RASD-v0.9

Francesco Spangaro - Tosetti Luca - Francesco Riccardi

22 December 2023

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

The purpose of the CodeKataBattle platform is to create a friendly and enjoyable environment for students to acquire new skills and improve the ones already owned in Software Development. This is done by allowing students to train and compete with each others by writing code in order to resolve problems. All of this under the supervision of educators who can challenge their students to take part to these competitions.

1.1.1 Goals

G1: ***Allow educators to create new tournaments:***

Educators have the possibility to create new tournaments. When creating a tournament, educators have the opportunity to create new badges. Badges have corresponding achievements, called "Rules", which are defined on badge creation. Badges are obtained by users on achievement completion. Obtained badges will be then displayed on the user's profile page.

G2: ***Allow educators to create new battles:***

Educators have the possibility to define new battles within tournament they crated or in tournaments they have been granted permission to do so. When creating a new battle educators have to set different parameters:

- upload project description;
- specify the programming language and build tool to utilize, including test cases and build automation scripts;
- set minimum and maximum number of students per group;
- set a registration deadline;
- set a final submission deadline;
- set additional configuration for scoring.

G3: ***Allow educators to administer different tournaments:***

Educators can grant other colleagues permission to create new battles in their tournaments. Educators have the possibility to close their tournaments, thus, not letting students submit new answers to any battle defined in the closed tournament, nor letting their colleagues create new battles in that tournament.

G4: ***Allow educators to administer different battles:***

Educators have the possibility, once a battle has expired, to manually evaluate through the platform each student's work, and then assign a corresponding score to each one of them, ranging from 0 to 100.

G5: ***Allow students to subscribe in tournaments:***

Students subscribed to the platform have the possibility to subscribe to different tournaments, in which they plan to participate in.

G6: ***Allow students to participate in battles:***

Students can join battles within a set deadline. They can do so by themselves, by inviting somebody else or by accepting someone's else invite.

G6.1: ***Allow students to form groups to participate with:***

Students have the possibility to send out invitations to other students, so that they can form a group to participate with. Groups need to follow the guidelines specified by the battle creator for it to be accepted.

G6.2: ***Allow students to submit their answers:***

When students have developed a solution to the battle, they can submit their answer to the platform. Groups are requested to send only one answer. Students can change their answer as they proceed, when uploading a new solution the older one is overwritten, since there can only be one answer for each battle.

G6.3: ***Allow students to see their scores and badges:***

After each answer submission, a new score is assigned to the students. The score can be manually created by an educator or automatically assigned to the students by the platform. Students can see the scores obtained in a battle and in a tournament they partecipated to. Finally students can even see the badges obtained in a tournament.

G7: ***Let the students be notified on important events:***

When a new tournament is created, all students subscribed to the platform are notified. A different notification will be sent when a new battle is created in a tournament they are subscribed to.

1.2 Scope

CodeKataBattle (CKB) is an easy-to-use platform, which aims to allow educators to propose homework and/or lessons in a new and fresh way to involve students more in acquiring and improving software developing skills. To do that, CKB offers educators the possibility to open several tournaments. Each

tournament is composed by several battles in which students can compete with each other, individually or in groups. In order to offer all of this CKB relies on the external platform GitHub. GitHub will take the role of a "bridge" between CKB platform and students, allowing them to upload their solutions on it. These solutions will be then taken by the CKB platform from GitHub and used to evaluate student's score in the battle for which they uploaded a specific solution.

1.2.1 Phenomena

Events that take place either in the real world, in machine world or in both. Used to describe respectively what cannot be observed by the machine, real world and event that connects the two.

1.2.1.1 World phenomena

Phenomena events that take place in the real world and are not observable by the machine

WP1: Students fork the GitHub repository of which they received a link by the platform.

WP2: Student write code on his personal device.

WP3: Students choose which tournament to join

WP4: Students choose which battle to join in a tournament precedently joined.

WP5: Student choose his teammates for a battle.

WP6: Educator choose whether and which colleagues to allow access to one of his tournaments

WP7: Student subscribed to a battle wait for its start (registration deadline expiration)

WP8: Educator decide to close a tournament

1.2.1.2 Shared phenomena

- Phenomena controlled by the world and observed by the machine

  ➤ Student related phenomena

  SP1: Student registration to the platform
  SP2: Student log in to the platform
  SP3: Student subscribe to a tournament within a deadline
  SP4: Student invite other students to form a team
  SP5: Student accept an invite from another student and join its group.

SP6: Student or a group of students join a battle in a tournament they are subscribed to within a deadline.

SP7: Student upload a new software solution for the battle's problem, in which he's partecipating

SP8: Student sees its, and others badges visualizing its or others profile page

SP9: Student or a group of it, push a new commit on GitHub repository

➤ Educator related phenomena

SP10: Educator create a new tournament

SP11: Educator grant access to his other collegues to create new battle within a tournament he created

SP12: Educator create a new battle

SP13: Educator set battle's setting while creating one of them

SP14: Educator manually evaluate the work done by students in a certain battle of a certain tournament during battle's consolidation phase

SP15: Educator closes a tournament

SP16: Educator defines new badges achievable in a tournament by students while creating it

SP17: Educator sees collected badges of a student by visualizing its profile page

- Phenomena controlled by the machine and observed by the World.

➤ Student related phenomena

SP18: Student registered to the platform gets notified when a new tournament is created

SP19: Student subscribed to a tournament gets notified of upcoming battle created in that tournament

SP20: Student receive from the platform a invite notification in order to join a group of students to join a battle.

SP21: Platform, when a battle's registration deadline expires, send every student that joined the battle a link to the GitHub repository created by the platform itself.

SP22: Platform at the end of each battle updates students' score in the tournament in which battle took place allowing all students and educators to see them

SP23: Students get notified when a tournament is closed

➤ Educator related phenomena

SP24: Educator receive a notification when allowed by a collegue to access its collegue's tournament.

SP25: Educator gets notified when submission deadline of solution for a battle expires, and start the consolidation phase.

1.2.1.3 Machine phenomena

Phenomena events that take place in the machine world and are not observable from the real world

MP1: The platform creates a GitHub repository containing the code kata when registration deadline of a battle expires.

MP2: The platform when notified by GitHub API pulls the latest sources of the repository of a battle

MP3: The platform analyzes the sources, by running tests on them.

MP4: The platform calculate the battle score of a team based on the executables uploaded by students for a battle. Score is automatically updated when the platform receive notification from GitHub about new push actions.

MP5: The platform at the end of each battle of a tournament, compute the tournament rank of each student in that tournament.

MP6: The platform automatically register badges acquirement from a student, when that student satisfy the rules to obtain them.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

Term	Definition
GitHub Repository	→ A place on the GitHub platform where a user can store code, files and each file's revision history.
Registration deadline	→ Maximum time within which a student can subscribe to a battle or to a tournament.
Submission deadline	→ Maximum time within which a student, or a group of student, can upload their solution to a battle problem

Term	Definition
Code Kata	→ The word kata refers to a karate exercise in which a form gets repeated many time, making little improvements in each tentative. In this context it's used to express the fact that the code need to be developed multiple times to reach a optimal solution to the battle problem.
Consolidation phase	→ Phase started at the end of a battle, used to consolidate the score of each student in the battle by eventually a manual evaluation of the students' code by an Educator.
View-only mode	→ An abstract modality (not an implementation detail) in which educator can be if he access to a tournament in which they have no rights to create new battles
Modify-enabled mode	→ An abstract modality (not an implementation detail) in which educator can be if he access to a tournament in which he go granted the access by tournament's creator, and thus can create new battles.

1.3.2 Acronyms

Acronym	Meaning
API	→ Application Programming Interface, indicates on demand procedure which supply a specific task
CKB	→ CodeKataBattle, the name of the platform described in this document
IT	→ Used as acronym for Information Technology which identify something, generally a computing or communication hardware, with information storage capability and closely related to the informatic world.

Acronym	Meaning
UML	→ Unified Modeling Language, a standard notation for modeling real world object, in an high level diagram representing OO components.
BPMN	→ Stands for Business Process Modeling Notation, standard notation to represent processes through diagrams.
OO	→ Object-Oriented is a programming paradigm based on the concept of object which can contain data and code, and represent usually real world object.
DB	→ Database, a set of data held in a computer which is accessible, modifiable, queryable in various ways.
GDPR	→ General Data Protection Regulation, an European law to protect the privacy and security of users' data.

1.3.3 Abbreviations

Abbreviation	Meaning
G#	→ Goal number #
WP#	→ World phenomena number #
SP#	→ Shared phenomena number #
MP#	→ Machine phenomena number #
D#	→ Domain assumption number #
R.#	→ Requirement number #
c.s.	→ Computer science
e.g.	→ Exempli gratia, latin phrase corresponding to "for example"

1.4 Revision history

- **Placeholder data**: version 1.0
- **Placeholder data**: version 1.1

- **Placeholder data**: version 1.2
- **Placeholder data**: version 2.0
- **Placeholder data**: version 2.1

1.5 Reference documents

GitHub references:

- Official documentation to get started with GitHub: → `https://docs.github.com/en/get-started/quickstart`
- Official documentation about fork process → `https://docs.github.com/en/get-started/quickstart/fork-a-repo`
- Official documentation about GitHub actions → `https://docs.github.com/en/actions`

UML official specification → `https://www.omg.org/spec/UML`

BPMN official specification → `https://www.omg.org/spec/BPMN/2.0`

Use case diagrams specification used → `https://it.wikipedia.org/wiki/Use_Case_Diagram`

Sequence diagrams specification → `https://www.uml-diagrams.org/sequence-diagrams.html`

1.6 Document structure

- ***Section 1: Introduction***
  This section introduces the problem and the platform/application to be developed in order to resolve it. It describes the major purpose of the project, every one of his goals, the analysis of the its domain and every world, shared and machine phenomena associated with it. In addition in this section are inserted the definitions, acronyms and abbreviations used in this ddocument, including even it's revision history and refereced documents or web pages.
- ***Section 2: Overall description***
  This section gives an overall description of the project and all the interactions that could occur between the platform and the final users (Students and educators). To do this, it will include different possibile scenarios that could happen, the different actors involved in the platform usage and all the assumptions, dependencies and eventual constraints that have to be considered in the development of the platform

- ***Section 3: Specific Requirements***
  This section is the most technical one, it contains more precise descriptions of each scenario called use cases. It describe the several functional and performance requirements of the project and their map to the goals of the project. Finally it contains also all the design constraint and system attributes that must be followed/guaranteeded while developing the platform.

- ***Section 4: Formal analysis using alloy***
  In this setion can be found a formal description of the platform/application. The formal description is done using the formal language Alloy (referenced in section 1)

- ***Section 5: Effor spent***
  A simple section in which are included all the information about the time requested by each group member to complete this document, and it's division by each section of the document.

2 Overall description

2.1 Product perspective

The following section contains the UML diagram of the platform and a list of meaningful scenarios in which the platform can be used, how and when users can interact with it.

2.1.1 Scenarios

2.1.1.1 Student sign up to the platform

Peter is a volenterous student of an informatic class, that want to improve his software development skills. He comes to know about the platform CKB one day, when his c.s. professor propose to his class, a software development tournament in substitution to the normal, boring and limited tests organized through the academic year. Peter is extremely interested, not only in the tournament, but also in the service offered by the platform itself, thus the same day he decide to try to register to it. Peter open his personal browser and go to the CKB site's homepage. Here he goes to the student dedicated page and clicks the "Sign up" button in order to register. He inserts all the required information in the mandatory fields of the page that showed up (e.g. name, surname, attended school, email, username, password, ...) and clicks on "Confirm" button. Peter now wait for the notification, through his email, about the correctness of his registration, and can now access all the CBK platform's features after loggin in with it's credentials.

2.1.1.2 Educator sign up to the platform

Vittorio is an innovative teacher who teaches c.s. in a school. He discovered, while searching the web for new ideas on how testing his students' abilities in software developing, the CKB platform and its services. Vittorio decide to try to register to the platform, as he's very intrigued about it. To do that Vittorio goes on the CKB Homepage and then on the page dedicated to educators and clicks the "Sign up" button. At this point he compiles all the fields in the form that showed up with the new page (e.g. name, surname, school in which he teaches, istitutional email, password, ...), especially the fields related to his profession, and in the end clicks on "Confirm" button. Finally Vittorio wait for the registration confirm email, and start using CKB platform's features after loggin in with it's credentials.

2.1.1.3 Educator creates a new tournament (and badges)

Laura is an c.s. educator who registered to CKB platform. She decide to create a new tournament to let her student compete with each other and improve their software developing skills. In order to do so, Laura log in her account on the platform and try to create a new tournament. While doing so, she decide which programming languages can be used to develop the solution to the battles that will be contained in the tournament, the name of the tournament, the method through which students can access it, eventually its maximum duration, and finally she decide if the tournament will contains some new or default badges, obtainable by the students by doing some achievements. Laura wants to create new particular badges, in order to encourage her students to participate more actively to the tournament. To do so, Laura access the appopriate section during tournament creations, and starts creating the badges, specifing their title, rules to obtain them, their icon and eventually their score, that will be added to the student's score that obtained them. At the end of the process Laura, confirm her choices and the tournament starts.

2.1.1.4 Educator creates a new battle

Laura is an c.s educator who registered to the CKB platform. After she created a new tournament, Laura wants to immediately create a new battle within it, to let her students compete with each other. To do so Laura, log in her account on the platform, access the tournament in which she wants to create a new battle through an appropriate web page. At this point Laura try to create a new battle within the tournament. While doing so, she decide the battle's name, the programming language allowed in the battle, maximum number of partecipant, dimension of group of students that can partecipate to it, a registration and submission deadline and eventually sets some personalized rules to calculate students' score in the battle. At the end of all of this process, Laura waits for the students of her class, subscribed to the tournament in which the battle has been created, to join the battle. At the end of the registration deadline, partecipant can start competing with each other, while Laura can supervise their work. At the end of the battle, which automaticcaly close itself at submission deadline, students' score gets calculated and added to their tournament score.

2.1.1.5 Educator close a tournament

Marco is an c.s educator who registered to the CKB platform. He has created a new tournament one mounth ago, today, after this mounth he decide to close it. In order to do so, Marco log in his account on the platform and access the tournament which he wants to close through an appropriate web page, at this point Marco control il there are battles that aren't altready closed and sees that Laura has a tournament that is still open, so he now control when is the submission deadline of this battle. After discovering that the submission deadline is in two days he close the possibility of other educators to creates new

battles. After three days Marco, after log in to the platform, control that there are'n still open battles or battles that are waiting for the manual evaluation. Then he proceed to close the tournament

2.1.1.6 Educator manually evaluate a battle after the end

Luca is an c.s educator who registered to the CKB platform. He has created a new battle in a tournament two weeks ago, choosing as the submission deadline yesterday and choosing to include a Manual evaluation in the total evaluation of the battle, according to the requirments of shared balaced partecipation, optimization and writing comments related to the code and the implementative choice. Today, Luca log in his account on the platform, access to the tournament in which he created the battle and see that this battle is closed and is waiting for a manual evaluation and decide to examine the codes that have been delivered in the battle by the different groups. The first group he analyzes is made of three students, and sees that they have a score of 90/100 in the automated evaluation. After entering on the resource made available by the group, Luca check if the number of commits and work has been done equally among the students in the group and not noticing large differences between the number of lines of code written and the number of commit made by each student he decide to go next to the comment section. Opening the code in the resource, he relizes that the group has limited himself to adding simple comments, without going to much in the details of the work done and choices made; so he decide to modify the grade assigned befor, by reducing it from 90/100 to 80/100, writing a little explanation of the reason for this change. It then continues to control all the others group and at the end he can close definitively the battle.

2.1.1.7 Student forms a group then joins a battle

Francesca signs in the platform with her credentials and sees that a new tournament has opened. She would like to participate in it, and so decides to form a group with two of her friends to compete in the tournament's battles. She clicks on the apposite menu and chooses the users she would like to invite to join her group. After both her friends accept her invite, she enters the tournament by signing up her group as participant.

2.1.1.8 Student joins a battle

Andres is subscribed to a tournament with his group. One day he is notified that a new battle for the tournament he is subscribed to is available. After reading the battle's description he decides that he wants to try and join in. He autonomously asks his group members if they want to join the battle too (through email or another external messaging platform). After reading their answers, he subscribes the group to the new battle. Once the registration deadline for the groups expires, Andres' group forks the automatically created GitHub reposi-

tory and sets up an automated workflow through GitHub Actions, to notify the platform of each of their commits.

2.1.1.9 Student uploads a solution

Piero is currently competing with his group in a tournament, they are working on a solution for a battle and they are confident that they are right with what they coded. Piero decides to upload a solution for the battle to the GitHub repository, so that it can be evaluated and they can see how much their score is for this first draft solution. Piero uploads the current solution to the forked GitHub repository. After the push, the GitHub Actions workflow is started, letting the platform know that Piero has uploaded a new solution for the battle he's currently competing in. The automated evaluation system integrated in the platform now starts, tests are automatically ran and no human evaluation is deemed needed, so the platform gives Piero's solution a score of 75/100. Piero sees the score on his profile on the platform and understands that his solution is found correct, but not the best that can possibly be done. He decides to continue working on it, until his solution will obtain a score of at least 95/100.

2.1.1.10 Student uploads a solution after deadline expiration time

Frank has been competing in a tournament for very long, and has finally found a solution for the last battle he struggled to solve. After having uploaded his solution on the forked repository, the GitHub Actions workflow starts and notifies the platform that Frank's has uploaded a new solution for his final battle. The CodeKataBattle platform then sees that the deadline for Frank's battle has expired. The platform notifies Frank that his last uploaded solution will not be considered on the final score he will get, and no further action is taken.

2.1.1.11 Students visualize his tournament's results and his badges

Paolo, Lucia e Alessandro are three friends that are subscribed in the CKB platform, one evening they decide to compare their tournament's result on the CKB platform. First, they want to compare their result in the last tournament, which ended the day before. Paolo, after logging in the site and entering his profile page, sees that he achived a score of 70/100. Lucia in last tournament was in the same group as Paolo, so she achived the same score. Alessandro, who was in a different group, sees that he has a score of 90/100, thus becoming the best of the three. Lucia now wants to see the best score they have ever achieved, so after logging in the site and searching for her best result, found out that she participated in a tournament two months prior, where she achived a score of 95/100. Alessandro wants to see his best result and after logging in the site, he sees that in his badges there is the "perfect score" badge, which is assigned

to a student who got a score of 100/100 in at least one tournament. Paolo then wants to know who out of the three has the most badges. After accessing his profile page, he sees that he has obtained a total of 57 badges, obtained in all of the tournaments in which he participated; in this badges Paolo sees that have achived badges for being the most committer of a tournament, one for having partecipated at 50 tournaments and one for having write the most number of code row in a tournament. Alessandro checks that too, and sees that he obtained only 13 badges.

2.1.2 Class diagram

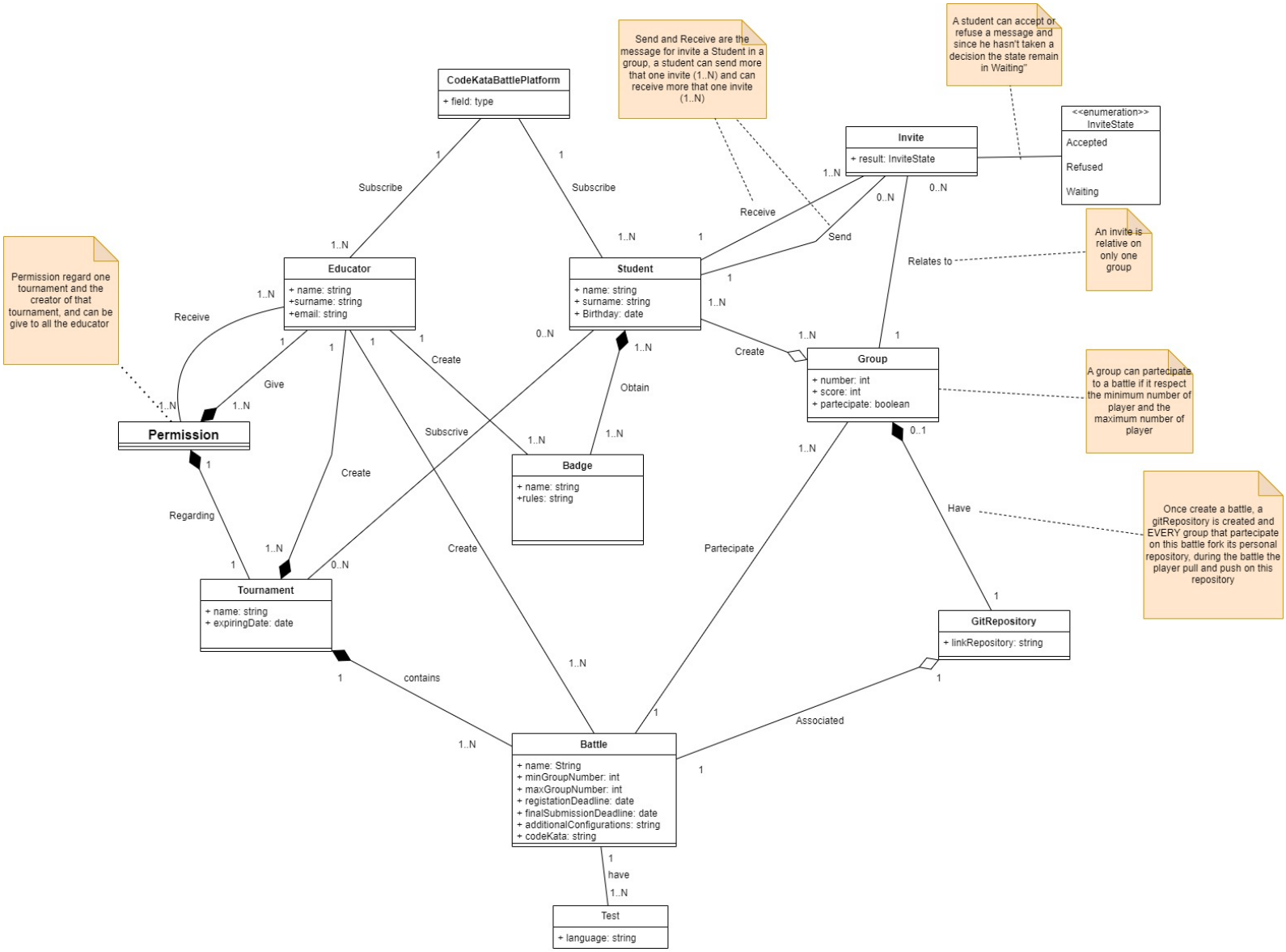

Figure 1: Class Diagram

2.2 Product functions

2.2.1 Shared functions

- **Sign-up:** Let the user (either students or educators) sign-up to the platform.

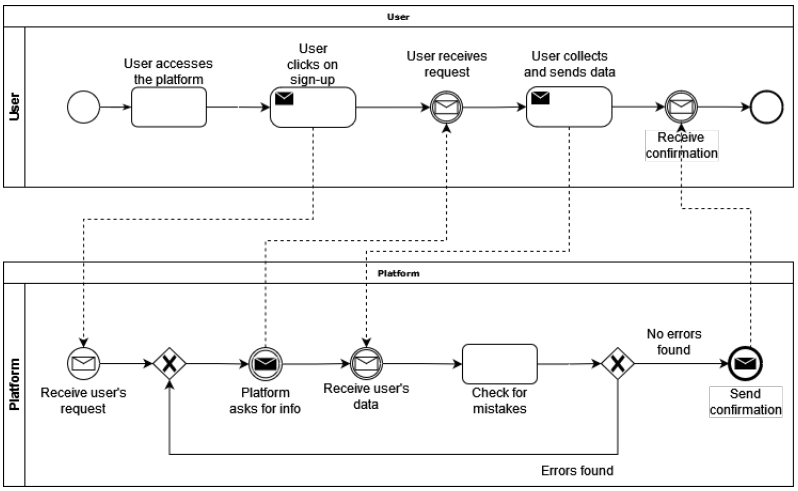

Figure 2: Sign-up BPMN

- **Visualize student's profile:** Let an user (either a student or an educator) to visualize the profile page of a specific student.

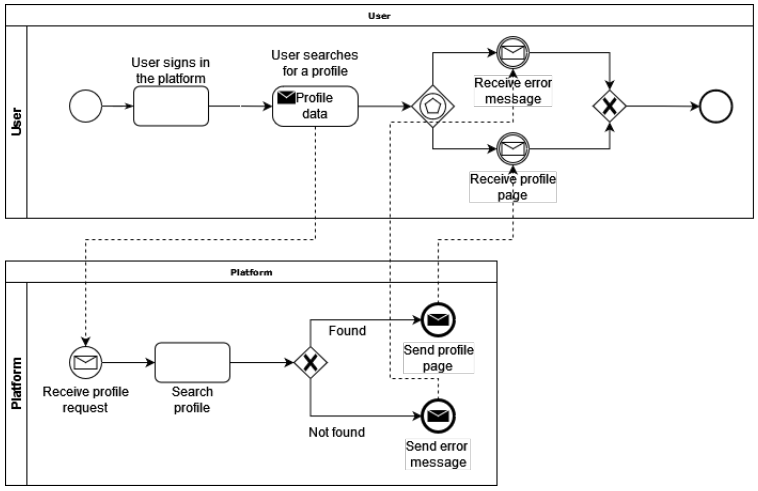

Figure 3: Profile visualization BPMN

2.2.2 Student functions

- **Student subscribes to a tournament:** Let a student search, according to some parameters (most used programming languages, creation date, number of partecipant, etc...) and subscribe to a tournament, after loggin in the platform.

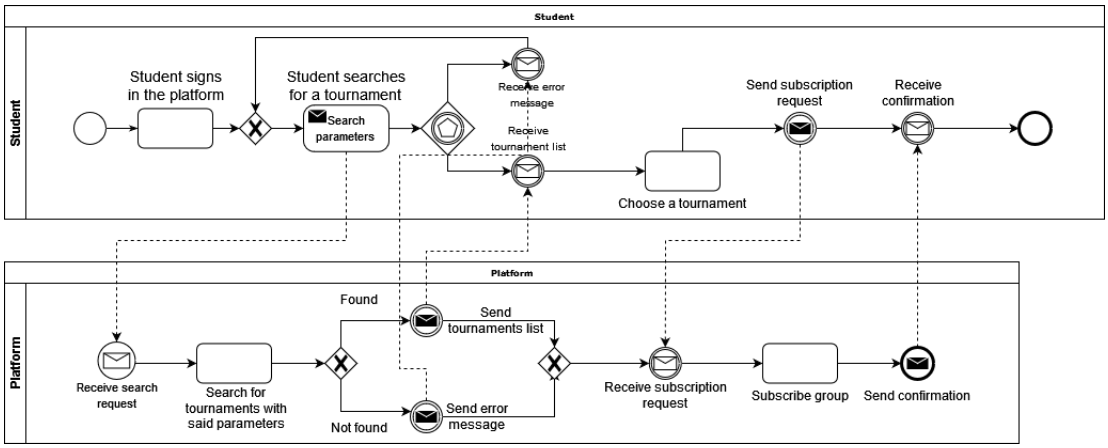

Figure 4: Student's tournament subscription BPMN

- **Student join a battle:** Let a student subscribed to a tournament search a battle in it according to some parameters (programming language requested, expiration date, difficulty, etc...) and join it alone or in a group.

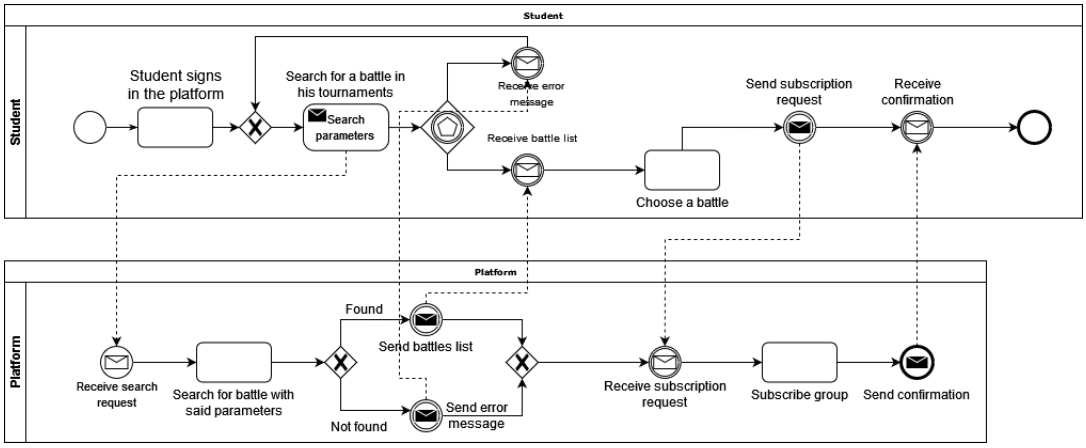

Figure 5: Student's battle joining BPMN

- **Student gets notified of new events:** Notify the student about new tournaments created, if signed to the platform, and new upcoming battles that take place in tournaments in which he has joined.

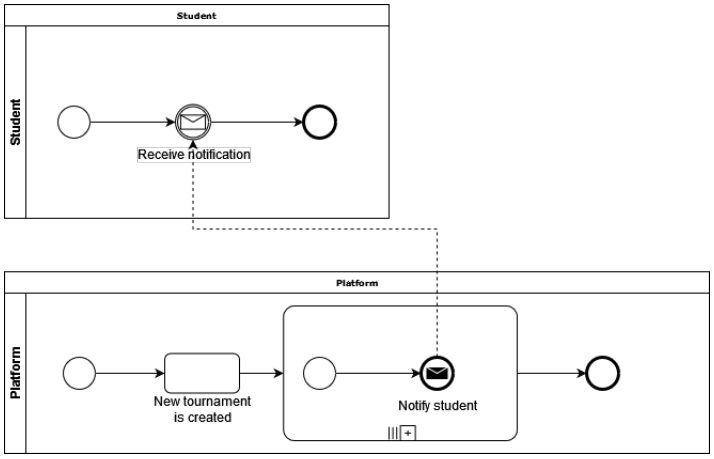

Figure 6: Student notification BPMN

- **Student's solutions get evaluated:** Let a student, who connects to his forked repository on GitHub, upload his solution and evaluate it according to some parameters.

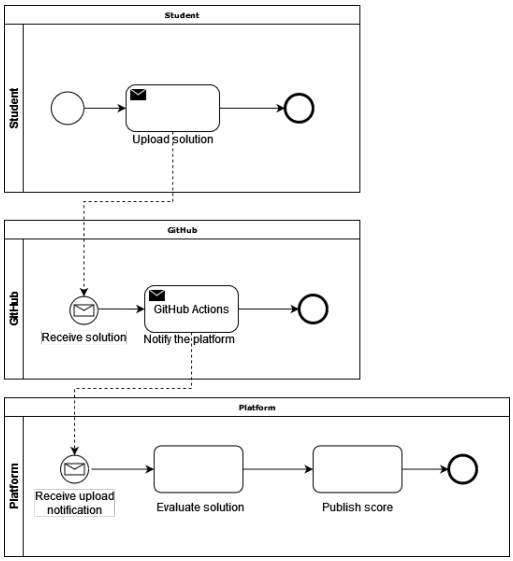

Figure 7: Student's solutions evaluation BPMN

- **Student form a group for a battle:** Let a student form a group with other students in order to face a battle, through invite messages.

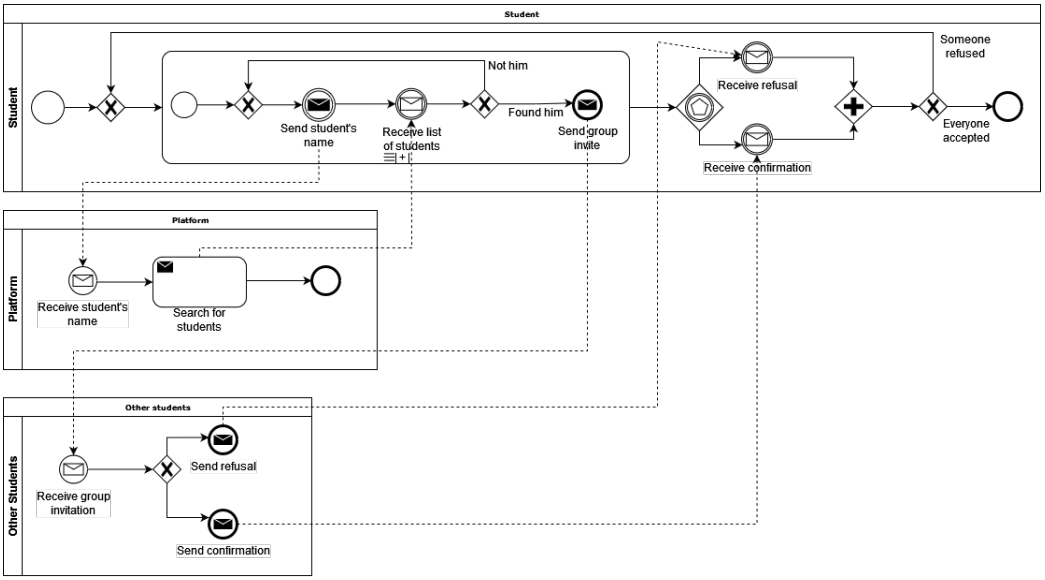

Figure 8: Formation of a group BPMN

2.2.3 Educator functions

- **Educator create a tournament:** Let an educator create a new tournament and set its parameters.

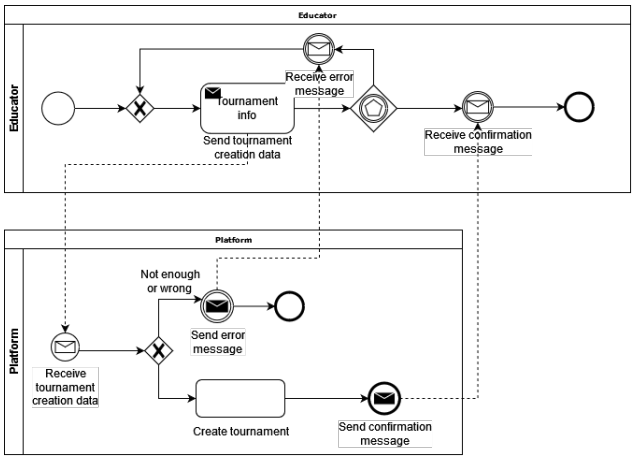

Figure 9: Tournament's creation BPMN

- **Educator grant access to a tournament:** Give an educator the possibility to grant the access to one of his tournaments to a colleague.

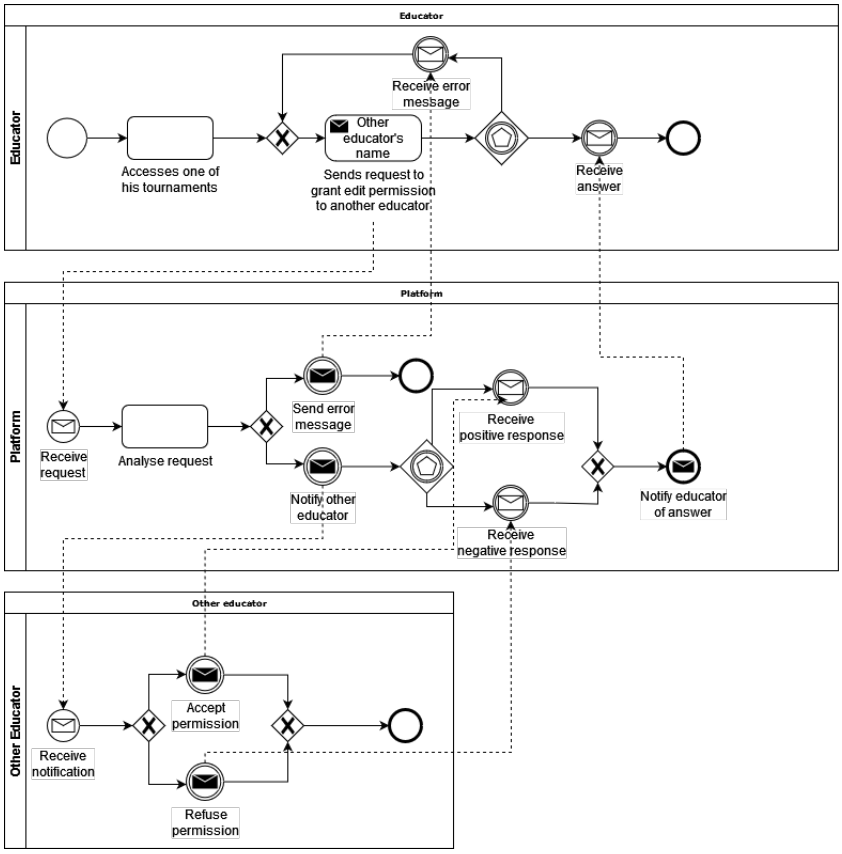

Figure 10: Grant access to tournament BPMN

- **Educator close a tournament:** Let an educator close one of his tournaments whenever he wants.

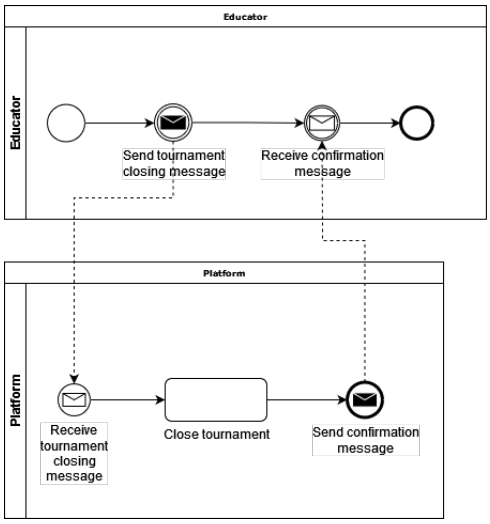

Figure 11: Tournament's closing BPMN

- **Educator creates a battle:** Grant an educator the possibility of creating a battle in tournaments where he has permission to.

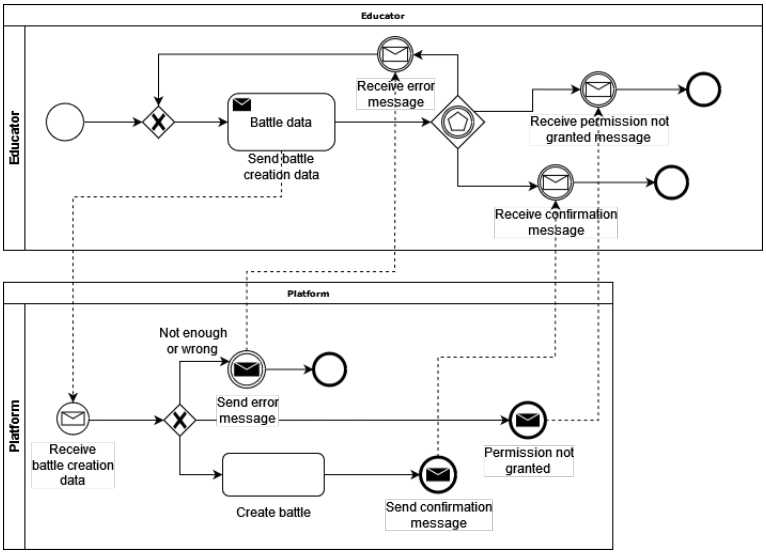

Figure 12: Create battle BPMN

- **Educator notified of battle's end:** Educator gets notified about the end of a battle in one of his tournaments, give him the possibility to manually evaluate the students' solutions.

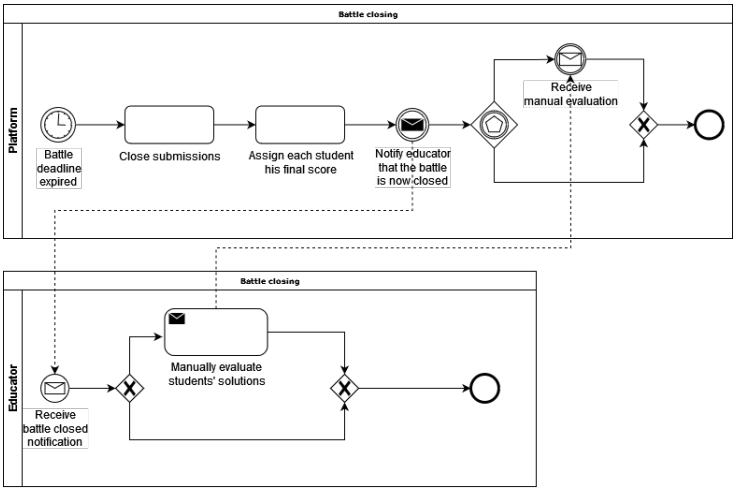

Figure 13: Ending of a battle BPMN

2.3 User characteristics

CKB platform has 2 different users:

- **Educators:** They are educators teaching in a school environment. They must be qualified to teach. They need to have medium/high programming skills, or at least enough to mostly understand a language on which they want to create a tournament and for which they have to write tests code. Finally they must be provided of an istitutional email in order to properly register to the platform.
- **Students:** They are students of a school that want to improve their development skills. They can be of any school and it's not relevant if they subscribe to a tournament created by an educator not part of their school. They must have some knowledge about GitHub platform and how to create GitHub Actions, fundamental for the correct functioning of the CKB platform.

2.4 Assumptions, dependencies and constraints

D1: Students and educators have access to internet while using the platform

D2: Students and educators have their own IT device to connect to the application

D3: Students and educators have to be subscribed to the platform in order to use its features

D4: Students know how to use GitHub actions

D5: Students know how to fork a repository in GitHub

D6: A student can join a battle only if subscribed to the tournament in which that battle take place

D7: GitHub platform offers reliable services through its API allowing to CKB platform to always get notified when new code is uploaded by students.

D8: Educator know how to create new badges, and new rules to obtain them, for the tournaments

D9: Time information about registration and submission deadlines for tournaments and battles are always correct.

D10: Code written by students can not make the platform crash while testing it.

D11: Educators upload ,when creating a battle, some correct, meaningful and faultproof test cases and automation scripts.

D12: Students score is always correctly calculated and meaningful.

D13: Educators access always access a tournament either in view-only mode, if not invited by the tournament's creator, or in modify-enabled mode if the tournament's creator has granted him the access. **(??? Verificare se potrebbe essere un goal piuttosto che una assumption ???)**

D14: Students can invite any other student to form a group, and in case the invited student is not registered to the tournament in which the Student that is inviting him has already registered, he get automatically registered by the platform.

//TODO: Complete the domain assumptions.

3 Specific requirements

3.1 External interface requirements

3.1.1 User Interfaces

The system should interface with the users (both Educator and Students) through their devices, such as: laptops, PC desktops and smartphones, whom must be connected to the internet. Every user, in order to access the platform, has to connect to an existing domain (like "www.codekatabattle.com").

3.1.2 Software Interfaces

The system has to use different software interfaces in order to properly function:

- **DB Interfaces:** The system has to interact with a DB, used to store all information necessary for the system to function.
- **GitHub Interfaces:** The system has to interact with the GitHub platform, in order to receive the students' solutions. This is done through "GitHub Actions" APIs made available by GitHub to automate tasks.
- **Testing Interfaces:** The system needs to test the students' solutions. In order to do this a Testing API must be used .

3.1.3 Hardware Interfaces

The system has to interface with the hardware components containing the DB.

3.1.4 Communication Interfaces

All communications from and to the CKB platform have to use the HTTPS protocol.

3.2 Functional requirements

3.2.1 Requirements

R.1 The CKB platform should allow an unregistered Student to create a new account.

R.2 The CKB platform should allow an unregistered Educator to create a new account.

R.3 The CKB platform must allow access to it's pages only if the used credentials are correct.

R.4 The CKB platform must not allow a Student to register more than once in the system.

R.5 The CKB platform must not allow an Educator to register more than once in the system.

R.6 Educators can access the platform's services only if they are registered to it.

R.7 Students can access the platform's services only if they are registered to it.

R.8 The CKB platform should not allow Students to create tournaments and/or battles.

R.9 The CKB platform should allow Educators to create battles within a tournament only to the tournament creator and to any other Educator that has been granted permission to do so by the tournament creator.

R.10 The CKB platform must allow Educators to personalize the tournaments they create.

R.11 The CKB platform must allow Edcuators to personalize the battles they create.

R.12 The CKB platform must allow Educators to define new obtainable badges for each tournament they create.

R.13 The CKB platform must allow Educators to manually evaluate the solutions uploaded by the students for the battles that the Educators created.

R.14 The CKB platform must allow Educators to delete or update badges before finalizing a tournament's creation.

R.15 The CKB platform must allow Educators to define rules to obtain badges in tournaments created by them.

R.16 The CKB platform must ensure that badges' characteristics respect guidelines regarding their name, icon format and rules to obtain them.

R.17 The CKB platform must allow Educators to create new tournaments.

R.18 The CKB platform must ensure that tournaments' characteristics respect guidelines regarding their name, deadline, access method, programming language.

R.19 The CKB platform should allow to educators to close tournaments they have created.

R.20 The CKB platform must ensure that when a tournament is closed, educators cannot create new battles within it.

R.21 The CKB platform must ensure that if a group uploads a solution to a battle after the submission's deadline, that solution will not be considered in the score computation, by preventing its upload.

R.22 The CKB platform must ensure that the score given to a group in a tournament is coherent with scores given to the same group in the battles they have partecipated in.

R.23 The CKB platform must ensure fair competition between group scores. In the tournament's evaluation, the final group score should be the average score of all the battles in the tournament for each group. Any battle with no solution submitted will count as 0 points.

R.24 The CKB platform must allow Students to subscribe to a tournament.

R.25 The CKB platform must allow Students to subscribe to a battle of a tournament within the registration deadline.

R.26 The CKB platform must allow Students to submit solutions to a tournament's battle within the battle's deadline.

R.27 The CKB platform must allow Students to send and receive group invitations to and from other students in order to form groups.

R.28 The CKB platform should allow Students to join a battle only if the group composition rules for that battle are complied with.

R.29 The CKB platform must ensure that solutions uploaded by a Student for a battle are evaluated.

R.30 The CKB platform must allow a Student to upload solutions to a battle relying on the external GitHub service.

R.31 The CKB platform must ensure that only the latest solution uploaded by a Student for a battle he is subscribed to will be taken into consideration for the final score.

R.32 The CKB platform must allow groups partecipating in a battle to change their solution, if the battle's deadline submission hasn't expired yet.

R.33 The CKB platform must ensure that the score given to a group's solution is updated as soon as possible.

R.34 The CKB platform must allow an Educator to modify the score for a Student's solution.

R.35 The CKB platform must ensure that when a new tournament is created, all students subscribed to the platform will receive a notification.

R.36 The CKB platform must ensure that when a new battle is created in a tournament, all students subscribed to the tournament will receive a notification.

R.37 The CKB platform must allow Students to visualize the score they obtained in a battle they partecipated in.

R.38 The CKB platform must allow Students to visualize the score they obtained in a tournament they partecipated in.

R.39 The CKB platform must allow Students to visualize the badges they obtained.

R.40 The CKB platform must ensures that battles' characteristics respect guidelines regarding their name, deadlines, programming language, number of member per group.

3.2.2 Use case diagrams

- **Student**

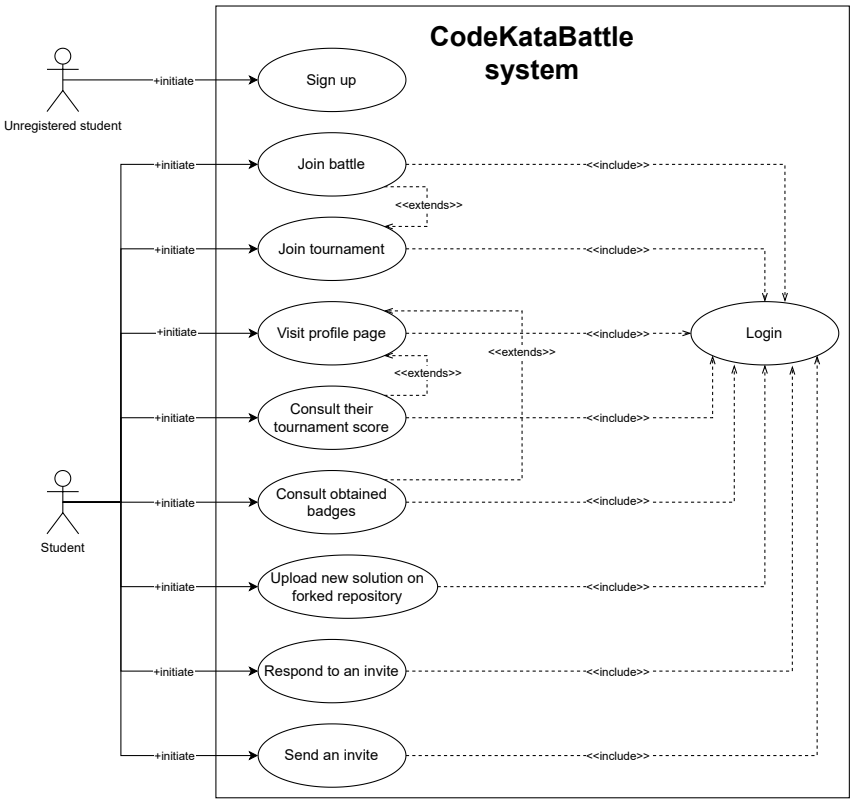

Figure 14: Student use case diagram

N.B: The repository citated in this diagram, on which the student upload their solution, it's not part of the platform. The system, in fact, relies on a third party service (GitHub) to handle the repository. This use case was specified in order to better clarify the tight interaction between the system and the student trought the GitHub service.

- **Educator**

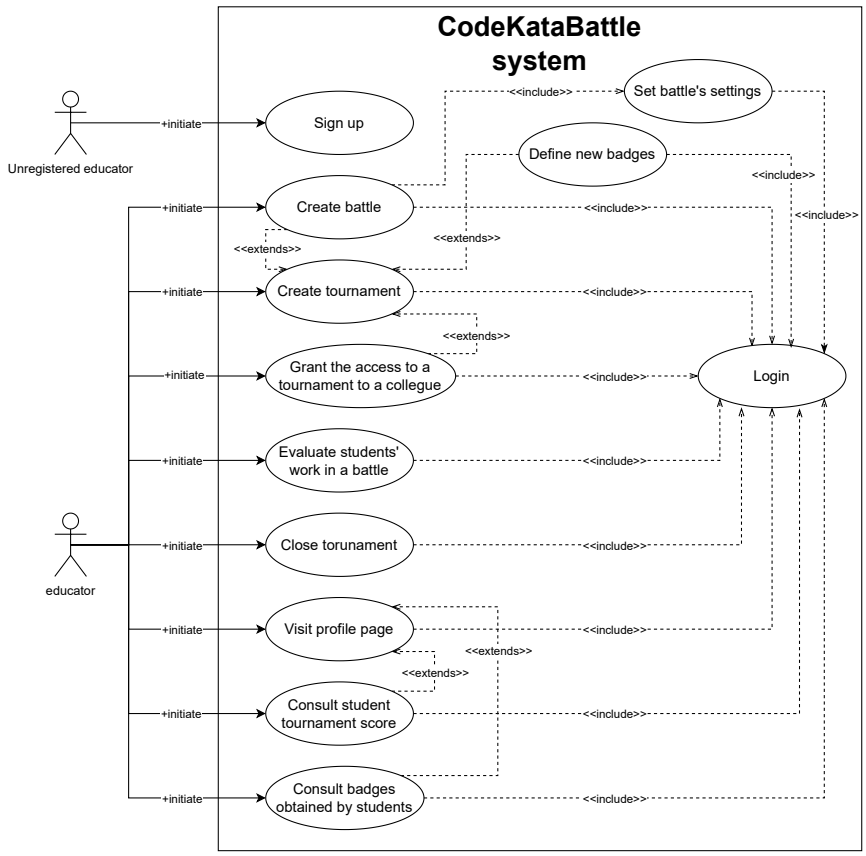

Figure 15: Educator use case diagram

- **GitHub**

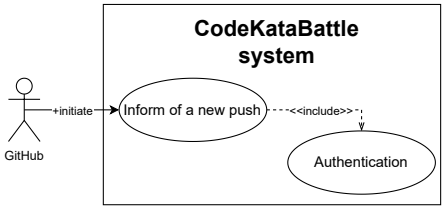

Figure 16: GitHub use case diagram

3.2.3 Use cases w/ sequence diagrams

1. Student sign up to the platform

Name	Student sign up
ID	UC.1
Actors	Unregistered student
Entry condition	Student want to register to the platform
Flow of events	1. Student opens CKB platform 2. Student presses the sign-up button 3. Student fills the form with all the required informations (name, surname, username attended school, email, password, ...), accepts the "Terms & Conditions". 4. Student clicks on a button to confirm. 5. CKB platform validates the personal information inserted by the student. 6. CBK platform display a confirmation message. 7. CKB platform send an email notification to the student regarding the registration outcome
Exit condition	Student's account is created, and its data are saved into the system

Exceptions	4.1 Email already used to register another account, or inexisting. 4.2 Inserted attended school non existings 4.3 Username contains forbidden characters 4.4 Password don't respect security standards → Unregistered student gets notified of the registration failure throught an error message and flow restart from point 3.

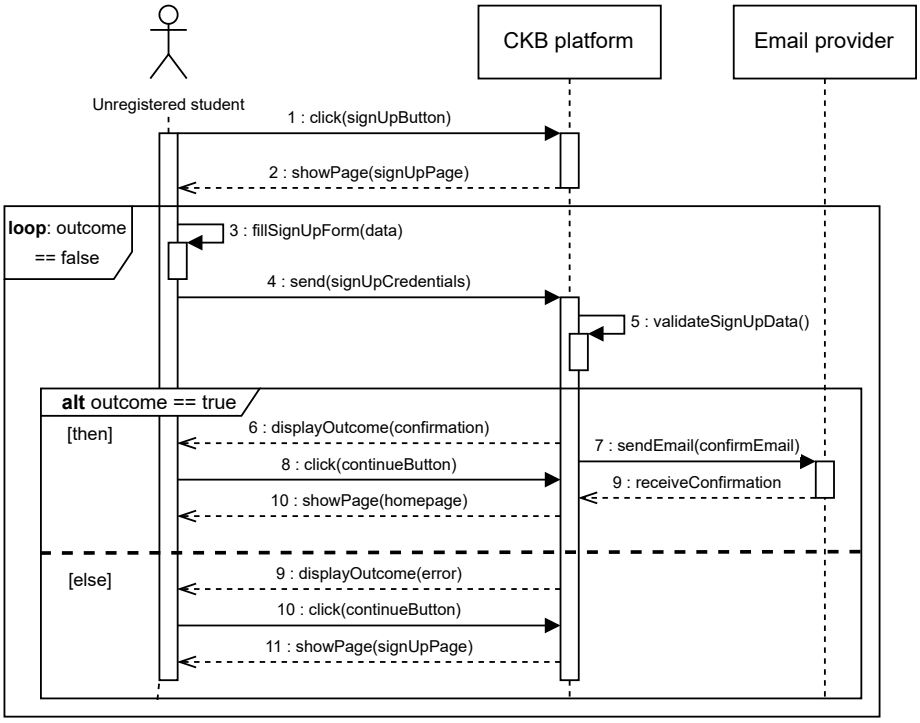

Figure 17: Student sign up sequence diagram

2. Educator sign up to the platform

Name	Educator sign up
ID	UC.2
Actors	Unregistered educator
Entry condition	Educator want to register to the platform
Flow of events	1. Educator opens CKB platform 2. Educator presses the sign-up button 3. Educator fills the form with all the required informations (name, surname, username school in which teaches, istitutional email, password, ...), accepts the "Terms & Conditions". 4. Educator clicks on a button to confirm. 5. CKB platform validates the personal information inserted by the student. 6. CBK platform display a confirmation message. 7. CKB platform send an email notification to the educator regarding the registration outcome
Exit condition	Student's account is created, and its data are saved into the system
Exceptions	4.1 Email already used to register another account, or inexisting. 4.2 Inserted school's details non correct 4.3 Username contains forbidden characters 4.4 Password don't respect security standards → Unregistered educator gets notified of the registration failure throught an error message and flow restart from point 3.

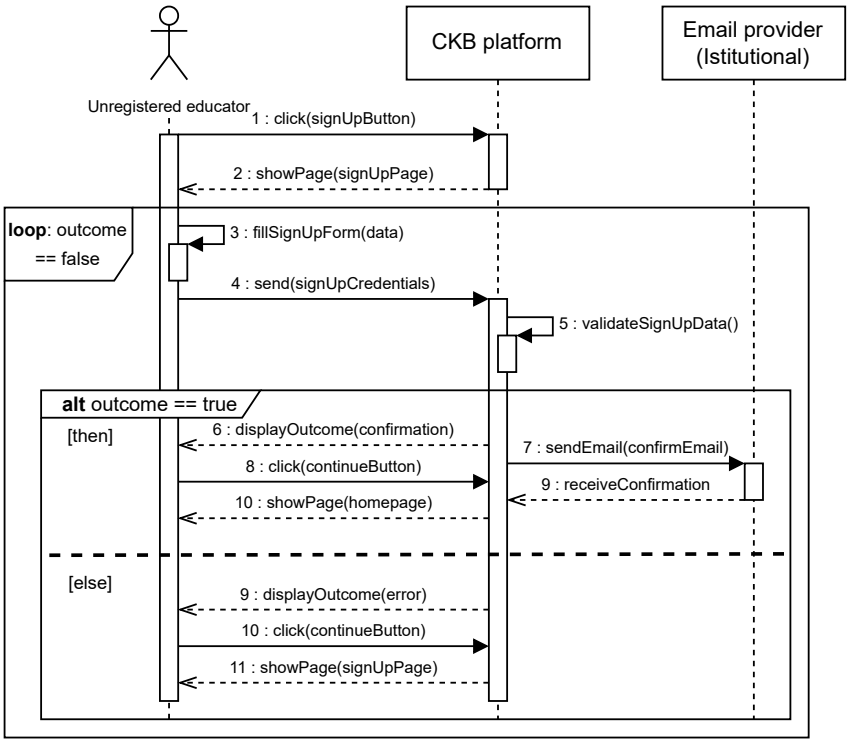

Figure 18: Educator sign up sequence diagram

3. Educator creates a new tournament

Name	Tournament creation
ID	UC.3
Actors	Educator, Student
Entry condition	Educator has logged in the platform and want to create a new tournament

Flow of events	1. Educator clicks on a "Create tournament" button 2. Educator fills a form of the page that has appeared with tournament details, such as: its deadline (eventually), allowed programming languages, name, access method and related information about how to use it, etc. 3. <<*eventually*>> Defines new badges for the tournament (see UC.4) 4. Educator clicks on a "Confirmation" button. 5. CKB platform checks the validity of the informations inserted by the Educator 6. CKB platform displays a message that confirms that the tournament has been created successfully
Exit condition	Tournament is created by saving it's data into the system and the platform sends a notification to all the Students registered to the platform itself. Educator is led back to the riepilogative page of the tournaments created by him.

Exceptions	5.1 Tournament's name contain forbidden characters. 5.2 Deadline inserted for the tournament is invalid. 5.3 Tournament's access method has been configured wrongly. 5.4 A programming language inserted is not recognized by the platform. → Educator gets notified about the failure of the creation of the tournament throught an error message displayed by the platform. The flow restart from point 2.

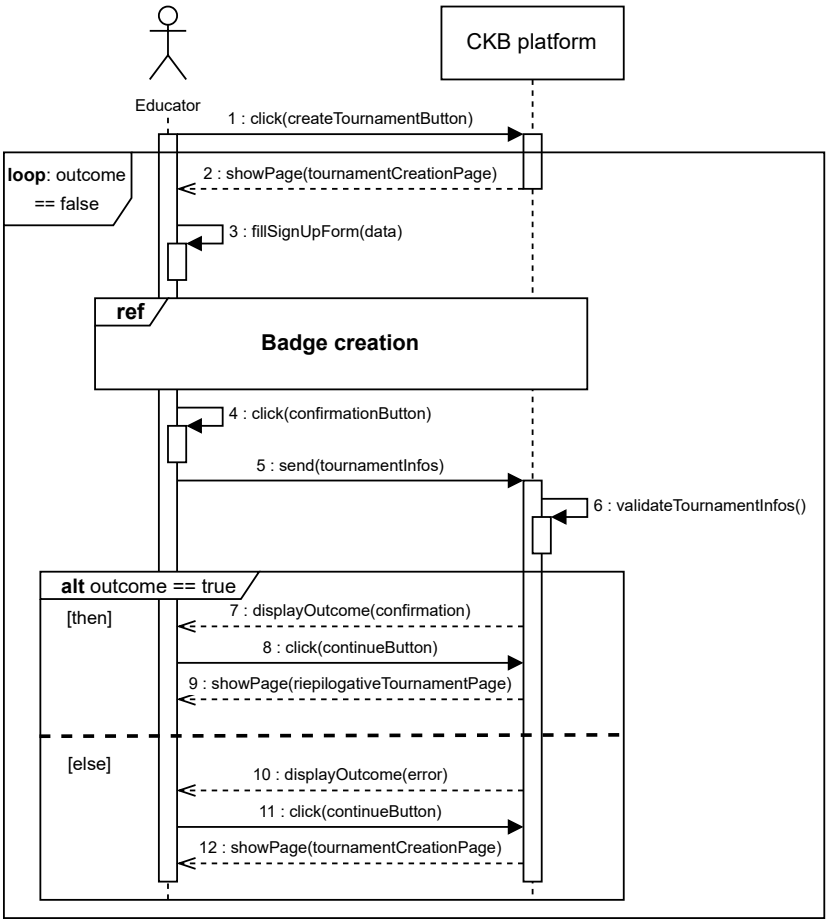

Figure 19: Educator create a new tournament sequence diagram

4. Educator creates new badges

Name	Badges creation
ID	UC.4
Actors	Educator
Entry condition	Educator has logged in the platform, he's creating a new tournament and want to define new badges for that tournament

Flow of events	1. Educator clicks on the "Create badge" button on the tournament creation page. 2. On the new page that appeared, the Educator defines badge's characteristics, such as: its name, value, icon, rules to obtain it, ... 3. Educator clicks on a button in order to confirm its choices. 4. CKB platform checks if all the badge informations are well defined 5. CKB platform displays a message that confirm the successfull creation of a new badge, and that the badge has been added to the tournament.
Exit condition	Badge is created, its data are saved and associated to the tournament which is being created by the Educator and Educator is led back to the create tournament page.
Exceptions	4.1 Badge's name contain forbidden characters 4.2 The image uploaded as icon image it's too big or have an unsupported format. 4.3 Badge's rules are not well defined. 4.4 Some badge's features has been left empty → Educator gets notified about the failure of the badge creation process throught an error message displayed by the platform. Flow restart from point 2.

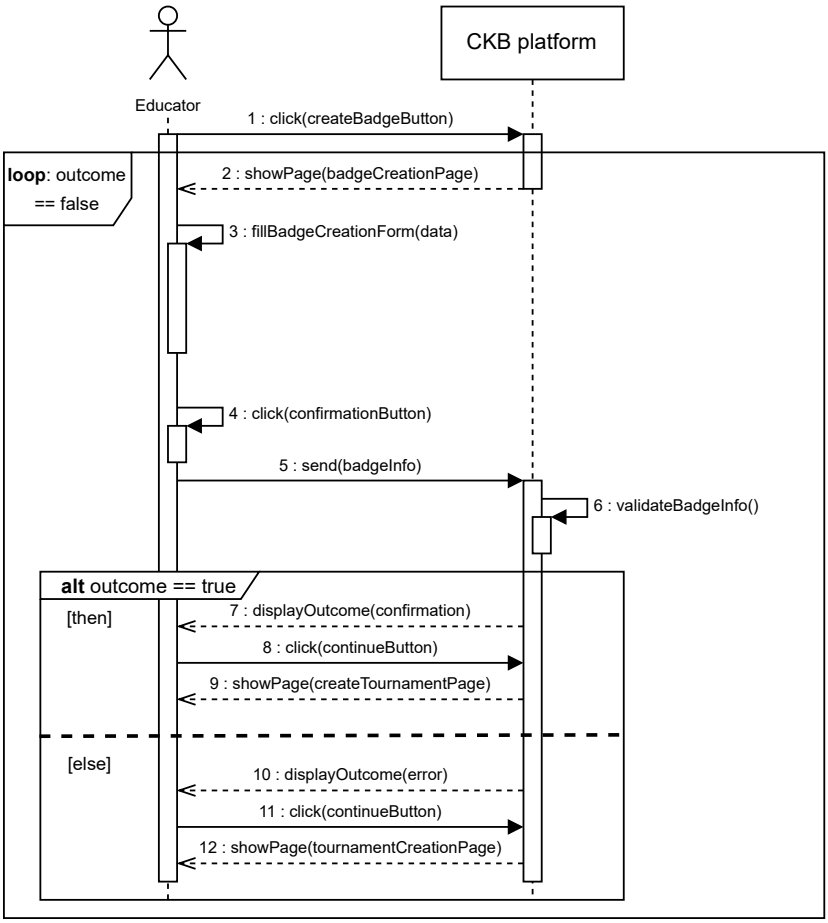

Figure 20: Educator creates new badges for a tournament, sequence diagram

5. Educator delete a badge

Name	Deletion & update of a badge
ID	UC.5
Actors	Educator
Entry condition	Educator has logged in the platform, has started the creation of a new tournament in which has already created at least 1 badge and want to delete and/or update one of them

Flow of events	1. Educator clicks on a update button related to the badge and present on the tournament creation page. 2. CKB platform show a modifiable form, filled with the badge's data which the Educator want to modify. 3. Educator proceeds to modify the badge's data 4. CKB platform tries to validate the new data 5. Update is confirmed with the display of a message by the CKB platform. 6. Educator clicks on a delete button related to the badge and present on the tournament creation page. 7. CKB platform show a message requiring confirmation for the Deletion 8. Educator presses a confirmation button 9. CKB platform delete the badge and bring the Educator to the previous page.
Exit condition	Deleted badges are removed from the tournament, and their data are removed from the system. The modifications of the updated ones are committed to the system. The Educator is led back to the tournament creation page.

Exceptions	4.1 Badge's updated name contain forbidden characters 4.2 The new image uploaded as icon, it's too big or have an unsupported format. 4.3 New Badge's rules are not well defined. 4.4 Some badge's features has been left empty → Badge it's not updated, Educator it's led to the tournament creation page (point 1.) 6.1 Educator presses the cancel button → Delete procedure is stopped, and the Educator it's led back to the tournament creation page (point 6.).

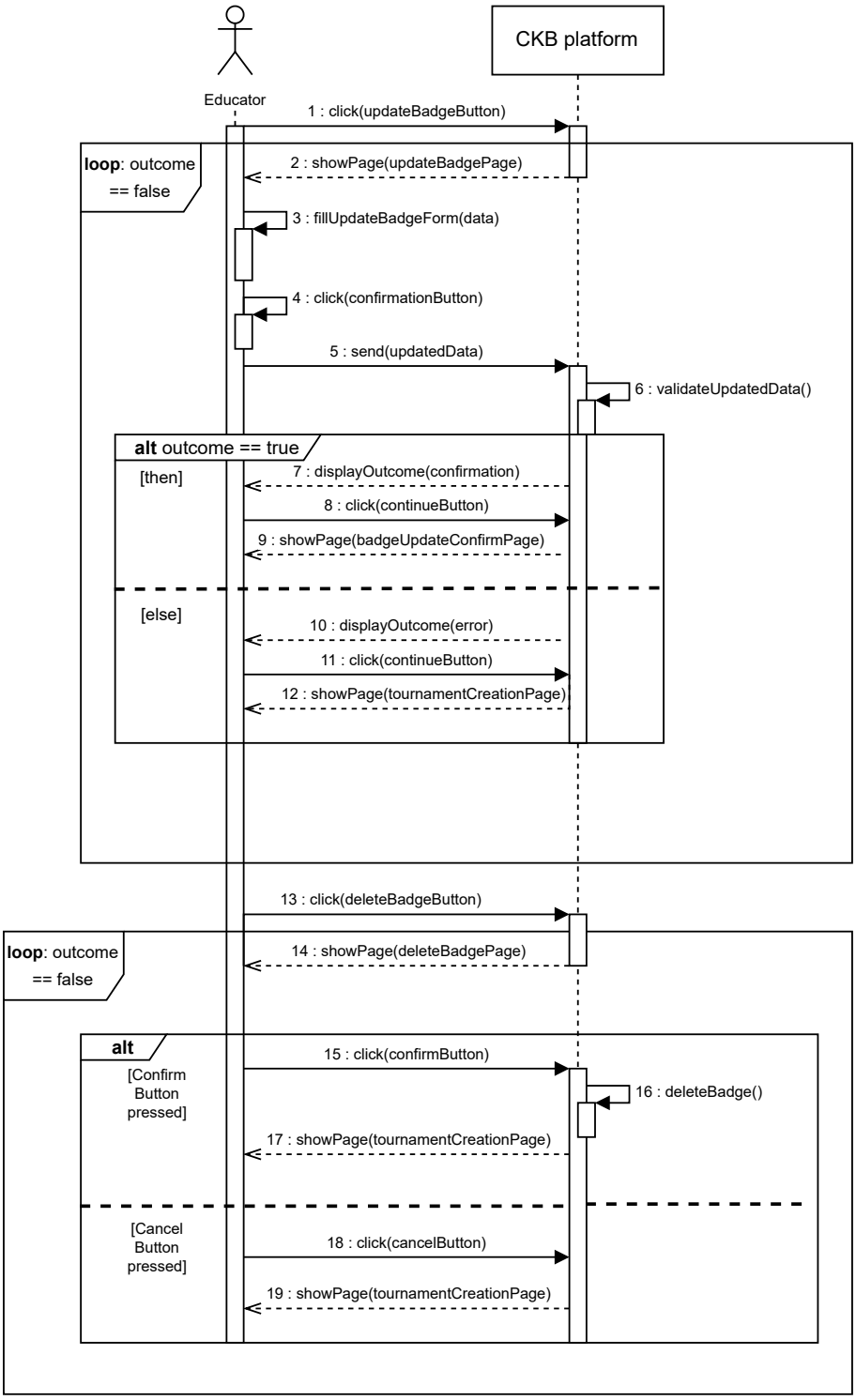

Figure 21: Educator delete and/or update badge(s) of a tournament, sequence diagram

6. Educator creates a new battle

Name	Battle creation
ID	UC.6
Actors	Educator
Entry condition	Educator has logged in the platform, has created at least 1 tournament and want to add a battle in a specific tournament
Flow of events	1. Educator open the page related to a specific tournament which has created. 2. Educator clicks on a button used to create a new battle. 3. Educator fills the form shown by the platform with informations related to the battle that he want to create (name, programming languages allowed, maximum and minimum number of students for each partecipating group, registration and submission deadlines, ...) 4. Educator clicks on a button to confirm. 5. CKB platform validates the battle characteristics 6. CKB platform display a confirmation message of the successfull creation of the battle.
Exit condition	New battle, with characteristics specified by the Educator, is created within the tournament chosen by the Educator itself. All the students subscribed to the tournament in which the battle has been created get notified. Educator it's then led to the riepilogative page of the battles within the tournament.

Exceptions	3.1 Battle's name contain forbidden characters 3.2 A programming language inserted it's not included in the tournament allowed programming languages. 3.3 The specified number of minimum or maximum students for each group it's beyond upper and/or lower limits imposed by the platform 3.4 Registration deadline inserted for the battle it's not valid or bigger than the submission one. 3.5 Submission deadline inserted for the battle it's not valid or smaller than the registration one. → Educator gets notified about battle creation failure by the platform throught an error message. The flow restart from point 3.

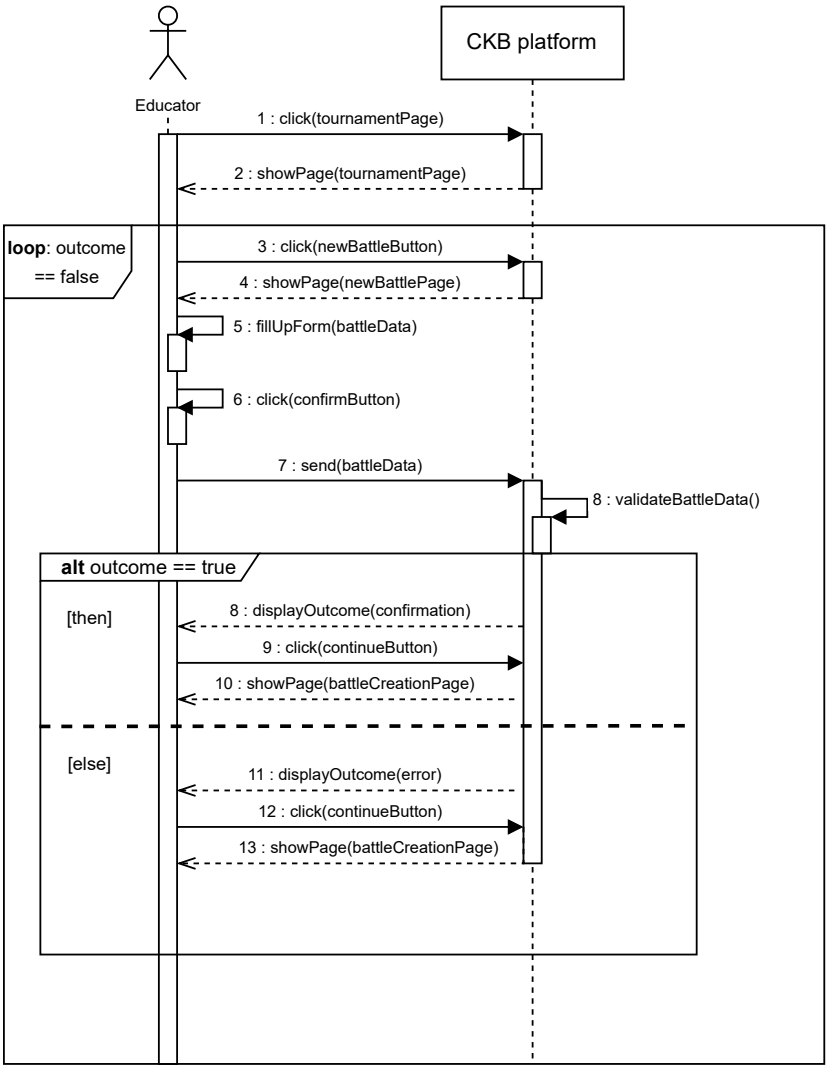

Figure 22: Educator creates a new battle, sequence diagram

7. Educator close a tournament

Name	Tournament closure
ID	UC.7
Actors	Educator

Entry condition	Educator has logged in the platform, has created at least 1 tournament and want to close one of them in order to end the competition
Flow of events	1. Educator access the page of the tournament that want to close. 2. Educator checks whether there are battles that aren't already closed. 3. Educator clicks on a "close" button related to the tournament and clicks on a confirm button. 4. CKB platform verifies if there are no battles still active within the tournament. 5. CKB platform notifies the Educator throught a message that confirms the deletion of the tournament.
Exit condition	The chosen tournament is closed, its data are kept saved in order to create an history of tournaments. Educator is led back to the riepilogative page of the tournaments created by him.
Exceptions	2.1 There are still active battles within the tournament. → Educator can close the possibility of other educators to create new battles if he's the tournament creator. The flow restart from point 1. 4.1 There are still active battles within the tournament. → CKB platform interrupts the tournament deletion and display an error message to the Educator. Flow restart from point 1.

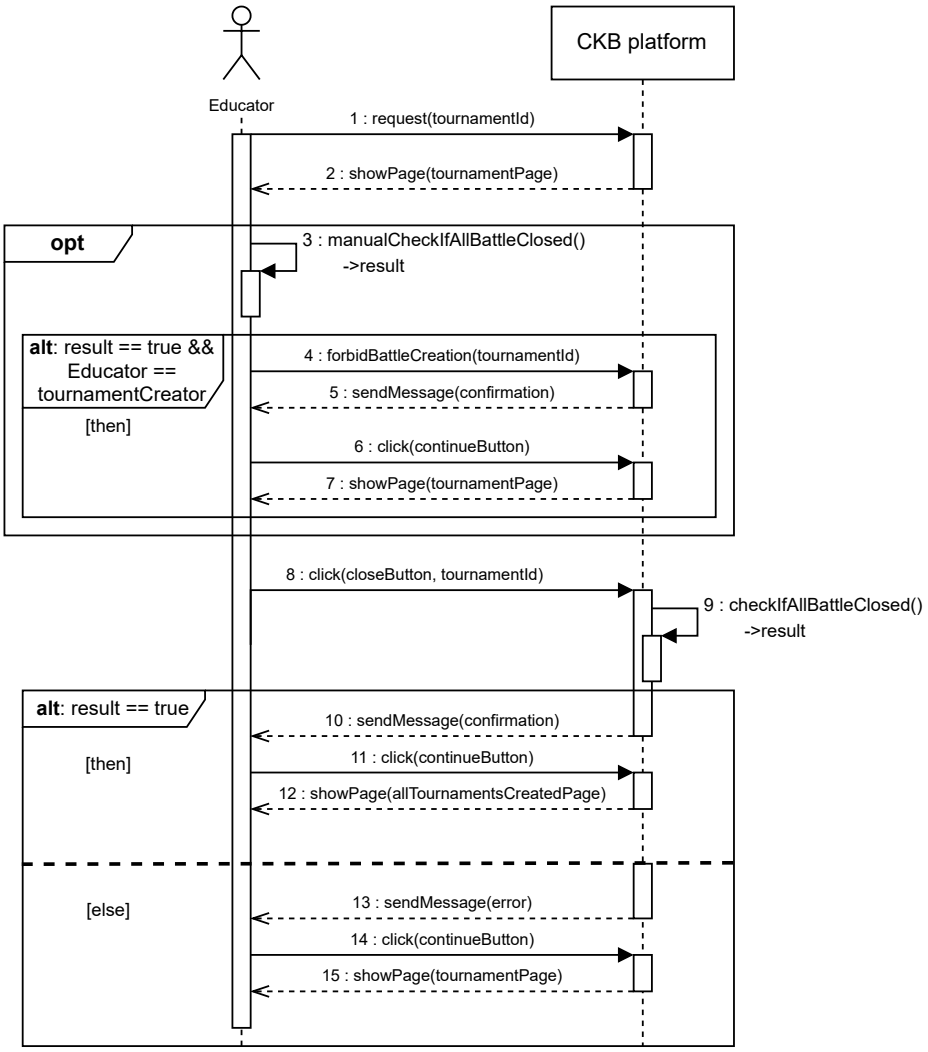

Figure 23: Educator close a tournament, sequence diagram

8. Educator evaluate battle's results

Name	Battle's results evaluation
ID	UC.8
Actors	Educator

Entry condition	Educator has logged in the platform, has created at least 1 battle within a tournament, that has terminated. Educator want to manually evaluate its results
Flow of events	1. Educator accesses the page related to tournament he has created. 2. Educator accesses the page related to a terminated battle within that tournament in which was included manual evaluation. 3. Educator manually evaluates groups' works by decreasing or increasing their scores according to his personal preferences, or the ones accorded with the students. 4. Educator after evaluating all the groups, proceeds to close definitively the battle.
Exit condition	Groups' scores within the battle get updated by the platform, battle is definitively closed and the Educator is led back to the riepilogative page of the tournament to whom the battle belonged.
Exceptions	2.1 In accessed battle was not included at the moment of the creation the possibility to manually evaluate the results. 2.2 Accessed battle isn't terminated. → CKB don't allow the Educator to manually evaluate the battle. Flow restart from point 2.

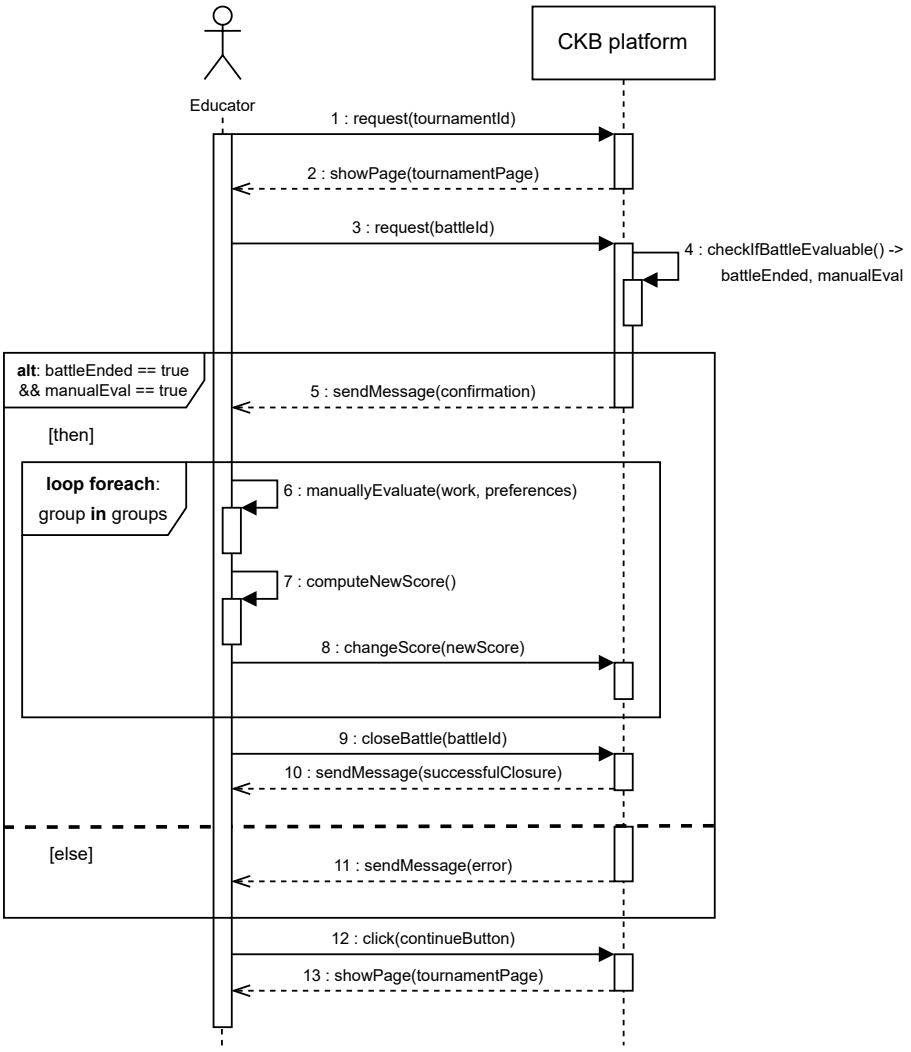

Figure 24: Educator evaluates manually a battle's score, sequence diagram

9. Student forms a group

Name	Group formation
ID	UC.9
Actors	Student

Entry condition	Student has logged in the platform, has subscribed to a tournament and want to form a group in order to later join a battle within that tournament.
Flow of events	1. Student clicks on a button that lead to an appropriate page to form a group between partecipant of a tournament. 2. Student choose the filters to seach one or more students to invite. 3. CKB platform compute all the filters and returns to the Student the available students according to the filters inserted. 4. Student choose the other students to which send the invite from the showed page. 5. Student clicks on a confirm button to send the invitations. 6. CKB platform proceeds to send the invitation to all students specified and send a notification with it in order to notify the receivers. 7. Invited students accept the invitation. 8. CKB platform send notification to the Student.
Exit condition	The group is formed and registered by the CKB platform. The Student is led back to the tournament riepilogative page.

Exceptions	3.1 The set of students returned applying the filters doesn't contain any available student. → CKB platform show an empty list, with a message explaining the absence of students in the list. Flow restart from point 2. 3.2 The set of students returned contains only students already in a group. → CKB platform shows the students returned from the computation but doesn't allow to the Student to select them. Flow restart from point 2. 7.1 One or more invited students don't accept the invitation → CKB platform notifies the Student that sent the invitation of the refusal of the invitation. Flow restart from point 1.

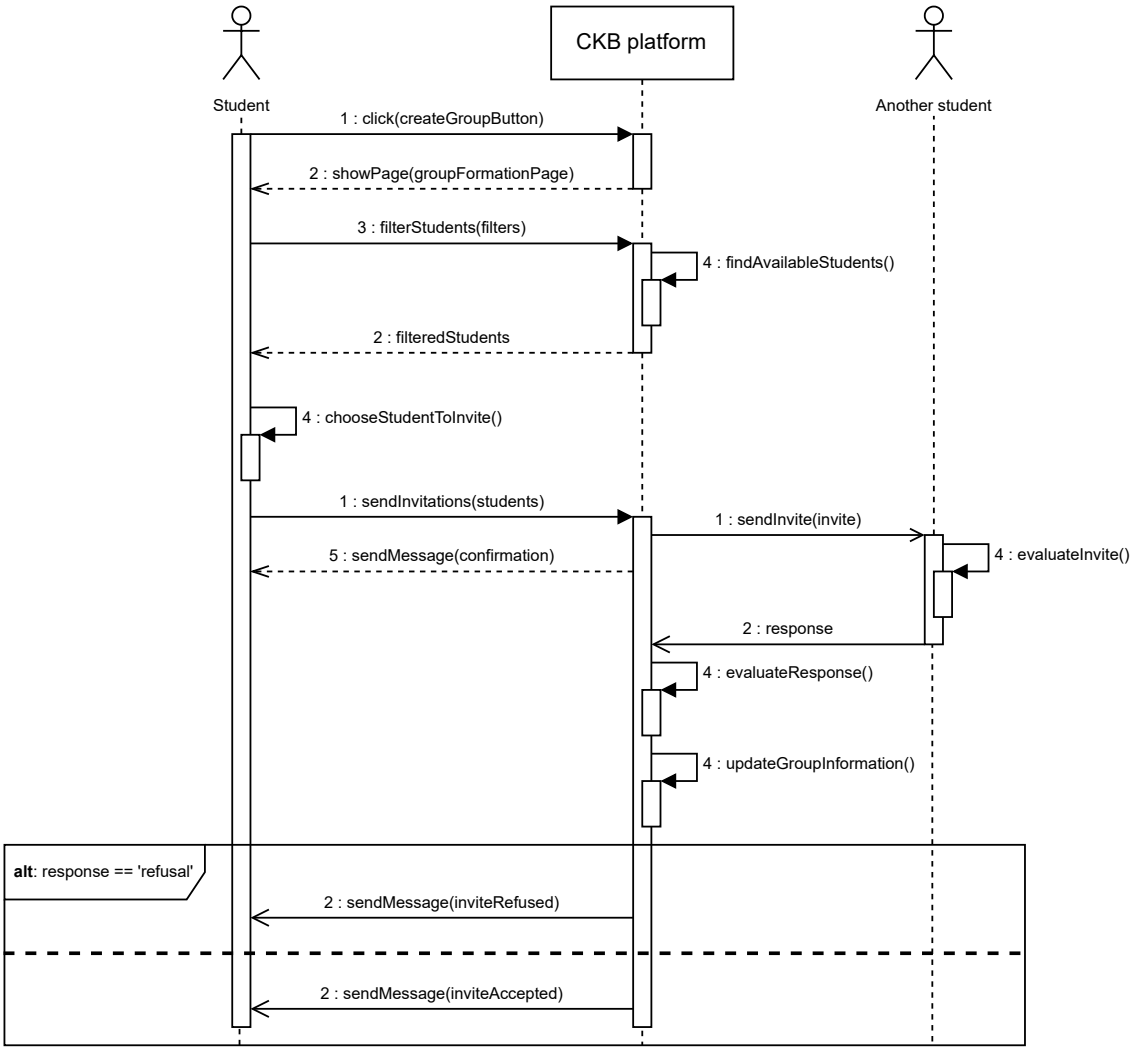

Figure 25: Students form a new group, sequence diagram

10. Student join a battle

Name	Battle joining
ID	UC.10
Actors	Student

Entry condition	Student has logged in the platform, has subscribed to a tournament and want to join a battle within that tournament.
Flow of events	1. Student enter the tournament's page and subsequently the battle's page. 2. Student clicks on a button to join the battle. 3. CKB platform shows a riepilogative page of the battle. 4. Student clicks on a button to subscribe to the battle his group. 5. CKB platform checks whether the group respect all the battle requirements. 5. CKB platform returns a confirmation message to all the group members. 6. CKB platform sends a notification to group members when subscription deadline expires and battle starts.
Exit condition	The Student and his group have joined the battle. CKB platform start saving all the solutions and statistics regarding the group's work. The student is led to the riepilogative page of the battle.
Exceptions	5.1 The group doesn't respect the requirement of the battle (too many or too few members). 5.2 The group contains members not subscribed to the tournament in which the battle is held. → CKB platform shows an error message.

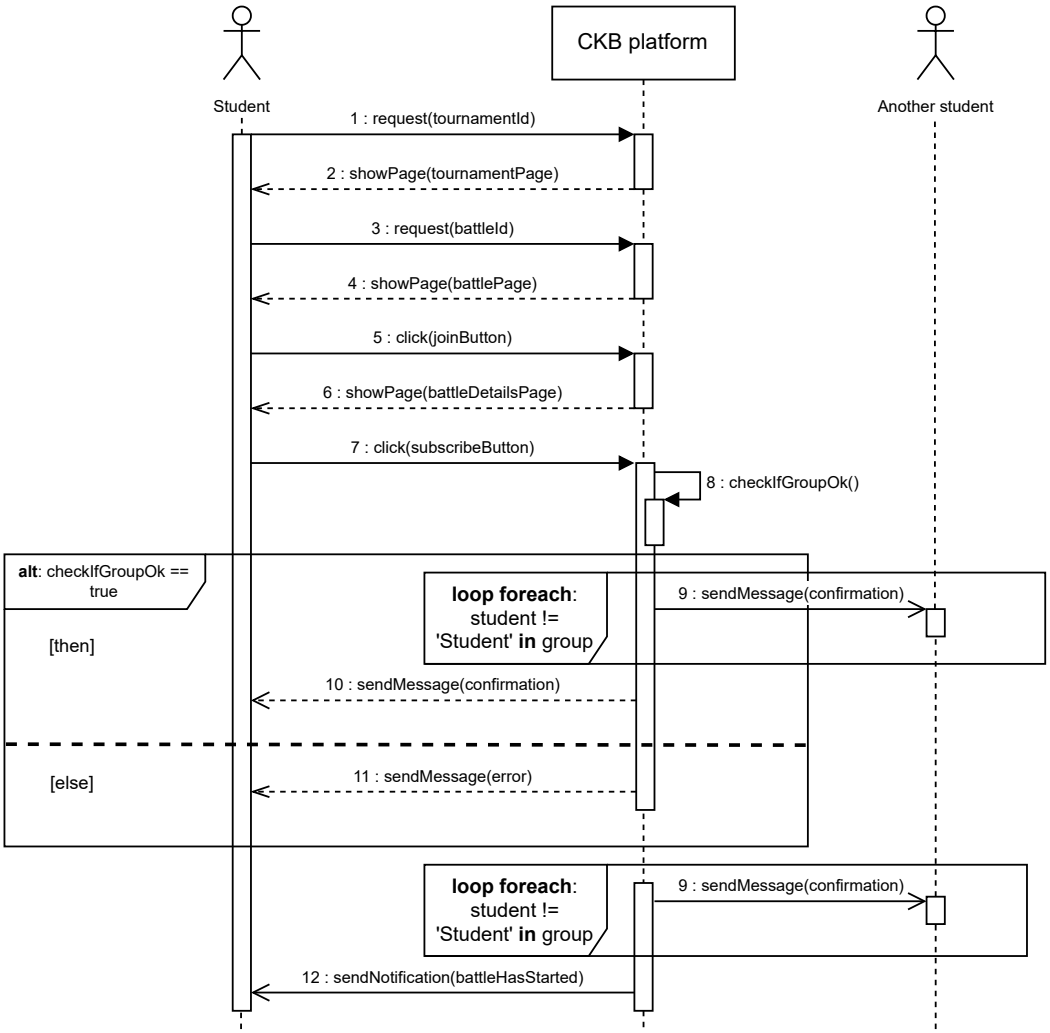

Figure 26: Student join a battle within a tournament, sequence diagram

11. Student upload a new solution

Name	Upload of a solution within deadline
ID	UC.11
Actors	Student, GitHub

Entry condition	Student has a GitHub account, has registered to at least 1 tournament and 1 battle within it, and want to upload a new solution to that battle.
Flow of events	1. Student access his GitHub account 2. Student upload his solution (or the group one) to his GitHub repository, precedently forked. 3. GitHub platform, throught his "GitHub Actions" notifies the CKB platform of the upload of the new solution by the Student. 4. CKB platform receives the new solution uploaded and starts testing it. 5. CKB platform gives a score to the solution based on the tests results. 6. CKB platform updates group score in the battle and tournament. 7. CKB platform notifies the group's members of the successfull testing.
Exit condition	Student's (or his group's) score is updated based on the uploaded solution.
Exceptions	2.1 Generic upload error. → Handled by GitHub, flow restart from point 2. 4.1 Tested code generates major issues, such as infinite loops, huge resource consumption, ... → CKB platform, terminates the testing phase and notifies the Student's group with an error message in the place of the solution score, flow ends.

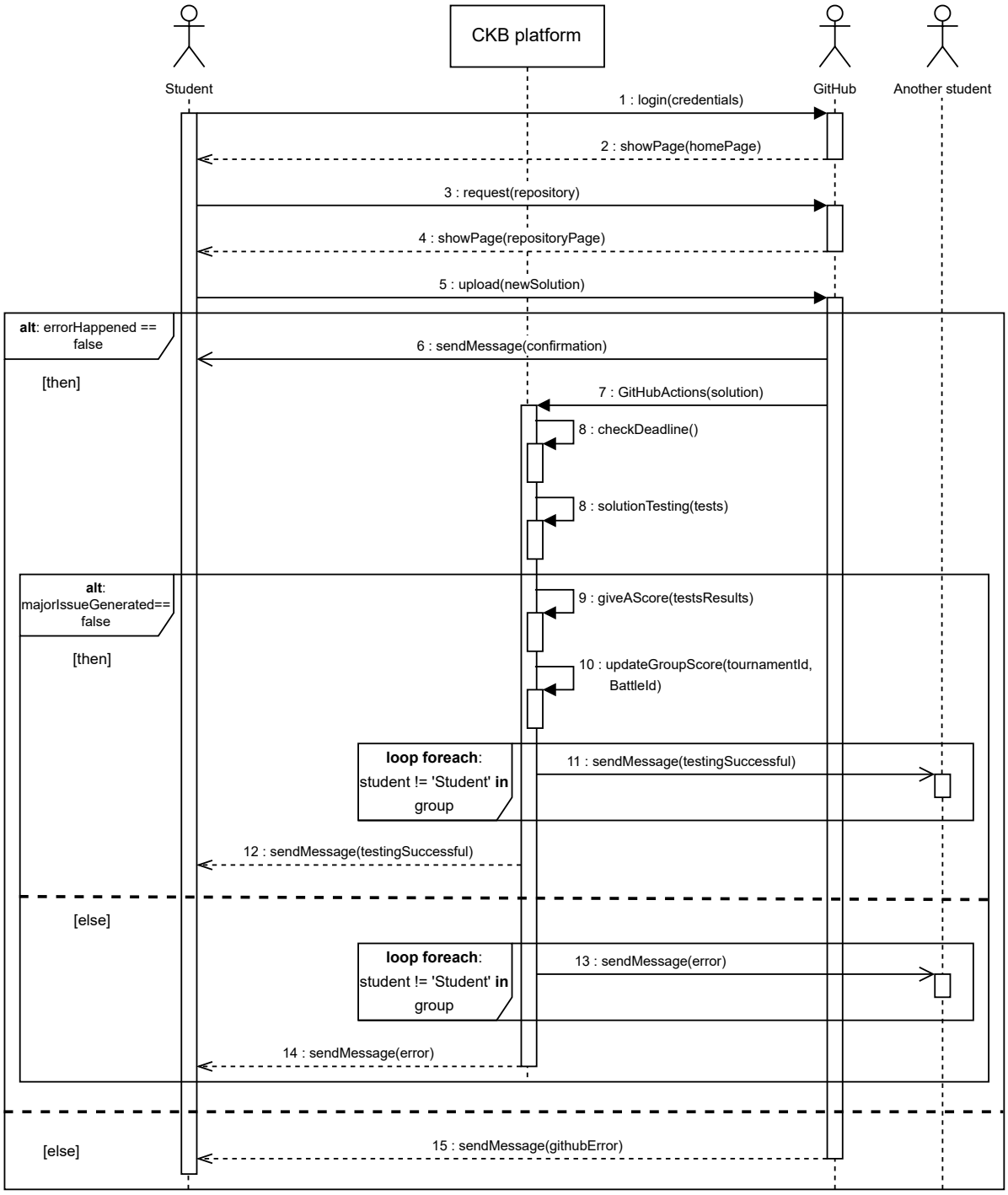

Figure 27: Student upload a new solution of the battle's problem within the deadline, sequence diagram

12. Student uploads a solution after submission deadline

Name	Solution upload exceeding deadline
ID	UC.12
Actors	Student
Entry condition	Student has logged in the platform, has registered to at least 1 tournament and 1 battle within it, and want to upload a new solution to that battle but the submission deadline expired.
Flow of events	1. Student access his GitHub account 2. Student upload his solution (the group one) to the group's GitHub repository, precedently forked. 3. GitHub platform, throught his "GitHub Actions" notifies the CKB platform of the upload of the new solution by the Student. 4. CKB platform receives the new solution uploaded and sees that the submission deadline of the battle for which the solution was uploaded has expired. 5. CKB platform notifies the group's members with an error message.
Exit condition	Student's (or his group's) receives a notification by the CKB platform stating that their last uploaded solution will not be considered on the final score since the submission deadline was expired.
Exceptions	2.1 Generic upload error. → Handled by GitHub, flow restart from point 2.

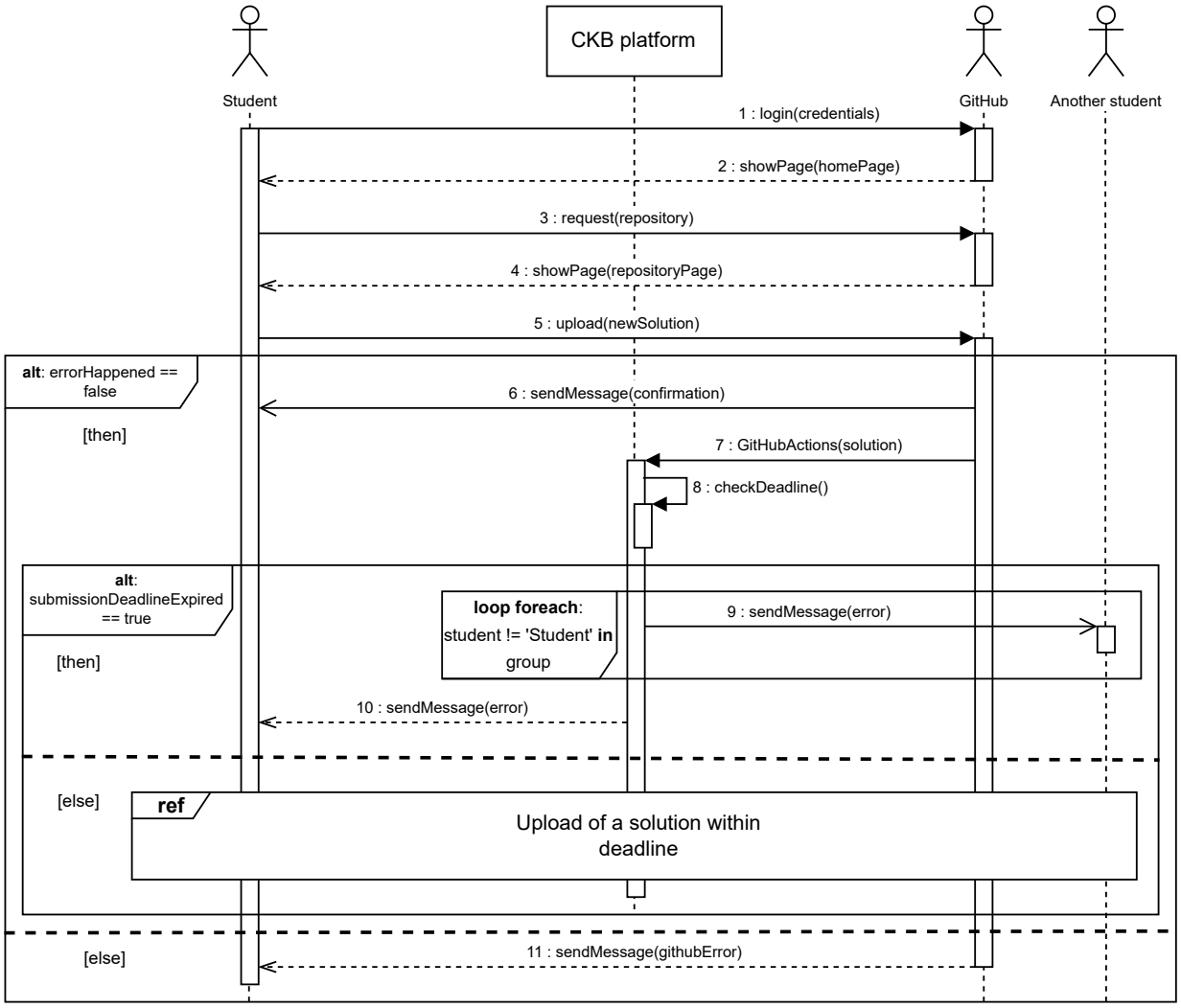

Figure 28: Student upload a new solution of the battle's problem beyond the deadline, sequence diagram

13. Student visualize his tournament's results and badges

Name	Viewing results and badges
ID	UC.13
Actors	Student

Entry condition	Student has logged in the platform and want to see his or others tournament score and badges.
Flow of events	1. Student enters in his profile page or in the profile page of the Student of which want to see his tournament score and badges. 2. Student choose the tournament of which want to see the details from the appropriate section. 3. Student consult his tournament score and badges
Exit condition	Student has consulted his tournament score and badges obtained during it, other than that the Student can see several details about all the tournament that he partecipated to. Flow ends when Student go back to a precedent page.
Exceptions	2.1 Student of the profile page accessed, doesn't have partecipated to any tournament. → CKB platform still allow the Student to access to the profile page of the Student that hasn't partecipated to any tournament, but doesn't show any tournament's score and badges. Flow ends.

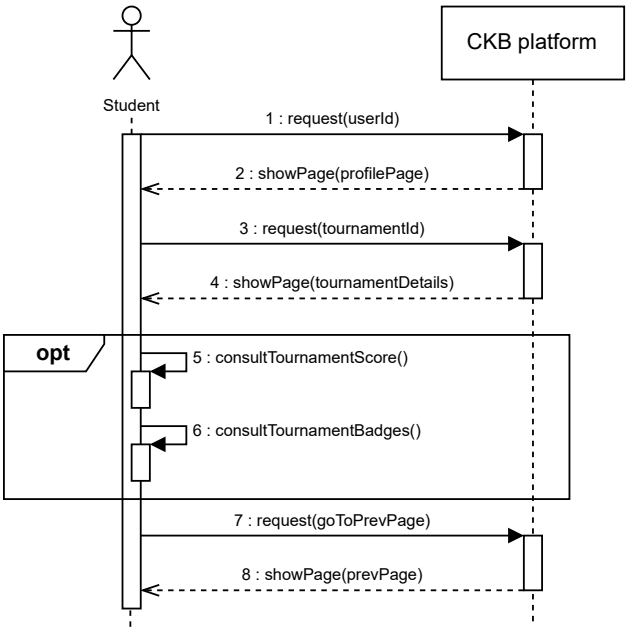

Figure 29: Student visualize his/other score(s) and badge(s), sequence diagram

3.2.4 Traceability matrix

Goal ID	Req. ID	Use case ID
G.1	R.2, R.3, R.5, R.6, R.8, R.10, R.12, R.14, R.15, R.16, R.17, R.18	UC.2, UC.3, UC.4, UC.5
G.2	R.2, R.3, R.5, R.6, R.8, R.9, R.11, R.40	UC.2, UC.6
G.3	R.2, R.3, R.5, R.6, R.19, R.20	UC.2, UC.7
G.4	R.2, R.3, R.5, R.6, R.13, R.34	UC.2, UC.8
G.5	R.1, R.3, R.4, R.7, R.24	UC.1
G.6	R.1, R.3, R.4, R.7, R.25, R.27, R.28	UC.1, UC.10

G.6.1	R.1, R.3, R.4, R.7, R.27	UC.1, UC.9
G.6.2	R.1, R.3, R.4, R.7, R.21, R.26, R.29, R.30, R.31, R.32	UC.1, UC.11, UC.12
G.6.3	R.1, R.3, R.4, R.7, R.22, R.23, R.33, R.37, R.38, R.39	UC.1, UC.13
G.7	R.1, R.3, R.4, R.7, R.35, R.36	UC.1, UC.3, UC.6

3.3 Performance requirements

We expect thousands of registrations in the first year of the platform's publication, due to the possible subscription of several private and public academic environments (especially high schools). In the subsequent years we expect to have progressively less and less subscriptions. For this reason scalability is not our main concern, althought it should not be overlooked considering the huge number of registered user expected in the first years. Response time and reliability are our main focuses. We need to guarantee a low response time in order to let students and educators interact actively with the platform without high delays that could compromise students' works. We have to take into consideration reliability since with a low reliability platform, students would not be able to upload their solutions consistently, educators would not be able to access the platform and manually evaluate students' solutions, and would be a bad experience overall. For data storing, we expect we'll need a high capacity DB. It's necessary for storing the user's data, tournaments and battles' data and statistics, informations and code about groups' commits. We'll need a high capacity DB, since being the platform built around students and schools, we expect to ha velots of users subscribing together at the same time, when school starts. These times are differenciated throught the year, for example in Italy schools starts in Semptember, while in New Zealand in starts in July. The platform being based around schools means that we'll have lots of user's data of old students that are now not using the platform anymore.

3.4 Design constraints

3.4.1 Hardware constraints

The users, both Educators and Students, to have access to the platform need to have a functioning internet access and a device connected to the net. The

Educators must have at least a smartphone, so that they can perform all of the different functions the platform offers them, like creating new battles and tournament, badges and manually evaluate groups' solutions. All of this can be done via smartphone. The Students are required to have a personal computer, either a desktop or a laptop, such that they can write and submit their solutions to the platform. Students can access the platform via smartphone too, for consulting their profile and looking at new tournaments and battles, but it would be best if everything that is concerning to coding and submitting a solution is done via personal computer.

3.4.2 Privacy constraints

Since the main application area is the european one, the platform must be compliant to the GDPR's law. Data must be encrypted before it is saved on the DB, no data must be accessed by any user other than the data's owner. For preventing cyber attacks, data must pass through secure connections and channels. In this case, no *Man in the Middle* attacks can occur. Autentication and login will be provided by the CKB platform, so all data must be treated accordingly to the platform's privacy policy. When a new User, both Educator and Student, wants to register to the platform, the privacy policy must be shown and accepted, before the new user is saved in the DB. If the privacy policy is not accepted, the user registration procedure is not completed and the new user is not saved in the DB, not granting access to the person that was trying to register to the platform.

3.5 Software system attributes

3.5.1 Usability

The platform must be easy to use. Users must not have doubts on which page to go to perform a precise task. Each button and link must be named accordingly, to limit time waste and useless interaction with the platform.

3.5.2 Reliability

The platform must prevent downtime in order to notify Students of new tournaments or battles as soon as they are created, in order to let Educators create new tournaments and battles whenever they want and for correctly publish new evaluations and grades on the platform for Students to see and access. We expect the highest number of simultaneous access to be whenever schools start and end, since more students will register on school starting, and more will rush to finish battles and projects as school is ending.

3.5.3 Availability

The platform must be available as much as possible, at least 99% of the time. Both Students and Educators must have access to the platform whenever they need to.

3.5.4 Security

Communication between Users and the Platform must pass through secure channels, and it must be encrypted. With a web certificate for the platform we can ensure that the SSL protocol is followed, and can ensure security for the platform and its Users. The DB guarantees that performed operations are all authorized, and permitted. *A Student cannot create new battles, for example.*

3.5.5 Web Browsing

The platform must be usable on any major web browser, since access to it is fully online.

3.5.6 Maintainability

The platform must be designed in a way that eases the future adding of new features with the least possible effort. Maintainability is also key for having a functioning platform, since the easier it is to maintain, the easier errors, reliability and availability failures can be corrected.

4 Formal analysis using Alloy

4.1 Signatures

```
//signature Educator used to represent the educator subscribed to the CBK
sig Educator {}

//A Permit is from one educator to a set of other educator, and regard only one
tournament
sig Permit {
        tournamentCreator : one Educator,
        battleCreators : some Educator,
        tournament : one Tournament
} {
        tournamentCreator not in battleCreators and
        tournament.creator = tournamentCreator and
        tournament.battleCreators = battleCreators
}

//A tournament have a creator and a set of educator that can create battle
sig Tournament {
        creator : one Educator,
        battleCreators : set Educator
}

//A battle have a creator and a tournament, the creator must be the creator or
an educator that can create tournament
sig Battle {
        tournament : one Tournament,
        creator : one Educator,
        tests : set Test
}{
        creator in tournament.battleCreators or
        creator = tournament.creator
}

//Test represent the test of a battle
sig Test {}

//Student in CKB can have a set of badges
sig Student{
        badges : set Badge,
        tournament : set Tournament
}

//A repository is assigned to a group and is relative to one battle
sig Repository{
```

```
        group : one Group,
        battle : one Battle
}{
        group.battle = battle
}

//A group is formed by n student and partecipate to one tournament
sig Group{
        groupCreator : one Student,
        students : set Student,
        battle : lone Battle
}{
        groupCreator not in students
}

//A badge is created by an educator
sig Badge{
        creator : one Educator,
        tournament : one Tournament
}{
        creator = tournament.creator
}

//An invite is made by a student of a group and direct to other student
sig Invite{
        sender : one Student,
        receiver : one Student,
        group : one Group
}{
        sender != receiver and
        sender = group.groupCreator
}
```

4.2 Facts

```
//The creator of a tournament can give the permission to creare battle in the
tournament only one time at educator
fact NoTwoPermissionAtSameEducator{
        all disj p1, p2 : Permit —
        (p1.tournament = p2.tournament anand d p1.tournamentCreator = p2.tournamentCreator)
        implies (all bC : p1.battleCreators — bC not in p2.battleCreators)
}
//A group can have at most only one repository for every battle
```

```
fact onlyOneRepository {
        all disj r1, r2 : Repository — all g : r1.group — all b : r1.battle —
        b != r2.battle or
        g != r2.group
}
//For every battle, a student can partecipate in only one group
fact noStudentsInMoreGroupInABattle{
        all s: Student — all disj g1, g2: Group —
        (s in g1.students and s in g2.students ) implies g1.battle != g2.battle
}

//The sender of an invite in a group and the receiver must be in the same
tournament
fact SenderAndReceiverInSameTournament{
        all i : Invite — all r : i.receiver — some t : r.tournament —
        t in i.sender.tournament
}

//All students in a group must partecipate at the tournament of the battle
they partecipate
fact memberOfAGroupSameTournament{
        all g : Group — all disj s1, s2 : g.students —
        some t1 : s1.tournament — some t2 : s2.tournament —
        t1 = t2
}

//Student must be in at least one battle in a tournament to receive badges
from it
fact StudentReceiveBadges{
        all s:Student — all b : s.badges — all g: Group —
        s in g.students and
        g.battle.tournament = b.tournament
}

//Student must be in the tournament where the battle is for partecipate at
them
fact StudentsGroupBattleInTournament{
        all g : Group — all s : g.students —
        g.battle.tournament in s.tournament and
        g.battle.tournament in g.groupCreator.tournament
}

//A student need to receive at least one invite from the leader of the group
to be into the group
fact StudentInAGroup{
        all g : Group — all s : g.students — some i : Invite —
```

i.sender = g.groupCreator and
i.receiver = s and
i.group = g
}

4.3 Examples of instances

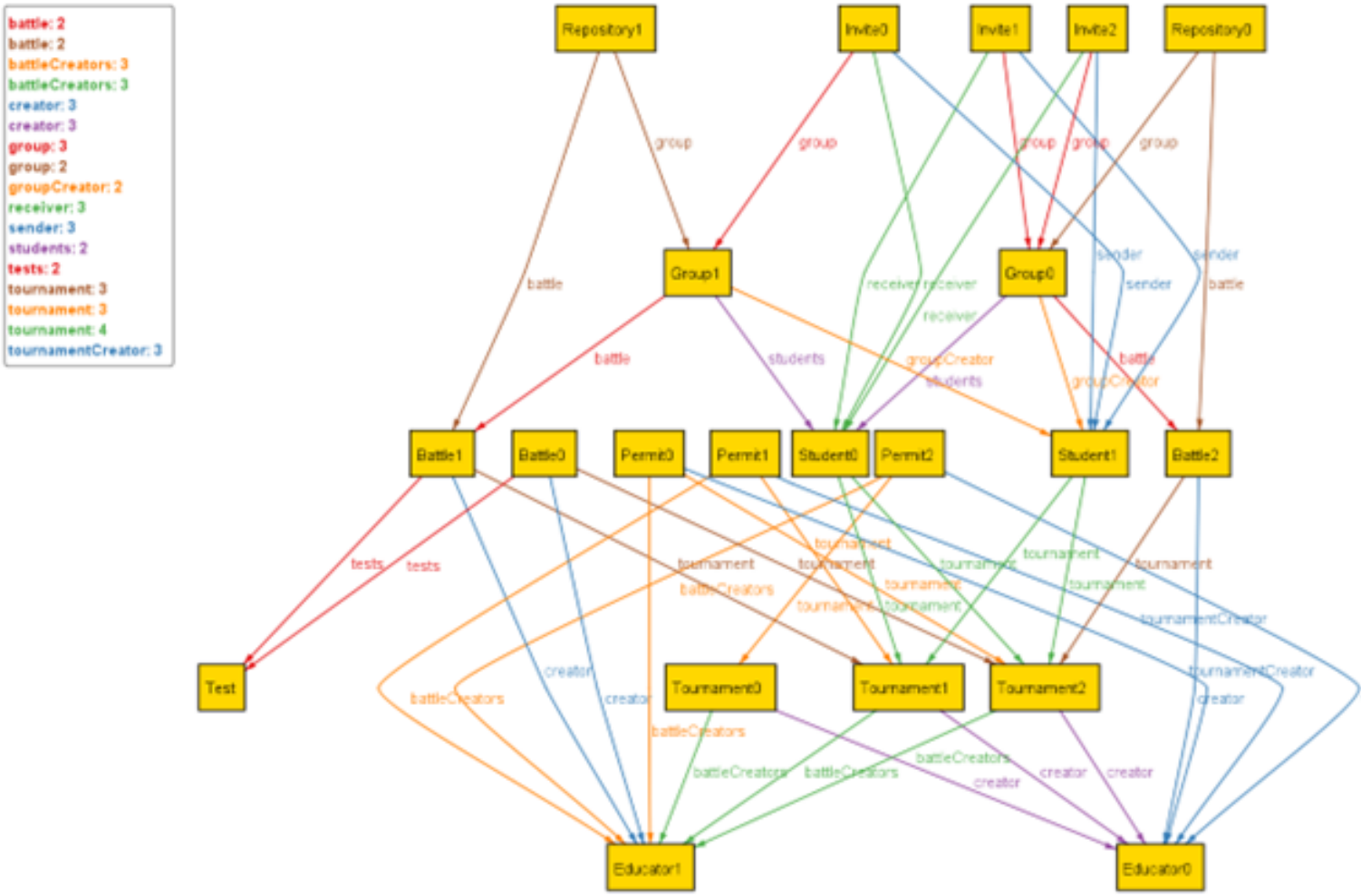

Figure 30: Picture of a general instance

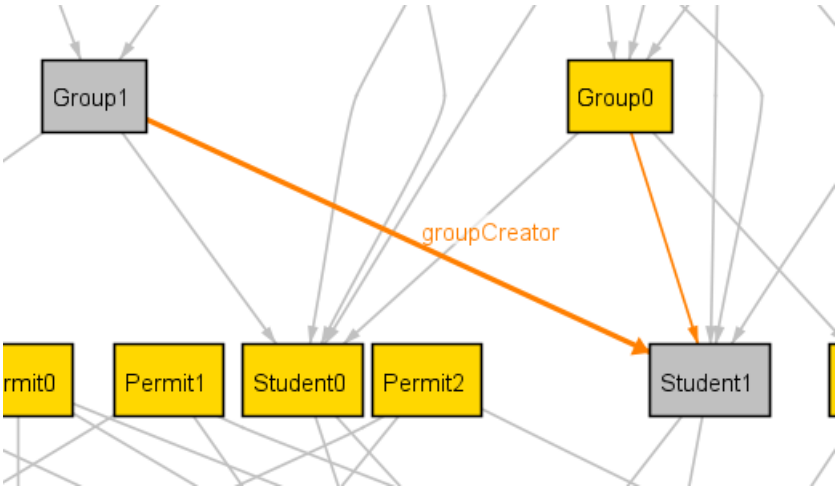

Figure 31: Student1 have created Group0 and Group1

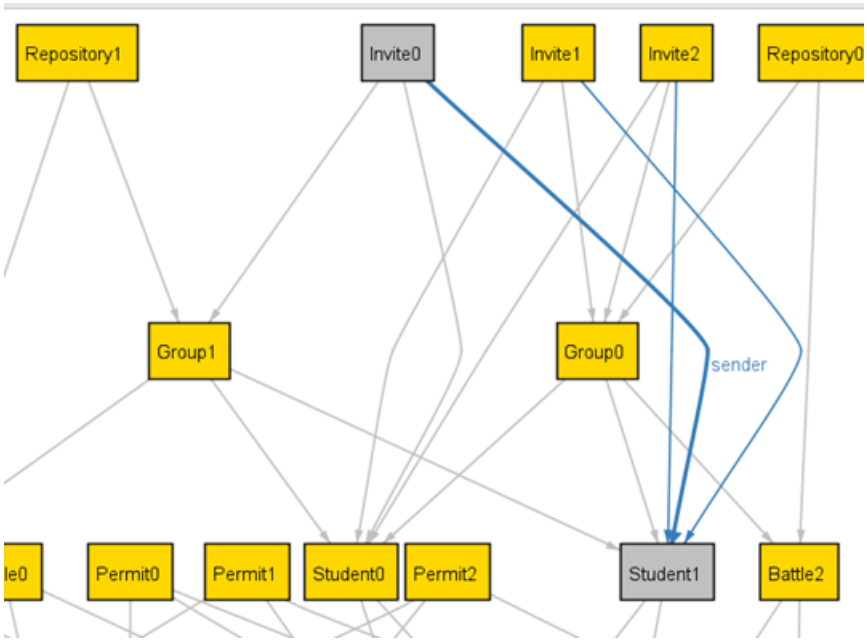

Figure 32: Student1 send 3 invite: Invite0, Invite1, Invite2

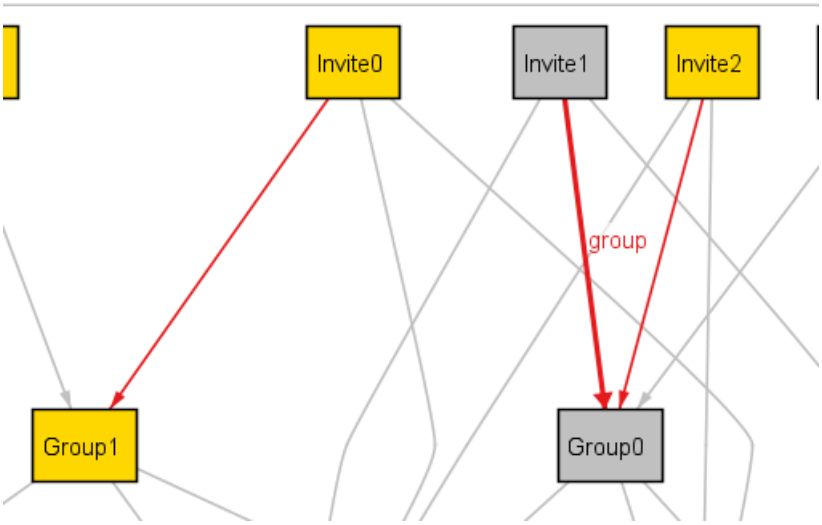

Figure 33: Each invite is relative to one group

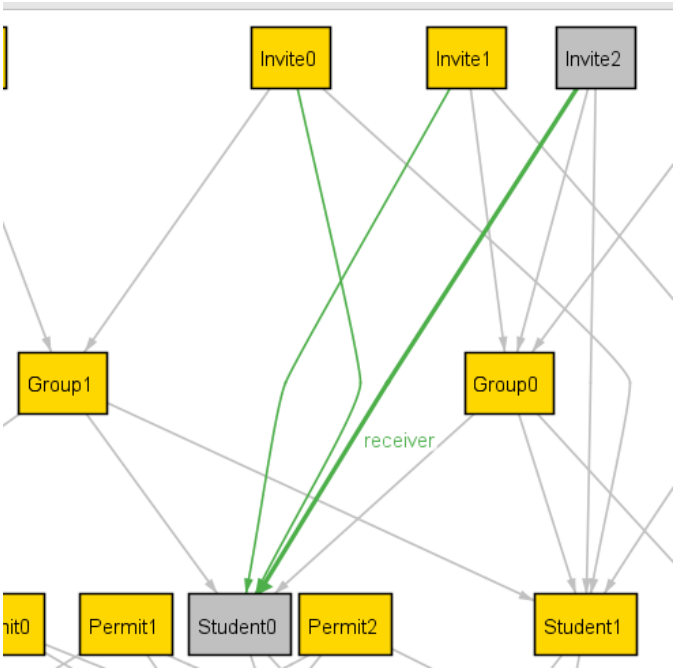

Figure 34: And all invite are being send to Student0

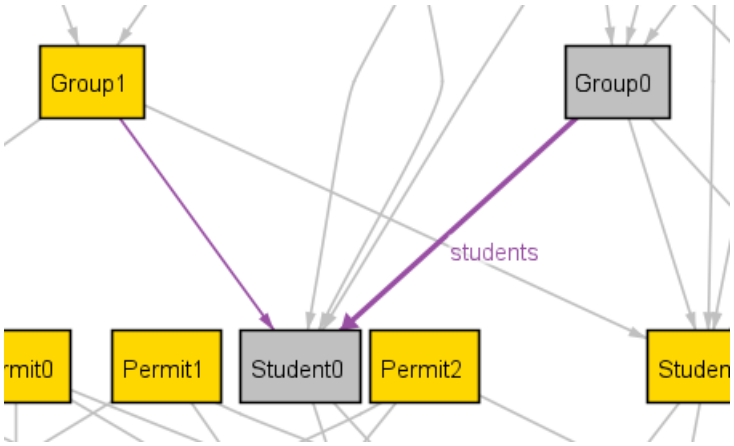

Figure 35: Here we can see Student0 in group1 and group0, since he have received two invite for group0 he have rejected the first one but accepted the second

Figure 36: tournament, educator and battle

Every permit give the possibility of create new tournaments to one Educator that isn't the creator of the tournament or have already the permit to create Battle (battleCreators)
The past description is propagated then to all the tournament (battleCreators)
Every battle is created by only one Educator that can be the creator of the tournament or also an educator that receive the permit to create battle in this tournament (creator) [Ex. Battle0 is created by Educator1 in Tournament2, that is created by Educator0. Educator0 give the Permit0 to Educator1 of create Battle in Tournament2]
Every tournament must be created by only one Educator (creator)
A test is relative to one or more battle, we can't exclude that one test can be only in one battle (tests)
A battle is only in one tournament (tournament)
A permission is relative to one tournament created by an educator (tournament)
A permission is also relative to the educator that created at least one tournament

(tournamentCreator)

5 Effort spent

Group member		Effort spent
Francesco Spangaro	Introduction Overall description Specific Requirements Formal analysis using Alloy	*Xh* *Xh* *Xh* *Xh*
Luca Tosetti	Introduction Overall description Specific Requirements Formal analysis using Alloy	*Xh* *Xh* *Xh* *Xh*
Francesco Riccardi	Introduction Overall description Specific Requirements Formal analysis using Alloy	*Xh* *Xh* *Xh* *Xh*

6 References

6.1 Paper references

- Specification document Assignment RDD AY 2023-2024.pdf
- //TODO:

6.2 Used tools

- GitHub for project versioning.
- Diagrams.net for UML,Use case,Sequence and BPMN diagrams.
- Visual Studio Code as LaTeXeditor.
- Alloy analyzer for formal analysis of alloy code.