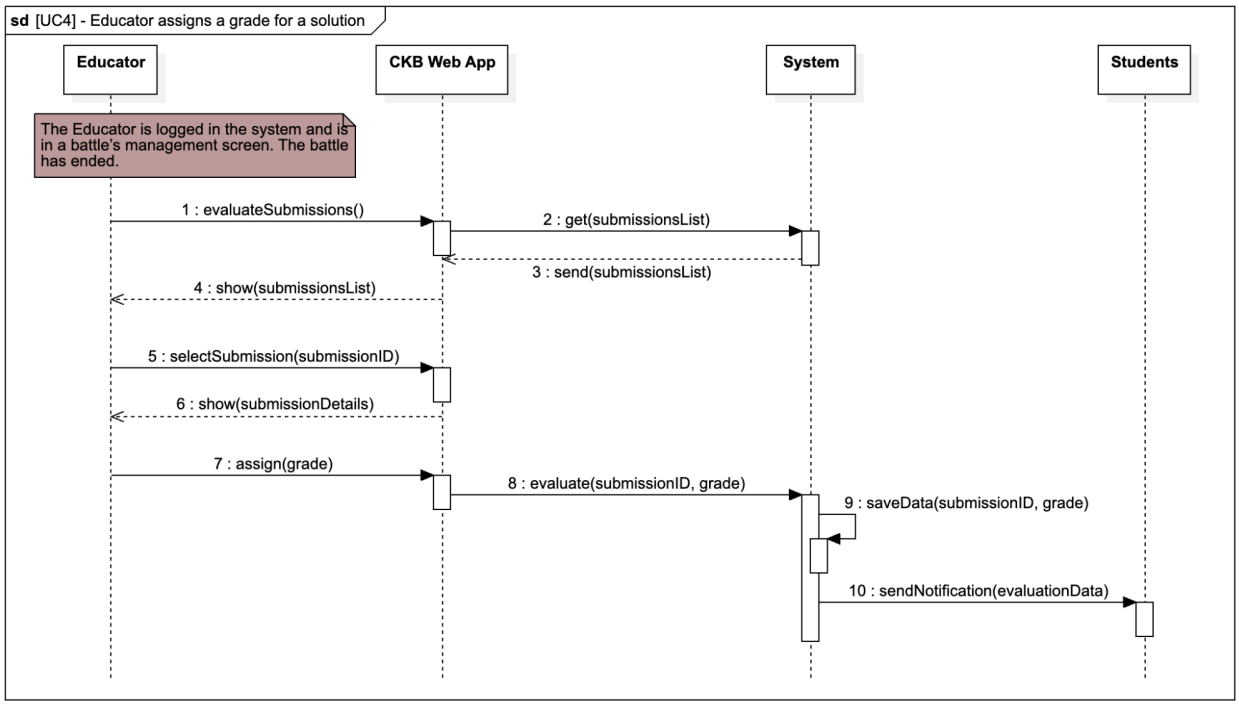

Figure 11: [UC4] Educator assigns a grade for a solution

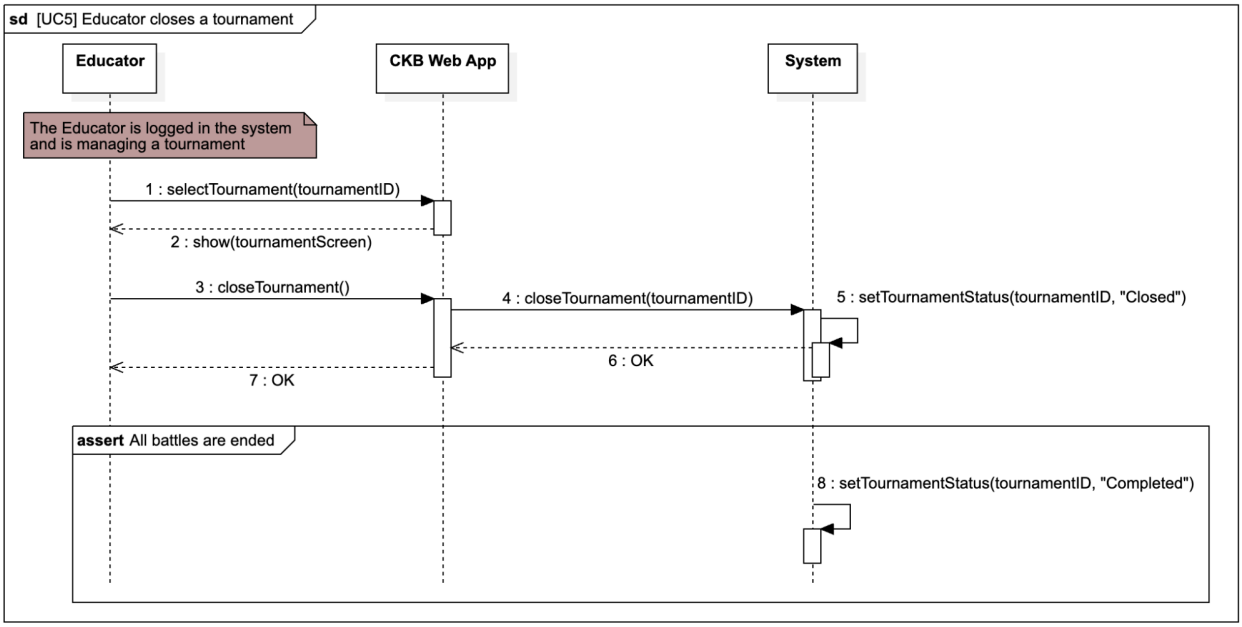

Figure 12: [UC5] Educator closes a tournament

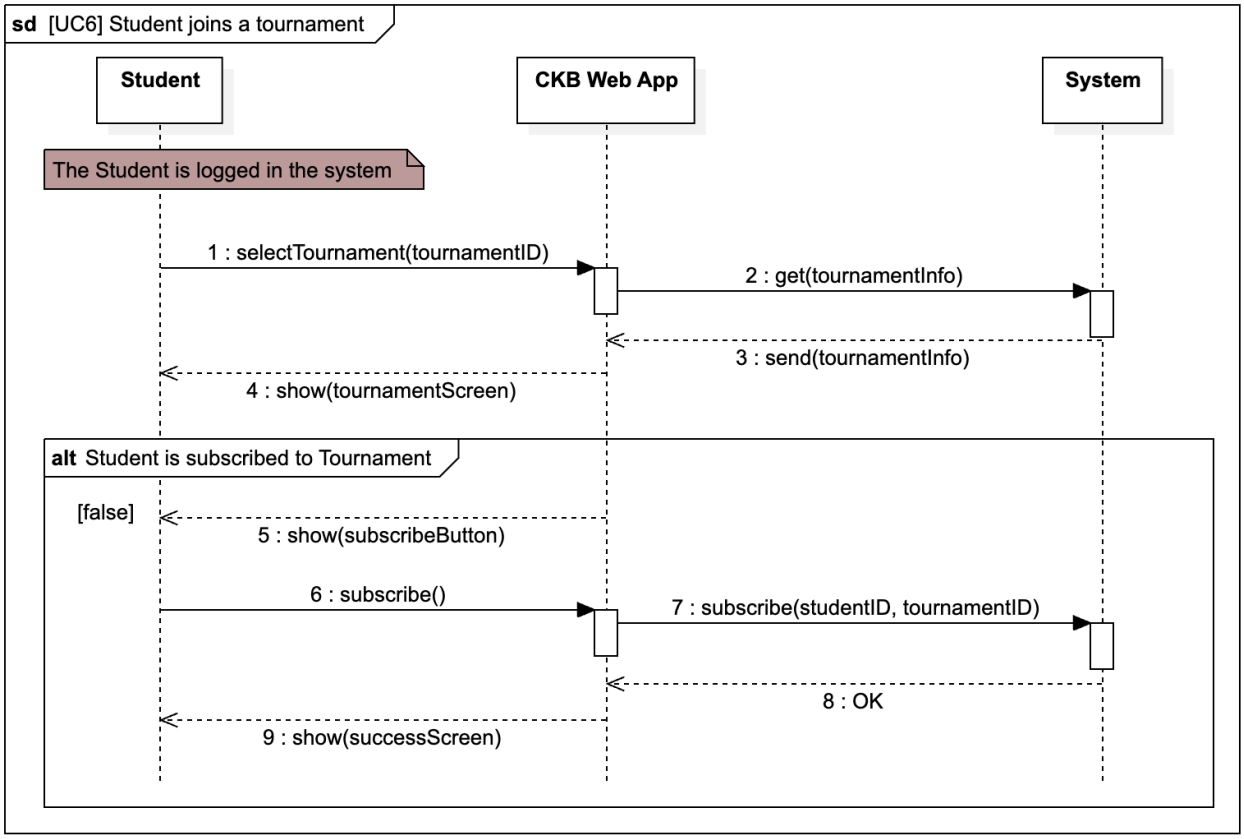

Figure 13: [UC6] Student joins a tournament

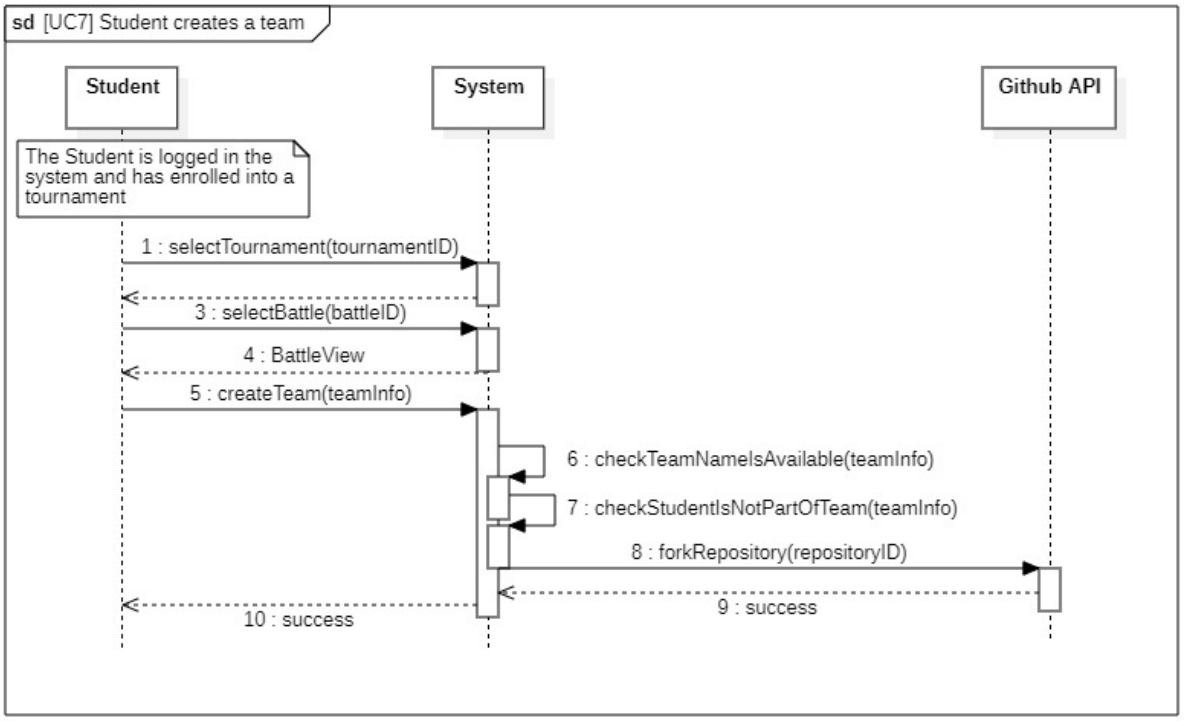

Figure 14: [UC7] Student creates a team

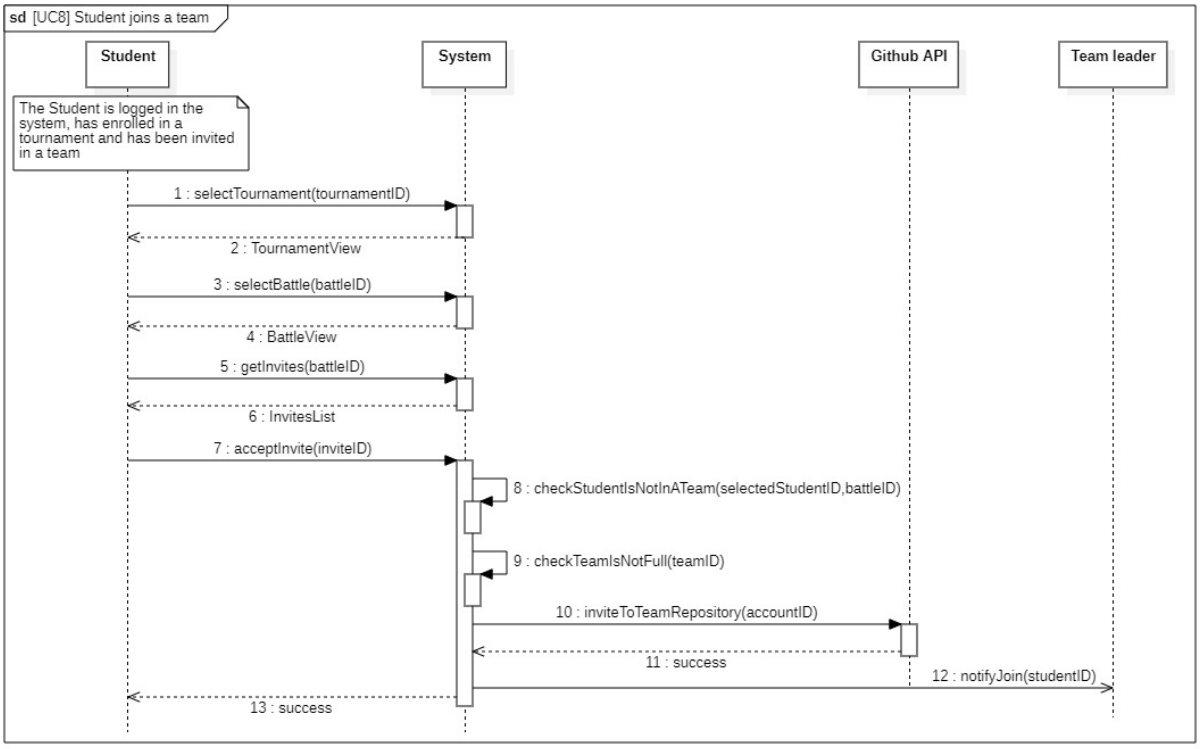

Figure 15: [UC8] Student joins a team

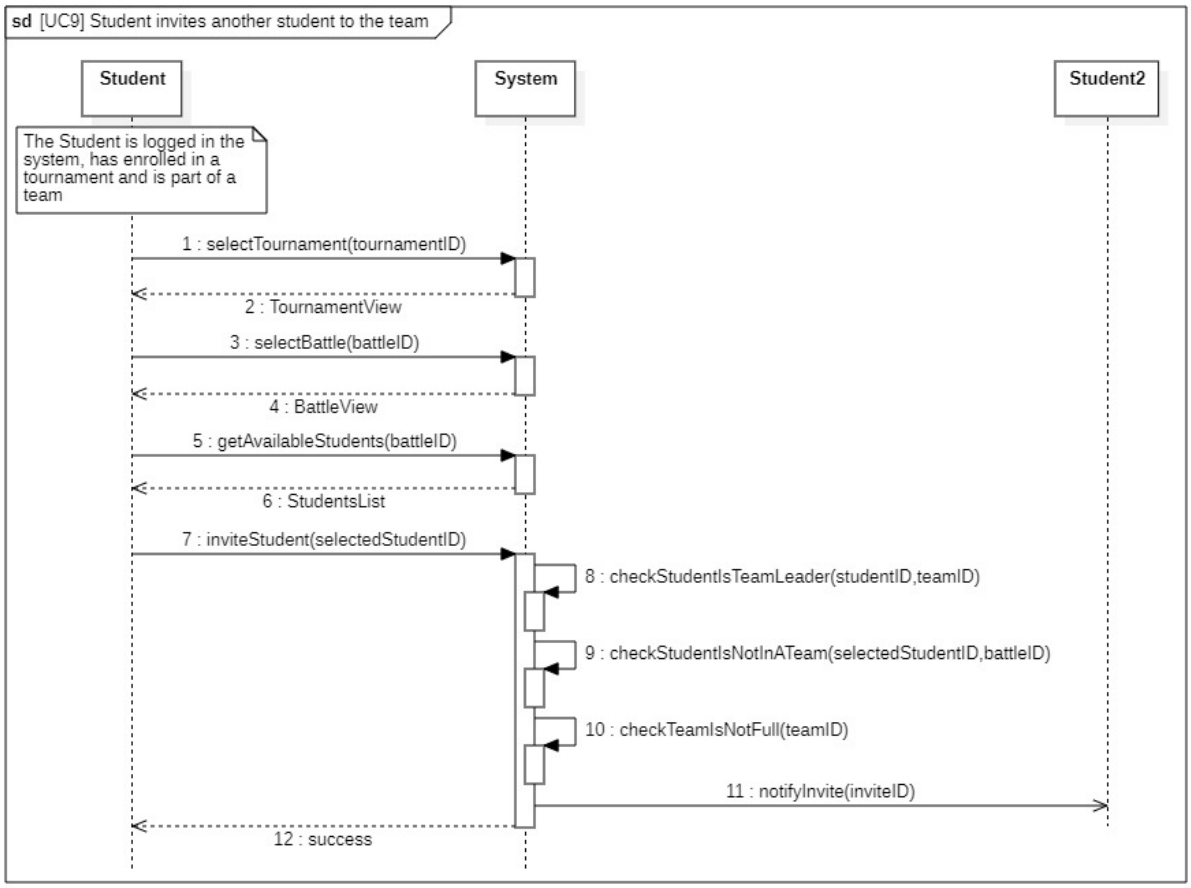

Figure 16: [UC9] Student invites another student to the team

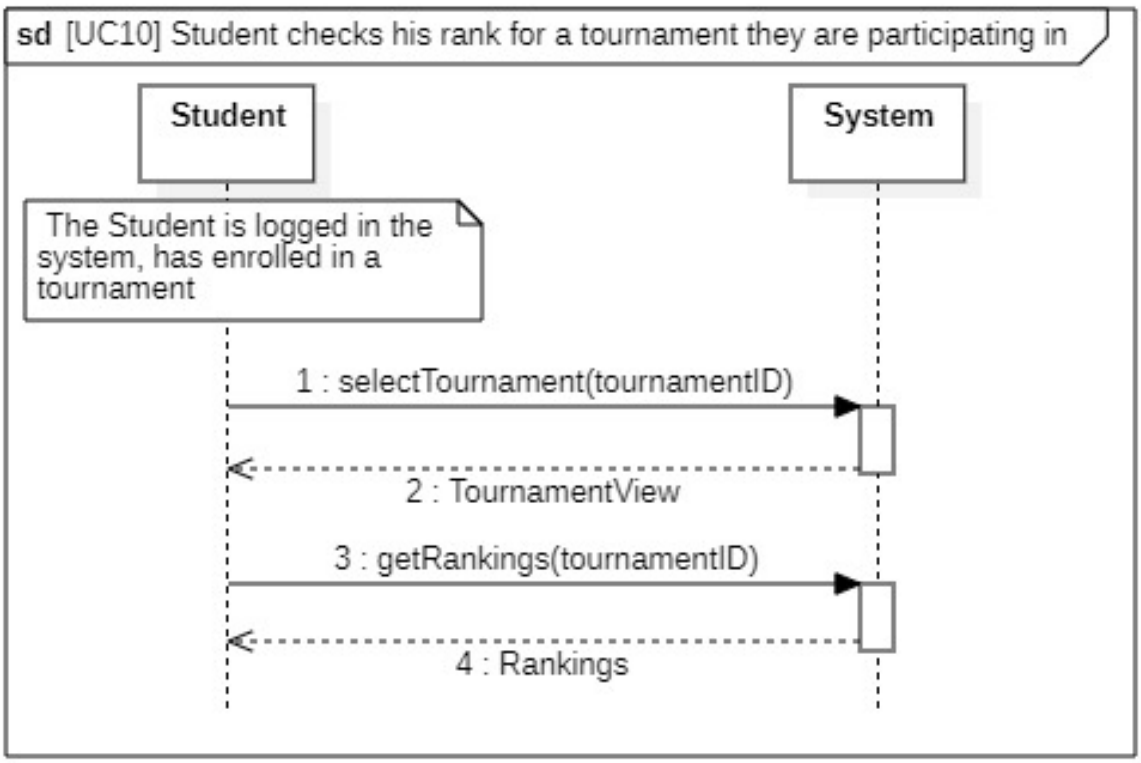

Figure 17: [UC10] Student checks his rank for a tournament they are participating in

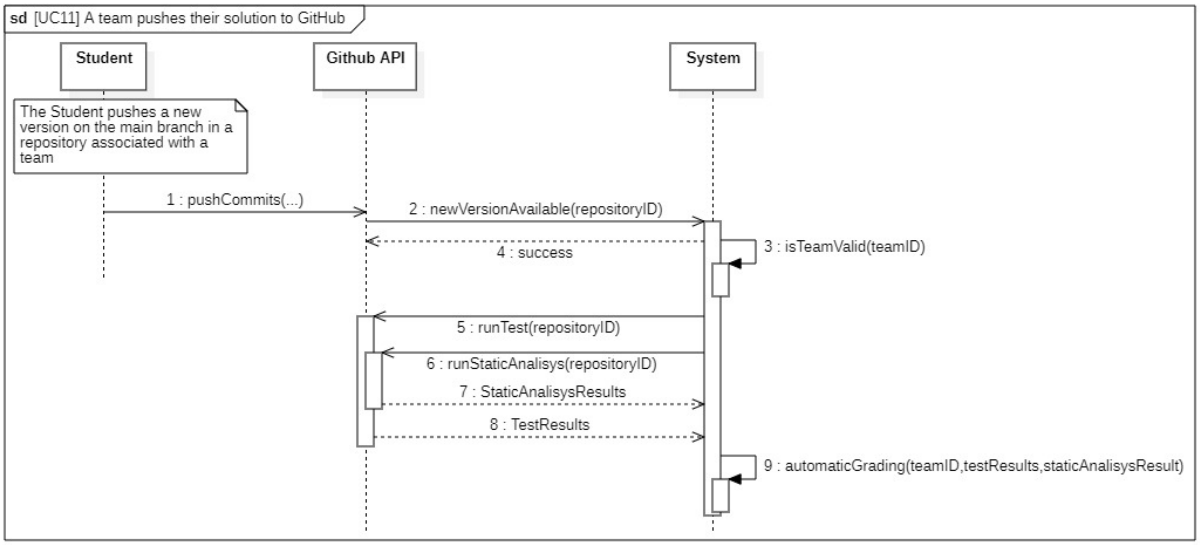

Figure 18: [UC11] A team pushes their solution to GitHub

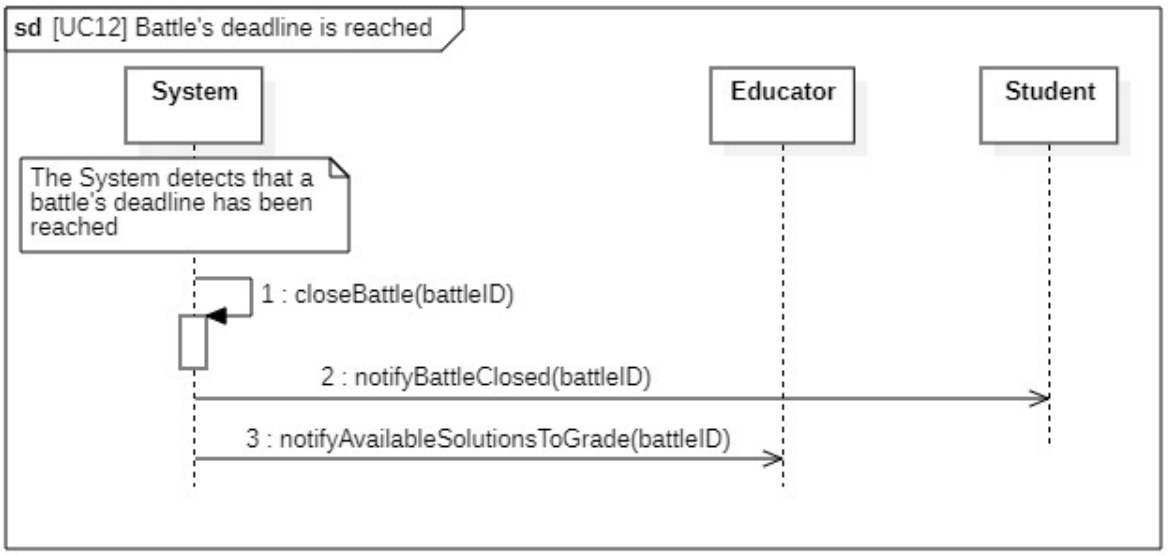

Figure 19: [UC12] Battle's deadline is reached

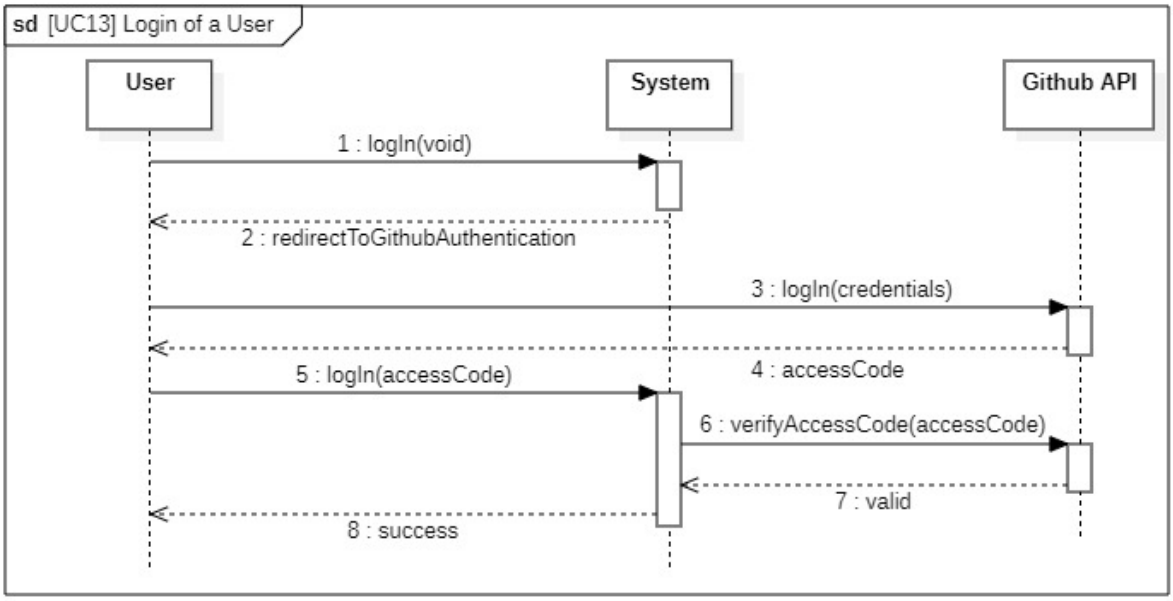

Figure 20: [UC13] - Login of a User

3.2.5 Requirements mapping

[G1] Educator is able to create and manage a tournament

[R2] The system allows Educators to log in via GitHub. [R3] The system allows Educators to create a tournament. [R4] The system allows Educators to view a list of only tournaments they manage. [R6] The system allows Educators to view a list of all Educators that manage the tournament they are managing. [R7] The system allows Educators to end a tournament they manage. [R8] The system allows Educators to view a list of Students that are enrolled in a tournament they manage. [R15] The system allows Educators to view the rank scoreboard of a tournament they manage. [R35] System notifies students a new tournament is available. [R36] System notifies a tournament has been closed.	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [D10] The system is able to connect to GitHub via internet connection. [D11] The system is reachable via the user's internet connection.

[G2] Educator is able to allow other educators to manage their tournament

[R2] The system allows Educators to log in via GitHub. [R4] The system allows Educators to view a list of only tournaments they manage. [D2] User has access to their GitHub account. [R5] The system allows Educators to add other Educators to a tournament they manage. [D10] The system is able to connect to GitHub via internet connection. [R6] The system allows Educators to view a list of all Educators that manage the tournament they are managing. [D11] The system is reachable via the user's internet connection.	[D1] User has a GitHub account.

[G3] Educator is able to create and manage code battles for a tournament they are involved in

[R2] The system allows Educators to log in via GitHub. [R4] The system allows Educators to view a list of only tournaments they manage. [R9] The system allows Educators to create a battle in a tournament they manage. [R10] The system allows Educators to view a list of teams that are participating to a battle they have created. [R11] The system allows Educators to view a list of submitted solutions for a battle they have created. [R12] The system allows Educators to view the repository of each solution submitted to a battle they have created. [R30] The system runs GitHub Actions on battles' repository. [R34] System notifies students a new battle is available. [R37] System notifies a student a battle is ended.	[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. [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.

[G4] Any user is able to authenticate with their GitHub account

[R1] The system allows Students to log in via GitHub. [R2] The system allows Educators to log in via GitHub.	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [D10] The system is able to connect to GitHub via internet connection. [D11] The system is reachable via the user's internet connection.

[G5] Student is able to enroll in tournament and join/create a team.

[R1] The system allows Students to log in via GitHub. [R16] The system allows Students to view a list of all available tournaments. [R17] The system allows Students to join a tournament. [R34] System notifies students a new battle is available. [R19] The system allows Students to view a list of all available battles of a given tournament. [R20] The system allows Students to create a team for a battle. [R21] The system allows Team Leaders to invite other students to their team. [R22] The system allows Students to view a list of invitations received. [R23] The system allows Students to accept an invite. [R24] The system allows Students to decline an invite. [R25] The system allows Students to view a list of battles they have joined, given a tournament. [R34] System notifies students a new battle is available. [R32] System notifies student they have been invited to a team. [R35] System notifies students a new tournament is available. [R40] System notifies students that the student they have invited to their team has accepted/rejected the invite. [R41] System is able to fork the educator's repository.	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [D7] Student accepts the invitation to the GitHub repository in time. [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. [D13] The Educator's repository is public.

[G6] Any Team participating in a battle is able to submit a solution.

[R13] The system allows Educators to grade a solution for a battle they have created, after the deadline of a battle. [R26] The system allows Students to view the description of a battle they are enrolled into. [R27] The system allows Students to view the repository of a battle they are enrolled into. [R30] The system runs GitHub Actions on battles' repository. [R31] The system detects when a new version of the main branch is available. [R33] System assign automatic grading.	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [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. [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.

[G7]Students are able to see their current rank and score in the tournaments they are taking part.

[R1] The system allows Students to log in via GitHub. [R18] The system allows Students to view a list of tournaments they are enrolled in. [R29] The system allows Students to view the rank scoreboard of a tournament they are enrolled in. [R36] System notifies a tournament has been closed. [R37] System notifies a student a battle has ended.	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [D10] The system is able to connect to GitHub via internet connection. [D11] The system is reachable via the user's internet connection.

[G8] Educators are able to see Student's current rank and score in the tournaments they are involved in.

[R2] The system allows Educators to log in via GitHub. [R4] The system allows Educators to view a list of only tournaments they manage. [R15] The system allows Educators to view the rank scoreboard of a tournament they manage. [R38] System notifies a team their solution has been graded.	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [D10] The system is able to connect to GitHub via internet connection. [D11] The system is reachable via the user's internet connection.

[G9] Educators are able to assign a grade to solutions submitted by Teams subscribed to a Battle created by the Educator.

[R2] The system allows Educators to log in via GitHub. [R4] The system allows Educators to view a list of only tournaments they manage. [R11] The system allows Educators to view a list of submitted solutions for a battle they have created. [R12] The system allows Educators to view the repository of each solution submitted to a battle they have created. [R13] The system allows Educators to grade a solution for a battle they have created, after the deadline of a battle. [R14] The system allows Educators to see the grade given by the automated tests to a solution of a battle. [R28] The system allows Students to view the score of the solution submitted, given a battle, and after the solution has been graded. [R33] System assign automatic grading. [R39] System notifies educator that a new	[D1] User has a GitHub account. [D2] User has access to their GitHub account. [D10] The system is able to connect to GitHub via internet connection. [D11] The system is reachable via the user's internet connection. [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. [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. [D14] The Student's repository is public.

3.2.6 Performance requirements

The system needs to be able to serve multiple users at the same time and to manage the grading and tracking of solutions in a manner that doesn't jeopardize the outcome of a battle.

3.2.7 Design Constraints

Being dependent on github the system will be able to manage users repositories and data only according to github api.
The system performance however is strictly connected with the github action performance.

3.3 Software System Attributes

3.3.1 Reliability

The system has to be able to run continuously without any interruptions for long periods. To be fault-tolerant the system backend deployment must take advantage of some sort of replication and redundancy. The system has to have offline backups of the data storage to exploit in disaster recovery after a data loss.

3.3.2 Availability

Given the fact that CodeKataBattle is not an emergency service or anything related to critical situations, the system must provide availability of 99

3.3.3 Security

The system doesn't store sensitive personal information about users, since every user of the system authenticates via GitHub. To communicate over the internet CodeKataBattle must use some sort of encryption to avoid traffic sniffing and spoofing, thus avoiding cheating attacks and guaranteeing privacy and consistency.

3.3.4 Maintainability

The system must guarantee a high level of maintainability. Appropriate design patterns should be used, together with good standards. The code must be well documented and hard-coding must be avoided. A testing routine has to be provided and it has to cover at least 85

3.3.5 Portability

The web application must run on any OS (like Windows, Mac OS, Linux, etc) that supports a web browser. Self hosted component of the stystem must be able to run on a Linux server.

4 Formal Analysis

With alloy we can formally verify that the constraints of the system are coherent and true

4.1 Signatures

With the signatures we describe entities that are part of the system with their relationships.

```
sig Student{}
sig Educator{}

sig Team{
    team_mates: some Student,
    battle: one Battle
}

sig Battle{
    creator: one Educator,
    tournament: one Tournament,
    teams: set Team,
}

sig Tournament{
    creator: one Educator,
    allowed_educators: set Educator,
    battles: set Battle,
    students: set Student
}
```

4.2 Predicates

```
pred CanCreateBattles[e : Educator, t:Tournament]{
    e in t.allowed_educators
}

pred CreatedTournament[e : Educator, t:Tournament]{
    t.creator = e
}

pred CanCloseTournament[e : Educator, t:Tournament]{
    t.creator = e
}

pred CanGradeSolutions[e: Educator, b: Battle]{
    e = b.creator
}

pred StudentInBattle[s: Student, b: Battle]{
    some t: b.teams | s in t.team_mates
}
```

4.3 Facts

```
fact CreatorIsAllowedEdcator{
    all e: Educator | all t: Tournament | t.creator = e implies e in t.
       ↪ allowed_educators
}

fact BattleIsPartOfTournament{
    all t: Tournament | all b: Battle | b.tournament = t iff b in t.battles
}

fact TeamIsPartOfBattle{
    all t: Team | all b: Battle | t.battle = b iff t in b.teams
}

fact StudentInTeamHasToBeEnrolled{
    all t: Team | all s: Student | s in t.team_mates implies s in t.battle.
       ↪ tournament.students
}

fact StudentCanOnlyBeInOneTeamPerBattle{
        all disj t1,t2: Team | t1.battle = t2.battle implies no (t1.team_mates
           ↪ & t2.team_mates)
}
```

4.4 Assertions

```
assert CreatorCanManageTournament{
    all e: Educator | all t: Tournament | CreatedTournament[e,t] implies (
       ↪ CanCloseTournament[e,t] and CanCreateBattles[e,t])
}

assert NoCommonTeammatesInBattle{
        all s: Student | all disj t1,t2: Team | (s in t1.team_mates and s in
           ↪ t2.team_mates) implies not (t1.battle = t2.battle)
}
```

4.5 Istance

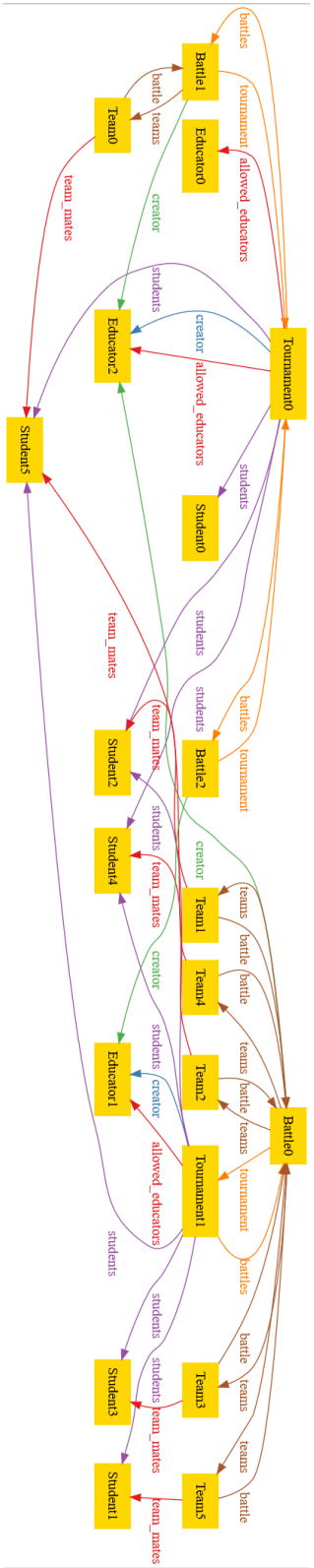

Figure 21: example of valid instance of the system

5 Effort Spent

Student	Section 1	Section 2	Section 3	Section 4
Merlino Lorenzo	3h	4h	5h	3h
Iodice Andrea	3h	5h	6h	?h

6 References

- UML diagrams: plantUML, starUML
- Alloy: vscode, alloytools