Software Engineering 2

CodeKataBattle
Requirements Analysis and Specifications Document
version 1.0

Francesco Galbiati, Chiara Fossà

16th December 2023

Contents

1 Introduction

1.1 Purpose

As several studies demonstrate, group project are key experiences in the formative processes and play a very important role in the development of psychological characteristics (such as a dynamic mindset) of students. The CodeKataBattle platform (or CKB) aims, among other things, to provide a support for academic contexts, giving teachers and professors a powerful tool to organize challenging coding competitions between students, thus improving both their "soft" and "hard" skills. Using CKB teachers can organize tournaments and battles, in which students can participate in groups and compete with other students in coding battles with a language of choice. Furthermore the code will be automatically evaluated through the use of GitHub, which will be directly connected to the CKB system.

1.1.1 Goals

G1. Students and educators can create a profile to access the CKB platform.

G2. Users can log into the CKB system.

G3. Educators can create tournaments.

G4. Educators can create battles within the tournaments.

G5. Educators can manually evaluate a group's work.

G6. Educators can check the partial and final results of the battles they organize.

G7. Educators can close tournaments.

G8. Students can register for tournaments.

G9. Students can register for battles within a tournament they are part of.

G10. Students can invite other students to join a battle.

G11. Students are notified when a new tournament is created.

G12. Students are notified when a new battle is added to a tournament they are participating in.

G13. Students are notified when a battle they are participating in ends.

G14. Students are notified and can check the final result of a tournament they participated in.

G15. Students receive the links to GitHub repositories to submit battle code.

G16. Users can review the list of ongoing tournaments and for each tournament, the current ranking of participants.

G17. Students are notified when the final score of the battle becomes available and can check them.

G18. Students can submit code to the system using GitHub.

1.2 Scope

The CodeKataBattle (CKB) platform is a modern educational tool that helps students enhance their software skills through teamwork and code kata challenges. Educators use this platform to set up battles where student teams compete, all to encourage using test-based methods. In a code kata battle, students create programming solutions using provided tests, focusing on a test-first approach. These challenges happen in specific tournaments, with set parameters like group sizes and deadlines. The platform automates evaluations, considering parameters like functionality, timely work, and code quality, with an option for manual assessment. With its connection to GitHub, the platform keeps battle scores updated in real-time, showing the current rankings as they evolve. At the end of each battle, individual student performances get compiled into personal tournament scores, showing their position in each tournament's ranking. All this info is visible to CKB users, displaying ongoing tournaments and their standings. When educators close tournaments, final rankings trigger notifications to all participating students.

1.2.1 World Phenomena

WP1. The educator design a battle to propose to the students.

WP2. The educator manually evaluates the projects.

WP3. The students fork the linked repository.

WP4. The students push code on GitHub.

1.2.2 Shared Phenomena

Phenomena controlled by the world and observed by the machine.

SP1. A user enters their data in the login.

SP2. An educator creates a tournament.

SP3. An educator creates a battle.

SP4. A student signs up for a tournament.

SP5. A student signs up for a battle.

SP6. A student invites other students to join a battle.

SP7. A student answers an invitation to a battle.

SP8. An educator closes a tournament.

SP9. An educator enters a manual assessment.

SP10. GitHub sends code to the CKB system through the automated workflow.

Phenomena controlled by the machine and observed by the World

SP11. The system notifies the students when a new tournament is created.

SP12. The system notifies the students when a new battle is created in a tournament they are part of.

SP13. The system notifies the students when a tournament they are part of concludes.

SP14. The system notifies the students participating to a battle when the battle ends.

SP15. The system creates the GitHub repository.

SP16. The system sends the repository link to the participants.

SP17. The system automatically evaluates a project.

SP18. The system assigns the final score to a project.

SP19. The system displays the list of available tournaments to the user.

SP20. The system displays the list of available battles to the user.

SP21. The system displays the rankings of ongoing tournaments.

SP22. The system displays the battle scores to the participants.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Battle**
  Challenge in which those who compete submit a project which will be subjected to an evaluation and scored from 0 to 100. The objective of the participants is to achieve the best possible score. Each battle has its own ranking.

- **Tournament**
  Context in which educators can propose battles to the students enrolled in it and the students can participate in them. Each tournament has its own ranking made up of the rankings obtained from the battles contained in it.

- **Subscription**
  It is a registration procedure that allows students to express their interest in participating in a tournament or battle. It occurs through the click of a button on your profile and allows the student who has signed up to access tournaments and battles.

- **Ranking**
  A list of ratings that are the result of a competition (Battle's ranking), or the sum of competition's results (Tournament's ranking) The list associates a student's username with their score and is sorted in descending order of grade. This can be the ranking drawn up at the end of a battle or the ranking of a tournament.

1.3.2 Acronyms

- CKB: Code Kata Battle
- GDPR: General Data Protection Regulation
- UI: User Interface
- EU: European Union
- EEA: European Economic Area
- SET: Static Evaluation Tool

1.4 Revision History

- Version 1.0: First Submit

1.5 Document Structure

Mainly the current document is divided in 4 chapters, which are:

1. Introduction: In this section the document presents the general aspects of the project, such as the goals that the system is supposed to achieve, and some information that clarify how the document is written (abbreviations and acronyms)

2. Overall description: This section is about the overall behaviour of the system and of the environment the system is supposed to operate, in particular it focuses on the main functions of the CKB system and on the actors that will use the system once it has been deployed. This section also takes into consideration the internal structure of the system with diagrams such as class diagrams and state charts.

3. Specific Requirements: This section focuses on the requirements of the system and on how the different components of it interact between themselves, ending with a brief description of some functional aspects of the system, which will be expanded adequately in the design document.

4. Formal Analysis: In this part of the document the system previously described is analyzed using the Alloy Analyzer, both statically and dynamically.

5. Effort Spent: This section highlights the effort in hours spent by the author of this document.

6. References: This section provides the references used to write this artifact, both documents and websites.

2 Overall description

2.1 Product perspective

2.1.1 Scenarios

Scenario 1 - An educator creates a tournament and a battle
Mario works as a professor at Politecnico di Milano and holds the Section A-D of the "Fundamentals of Computer Science" course for the freshman students of the Engineering in Computer Systems degree. Aiming to improve the interaction between students in his class, he decides to employ the CKB platform to challenge his students with various problems that can be resolved using the C programming language. After registering to the CKB platform as an educator and understanding the benefits that its use can provide to the students, Mario invites his colleagues to do the same and suggests them to talk about the upcoming tournaments during their lectures. The next week Mario presents the CKB initiative to his class and invites every student to register immediately to the CKB platform using their university email address to be notified when the tournament will be created, after the lecture Mario creates a tournament called "SuperMarioCode", to which the students attending his lectures will be able to compete. Mario also creates a tournament called "SuperPoliCode" for all the students enrolled in the whole course and grants the permission to create battles to every colleague registered to CKB teaching in the classes parallel to his. After a week all Mario's students subscribed to "SuperMarioCode" tournament and Mario decides to create the first battle for SuperMarioCode and inserts in the system the necessary parameters (description of the problem, test cases, minimum and maximun number of students for per team, registration and submission deadlines), when the battle is created all his students participating in the tournament are notified and can subscribe to the first battle.

Scenario 2 - An educator is granted the permission to create a battle for a tournament
Professor Agenore, a professor at Politecnico di Milano, holds the Section E-L of the "Fundamentals of Computer Science" course for the freshman students of the "Engineering in Computer Systems" degree. He would like to challenge students in solving computer problems through the use of the C language and to do so he would like to use the CKB platform. One day, Professor Mario offers him to collaborate in creating challenges for students on the CKB platform. It seems like a wonderful idea to Agenore and so he signs up to the platform as an educator. Some time later he receives a notification informing him that Professor Mario has invited him to contribute to the creation of the battles in the "SuperPoliCode" tournament, a tournament involving every section of the "Fundamentals of Computer Science" course. Agenore accepts the invitation: he can now create a battle for the students registered in the "SuperPoliCode" tournament. After two week Agenore decides to create his battle by inserting into the system the requested parameters (description of the problem, test cases,

minimum and maximum number of students for per team, registration and submission deadlines). Once the battle is created all his students participating in the tournament are notified and can subscribe to Agenore's battle.

Scenario 3 - A student subscribes to a tournament and to a battle
After hearing about an upcoming tournament on the CKB platform, organized by his professor, Linda, a freshman student at Politecnico di Milano, signs up for the CodeKataBattle (CKB) platform using her university email address. After some time, she receives a notification that registrations are open for the "SuperMarioCode" tournament, a coding challenge designed by Professor Mario for the students of his class. Determined to improve her skills in the C programming language and to pass her first exam with full marks, Linda logs into the platform and successfully enrolls in the SuperMarioCode tournament before the deadline. After the enrollment, one day Linda recieve the notification that signals the beginning the first battle within the tournament. Reading the battle's description and instructions Linda find out that this battle allows teams made up of a single person. Linda wants to test her skills so she signs up for the battle and starts competing.

Scenario 4 - A student subscribes to a battle and invites a colleague to form a team
Simone is a freshman student in the "Engineering of Computer Systems" degree and attends the lectures held by professor Agenore, which the "Fundamentals of Computer Science" course. After hearing from his professor about the "SuperPoliCode" tournament, a coding competition held between all the freshman students of "Engineering of Computer Systems", he registers to the CKB system and subscribes to the tournament. After the lecture he informs his fellow student Elisa, who was not present in class that day, of "SuperPoliCode" and suggests her to subscribe to the platform and to the "SuperPoliCode" tournament as the subscriptions deadlines expired yet, so that they can form a team and enroll in the next battle, Elisa accepts and immediately subscribes to CKB. When the first battle of "SuperPoliCode" is created Simone subscribes to it and sends an invitation to Elisa, requesting to form a team with him in order to compete in the "SuperPoliCode".

Scenario 5 - A student is invited by a colleague to form a team in an upcoming battle
Elisa is a student attending the "Fundamentals of Computer Science" course of professor Agenore and while talking to her friend and colleague Simone about a lesson she couldn't attend, Simone advises her to sign up to the CKB platform suggested by the professor and asks her to form a team with him for the next battle in the "SuperPoliCode" tournament. Elisa accepts very willingly and informs Simone of her decision, so she is not surprised to receive a notification from the CKB platform after a short time in which Simone invites her to form a team with him. Elisa logs into the platform and accepts Simone's request.

Now the two students are a team ready to face the battle proposed by Professor Agenore.

Scenario 6 - A group pushes the code on GitHub
Marina, Linda and Daniel work as a team on a battle proposed by Professor Mario, which is part of his "SuperMarioCode" tournament. After hours of work they reach a satisfying solution and decide to submit the code to the automatic evaluator, to do this they access their GitHub repository and push their code. In this way GitHub automatically sends the submitted code to the CKB platform, that will evaluate it and assign it a partial score between 0 and 100, which will be immediately visible to the team. Unfortunately the group is not satisfied with the result and they decide to work on it a bit more, so after some time they push the new code on GitHub, but the result of the scoring is even worse. After figuring out the errors in their code the group agrees on one last version, and, since the deadline is drawing near, the immediately push the corrected code, in this way the system updates their score, leaving the group satisfied with the obtained result.

Scenario 7 - An educator manually evaluates a project
The battle proposed by Professor Mario has reached its deadline. When Mario created it, he requested to be able to manually evaluate a portion of the project, so now that the time is up, Professor Mario can proceed with the manual correction by accessing the platform. For each team that participated in the battle, Mario carries out a correction and evaluation of the submitted projects, then he verbalizes the final evaluations. The results of the battle obtained, combining the automatic and the manual evaluation, become available for consultation and, for each student involved, the respective tournament score is updated. Thee students that participated in the battle are notified about the availability of the battle ranking .

Scenario 8 - An educator closes the tournament and the results become available
The semester ha come to an end and the "Fundamentals of Computer Science" course will conclude shortly. There are no more battles underway in the "SuperPoliCode" tournament so Mario, the tournament organizer, logs in to the platform and closes the tournament. All participants in the tournament receive a notification regarding the closure and the ranking of the scores that each student achieved during the tournament. Said score is available for consultation by all users now that the tournament is officially over.

Scenario 9 - After a tournament has been closed a student consults the final results
After the "SuperMarioCode" has officially ended the student Fragab receives a notification informing the participants the closure of the tournament. Since the final scores are now available for consultation the student logs into the CKB

platform, selects the "SuperMarioCode" tournament and looks for his username in the ranking of all the participants.

2.1.2 Domain class diagram

The figure represents the domain class diagram related to the CKB system, in which the main elements of the system are described with the interactions that take place between them.

In particular:

- The users of the system are divided in two types, Educators, which can manage tournaments and battle, and Students, that can participate in challenges created by educators. Students and Educators mainly differ in the role they play in the functioning of the system, without much difference in the information provided to the system.

- The main features of the system are tournamenst and battles, tournaments are created by educators, and students can subscribe to them. A tournament is created by only one educator, but the creator can also grant other colleagues some permissions that will give the colleague the possibility of managing some aspects of the tournament (creating battles). Tournaments are composed of battles, created by the educators managing of the tournament. To create a battle some information need to be inserted (discussed in the next section). Student have to form teams, according to the rules inserted by the managing educator, with other students subscribed to the same tournament. More about the invitation process is provided in 2.2.3. Once started, students produce solutions to the problem (generally multiple solutions for a single problem, given the *test-first* approach that characterize the CKB platform), solutions are immediately evaluated by the system, and at the end of the battle the last submitted solution will be considered the final one for the group and will be manually evaluated by the educator that manages the battle.

- Battles consist in a problem formulated in natural language that has to be solved using a programming language of of the battle creator's choice, when a battle is created the educator has to insert the parameters that characterize the challenge, first among them is the specification of the problem, then the rules for team composition (minimum and maximum number of students in a single team), the registration and submission deadline, the programming language to use in the battle and if the problem requires a manual evaluation by the educator. Lastly the creator needs also to provide the test cases, which will be used by the students to verify the correctness of their solution and by the system to evaluate each solution proposed by the teams. The final scores of a battles contributes to the global ranking of the tournament the battle belongs to.

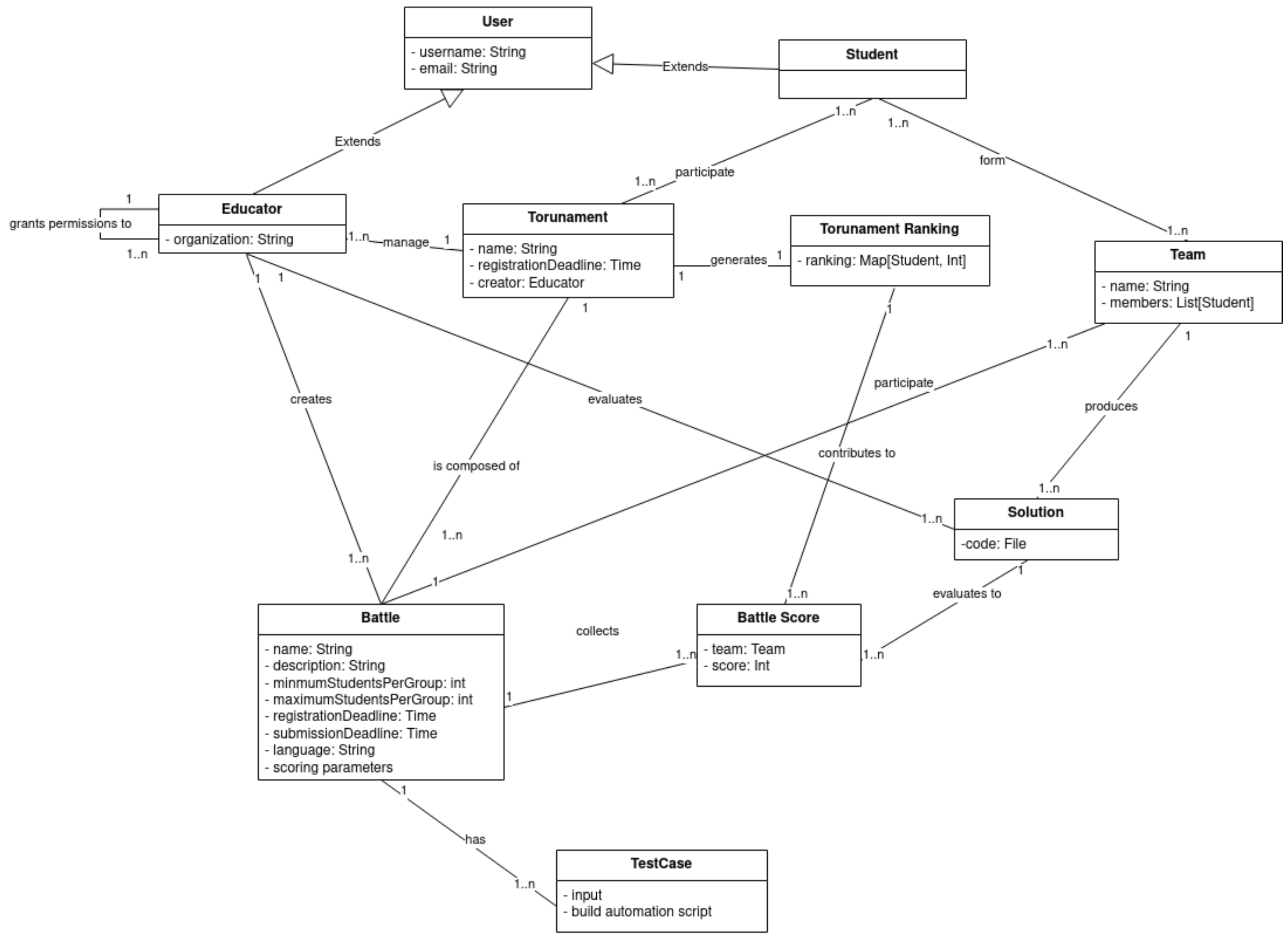


Figure 1: Domain Class Diagram of the CKB system

2.1.3 State-charts

The following state-charts describe the internal behaviour of the tournament and battle classes, which are the two main classes described in the domain diagram.

Tournament class When a tournament is created it enters the first state of the chart, called *Created*, while in this state the tournament can provide registration functions for the students subscribing to the tournament and a way for the creator to grant permissions to other colleagues. Once the registration deadline is reached the tournament enters in the *Ongoing* state, in this state the tournament is started and battles can be created by the educators having the relative permissions, moreover the creator of the tournament can still provide managing permissions to other colleagues also while the tournament is ongoing. While the tournament is in this state results from battles which have ended will be published, once published this results are used to update the global ranking of the tournament. The tournament ends when the creator decides so (if there are no ongoing battles in the tournament), the closure is registered as a state passage from *Ongoing* to *Closed*, lastly when the educators decides to publish the definitive results of the tournament the system passes into the last state, where no more computations are needed and can pass in the termination state.

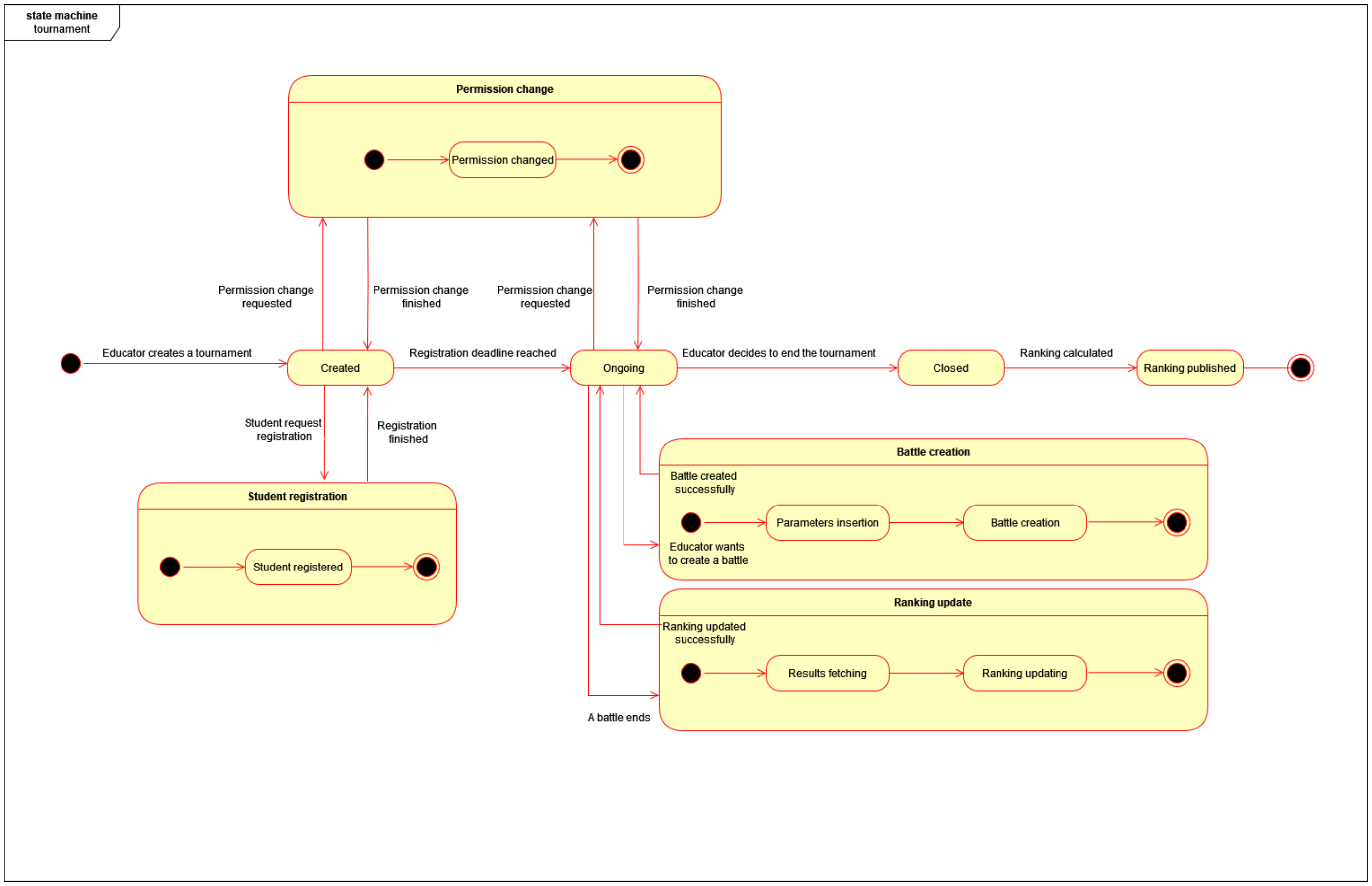
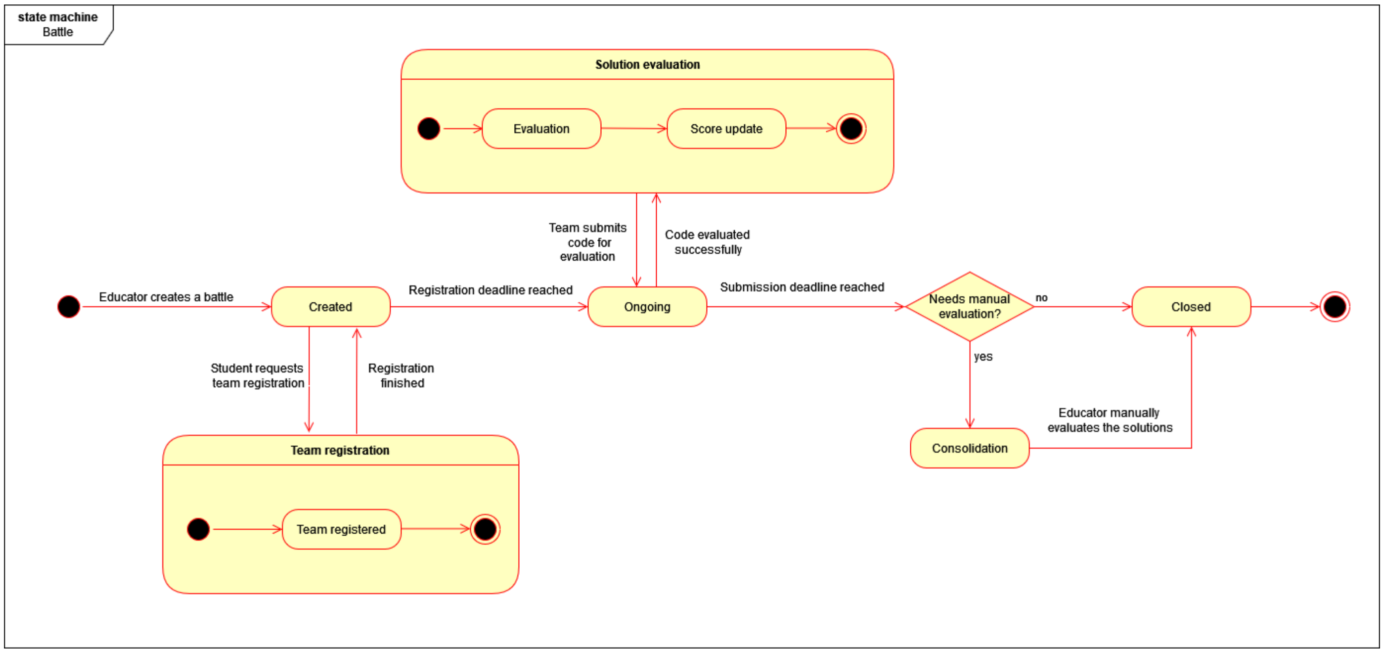

state machine
tournament
Permission change
Permission changed
Permission change requested
Permission change finished
Permission change requested
Permission change finished
Educator creates a tournament
Created
Registration deadline reached
Ongoing
Educator decides to end the tournament
Closed
Ranking calculated
Ranking published
Student request registration
Registration finished
Student registration
Student registered
Battle creation
Battle created successfully
Parameters insertion
Battle creation
Educator wants to create a battle
Ranking update
Ranking updated successfully
Results fetching
Ranking updating
A battle ends

Figure 2: State chart of the torunament class

Battle class A battle belongs to a specific tournament and can be created by an educator who has the corresponding authorizations for that tournament. To create a battle an educator is required to specify some information regarding that battle, in particular the system needs to know:

- The specification of the problem, expressed in natural language
- The programming language to use for the specific battle
- The minimum and maximum number of students for each team
- The registration and submission deadlines
- If the battle needs manual evaluation
- The test cases to use in the evaluation process

Once all this information are inserted the battle can be created (*Created* state in the chart), until the registration deadline expires students can form teams and subscribe their team to the battle. When the registration deadline is reached the battle starts (*Ongoing* state in the chart), in this phase teams can submit solutions to the problem, solutions are immediately evaluated by the system and the computed results are published on the CKB platform, where they become visible by everyone involved in the ongoing battle, both educators and students. After the submission deadline passes the CKB platform validates the partial scores of the participants as definitive, at this point if manual evaluation is needed the battle enters into the *Consolidation* state, where the platform waits for the evaluation by the educator, otherwise the battle simply ends, adding the scores to the global ranking of the tournament.



state machine Battle
Solution evaluation
Evaluation
Score update
Team submits code for evaluation
Code evaluated successfully
Educator creates a battle
Created
Registration deadline reached
Ongoing
Submission deadline reached
Needs manual evaluation?
no
Closed
yes
Consolidation
Educator manually evaluates the solutions
Student requests team registration
Registration finished
Team registration
Team registered

Figure 3: State chart of the battle class

2.2 Product functions

2.2.1 Sign up and login

All users must perform these two actions to have access to the platform. With the sign up functionality the user registers in the system and the user's personal homepage is created. When registering to CKB the user must first specify whether he intends to register as a student or as an educator through to correspondent button on the UI. Both of these options lead to a specific registration form, which can ask different kinds of data depending on the choice. During the registration process, each user must provide their name, surname and email address (which will be used as the user's username) and must choose a password. Educators could also be asked to insert data regarding the organization they are part of, other aspects of their job or about the context in which they intend to use CKB. As the registration request has been sent, the user will receive a verification email to the email address provided. Once the email address has been verified, the user can carry out the login procedure: by entering the email address and password the user will be able to access the Homepage.

2.2.2 Manage the battles' creation

The core of the CodeKata platform is the management of battles in which users registered as students can compete. Battles can only be created by users registered as educators. To create a battle, an educator must access his homepage and select a tournament created by her or a tournament created by another educator in which she has been invited to participate in the creation of battles. Once the tournament has been selected, the educator creates the new battle (entering all the required details: the code kata (description and software project, including test cases and build automation scripts), minimum and maximum number of students per group, a registration deadline, additional configurations for scoring (such as criteria for automatically evaluating the quality of sources)). The system notifies all tournament participants the creation of the new battle.

2.2.3 Manage the battles' subscription

At this point, the "students" registered for the tournament can, by accessing their homepage on the platform (where all the tournaments and battles still in progress and in which they are participating are listed), register for the battle. When a student signs up for a battle she must specify (if required by the battle specifications) whether she wants to face it alone or with other students. If she wants to create a team with other students she will have to provide the emails of her teammates to allow the system to send an invitation to them. The number of teammates must not exceed the limit decided by the educator for that battle, so a team can only have a number of pending invitations equal to the remaining available slots. When a student receives an invitation they have 2 days to accept it, any unanswered invitation is considered as declined after that period of time. When an invitation is accepted by a student, any other invitation sent to that student is automatically declined and when an invitation from a team is declined a slot is freed for the team, so that another one can be sent. When the registration deadline is reached, every team with a number of "confirmed" members greater than the minimum number of allowed participants is subscribed to the battle, the teams that do not meet this requirement are not enrolled.

2.2.4 Manage the battles' competition

When the registration deadline expires, the platform creates a GitHub repository containing the code kata and then sends the link to all subscribed teams (a team can also be composed by one student). Students must fork the repository and set up an automated workflow through GitHub Actions that informs the CKB platform (through proper API calls) as soon as students push a new commit into the main branch of their repository. Each push before the deadline activates the CKB platform that analyzes the latest sources, and runs the tests on the corresponding executables to calculate and update the battle score of the team. The score ranges from 0 to 100 and the following are evaluated:

- The functional aspects based on the number of test cases passed
- The time taken from the registration to the last commit
- The quality of the sources, evaluated according to the criteria provided by the creator of the battle

The last updated score before the deadline will be considered for the final ranking. If a team has not pushed any work the score will automatically be 0.

2.2.5 Manage the battles' consolidation stage and valutation

When the time expires a consolidation stage begins which, if manual evaluation is required, the educator uses the CKB platform to go through the sources produced by each team to assign the manual evaluation score. Once the consolidation stage finishes, all students participating in the battle are notified when the final battle rank becomes available.

2.2.6 Manage the tournament

The function of creating and managing tournaments, as well as that which concerns the creation and management of battles, is reserved for educators. From the homepage the educator, in addition to viewing the active tournaments in which he can create a battle (these are the tournaments he created and those in which he was invited by another educator), can create a new tournament. When a new tournament is created, all students subscribed to the CKB platform are notified and they can subscribe by a given deadline. If they subscribe, they are notified of all upcoming battles created within that tournament. Once the tournament has been created, the educator has the possibility to create battles and invite other educators. At the end of each battle, the platform updates each student's personal tournament score, i.e. the sum of the scores for each battle played in that tournament. For each tournament there is a ranking visible to all members of the platform, which reports and compares the tournament scores of each participating student

2.3 User characteristics

2.3.1 User

In general, a user is a person in possession of a device capable of connecting to the internet and accessing the CKB platform. To have access to the platform's features the user must register.

2.3.2 Student

A student is a user registered as student. In addition to the requirements of a user, the student must have access to a GitHub account connected to the same email with which they registered on the CKB platform. The user must also have a text editor available on which to write the code which will then be pushed to GitHub for evaluation. The student is the user who can register and compete in tournaments and battles and receive a rating.

2.3.3 Educator

An educator is a user registered as educator. It does not require any further prerequisites than the user definition. The educator is the user who can create and close tournaments and create and evaluate battles.

2.4 Assumptions, dependencies and constraints

2.4.1 Regulatory policies

The CKB application requires personal user data such as name, surname and mail address. Email addresseses won't be used for marketing purposes. Personal information will be handled in accordance with the GDPR.

2.4.2 Domain assumption

- D1. Users must have access to a device with internet connection.
- D2. Users must have access to the software interfaces required to write and send code to the platform (GitHub and a text editor).
- D3. Connection in reliable.
- D4. GitHub services must be available.
- D5. Information showed by the user interface must be correct.

D6. Automatic evaluation must be consistent with the specified parameters.

D7. Educators will eventually manually evaluate the solutions, if specified at the creation stage of the battle.

D8. Students successfully fork the GitHub repository.

3 Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

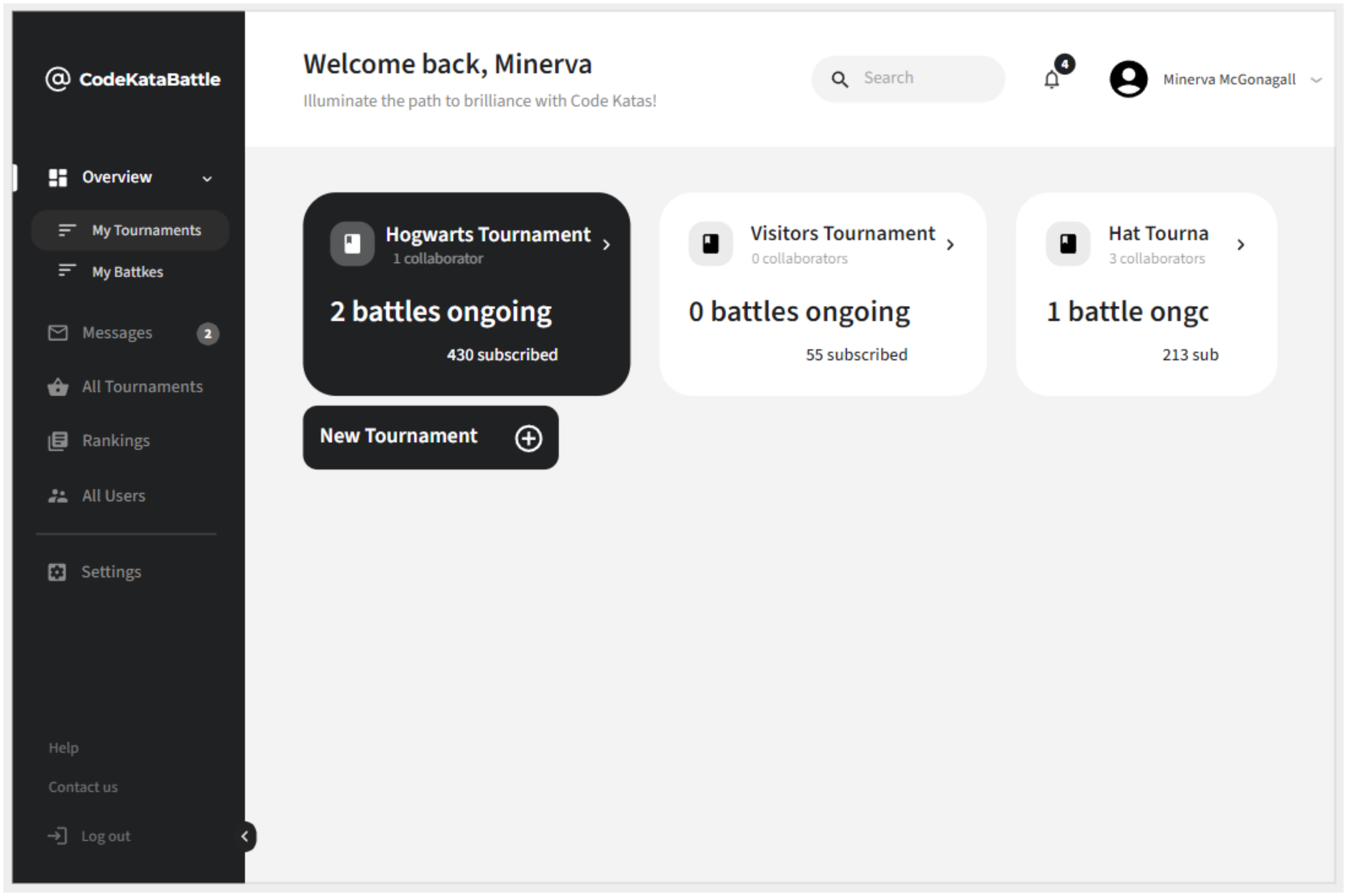
This section of the document represents the user interface from the point of view of an educator user and from the point of view of a student user. The interfaces that are shown to the two types of users are very similar: what changes are essentially some details that are shown in one interface rather than the other and the possibility or not of interacting with the interface to make changes to the platform.

Figure 4 shows the interface in which the educator can view and have access to all the tournaments she created or in which she was invited to participate. In this section the educator can also create a new tournament. For each tournament, the number of collaborators, the students registered for the tournament and the battles currently underway in that tournament are indicated. Tournament battles that have already concluded will be viewable both in the My Battles section and by clicking on a tournament in this section. By clicking on a tournament you can view all the details of the tournament.

Figure 5 represents the My Battles section of the educator interface. Here the user can view all the battles created by her whether they are finished or not. For each battle the tournament it is part of is indicated. Furthermore, battles can be ongoing or already concluded. For battles still in progress, the time remaining before the first deadline and the number of students registered for the battle are indicated. If the battle is concluded it will be indicated with the wording "ENDED" and the number of students who took part will be attached. To get more information and view all the details regarding a battle, you must click on it.

Clicking on a tournament opens the detailed interface relating to that tournament. As shown in Figure 6, the educator can view who the creator of the tournament is and who are the other educators who have been invited to participate. You can see all the battles that make up the tournament, both the finished ones and those still in progress. For each battle, the educator who created it, the number of participating students and the deadline are indicated. The ranking of the students participating in the tournament is also available. The rank is updated with battle results as soon as a battle ends. The data listed so far is also visible from the student interface. The privilege of the educator who created the tournament is the button to close the tournament and the button to manage the permissions granted to other educators to add battles in that tournament. All educators collaborating in the tournament see the button to add a battle to the tournament.

Figure 7 and Figure 8 are the student interface.



Welcome back, Minerva

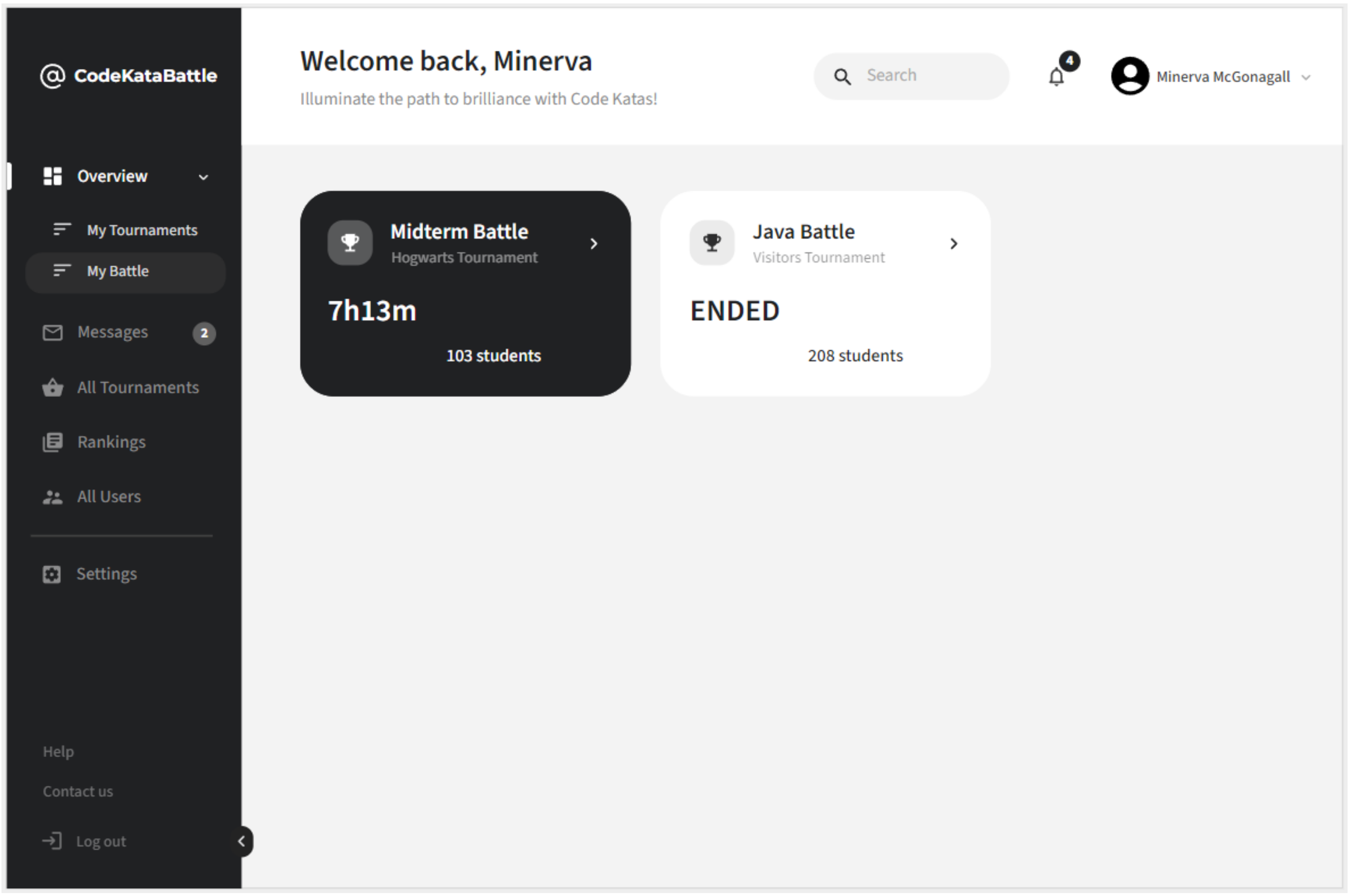
Illuminate the path to brilliance with Code Katas!

Hogwarts Tournament
1 collaborator
2 battles ongoing
430 subscribed

Visitors Tournament
0 collaborators
0 battles ongoing
55 subscribed

Hat Tourna
3 collaborators
1 battle ongc
213 sub

Figure 4: Educator: My Tournaments



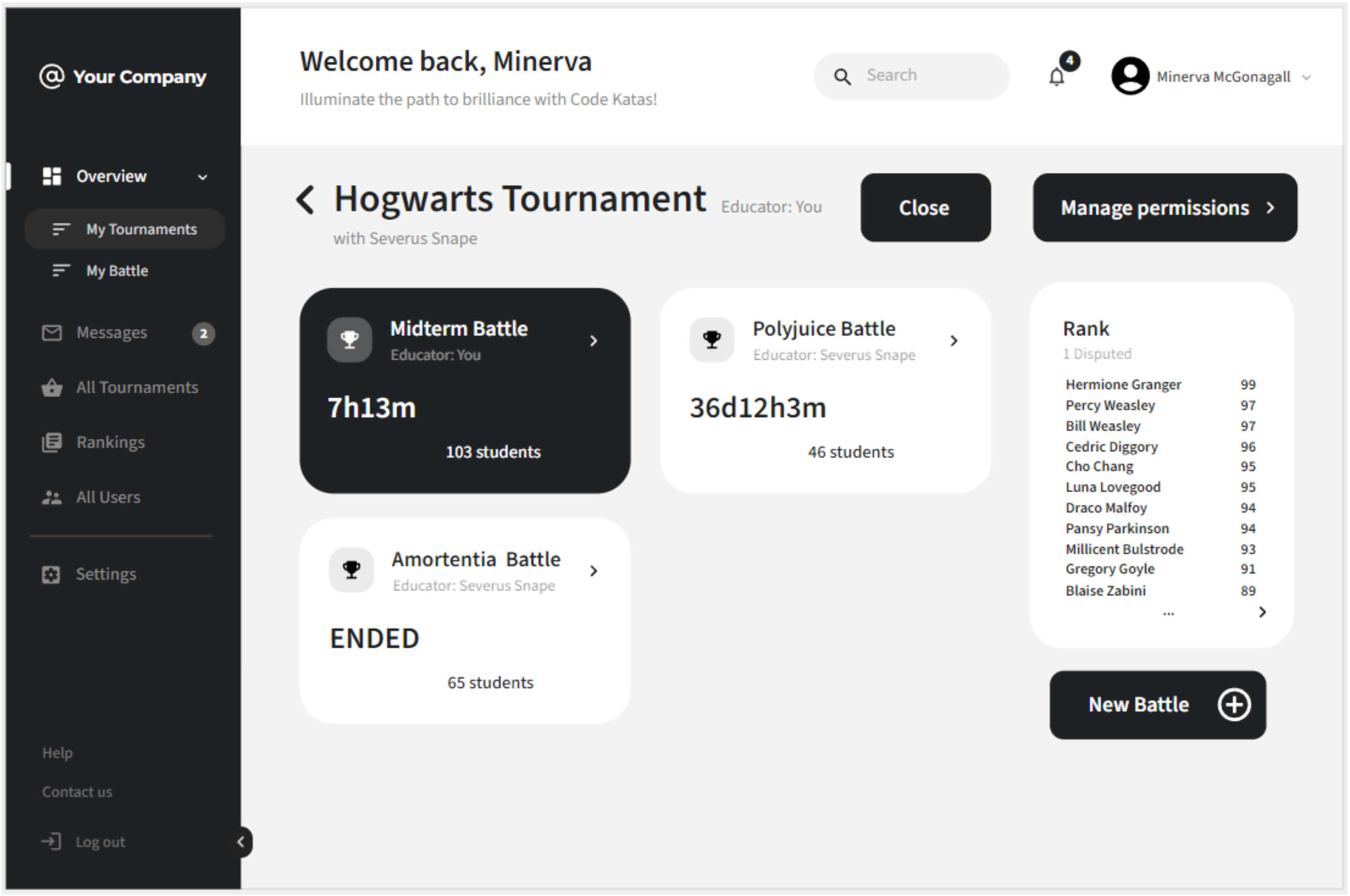
Welcome back, Minerva

Illuminate the path to brilliance with Code Katas!

Midterm Battle
Hogwarts Tournament
7h13m
103 students

Java Battle
Visitors Tournament
ENDED
208 students

Figure 5: Educator: My Battles



Welcome back, Minerva

Illuminate the path to brilliance with Code Katas!

Hogwarts Tournament Educator: You

with Severus Snape

Midterm Battle
Educator: You
7h13m
103 students

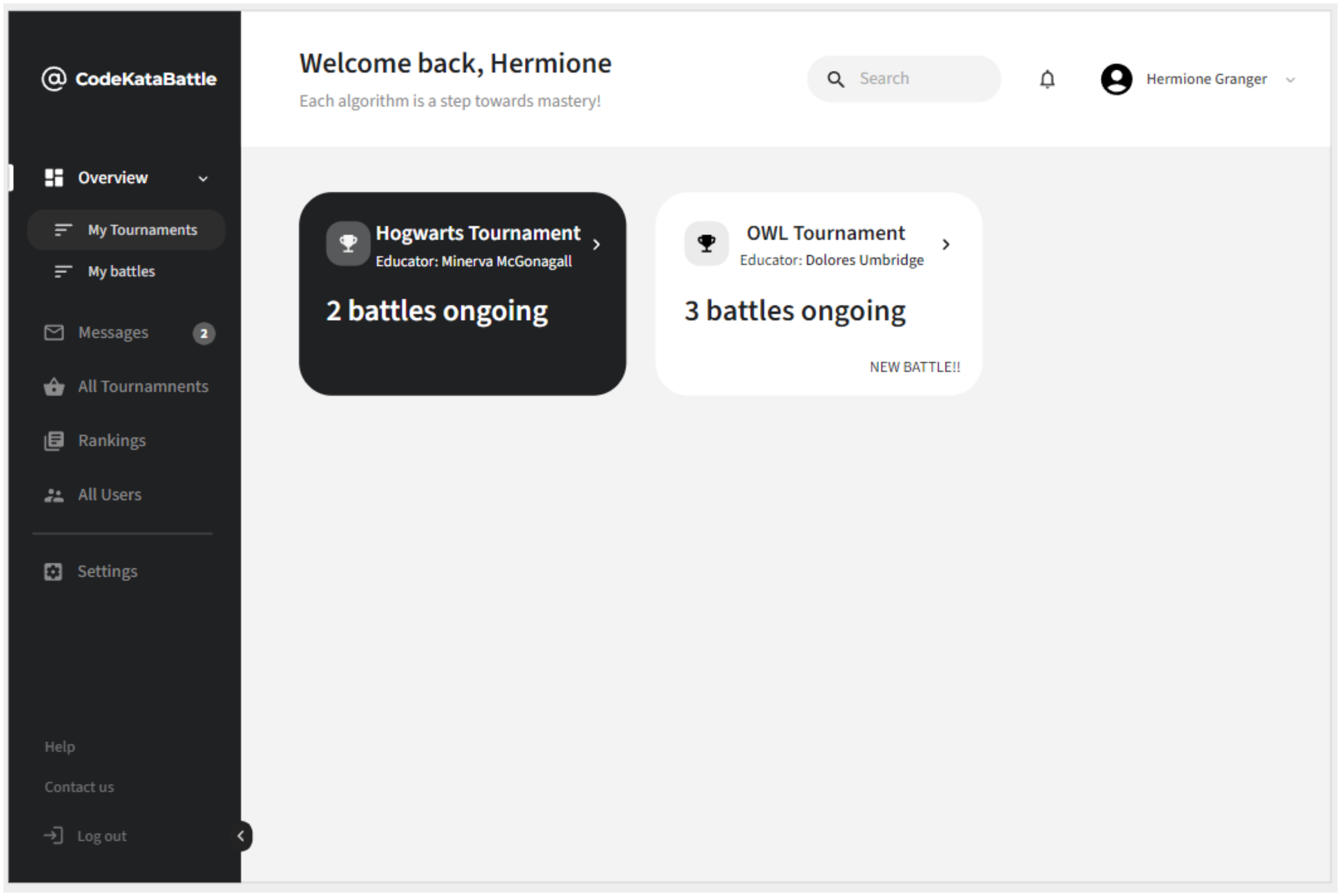
Polyjuice Battle
Educator: Severus Snape
36d12h3m
46 students

Amortentia Battle
Educator: Severus Snape
ENDED
65 students

Rank
1 Disputed

Hermione Granger	99
Percy Weasley	97
Bill Weasley	97
Cedric Diggory	96
Cho Chang	95
Luna Lovegood	95
Draco Malfoy	94
Pansy Parkinson	94
Millicent Bulstrode	93
Gregory Goyle	91
Blaise Zabini	89
...	

Figure 6: Educator: A Tournament



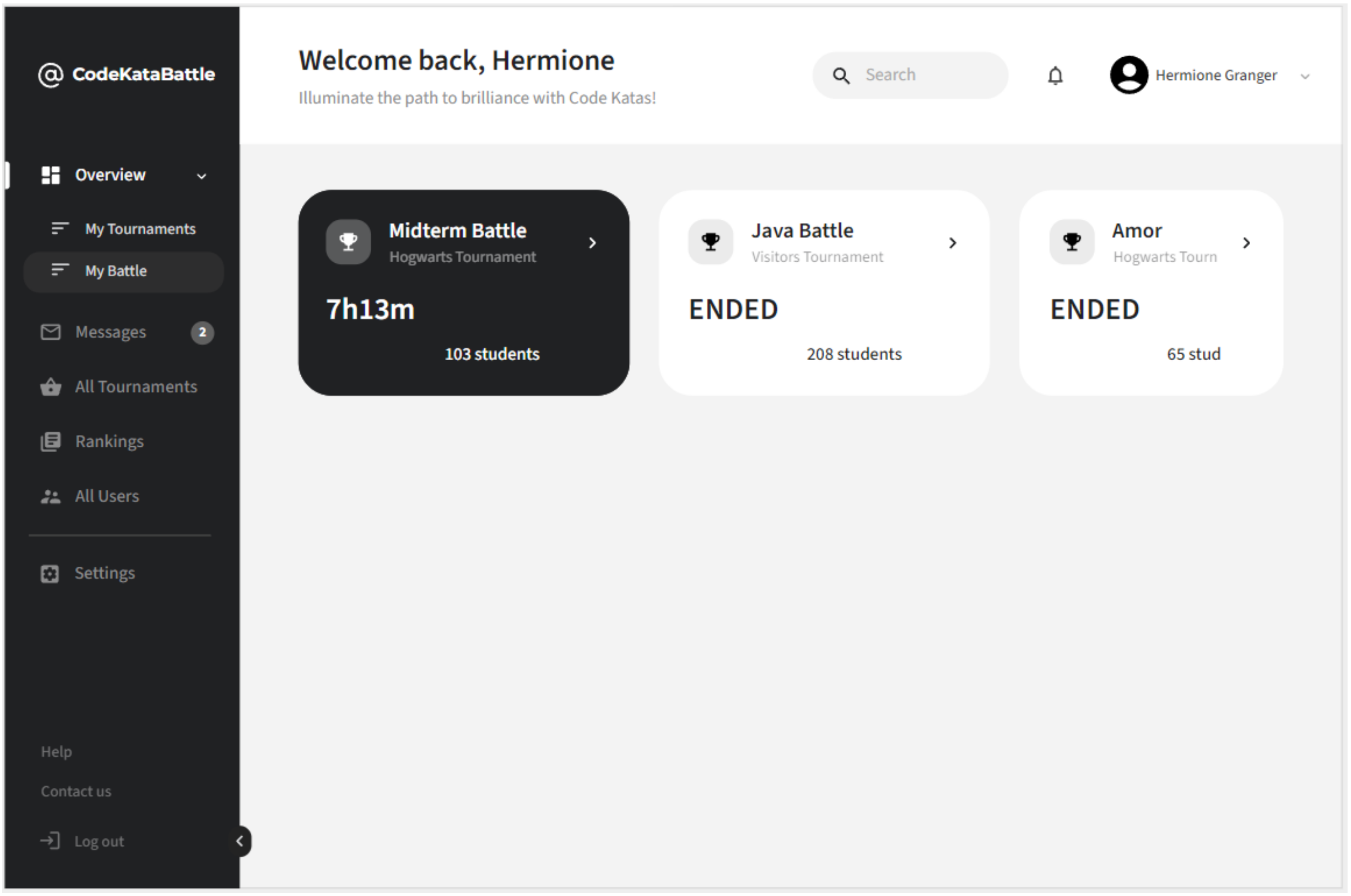
Welcome back, Hermione

Each algorithm is a step towards mastery!

Hogwarts Tournament
Educator: Minerva McGonagall
2 battles ongoing

OWL Tournament
Educator: Dolores Umbridge
3 battles ongoing
NEW BATTLE!!

Figure 7: Student: My Tournaments



Welcome back, Hermione

Illuminate the path to brilliance with Code Katas!

Midterm Battle
Hogwarts Tournament
7h13m
103 students

Java Battle
Visitors Tournament
ENDED
208 students

Amor
Hogwarts Tourn
ENDED
65 stud

Figure 8: Student: My Battles

3.1.2 Hardware interfaces

All users have access to their personal homepage of CKB which is provided via a web page, hence they are required to have just a device with a working internet connection.

3.1.3 Software interfaces

To function correctly, the system requires some software interfaces that allow the user and the platform to communicate more efficiently.

Mailbox Mailbox is essential for timely communication on platforms. It allows users to confirm registration and receive notifications about new battles, tournaments and rankings.

GitHub account GitHub is a collaborative platform that enables students to fork the kata code repository supporting collaboration and communication with the CKB platform.

Text editor A text editor is useful for students to efficiently manage the code, make changes on it, add commits to GitHub and deal with the code in general, A specialized text editor also offers practical features like syntax highlighting ,debugging tools and auto-complete.

3.1.4 Communication interfaces

The CKB system exploits internet connection to connect the clients' devices with the platform. The system is also connected to the GitHub platform and to some static analysis tools, in order to provide the automatic evaluation functions.

3.2 Functional requirements

The following sections concern the different functional requirements that have been individuated with respect to the CKB system.

R1. The system allows allow all users to log in.

R2. The system allows users to register with the proper account.

R3. The system allows educators to create tournaments.

R4. The system allows educators to grant permissions to manage a tournament to other educators, but not to themselves.

R5. The system allows educators with the proper permissions to create a battle for a tournament.

R6. The system allows educators to insert all the information regarding a battle during the battle creation phase.

R7. The system allows students to register to a tournament.

R8. The system notifies students when a new tournament is created.

R9. The system allows students to register for a battle.

R10. The system doesn't allow a student already registered in a battle with a team to be registered in the same battle with another team.

R11. The system allows students to send invitation to other students but not to themselves.

R12. The system allows invited students to accept or decline an invitation.

R13. The system creates a GitHub repository for each battle.

R14. The system allows GitHub to create an automated workflow connected to the CKB platform.

R15. The system allows GitHub to send submitted solutions to the CKB platform.

R16. The system automatically evaluates every proposed solution with reference to the specified parameters.

R17. The system allows educators to manually evaluate solutions at the end of the battle, if specified.

R18. The system notifies students registered to a tournament when a new battle belonging to that tournament is created.

R19. The system notifies students when a battle they are registered to ends.

R20. The system notifies students when a tournament they are registered to ends.

R21. The system allows educators to close tournaments they created.

R22. The system allows users involved in a battle to consult the partial score of that battle.

R23. The system allows users to consult the ranking of a tournament.

R24. The system updates the ranking of a tournament at the end of each battle belonging to that tournament.

R25. The system doesn't allow a tournament to be closed if a battle is still ongoing.

R26. The system doesn't allow to create a tournament with the same name of an already existing one.

R27. The system automatically declines unanswered invitations that are older than 2 days.

R28. When a student answers an invitation the system automatically declines all the other invitations sent to that student.

R29. The system doesn't allow students to subscribe to tournaments or battle after the registration deadline.

R30. The system cannot receive submitted solutions after the deadline.

3.2.1 Use-Case Diagrams

In the following section use-case diagrams are provided, the diagrams give a brief overview of the different uses that an user, educator or student, can make of the CKB system.

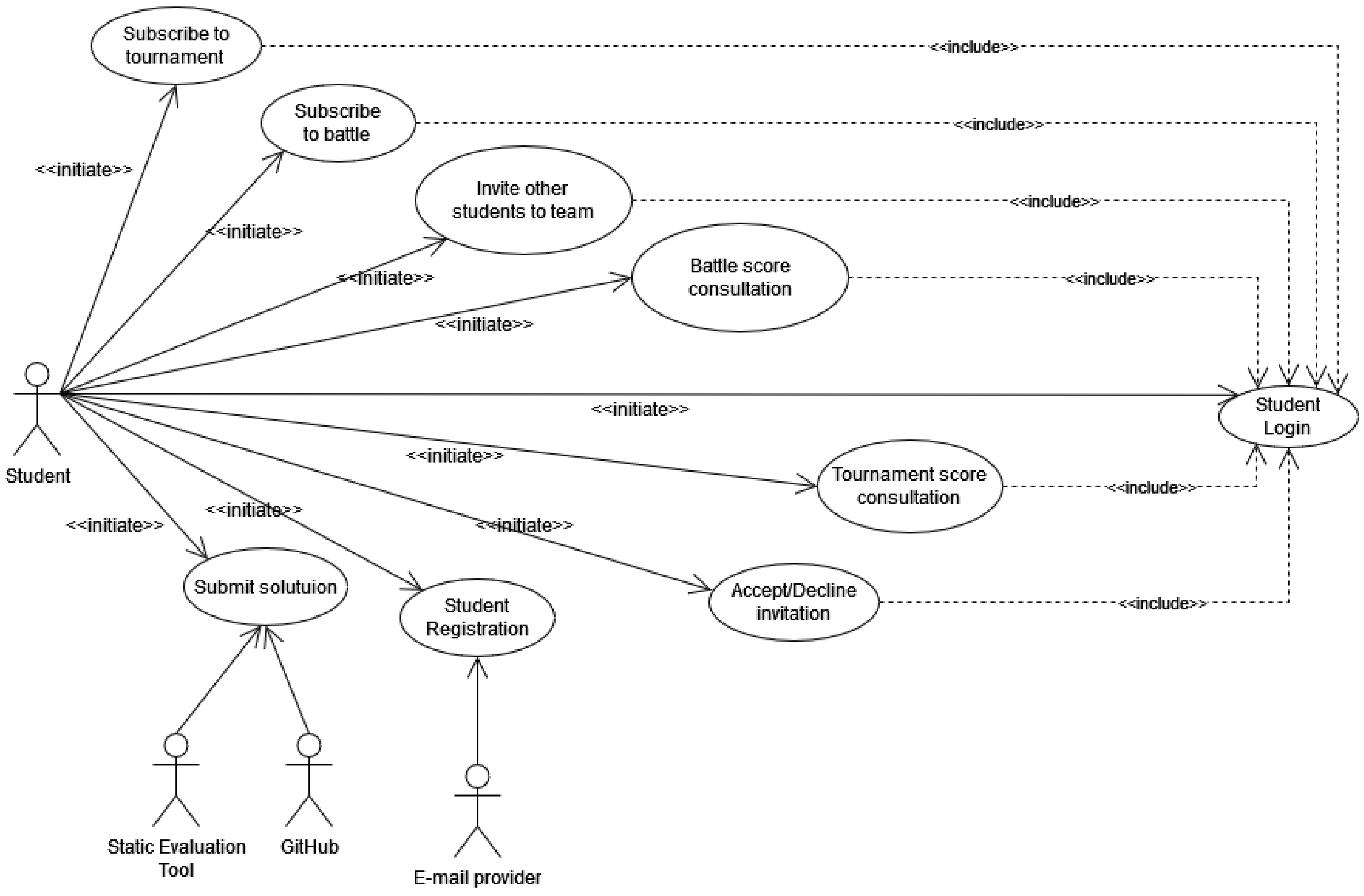


Figure 9: Use Case Diagram for the student class of users

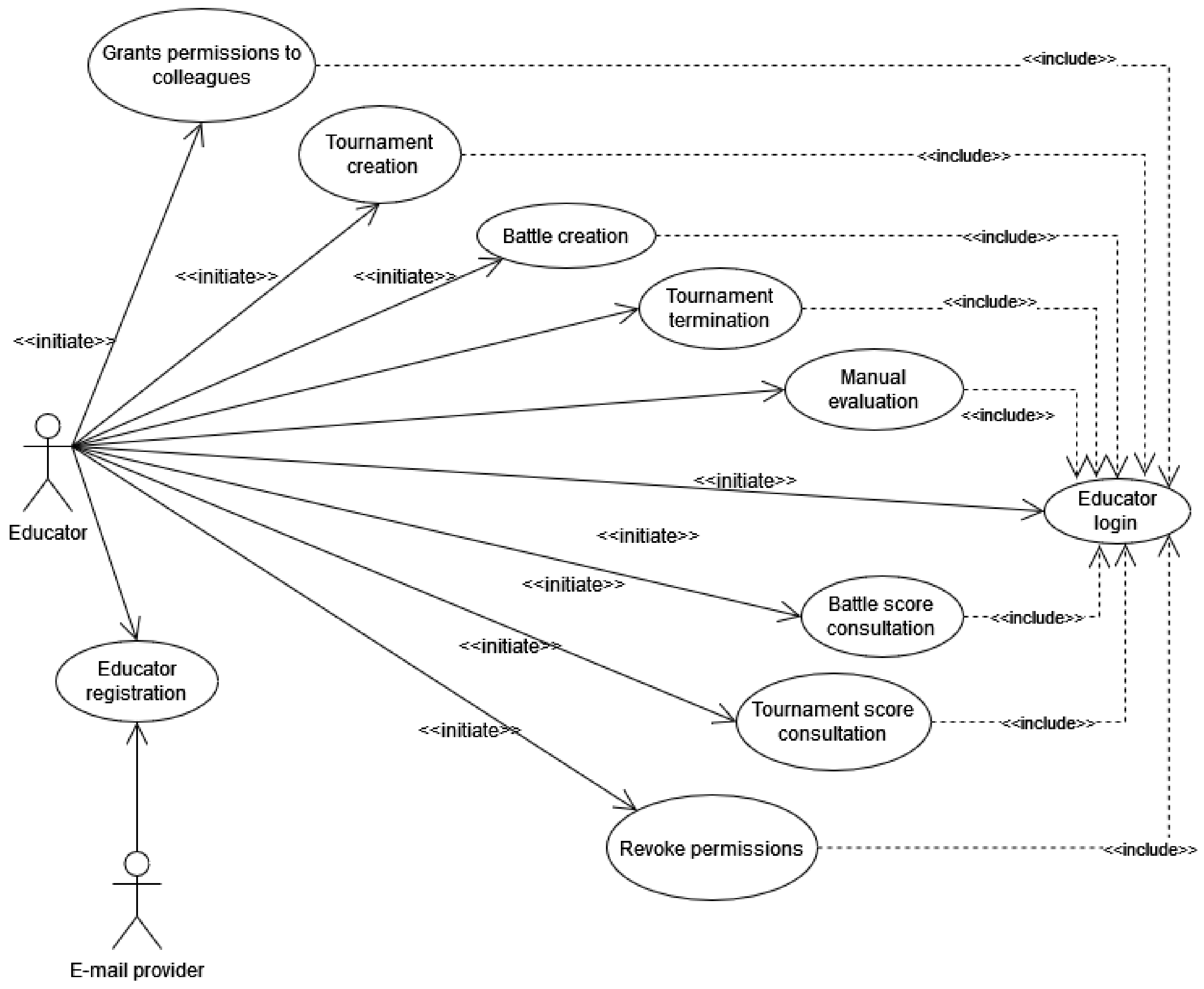


Figure 10: Use Case Diagram for the educator class of users

3.2.2 Use Cases

The following section provides a in-depth description of the use cases defined in the preceding diagrams and scenarios.

[UC1]. Student registration

Name	Student registers to CKB
Actors	Student, e-mail provider
Entry Conditions	The student is not already registered into the system
Event Flow	1. The student clicks the student registration button in the homepage of CKB 2. The system provides the form in which to insert the account information 3. The student inserts an email and a password to create the account 4. The student inserts additional anagraphic data 5. The system sends a verification e-mail to the specified address 6. The user verifies the account creation with the e-mail
Exit Conditions	The system creates the account
Exception	If the user does not verify the account through the e-mail, the account is not created

[UC2]. Education registration

Name	Educator registers to CKB
Actors	Educator, e-mail provider
Entry Conditions	The educator is not already registered into the system
Event Flow	1. The educator clicks the educator registration button in the homepage of CKB 2. The system provides the form in which to insert the account information 3. The educator inserts an email and a password to create the account 4. The educator inserts additional anagraphic data 5. The system sends a verification e-mail to the specified address 6. The user verifies the account creation with the e-mail
Exit Conditions	The system creates the account
Exception	If the user does not verify the account through the e-mail, the account is not created

[UC3]. User login

Name	User logs in CKB
Actors	User (Student or Educator)
Entry Conditions	The user is already registered to CKB
Event Flow	1. The user clicks the login button in the homepage of CKB 2. The system provides the login form 3. The user inserts the login credentials 4. The system checks the credentials
Exit Conditions	The system grants the user access to the account
Exception	If the login credentials are faulty the system does not provide access to the user

[UC4.] Tournament creation

Name	Educator creates a new tournament
Actors	Edcuator
Entry Condition	The educator is logged into CKB
Event Flow	1. The educator clicks the ”Create New Tournament” button in the ”my tournaments” page of his account on CKB 2. The system provides the form to insert the main information about the tournament 3. The educator inserts name of the new tournament and its details and clicks on the ”Create tournament” button 4. The system checks if the name of the tournament is valid
Exit Conditions	The system creates the tournament and notifies every student registered to CKB
Exception	If the name is already in use by another tournament the system does not allow the creation of the new tournament

[UC5.] Permission granting

Name	Educator grants permission to a colleague to manage a tournament
Actors	Edcuator
Entry Conditions	he educator is logged into CKB The educator has already created the tournament T
Event Flow	1. The educator goes to the ”My tournaments” section in the CKB online platform 2. The system provides the list of the tournament managed by the educator 3. The educator selects the tournament T from the list provided by the UI 4. The system provides the data regarding the selected tournament 5. The educator clicks the ”Manage permissions” button on the UI 6. The educator selects the ”Add permission” option 7. The system asks the educator to insert the e-mail with which the colleague is registered in the CKB system 8. The educator inputs the e-mail and clicks the ”Save permissions” button 9. The system checks the e-mail.
Exit Conditions	The system provides the authorization to the specified account and notifies the educator of the permission change
Exception	If the inserted e-mail is not valid the system does not allow the educator to save the permissions

[UC6.] Permission revoking

Name	An educator revokes the permissions from a colleague
Actors	Edcuator
Entry Conditions	The educator is logged into CKB The educator has already created the tournament T The educator has previously granted the permissions to the colleague
Event Flow	1. The educator goes to the ”My tournaments” section in the CKB online platform 2. The system provides the list of the tournament managed by the educator 3. The educator selects the tournament T from the list provided by the UI 4. The system provides the data regarding the selected tournament 5. The educator clicks the ”Manage permissions” button on the UI 6. The educator selects the ”Delete permission” option 7. The system provides the list of all the educators with permissions for tournament T 8. The educator selects the colleague and presses the ”Save” button
Exit Conditions	The system revokes the authorization from the specified account and notifies the educator

[UC7.] Battle creation

Name	An educator creates a battle for a tournament
Actors	Edcuator
Entry Conditions	The educator is logged into CKB The educator is either the creator of the tournament or has the permission to create battles for it
Event Flow	1. The educator clicks the ”My tournaments” button in the CKB online platform 2. The system provides the list of the tournament managed by the educator 3. The educator selects the tournament from the list provided by the UI 4. The system provides the data regarding the selected tournament 5. The educator clicks the ”New battle” button on the UI 6. The system provides the form in which to insert the information regarding the battle 7. The educator provides the requested information 8. The systems checks the validity of the information
Exit Conditions	The system creates the new battle and notifies all the student involved in the tournament
Exception	The educator inserts invalid information, the system does not allow the battle to be created

[UC8.] Subscribing to tournament

Name	A student subscribes to a tournament
Actors	Student
Entry Conditions	The student is logged into CKB
Event Flow	1. The students enters the ”All tournaments” section of the UI 2. The system provides a list of all the available tournaments 3. The students selects the tournament to subscribe to 4. The system provides the tournament information through the UI 5. The student clicks the ”Subscribe” button
Exit Conditions	The system allows the student to participate to the selected tournament

[UC9.] Subscribing to a battle

Name	A student subscribes to a battle on his own
Actors	Student
Entry Conditions	The student is logged into CKB The student is already subscribed to the tournament the battle belongs to
Event Flow	1. The students enters the ”My tournaments” section of the UI 2. The system provides a list of all the tournaments the student is subscribed to 3. The students selects the tournament the battle belongs to 4. The system provides the tournament information 5. The student selects the battle to subscribe to 6. The system provides the battle information through the UI 7. The student clicks the ”Subscribe on my own” button
Exit Conditions	The system allows the student to participate to the selected battle

[UC10.] Subscribing a team to a battle

Name	A student subscribes to a battle with a team
Actors	Student
Entry Conditions	The student is logged into CKB The student is already subscribed to the tournament the battle belongs to
Event Flow	1. The students enters the ”My tournaments” section of the UI 2. The system provides a list of all the tournaments the student is subscribed to 3. The students selects the tournament the battle belongs to 4. The system provides the tournament information 5. The student selects the battle to subscribe to 6. The system provides the battle information through the UI 7. The student clicks the ”Subscribe with a team” button 8. The system asks the student the e-mail addresses of the teammates 9. The students provides the addresses 10. The system checks the addresses and sends the invitation(s)
Exit Conditions	Once a team has all members confirmed it is subscribed to the battle, otherwise after the registration deadline is reached every team with a sufficient number of confirmed student is subscribed.
Exception	If the e-mail addresses are not valid the system does not allow the student to continue with the registration

[UC11.] Answering to an invitation

Name	A student answers to an invitation
Actors	Student
Entry Conditions	The student is logged into CKB The student is already subscribed to the tournament the battle belongs to
Event Flow	1. The students enters the ”My messages” section of the UI 2. The system provides a list of all the notification received by the account 3. The students selects the invitation message 4. The system displays the invitation message
Alternative 1	5. The student clicks on the ”accept” button
Alternative 2	5. The student click on the ”decline” button
Exit Condition	The system registers the answer
Exception	If the e-mail addresses are not valid the system does not allow the student to continue with the registration

[UC12.] Submitting a solution

Name	A student submits a solution
Actors	Student, GitHub, SET
Entry Conditions	The student is logged into GitHub The student is already subscribed to the a battle and to the corresponding tournament
Event Flow	1. The students pushes a solution on GitHub 2. Through the automated workflow GitHub forwards the submitted code to CKB 3. CKB sends the solution to the SET 4. The SET evaluates the solution according to the parameters specified by the creator of the battle 5. The SET sends the results back to CKB
Exit Condition	The system updates the partial score and notifies the students of the team
Exception	If the code is not executable (e.g. compilation error) CKB refuses to evaluate the submission

[UC13.] Checking the battle's ranking

Name	An user checks the partial scores of a battle
Actors	User (Student or Educator)
Entry Conditions	The user is logged into CKB The user is either subscribed to the battle or the creator of the battle
Event Flow	1. The user selects the "My battles" button on the CKB UI 2. The system displays the battles in which the user is involved 3. The user selects the battle 4. The system displays the battle's information
Exit Condition	The user can check the current ranking of the battle

[UC14.] Checking the tournament's ranking

Name	An user checks the tankings of a tournament
Actors	User (Student or Educator)
Entry Conditions	The user is logged into CKB The user is either subscribed to the tournament or the creator of the tournament
Event Flow	1. The user selects the "My tournaments" button on the CKB UI 2. The system displays the tournaments in which the user is involved 3. The user selects the tournament 4. The system displays the tournament's information
Exit Condition	The user can check the current ranking of the tournament

[UC15.] Closing a tournament

Name	An educator closes a tournament
Actors	Educator
Entry Conditions	The educator is logged into CKB The user is the creator of the tournament
Event Flow	1. The educator selects the "My tournaments" button on the CKB UI 2. The system displays the tournaments managed by the educator 3. The educator selects the tournament 4. The system displays the tournament's information 5. The educator clicks on the "Close tournament" button
Exit Condition	The system closes the tournament and sends a notification to all of its participants

[UC16.] Manually evaluating a battle

Name	An educator manually evaluates the solutions of a battle
Actors	Educator
Entry Conditions	The educator is logged into CKB The user is the creator of the battle The battle must have ended
Event Flow	1. The educator selects the ”My battles” button on the CKB UI 2. The system displays the battles managed by the educator 3. The educator selects the battle to evaluate 4. The system displays the battle’s information 5. The educator clicks on the ”Evaluate” button 6. The educator inserts the evaluation for every solution 7. The educator clicks the ”Submit evaluation” button
Exit Condition	The system registers the evaluation and updates the final scores of the battle

3.2.3 Sequence Diagrams

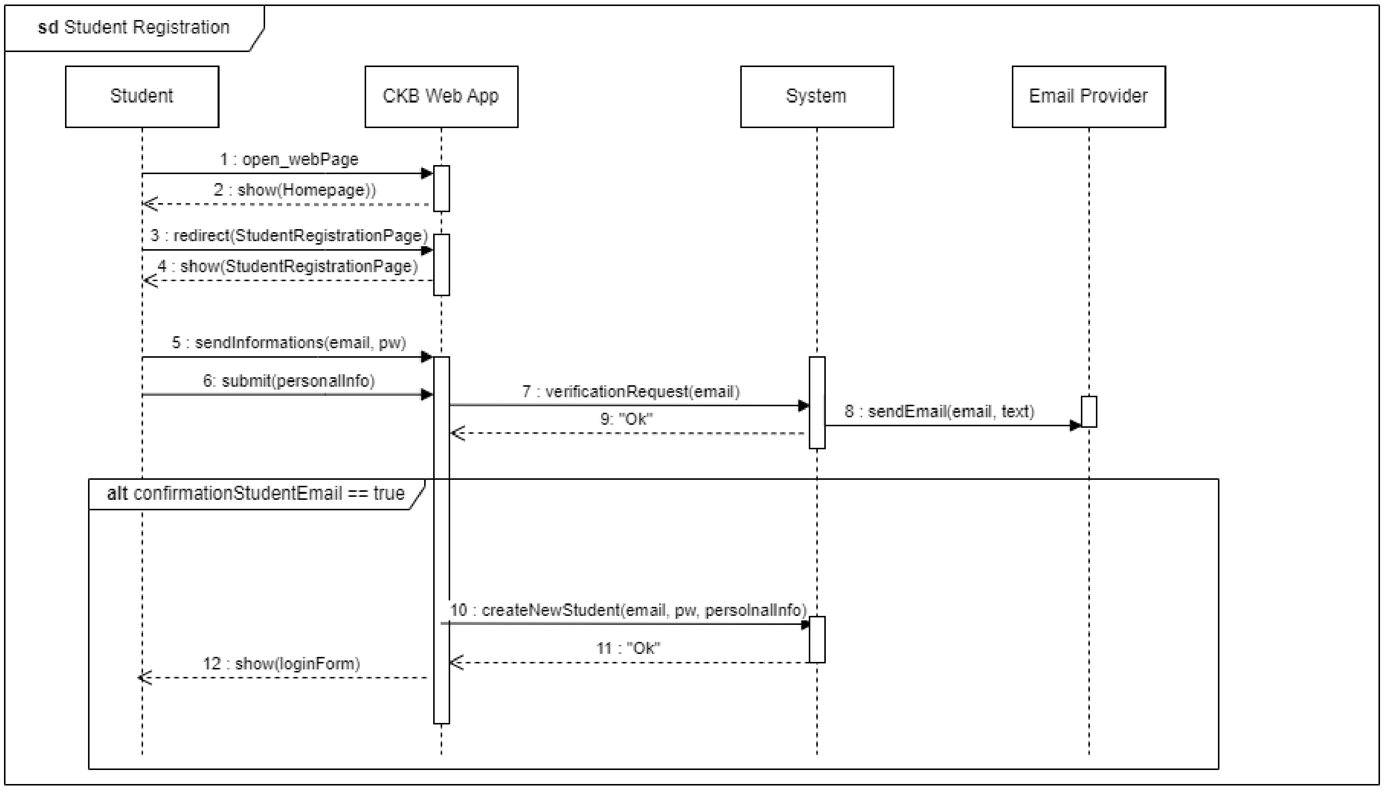

sd Student Registration
Student
CKB Web App
System
Email Provider
1 : open_webPage
2 : show(Homepage))
3 : redirect(StudentRegistrationPage)
4 : show(StudentRegistrationPage)
5 : sendInformations(email, pw)
6: submit(personalInfo)
7 : verificationRequest(email)
8 : sendEmail(email, text)
9: "Ok"
alt confirmationStudentEmail == true
10 : createNewStudent(email, pw, persolnalInfo)
11 : "Ok"
12 : show(loginForm)

Figure 11: Sequence diagram for [**UC1.**]

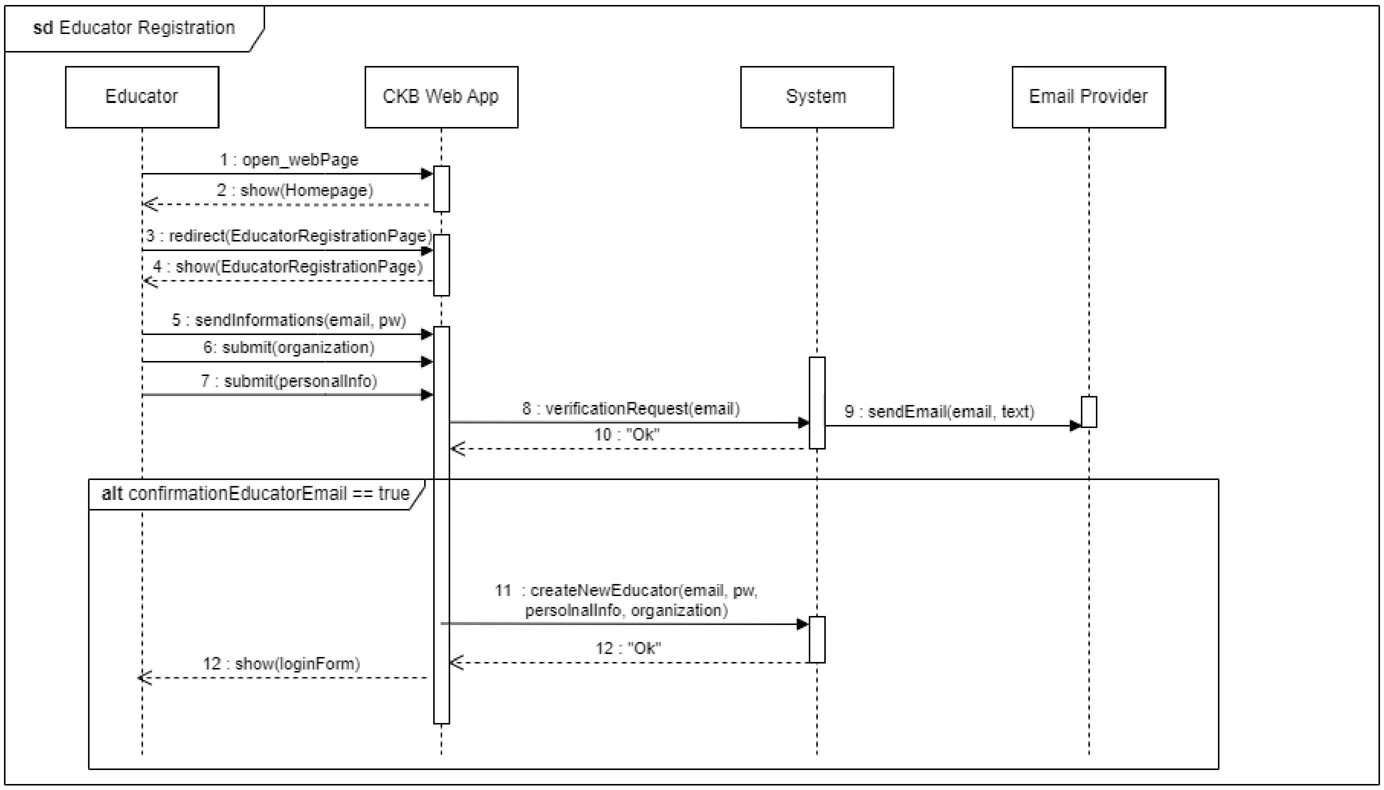

sd Educator Registration
Educator
CKB Web App
System
Email Provider
1 : open_webPage
2 : show(Homepage)
3 : redirect(EducatorRegistrationPage)
4 : show(EducatorRegistrationPage)
5 : sendInformations(email, pw)
6: submit(organization)
7 : submit(personalInfo)
8 : verificationRequest(email)
9 : sendEmail(email, text)
10 : "Ok"
alt confirmationEducatorEmail == true
11 : createNewEducator(email, pw, persolnalInfo, organization)
12 : "Ok"
12 : show(loginForm)

Figure 12: Sequence diagram for [**UC2.**]

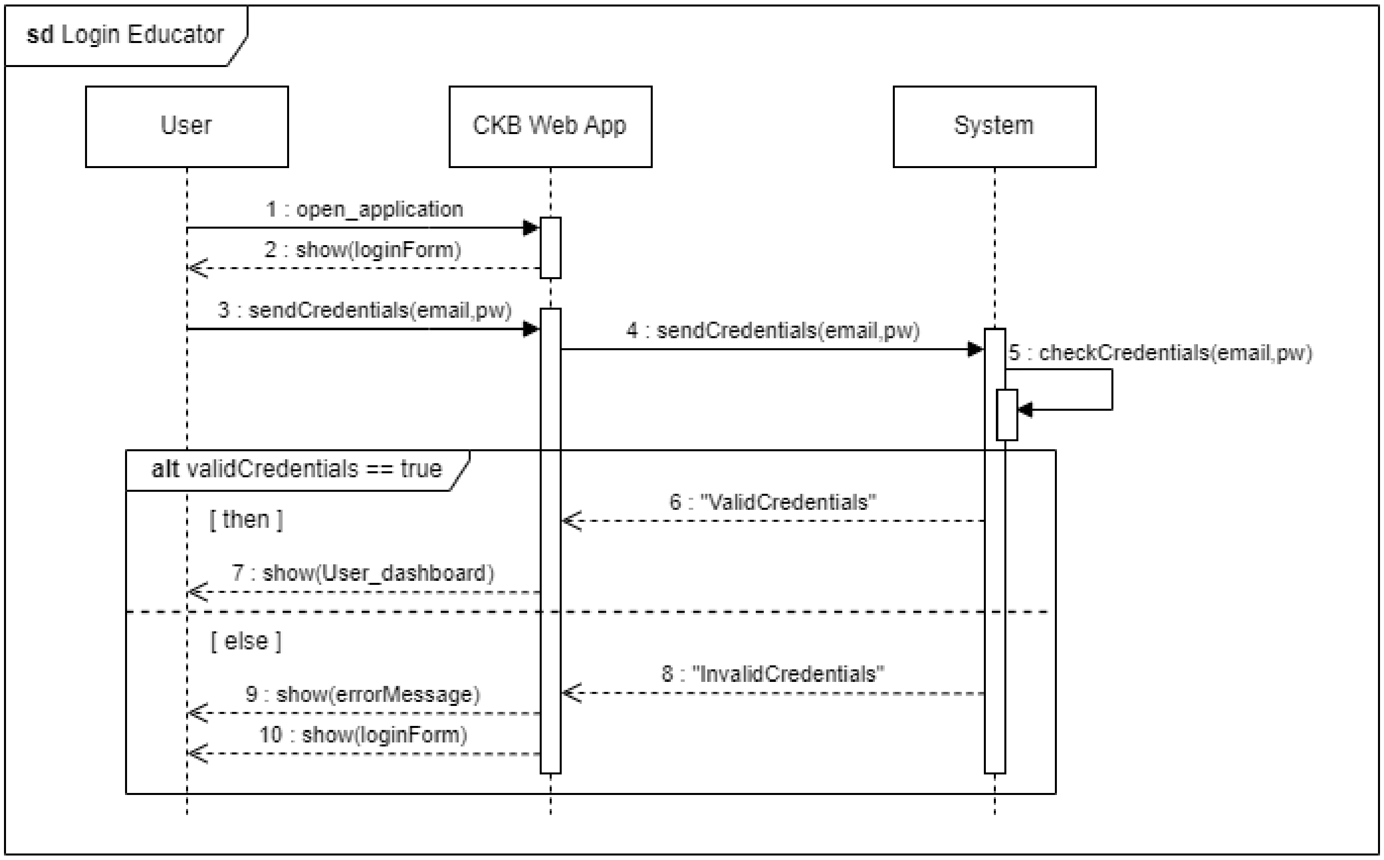


Figure 13: Sequence diagram for [**UC3.**]

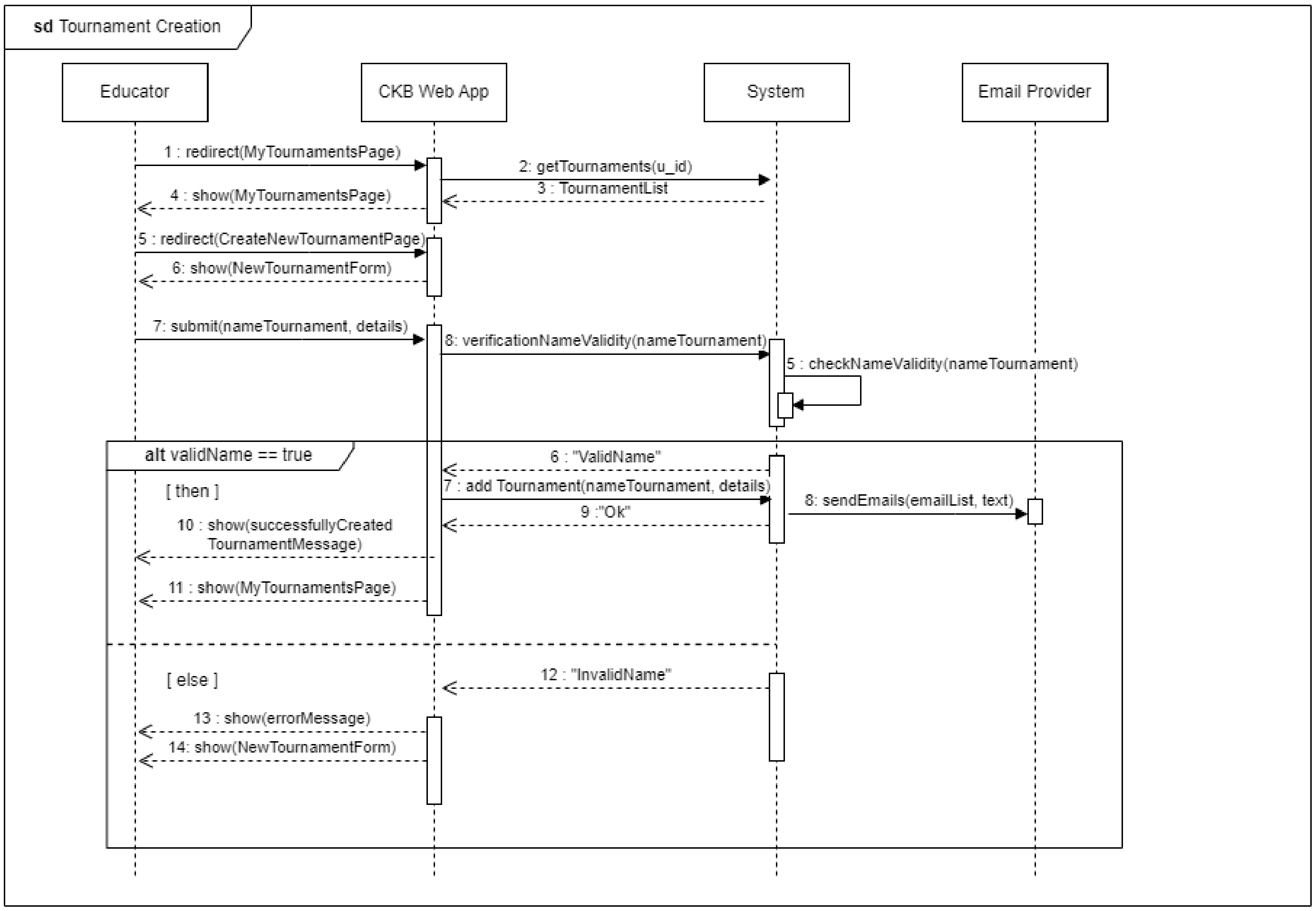

sd Tournament Creation
Educator
CKB Web App
System
Email Provider
1 : redirect(MyTournamentsPage)
2: getTournaments(u_id)
3 : TournamentList
4 : show(MyTournamentsPage)
5 : redirect(CreateNewTournamentPage)
6: show(NewTournamentForm)
7: submit(nameTournament, details)
8: verificationNameValidity(nameTournament)
5 : checkNameValidity(nameTournament)
alt validName == true
[then]
6 : "ValidName"
7 : add Tournament(nameTournament, details)
8: sendEmails(emailList, text)
9 :"Ok"
10 : show(successfullyCreated TournamentMessage)
11 : show(MyTournamentsPage)
[else]
12 : "InvalidName"
13 : show(errorMessage)
14: show(NewTournamentForm)

Figure 14: Sequence diagram for [**UC4.**]

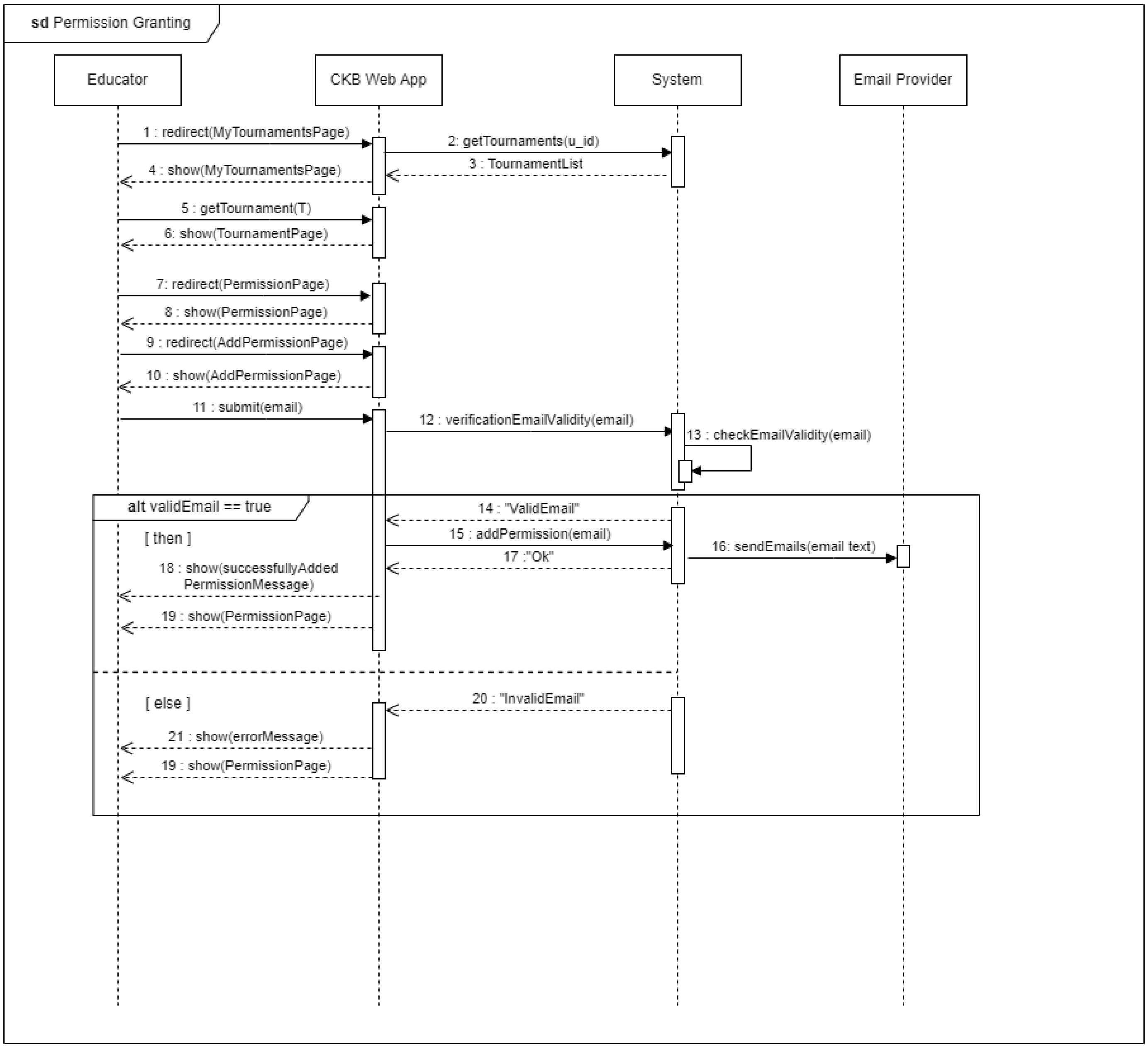

sd Permission Granting
Educator
CKB Web App
System
Email Provider
1 : redirect(MyTournamentsPage)
2: getTournaments(u_id)
3 : TournamentList
4 : show(MyTournamentsPage)
5 : getTournament(T)
6: show(TournamentPage)
7: redirect(PermissionPage)
8 : show(PermissionPage)
9 : redirect(AddPermissionPage)
10 : show(AddPermissionPage)
11 : submit(email)
12 : verificationEmailValidity(email)
13 : checkEmailValidity(email)
alt validEmail == true
[then]
14 : "ValidEmail"
15 : addPermission(email)
16: sendEmails(email text)
17 :"Ok"
18 : show(successfullyAdded PermissionMessage)
19 : show(PermissionPage)
[else]
20 : "InvalidEmail"
21 : show(errorMessage)
19 : show(PermissionPage)

Figure 15: Sequence diagram for [**UC5.**]

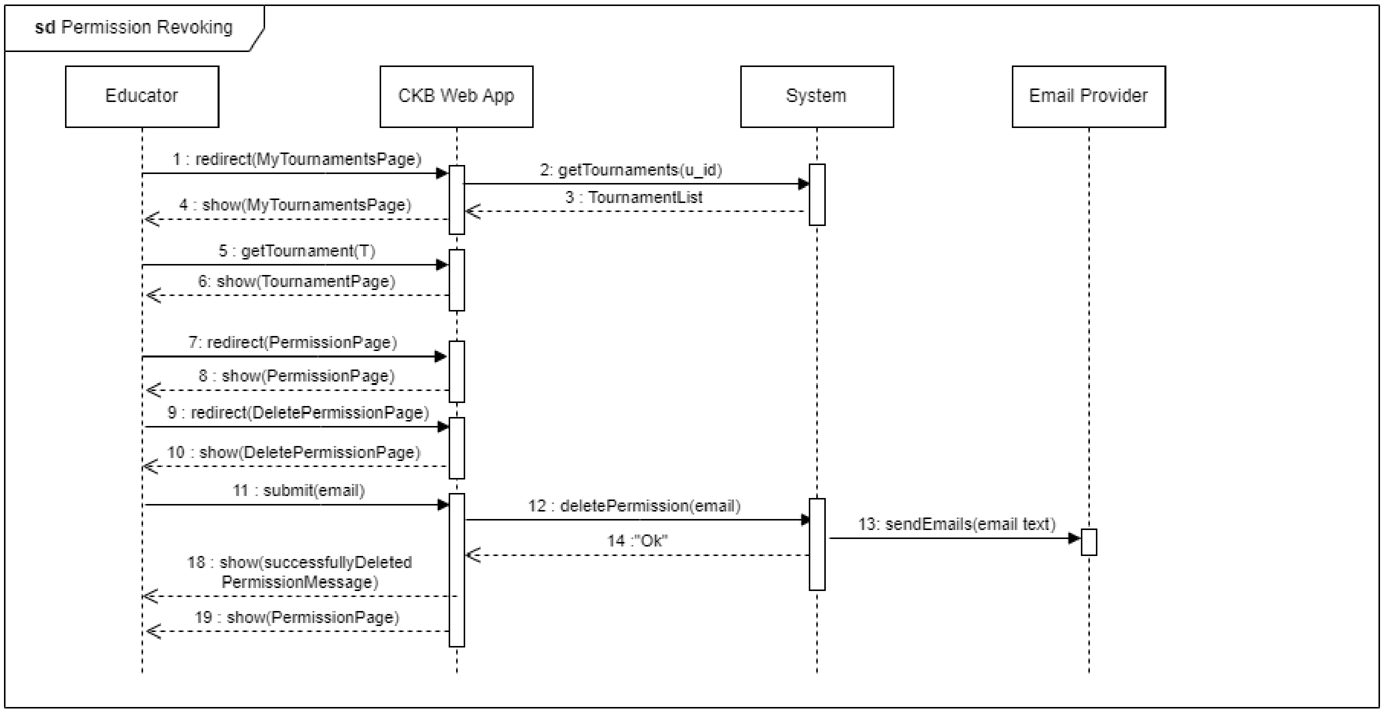

sd Permission Revoking
Educator
CKB Web App
System
Email Provider
1 : redirect(MyTournamentsPage)
2: getTournaments(u_id)
3 : TournamentList
4 : show(MyTournamentsPage)
5 : getTournament(T)
6: show(TournamentPage)
7: redirect(PermissionPage)
8 : show(PermissionPage)
9 : redirect(DeletePermissionPage)
10 : show(DeletePermissionPage)
11 : submit(email)
12 : deletePermission(email)
13: sendEmails(email text)
14 :"Ok"
18 : show(successfullyDeleted PermissionMessage)
19 : show(PermissionPage)

Figure 16: Sequence diagram for [**UC6.**]

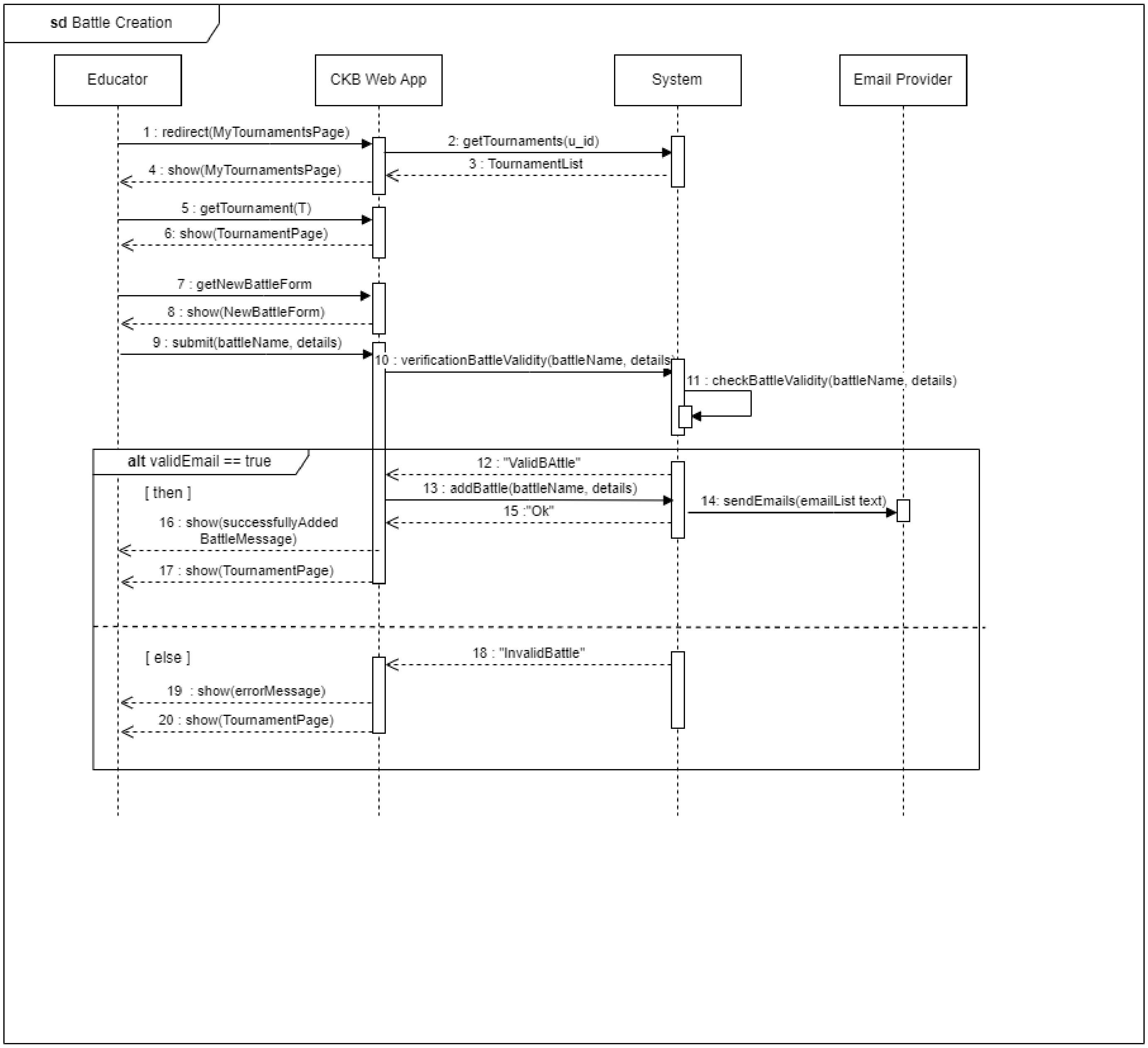

sd Battle Creation
Educator
CKB Web App
System
Email Provider
1 : redirect(MyTournamentsPage)
2: getTournaments(u_id)
3 : TournamentList
4 : show(MyTournamentsPage)
5 : getTournament(T)
6: show(TournamentPage)
7 : getNewBattleForm
8 : show(NewBattleForm)
9 : submit(battleName, details)
10 : verificationBattleValidity(battleName, details)
11 : checkBattleValidity(battleName, details)
alt validEmail == true
[then]
12 : "ValidBAttle"
13 : addBattle(battleName, details)
14: sendEmails(emailList text)
15 :"Ok"
16 : show(successfullyAdded BattleMessage)
17 : show(TournamentPage)
[else]
18 : "InvalidBattle"
19 : show(errorMessage)
20 : show(TournamentPage)

Figure 17: Sequence diagram for [**UC7.**]

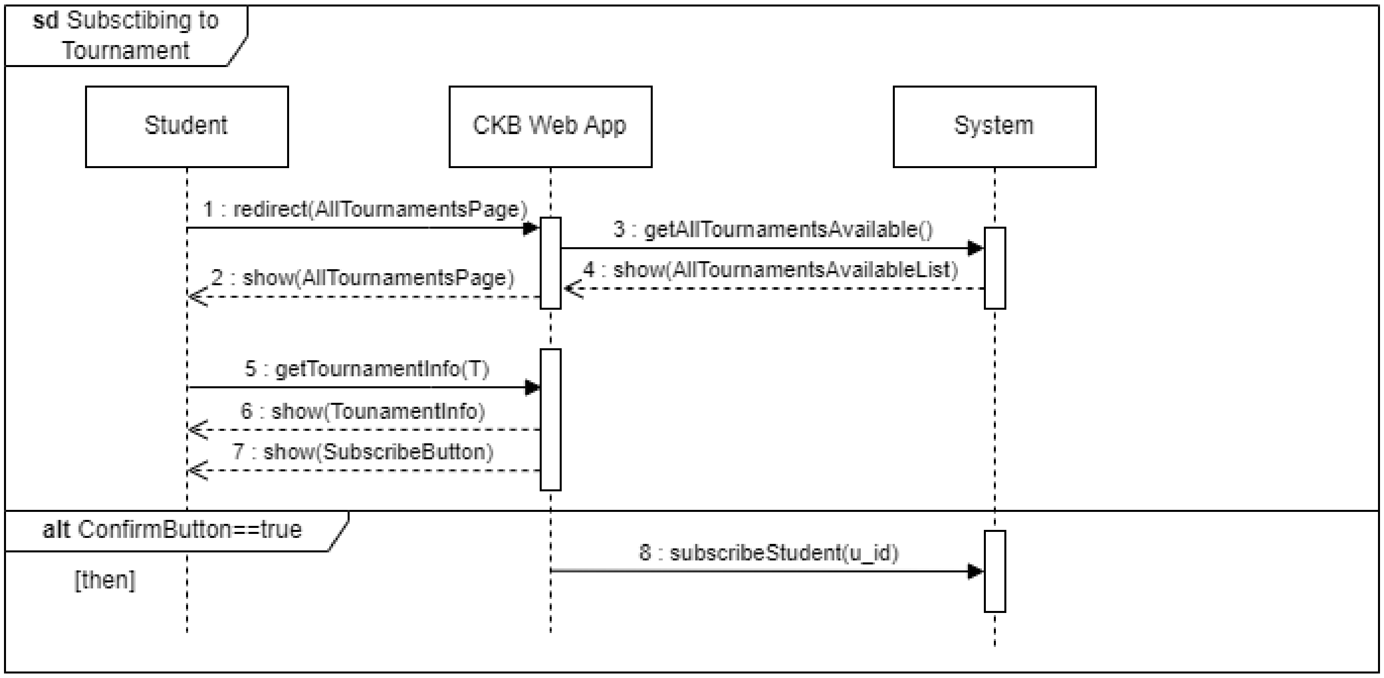

sd Subsctibing to Tournament
Student
CKB Web App
System
1 : redirect(AllTournamentsPage)
3 : getAllTournamentsAvailable()
4 : show(AllTournamentsAvailableList)
2 : show(AllTournamentsPage)
5 : getTournamentInfo(T)
6 : show(TounamentInfo)
7 : show(SubscribeButton)
alt ConfirmButton==true
[then]
8 : subscribeStudent(u_id)

Figure 18: Sequence diagram for [**UC8.**]

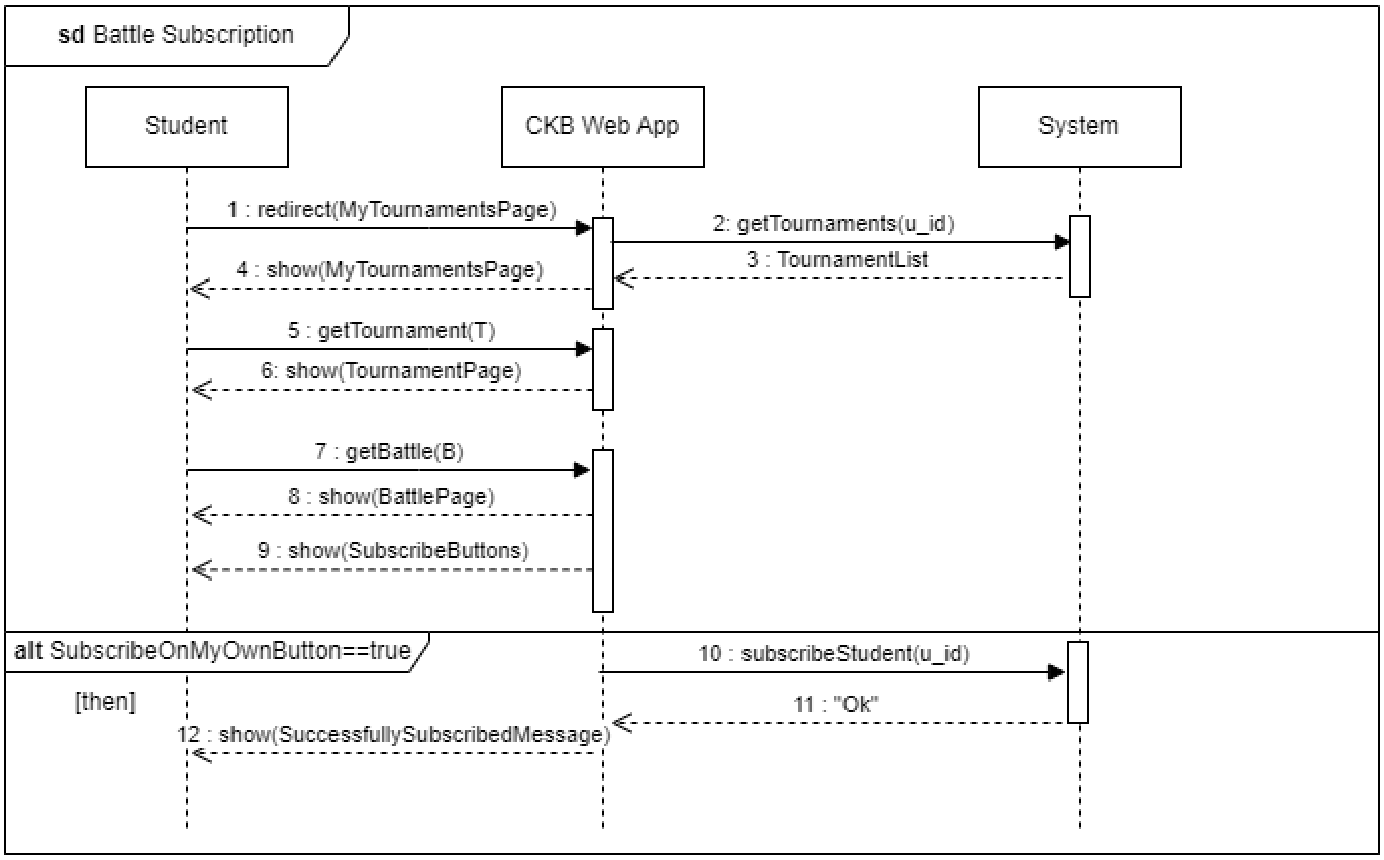

sd Battle Subscription
Student
CKB Web App
System
1 : redirect(MyTournamentsPage)
2: getTournaments(u_id)
3 : TournamentList
4 : show(MyTournamentsPage)
5 : getTournament(T)
6: show(TournamentPage)
7 : getBattle(B)
8 : show(BattlePage)
9 : show(SubscribeButtons)
alt SubscribeOnMyOwnButton==true
[then]
10 : subscribeStudent(u_id)
11 : "Ok"
12 : show(SuccessfullySubscribedMessage)

Figure 19: Sequence diagram for [**UC9.**]

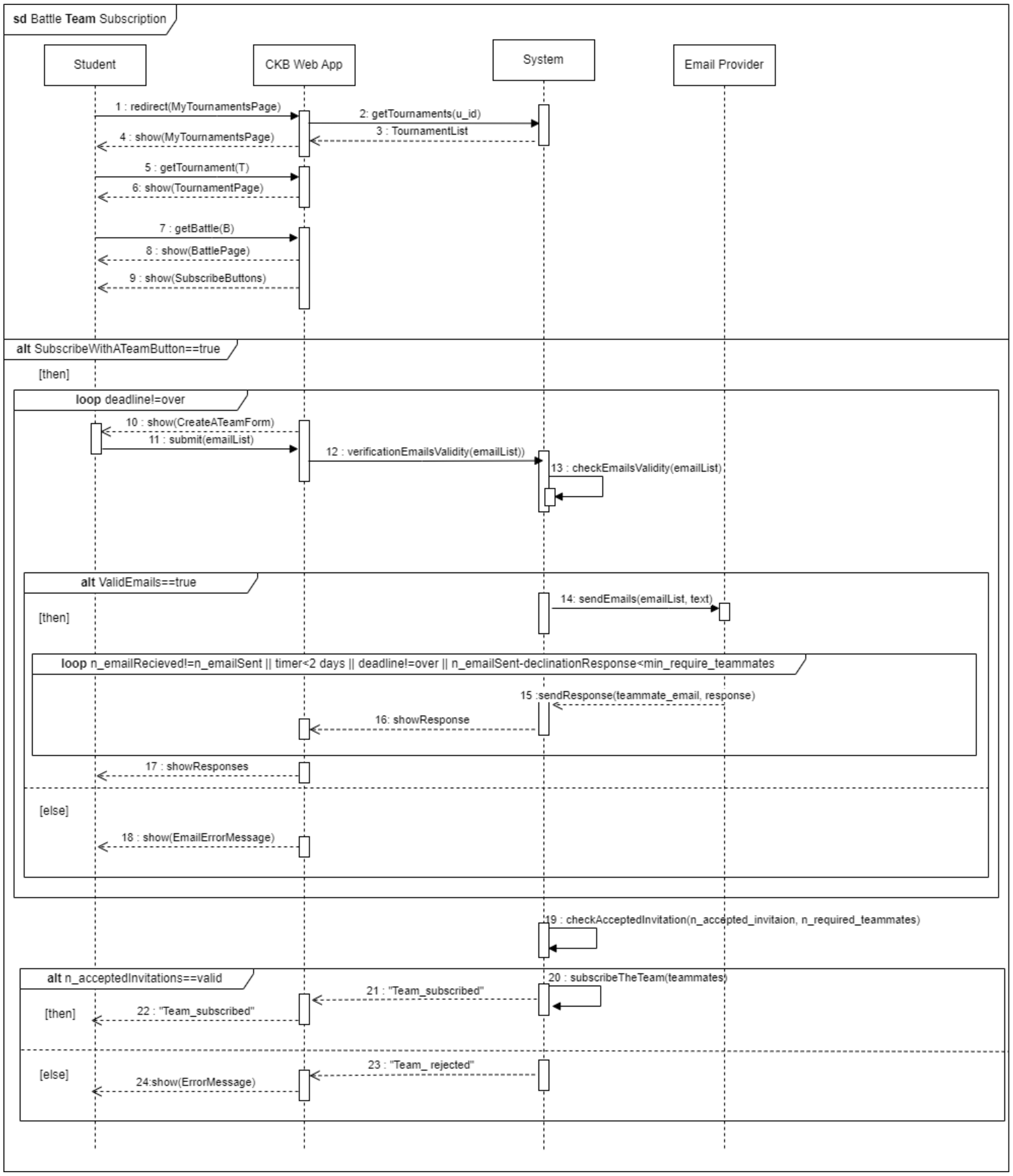

sd Battle Team Subscription
Student
CKB Web App
System
Email Provider
1 : redirect(MyTournamentsPage)
2: getTournaments(u_id)
3 : TournamentList
4 : show(MyTournamentsPage)
5 : getTournament(T)
6: show(TournamentPage)
7 : getBattle(B)
8 : show(BattlePage)
9 : show(SubscribeButtons)
alt SubscribeWithATeamButton==true
[then]
loop deadline!=over
10 : show(CreateATeamForm)
11 : submit(emailList)
12 : verificationEmailsValidity(emailList))
13 : checkEmailsValidity(emailList)
alt ValidEmails==true
[then]
14: sendEmails(emailList, text)
loop n_emailRecieved!=n_emailSent || timer<2 days || deadline!=over || n_emailSent-declinationResponse<min_require_teammates
15 :sendResponse(teammate_email, response)
16: showResponse
17 : showResponses
[else]
18 : show(EmailErrorMessage)
19 : checkAcceptedInvitation(n_accepted_invitaion, n_required_teammates)
alt n_acceptedInvitations==valid
[then]
20 : subscribeTheTeam(teammates)
21 : "Team_subscribed"
22 : "Team_subscribed"
[else]
23 : "Team_ rejected"
24:show(ErrorMessage)

Figure 20: Sequence diagram for [**UC10.**]

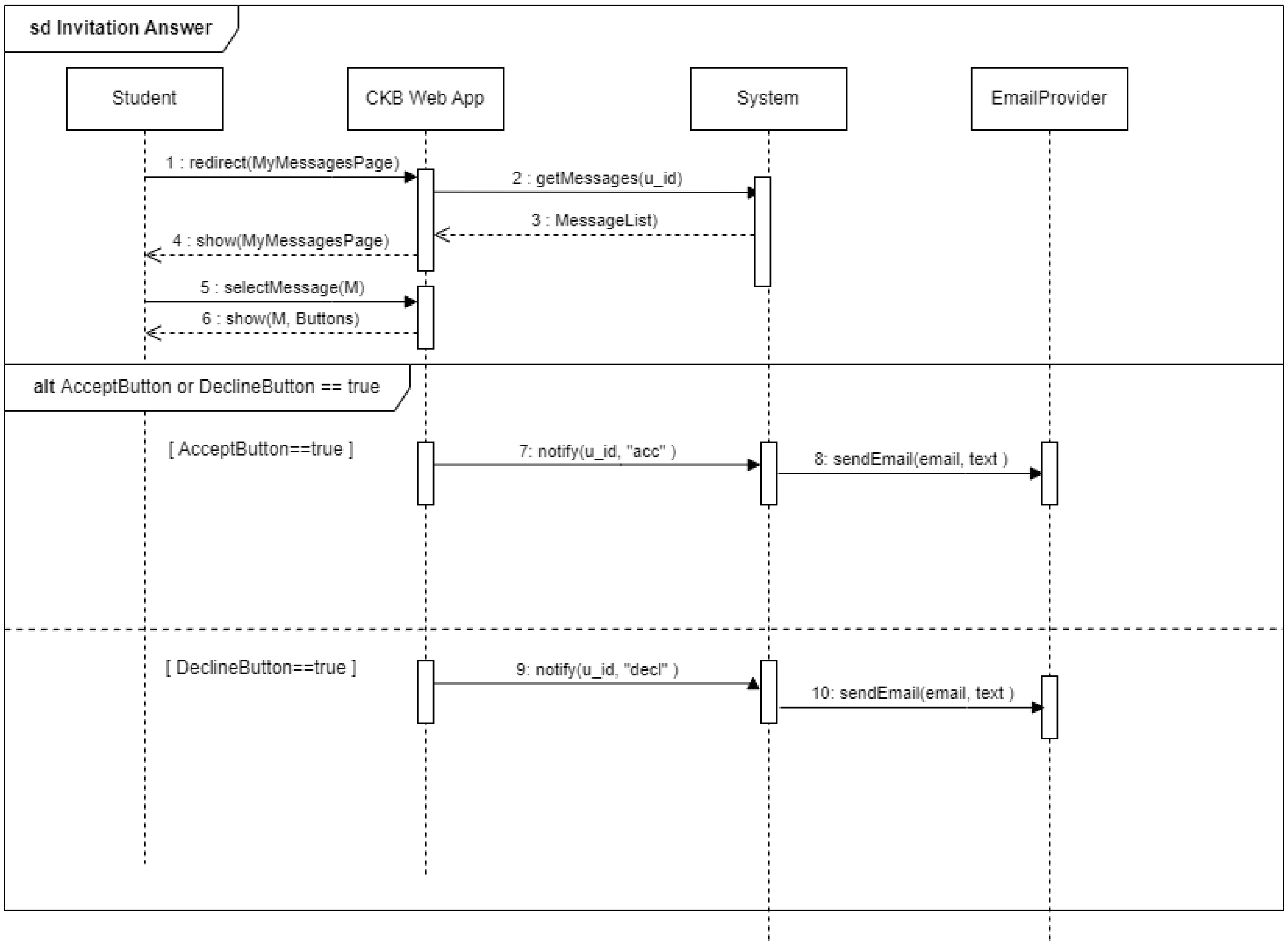


Figure 21: Sequence diagram for [**UC11.**]

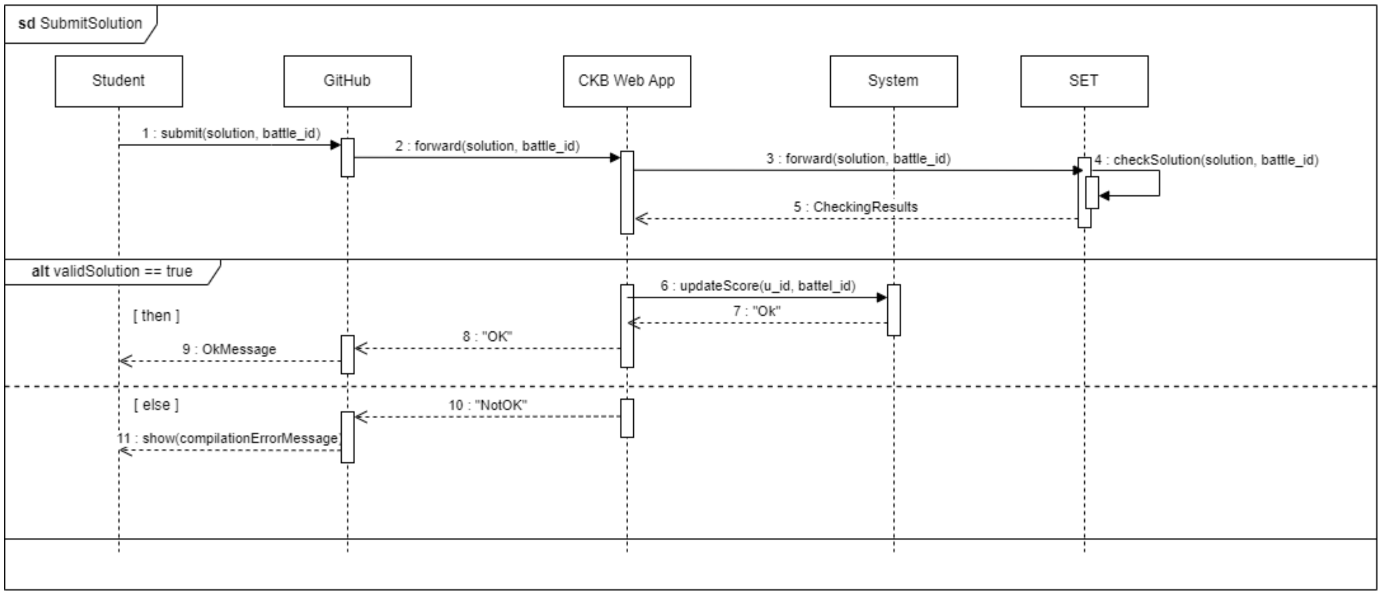

sd SubmitSolution
Student
GitHub
CKB Web App
System
SET
1 : submit(solution, battle_id)
2 : forward(solution, battle_id)
3 : forward(solution, battle_id)
4 : checkSolution(solution, battle_id)
5 : CheckingResults
alt validSolution == true
[then]
6 : updateScore(u_id, battel_id)
7 : "Ok"
8 : "OK"
9 : OkMessage
[else]
10 : "NotOK"
11 : show(compilationErrorMessage)

Figure 22: Sequence diagram for [**UC12.**]

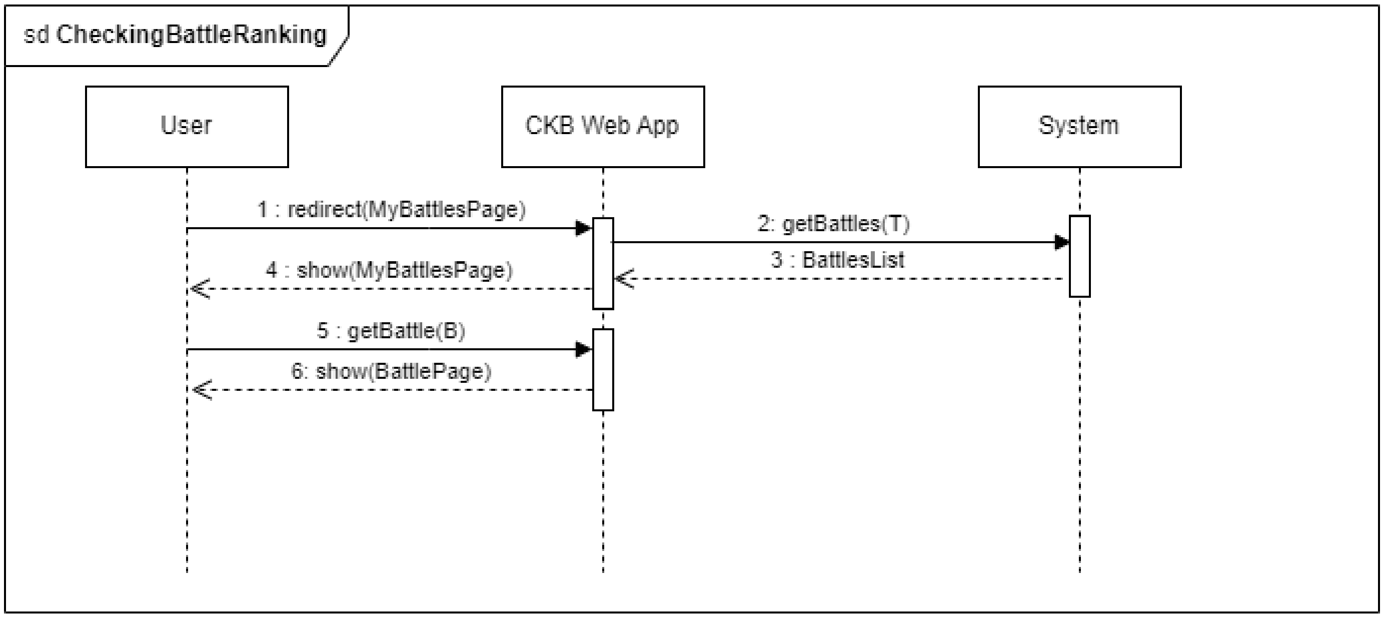


Figure 23: Sequence diagram for [**UC13.**]

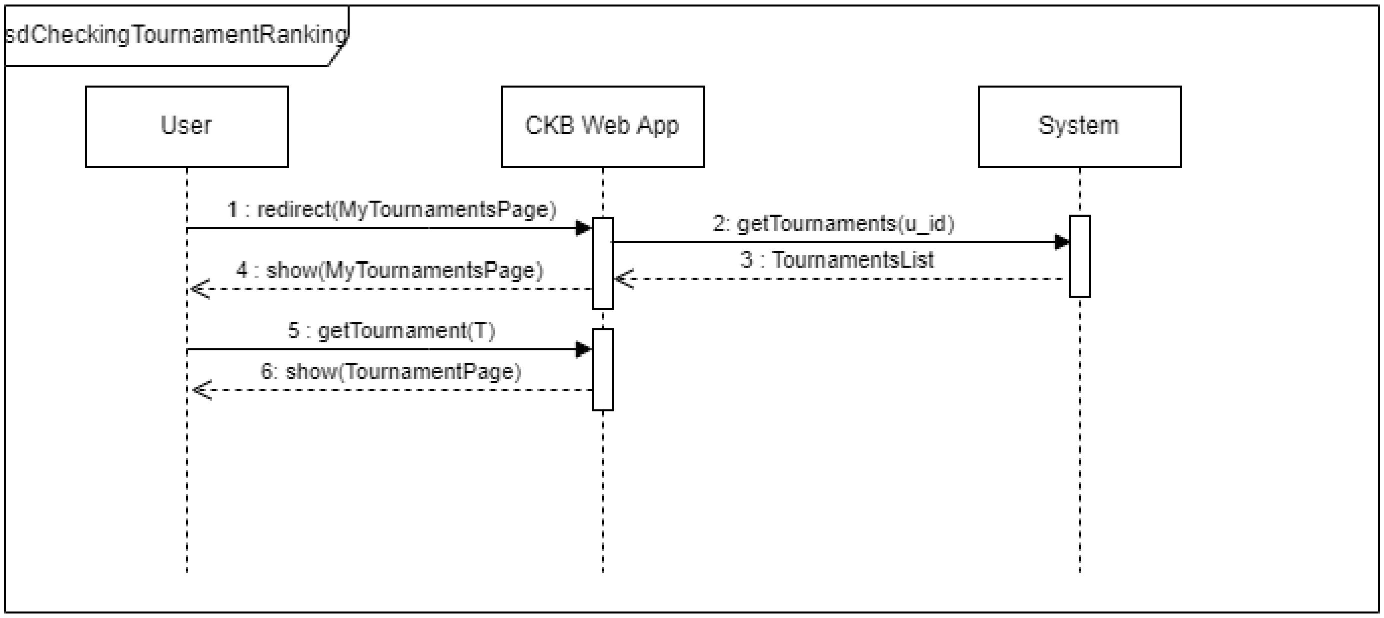


Figure 24: Sequence diagram for [**UC14.**]

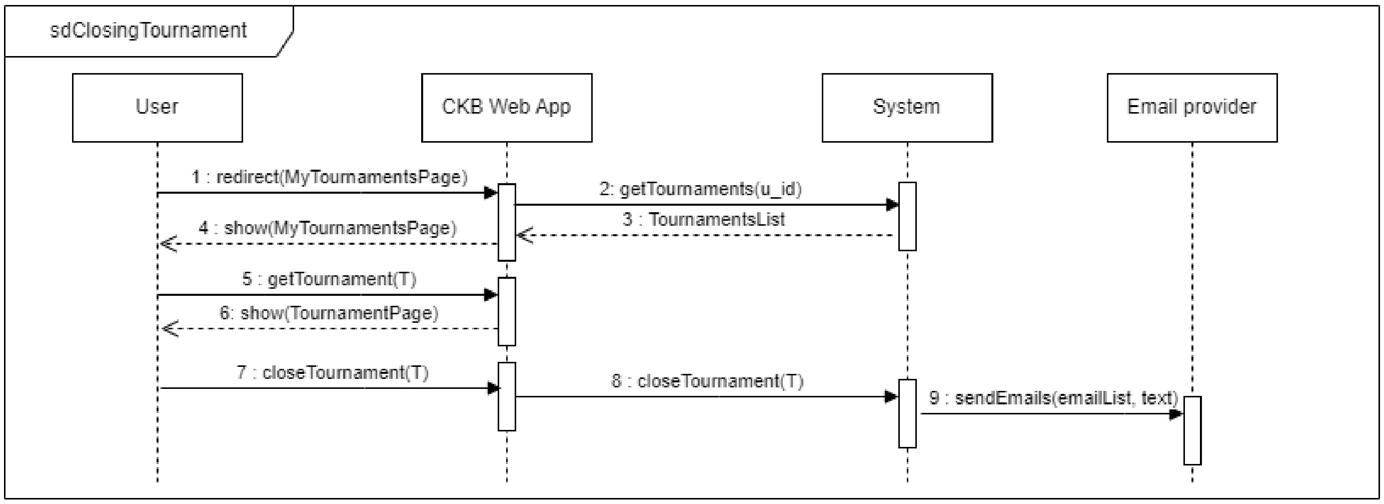


Figure 25: Sequence diagram for [**UC15.**]

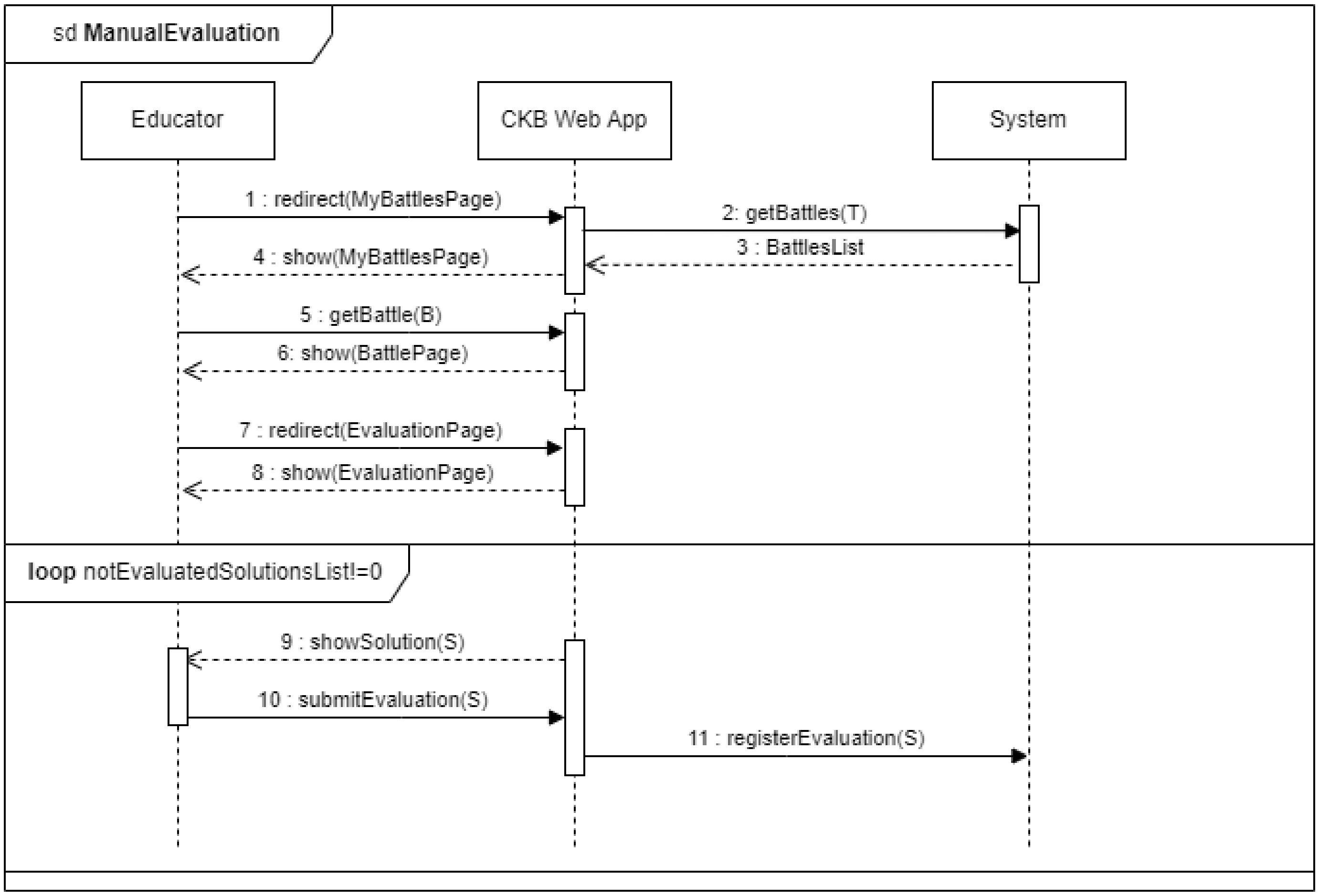


Figure 26: Sequence diagram for [**UC16.**]

3.2.4 Requirements mapping

[G1.]Students and educators can create a profile to access the CKB platform	
[R2.]The system allows users to register with the proper account	**[D1.]**Users must have access to a device with internet connection

[G2.]Users can log into the CKB system	
[R1.] The system allows allow all users to log in. **[R2.]** The system allows users to register with the proper account	**[D1.]** Users must have access to a device with internet connection

[G3.] Educators can create tournaments	
[R1.] The system allows allow all users to log in. **[R3.]** The system allows educators to create tournaments. **[R26.]** The system doesn't allow to create a tournament with the same name of an already existing one.	**[D1.]** Users must have access to a device with internet connection **[D5.]** Information showed by the user interface must be correct

[G4.]Educators can create battles within the tournaments	
[R1.] The system allows allow all users to log in **[R4.]** The system allows educators to grant permissions to manage a tournament to other educators, but not to themselves **[R5.]** The system allows educators with the proper permissions to create a battle for a tournament **[R6.]** The system allows educators to insert all the information regarding a battle during the battle creation phase **[R13.]** The system creates a GitHub repository for each battle **[R14.]** The system allows GitHub to create an automated workflow connected to the CKB platform	**[D1.]** Users must have access to a device with internet connection **[D2.]**Users must have access to the software interfaces required to write and send code to the platform (GitHub and a text editor) **[D3.]**Connection in reliable. **[D4.]**GitHub services must be available. **[D5.]** Information showed by the user interface must be correct

[G5.]Educators can manually evaluate a group's work	
[R1.] The system allows allow all users to log in **[R6.]** The system allows educators to insert all the information regarding a battle during the battle creation phase **[R9.]** The system allows students to register for a battle. **[R13.]** The system creates a GitHub repository for each battle **[R14.]** The system allows GitHub to create an automated workflow connected to the CKB platform **[R15.]** The system allows GitHub to send submitted solutions to the CKB platform **[R17.]** The system allows educators to manually evaluate solutions at the end of the battle, if specified	**[D1.]** Users must have access to a device with internet connection **[D3.]**Connection in reliable. **[D5.]** Information showed by the user interface must be correct **[D7.]** Educators will eventually manually evaluate the solutions, if specified at the creation stage of the battle

[G6.]Educators can review the partial and final results of the battles they organize	
[R1.] The system allows allow all users to log in **[R6.]** The system allows educators to insert all the information regarding a battle during the battle creation phase **[R9.]** The system allows students to register for a battle. **[R16.]** The system automatically evaluates every proposed solution with reference to the specified parameters **[R17.]** The system allows educators to manually evaluate solutions at the end of the battle, if specified **[R22.]** The system allows users involved in a battle to consult the partial score of that battle	**[D1.]** Users must have access to a device with internet connection **[D5.]** Information showed by the user interface must be correct **[D6.]** Automatic evaluation must be consistent with the specified parameters **[D7.]** Educators will eventually manually evaluate the solutions, if specified at the creation stage of the battle

[G7.]Educators can close tournaments.	
[R3.]The system allows educators to create tournaments **[R21.]**The system allows educators to close tournaments they created **[R25.]**The system doesn't allow a tournament to be closed if a battle is still ongoing	**[D1.]**Users must have access to a device with internet connection **[D5.]** Information showed by the user interface must be correct

[G8.]Students can register for tournaments.	
[R1.]The system allows allow all users to log in. **[R3.]**The system allows educators to create tournaments **[R7.]**The system allows students to register to a tournament **[R29.]**The system doesn't allow students to subscribe to tournaments or battle after the registration deadline.	**[D1.]**Users must have access to a device with internet connection **[D5.]**Information showed by the user interface must be correct

[G9.]Students can register for battles within a tournament they are part of	
[R1.]The system allows allow all users to log in **[R5.]**The system allows educators with the proper permissions to create a battle for a tournament **[R6.]**The system allows educators to insert all the information regarding a battle during the battle creation phase **[R7.]**The system allows students to register to a tournament **[R9.]**The system allows students to register for a battle **[R10.]**The system doesn't allow a student already registered in a battle with a team to be registered in the same battle with another team **[R11.]**The system allows students to send invitation to other students but not to themselves **[R12.]**The system allows invited students to accept or decline an invitation **[R27.]**The system automatically declines unanswered invitations that are older than 2 days **[R28.]**When a student answers an invitation the system automatically declines all the other invitations sent to that student **[R29.]**The system doesn't allow students to subscribe to tournaments or battle after the registration deadline	**[D1.]**Users must have access to a device with internet connection **[D5.]**Information showed by the user interface must be correct

[G10.]Students can invite other students to join a battle	
[R1.]The system allows allow all users to log in **[R5.]**The system allows educators with the proper permissions to create a battle for a tournament **[R6.]**The system allows educators to insert all the information regarding a battle during the battle creation phase **[R7.]**The system allows students to register to a tournament **[R9.]**The system allows students to register for a battle **[R10.]**The system doesn't allow a student already registered in a battle with a team to be registered in the same battle with another team **[R11.]**The system allows students to send invitation to other students but not to themselves. **[R29.]**The system doesn't allow students to subscribe to tournaments or battle after the registration deadline	**[D1.]**Users must have access to a device with internet connection **[D5.]**Information showed by the user interface must be correct

[G11.]Students are notified when a new tournament is created	
[R2.]The system allows users to register with the proper account [R3.] The system allows educators to create tournaments [R8.] The system notifies students when a new tournament is created	**[D1.]**Users must have access to a device with internet connection

[G12.]Students are notified when a new battle is added to a tournament they are participating in	
[R2.]The system allows users to register with the proper account **[R5.]**The system allows educators with the proper permissions to create a battle for a tournament **[R7.]**The system allows students to register to a tournament **[R18.]**The system notifies students registered to a tournament when a new battle belonging to that tournament is created	**[D1.]**Users must have access to a device with internet connection

[G13.]Students are notified when a battle they are participating in ends	
[R2.]The system allows users to register with the proper account **[R5.]**The system allows educators with the proper permissions to create a battle for a tournament **[R6.]**The system allows educators to insert all the information regarding a battle during the battle creation phase **[R9.]**The system allows students to register for a battle **[R19.]**The system notifies students when a battle they are registered to ends	**[D1.]**Users must have access to a device with internet connection

[G14.]Students are notified and can review the final result of a tournament they participated in	
[R1.]The system allows allow all users to log in **[R3.]**The system allows educators to create tournaments **[R7.]**The system allows students to register to a tournament **[R20.]**The system notifies students when a tournament they are registered to ends **[R23.]**The system allows users to consult the ranking of a tournament **[R25.]**The system doesn't allow a tournament to be closed if a battle is still ongoing	**[D1.]**Users must have access to a device with internet connection **[D5.]**Information showed by the user interface must be correct

[G15.]Students receive the links to GitHub repositories to submit battle code	
[R2.]The system allows users to register with the proper account **[R5.]**The system allows educators with the proper permissions to create a battle for a tournament **[R6.]**The system allows educators to insert all the information regarding a battle during the battle creation phase **[R9.]**The system allows students to register for a battle **[R13.]**The system creates a GitHub repository for each battle	**[D1.]**Users must have access to a device with internet connection **[D3.]**Connection in reliable **[D4.]**GitHub services must be available **[D5.]**Information showed by the user interface must be correct

[G16.]Users can review the list of ongoing tournaments and for each tournament, the current ranking of participants	
[R1.]The system allows allow all users to log in **[R3.]**The system allows educators to create tournaments **[R7.]**The system allows students to register to a tournament **[R23.]**The system allows users to consult the ranking of a tournament **[R24.]**The system updates the ranking of a tournament at the end of each battle belonging to that tournament	**[D1.]**Users must have access to a device with internet connection **[D5.]**Information showed by the user interface must be correct

[G17.]Students are notified when the final score of the battle becomes available and can review them	
[R2.]The system allows users to register with the proper account **[R5.]**The system allows educators with the proper permissions to create a battle for a tournament **[R6.]**The system allows educators to insert all the information regarding a battle during the battle creation phase **[R9.]**The system allows students to register for a battle **[R16.]**The system automatically evaluates every proposed solution with reference to the specified parameters **[R17.]**The system allows educators to manually evaluate solutions at the end of the battle, if specified **[R19.]**The system notifies students when a battle they are registered to ends	**[D1.]**Users must have access to a device with internet connection **[D5.]**Information showed by the user interface must be correct **[D6.]**Automatic evaluation must be consistent with the specified parameters **[D7.]**Educators will eventually manually evaluate the solutions, if specified at the creation stage of the battle

[G18.]Students can submit code to the system using GitHub	
[R2.]The system allows users to register with the proper account **[R5.]** The system allows educators with the proper permissions to create a battle for a tournament **[R6.]** The system allows educators to insert all the information regarding a battle during the battle creation phase **[R9.]** The system allows students to register for a battle **[R13.]** The system creates a GitHub repository for each battle **[R14.]** The system allows GitHub to create an automated workflow connected to the CKB platform **[R15.]** The system allows GitHub to send submitted solutions to the CKB platform **[R30.]**The system cannot receive submitted solutions after the deadline	**[D1.]**Users must have access to a device with internet connection **[D2.]**Users must have access to the software interfaces required to write and send code to the platform (GitHub and a text editor) **[D3.]**Connection in reliable **[D4.]**GitHub services must be available **[D8.]** Students successfully fork the GitHub repository

3.3 Performance requirements

The CKB system is required to be able to provide a good level of performance in order to grant the access to the platform to a great number of users, as can happen in the context of a university course. The system, assuming a functioning internet connection, must serve the requesting clients with a short latency and is required to be able to connect with GitHub fast and reliably to guarantee a fair evaluation of the solutions without the influence of errors beyond the control of the users.

3.4 Design constraints

3.4.1 Standard compliance

The CKB system is compliant with the GDPR regulation in handling the data of its users within the EU and the EEA.

3.4.2 Hardware limitations

The users (both students and educators) are required to have access to a device with an internet connection, either Wi-Fi or mobile (3G/4G/5G).

3.4.3 Any other constraints

The CKB system must present, in all its realization, a user-friendly interface, providing options to scale the font size and a choice between light and dark theme, to best suit the need of each user. Moreover, the system should provide a short tutorial on how to use the application at the first access by the users, this tutorial should also be replayable from the application settings.

3.5 Software system attributes

3.5.1 Reliability

The CKB system is required to provide reliable communication between itself and both the users and the GitHub platform. To achieve a suitable degree of fault tolerance the data concerning the system must be adequately replicated among different machines, which must keep consistent data between each other.

3.5.2 Availability

The system should provide access to the platform as much as possible, which would ideally be 100%, of course obtaining such a result is impossible, so the system should aim to provide a 4-nines availability (99.99%), meaning 52 minutes per year of unavailability.

3.5.3 Security

Since the system stores personal data about the users registered to the CKB platform, some form of security must be ensured at least on all the sensible parts of the central data storage, if not to every part of it. Other security measures include data encryption during communication on passwords and other sensible data, in order to limit the damage that can be done by sniffing and spoofing attacks, guaranteeing privacy for all users of CKB. Moreover, the data center has to restrict access to sensible data about the subscribed users, so that an user cannot see private information about other users, ranging from access credentials to complete lists of battles and tournaments in which an user can take part.

3.5.4 Maintainability

The CKB platform has to ensure maintainability, this is achieved through the use of design patterns during the software development, thus providing a well structured code that can be updated and modified without the risk of damaging other parts of the system. To guarantee further maintainability the code should be thoroughly documented during the development process (e.g. using JavaDoc if the system is implemented in Java), this ensures that people who did not develop the system can also understand the main parts of the code without effort.

3.5.5 Portability

Given it's Web-App nature, the CKB system will be accessible by any device connected to the internet, also implementing a variation of the UI that adapts to the screen of a mobile device. The Web-App must be able to be executed on the main browsers in use by the general public (Google Chrome, Microsoft Edge, Safari, Mozilla Firefox, Opera, etc.) on both computers and mobile devices.

4 Formal analysis using Alloy 6

This section of the document is about the formal analysis of the described system with the use of the Alloy Analyzer. The analysis done on the CKB system aims to highlight the main characteristics of the system, because of this the Alloy code may not include some trivial aspects, such as string parameters, that would make the diagrams more difficult to read, without providing any real insight on how the logic of the system was verified.

4.1 Signatures

In this first part the main signatures are presented, signatures are the main entities of the system and the code will define relations between them. For the sake of readability redundant relations are omitted (such as having both the tournament the battle belongs to in the **battle** signature and the set of the battles in a tournament in the **tournament** signature), this of course may not be the case in the real implementation, but, as stated before, the aim of this section is to provide a logic model of the system.

```
// User are identifyied by a user id and divide themselves in
// students and educators
abstract sig User {
    u_id: one Int
}

sig Student extends User{}

sig Educator extends User{}

// Battles are created by educators for tournaments and
// students can form teams and participate to the battles
sig Battle{
        creator: one Educator,
        tournament: one Tournament,
        var subscribedTeams: set Team,
        var status: one Status
}
```

```
// Tournaments are created by an educator, who can grant managing
// permissions to other educators
// Students ccan subscribe to tournaments and take part in battles
// created by the managers
sig Tournament{
        creator: one Educator,
        var managers: some Educator,
        var students: set Student,
        var status: one Status,
}{
        creator in managers
}

// A team is composed of students and can participate to a battle
sig Team{
        members: some Student,
        battle: lone Battle,
}

// Teams propose solutions to the evauator, that assigns a score
// to each solution
sig Solution{
        team: one Team,
        score: one Int
}{
        score >= 0
        //score <= 100, Alloy can have integers of maximum value 7
}

// This enumeration represents the status of a battle or of a
// tournament:
// - Created status means that the battle/tournament has been
//   created and students can subscribe to it
// - Ongoing means that the registration deadline has expired
// - Closed means that the submission deadline of the battle
//   has expired or the creator of the tournamen has closed
//   the competition
enum Status {Created, Ongoing, Closed}
```

4.2 Predicates

In this section are reported the predicates describing the evolution of the main components of the system over time. Predicates will then be used to highlight the dynamic characteristics of the system using the features added in Alloy 6.

```
// Starts a tournament
pred startTournamentt: Tournament{
        t.status = Created
        t.status’ = Ongoing
}

// Closes a tournament
pred closeTournamentt: Tournament{
        t.status = Ongoing
        t.status’ = Closed
}

// Starts a battle
pred startBattleb: Battle{
        b.status = Created
        b.status’ = Ongoing
}

// Closes a battle
pred closeBattleb: Battle{
        b.status = Ongoing
        b.status’ = Closed
}

// Enrolls a student to a tournament
pred enrollStudents: Student, t: Tournament{
        t.status = Created
        t.status’ = Created
        s not in t.students
        t.students’ = t.students + s
}
```

```
// Subscribes a team to a battle
pred teamJoinsBattlet: Team, b: Battle{
        b.status = Created
        b.status' = Created
        (all s: Student | s in t.members implies s in b.tournament.students)
        t not in b.subscribedTeams
        b.subscribedTeams' = b.subscribedTeams + t
    no t.battle
}

// Grants managing permissions to an educator
pred addManagere: Educator, t: Tournament{
        t.status = Created or t.status = Ongoing
        t.status' = t.status
        e not in t.managers
        t.managers' = t.managers + e
}

// Revokes managing permissions from a manager
pred removeManagere: Educator, t: Tournament{
        t.status = Created or t.status = Ongoing
        t.status' = t.status
        e != t.creator
        e in t.managers
        t.managers' = t.managers - e
}
```

4.3 Facts

Facts are the invariants of the system, in other words the constraints that a valid instance of it should always respect.

```
// A student can participate in a battle in only one team
fact noOverlappingTeams{
        all s1: Student, t1, t2: Team |
            (s1 in t1.members and
         s1 in t2.members and
         t1.battle = t2.battle) implies
            t1 = t2
}
```

```
// There cannot be two users with the same name
fact usernameUnicity{
        all disj u1, u2: User | u1.u_id != u2.u_id
}

// If a student is part of a team and that team is subscribed to
// a battle belonging to a tournament, then the student must be
// subscribed to the tournament
fact teamSubscribedToTournament{
        all s: Student, tm: Team, tr: Tournament, b: Battle|
                (s in tm.members and
         tm in b.subscribedTeams and
         b.tournament = tr) implies
            (s in tr.students)
}

// If an educator creates a battle for a tournament, it must be
// a manager of that tournament
fact battleCreatorIsManager{
        all e: Educator, b: Battle, t: Tournament |
                (b.creator = e and b.tournament = t) implies
                        (e in t.managers)
}

// A team can take part only in one battle
fact TeamBattleConsistency{
        all t: Team, b: Battle |
                t in b.subscribedTeams iff b = t.battle
}

// A tournament can be closed only when all its battles are closed
fact closedTournamentWhenClosedBattles{
        all t: Tournament |
                t.status = Closed implies
            (all b: Battle | b.tournament = t implies
                b.status = Closed)
}

// If a team is subscribed to a battle, then all of its members
// are subscribed to the tournament of the battle
fact studentSubscriptionConsistency{
        all s: Student, t: Team, b: Battle |
                (s in t.members and t in b.subscribedTeams) implies
                        s in b.tournament.students
}
```

```
// If a battle is in the Created status, then no team can provide
// solution for that battle
fact noSolutionForCreatedBattles{
        all b: Battle |
                b.status = Created implies
            (no s: Solution | s.team in b.subscribedTeams)
}

// If a tournament is in the Created status, then it cannot have
// any battle
fact noBattleForCreatedTournament{
        all t: Tournament |
                t.status = Created implies
            (no b: Battle | b.tournament = t)
}




// - Once a tournament is in the Ongoing state it can never
//   return in the Created state
// - Once a battle is in the Closed state it will always be
//   in the Closed state
// - Once a tournament is in the Ongoing state it can never
//   return in the Created state
// - Once a battle is in the Closed state it will always be
//   in the Closed state
fact statusConsistency{
        all t: Tournament |
                t.status = Ongoing implies
            always t.status != Created


        all t: Tournament |
                t.status = Closed implies
            always t.status = Closed

        all b: Battle |
                b.status = Ongoing implies
            always b.status != Created

        all b: Battle |
                b.status = Closed implies
            always b.status = Closed
}
```

```
// - Once a student has subscribed to a tournament, it will
//   always be subscribed to that tournament
// - Once a team has subscribed to a battle, it will always
//   be subscribed to that battle
fact onceSubscribedAlwaysSubscribed{
        all s: Student, t: Tournament |
                s in t.students implies
            always s in t.students

        all t: Team, b: Battle |
                t in b.subscribedTeams implies
            always t in b.subscribedTeams
}
```

4.4 Static Model

The static model shown below is generated using the following command:

```
pred show{
        all t: Tournament | #t.students >= 2
        one t: Tournament | t.status = Created
        one b: Battle | b.status = Created
        all t: Team | (some s: Solution | s.team = t)
}

run show for exactly 5 Student,
             exactly 2 Educator,
             exactly 4 Battle,
             exactly 3 Tournament,
             exactly 4 Team,
             exactly 5 Solution
```

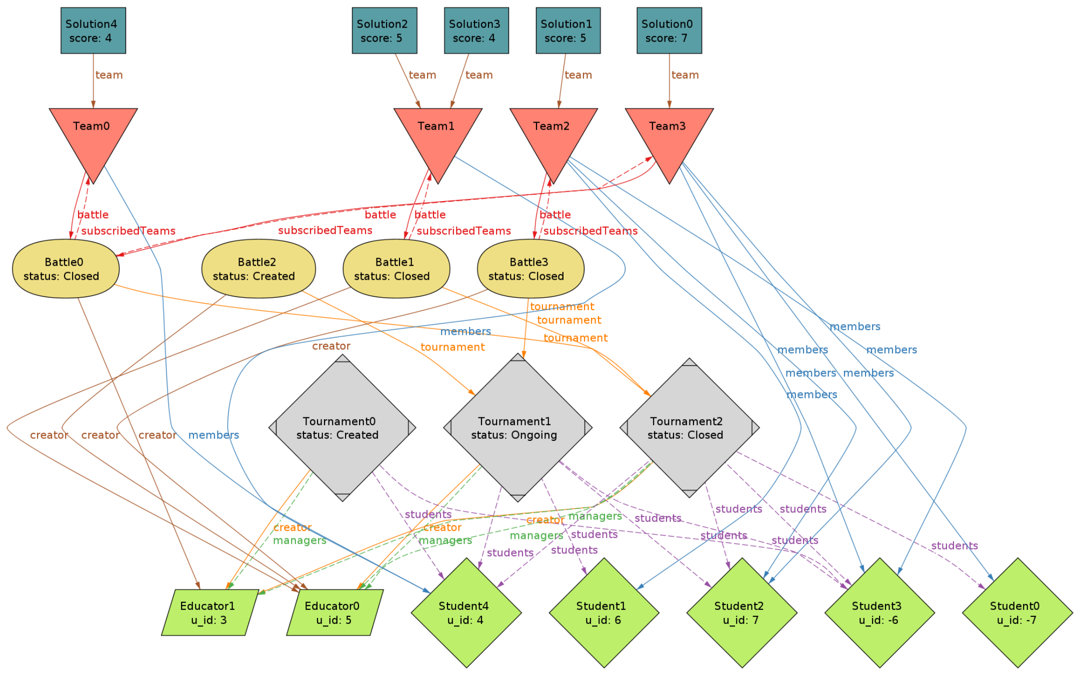


Figure 27: Model generated by the Alloy Analyzer

The generated model represents a valid instance of the CKB system as it was described in the Alloy code above. The generated world's composition is described in the run command, the schema differentiates the shape of the different kinds of atoms, in order to make them more recognizable, and the colours of the relations, for the same reason. An example of non-trivial constraint that can be noticed in the model is the fact that the members of a team subscribed to a battle are subscribed to the tournament the battle belongs to. Other examples is the fact that a tournament in the Created state doesn't have any battle associated and that a tournament in the CLosed state only has closed battles.

4.5 Dynamic Model

In this section are presented two examples of how some aspects of the model evolve during time, in particular are provided the schema about the enrollment of a student in a tournament and the granting of managing permissions to an educator. Of course similar results are obtainable by running the other predicates, but for the sake of simplicity only two are provided.

4.5.1 Enrollment of a student

The following instances are obtained by running the enrollStudent predicate, the first image represents the state before the enrollment, while the second one represents the state after the enrollment.

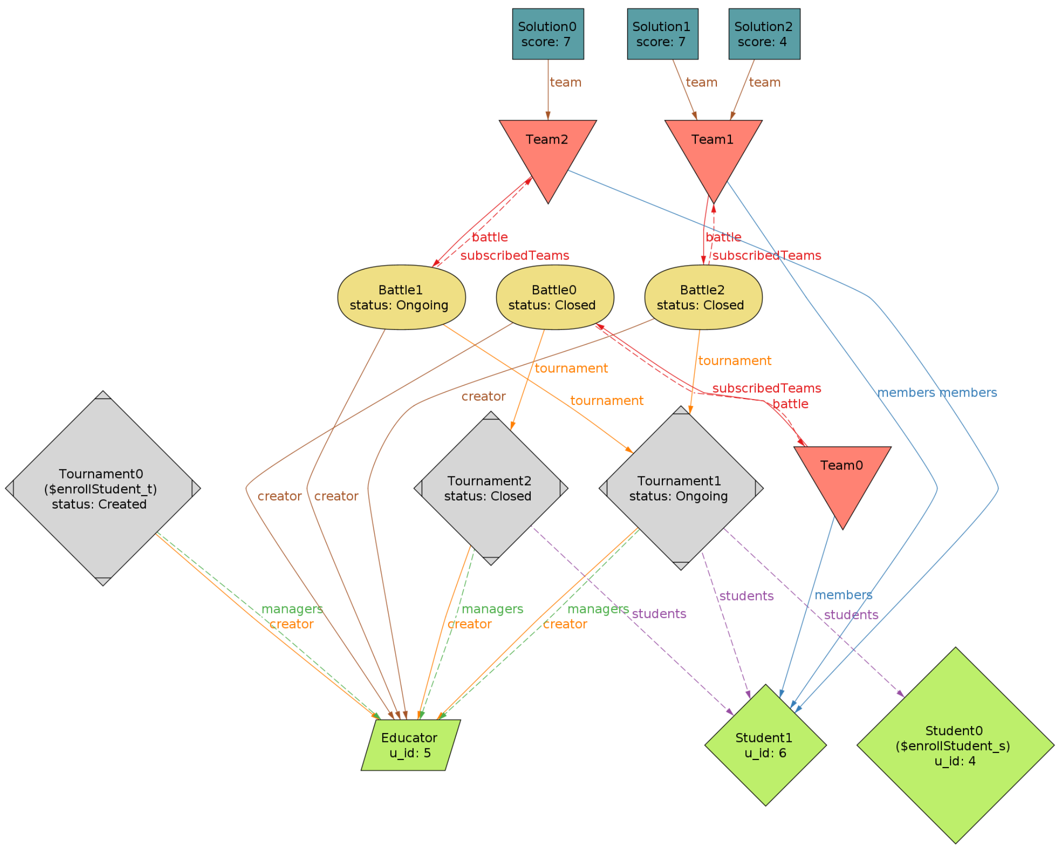


Figure 28: Model before the enrollment

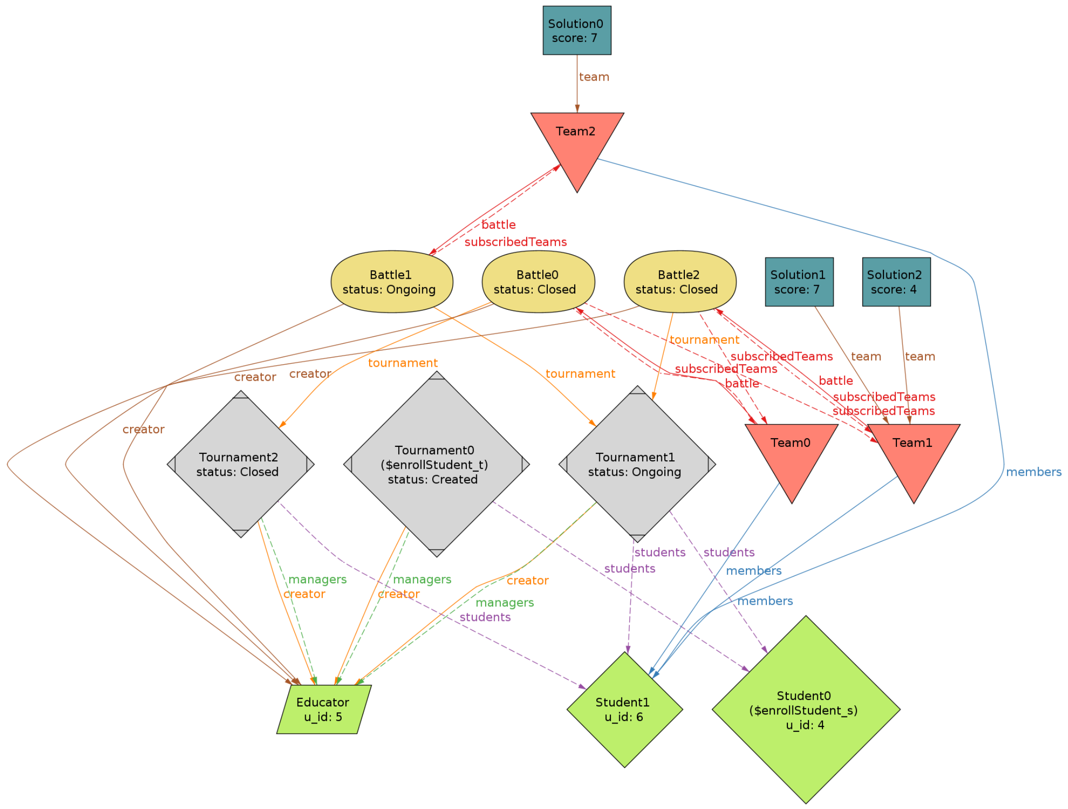


Figure 29: Model after the enrollment

4.5.2 Adding a manager

The following instances are similar to the previous ones, both are obtained by running the addManager predicate and, as before, the two pictures represent two consecutive states.

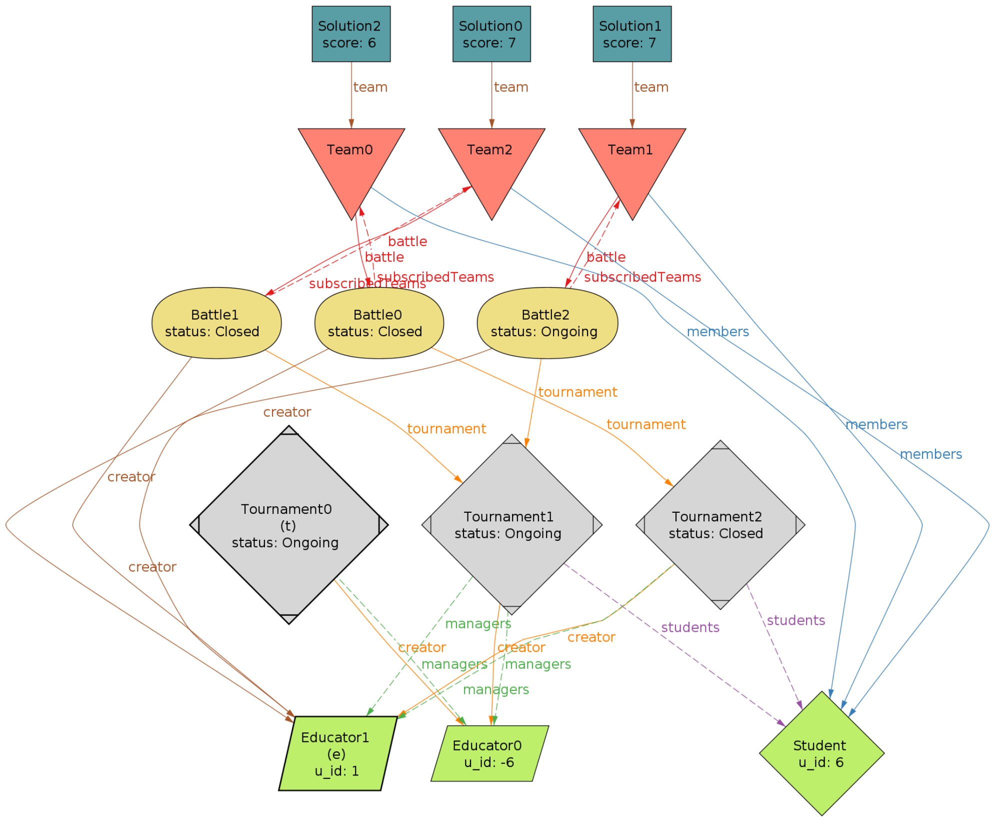


Figure 30: Model before the permission granting

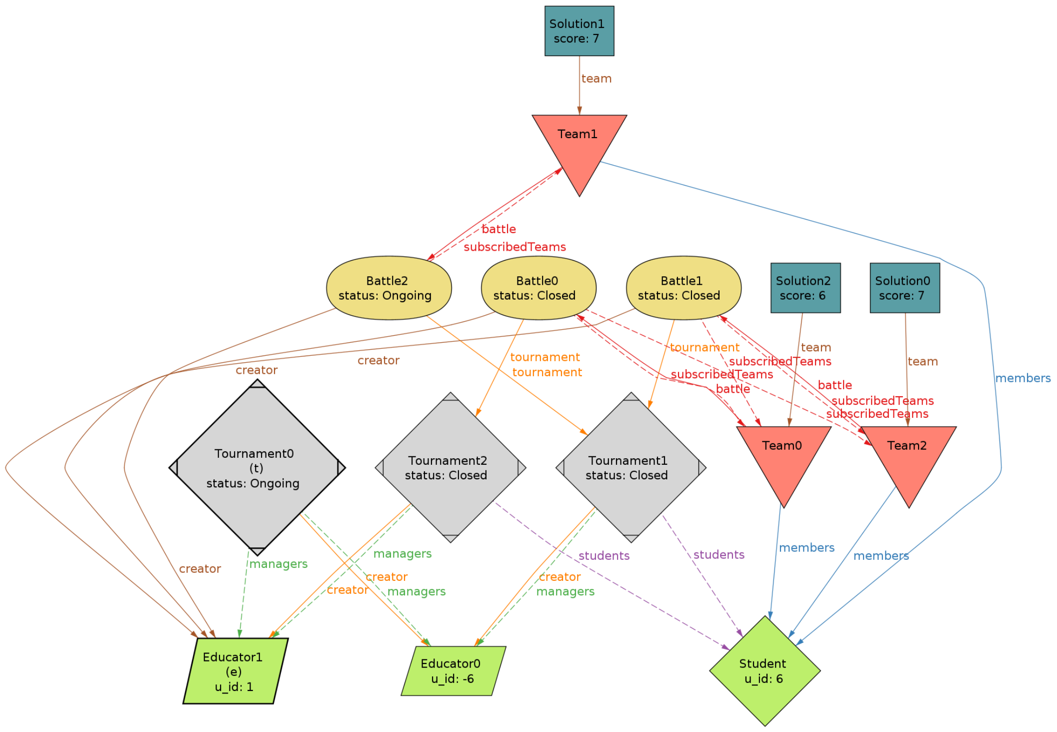


Figure 31: Model after the permission granting

5 Effort spent

5.1 Francesco Galbiati

Chapter 1: 3h00m
Chapter 2: 9h00m
Chapter 3: 14h45m
Chapter 4: 14h45m

5.2 Chiara Fossà

Chapter 1: 3h15m
Chapter 2: 9h30m
Chapter 3: 27h30m
Chapter 4: 3h

6 References

Document structure: Project assignment document and previous year project (https://webeep.polimi.it/mod/folder/view.php?id=219353)
UI mockups: Moqups (https://moqups.com/it/templates/wireframes-mockups/)
Diagrams and schemes: Draw.io(http://draw.io/)
Lecture material from the Software Engineering 2 course at Politecnico di Milano
Alloy 6 documentation (https://alloy.readthedocs.io/en/latest/)