POLITECNICO
MILANO 1863

CodeKataBattle

Requirements and Specification Document

Alessandro Griffanti
Luca Masiero

—

22 December 2023

Table of contents

1 | Introduction

1.1 Purpose

As the years go by, programming takes on an increasingly central position in society and this leads more and more students to enroll in university faculties inherent to this world. In this context, educational platforms like CodeKataBattle play a crucial role in helping future professional developers to hone their knowledge and skills.

This platform aims at doing so by offering to university students several coding challenges dealing with different programming concepts, like linked lists, trees, graphs and many more.

Educators will have the possibility to create tournaments and battles which will challenge every student who wants to grapple with them in a fun and collaborative way. In fact, the platform will not only enable students to develop their programming skills, but also allow them to do so with peers, promoting both the development of programming skills and the communication and teamwork abilities.

Hence, the main goals of CodeKataBattle are:

ID	Description
G1	All students and educators can register and then log into the CKB platform for, respectively, participating and creating tournaments and battles
G2	Students can improve their software development skills by joining the coding battles available for each tournament
G3	Educator can create and manage tournaments, battles within each of them and evaluate the code submitted by students

Table 1.1: The goals

1.2 Scope

To represent the scope of the project we use the “World and the Machine” model by M. Jackson. It presents a clear description of the events which cannot be observed by the system (the “world” phenomena), those strictly related to the system (the “machine” phenomena) and those in common (the “shared” phenomena).

1.2.1 World Phenomena

ID	Description
WP1	A student wants to join CodeKataBattle
WP2	An educator wants to join CodeKataBattle
WP3	An educator wants to create a tournament
WP4	A student wants to join a tournament
WP5	An educator wants to create a battle inside a his/her tournament
WP6	A student wants to join a battle
WP7	A student wants to invite some friends to join a battle with him/her
WP8	A student works on the source code
WP9	A student commits a solution to a battle on GitHub
WP10	A student wants to view the current battle and tournament ranking
WP11	An educator wants to view the current battle and tournament ranking
WP12	A student wants to see the final ranking of a battle or a tournament
WP13	An educator wants to see the final ranking of a battle or a tournament
WP14	An educator wants to personally evaluate the sources submitted by the teams

Table 1.2: World Phenomena

1.2.2 Shared Phenomena

ID	Description	Controller	Observer
SP1	A student registers and logs in	Student	Machine
SP2	An educator registers and logs in	Educator	Machine
SP3	An educator creates a tournament	Educator	Machine
SP4	An educator is given the permissions to create	Educator	Machine

	battles in a tournament		
SP5	The machine notifies the students when a new tournament is created	Machine	Students
SP6	A student joins a tournament	Educator	Machine
SP7	An educator creates a battle	Educator	Machine
SP8	The machine notifies the students when a new battle is created	Machine	Students
SP9	A student joins a battle by himself	Student	Machine
SP10	A student joins a battle with other peers	Student	Machine
SP11	The machine notifies the students when a battle begins	Machine	Students
SP12	The machine updates a team battle score and ranking	Machine	Students, Educator
SP13	An educator evaluates the teams' work	Educator	Machine
SP14	The machine notifies the students when a battle final ranking is available	Machine	Students
SP15	The machine notifies the students when a tournament final ranking is available	Machine	Students

Table 1.3: Shared Phenomena

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

User	Everyone who uses the CodeKataBattle platform: students and educators
Code Kata	It represents the description of the project, including test cases and build automation scripts

Table 1.4: Definitions

1.3.2 Acronyms

CKB	CodeKataBattle

Table 1.5: Acronyms

1.3.3 Abbreviations

Gn	N-th goal
Rn	N-th functional requirement
Dn	N-th domain assumption
WPn	N-th world phenomena
SPn	N-th shared phenomena
UCn	N-th use case

Table 1.6: Abbreviations

1.4 Revision History

Date	**Description**
22 December 2023	First Version

Table 1.7: Revision History

1.5 Reference Documents

- Specification document: “Assignment RDD AY 2023-2024”

- Lecture slides from the Software Engineering 2 course

1.6 Document Structure

The document is structured into the following main six chapters:

- **Chapter 1: Introduction.** This chapter includes the goals of the system to be (*purpose*) and an analysis of the world and machine phenomena (*scope*). It also includes a section where all the definitions, acronyms and abbreviations used in the document are described (*definitions, acronyms, abbreviations*). Lastly, there is a revision history and a reference list.

- **Chapter 2: Overall description.** This chapter includes scenarios, the main state diagrams and class diagram (*product perspective*). Then the most important high level functions of the system are presented (*product functions*). After that the users' needs and characteristics are clarified (*user characteristics*) as well as, in the end, the domain assumptions (*assumptions, dependencies and constraints*).

- **Chapter 3: Specific requirements.** This chapter is the main body part of the document. Firstly it presents mockups interfaces (*external interface requirements*). Then, all the functional requirements are listed along with use case diagrams, use cases and sequence diagrams (*functional requirements*). Lastly, it is presented a brief section on non functional requirements (*performance requirements, design constraints, software attributes*).

- **Chapter 4: Formal analysis using alloy.** This chapter presents a formal analysis using alloy, which is really important to prove the correctness of the model presented in the previous sections.

- **Chapter 5: Effort spent.** This chapter reports approximately the amount of hours each member of the group worked to realise the document.

- **Chapter 6: References.** This section simply lists references and other resources used to redact this document.

2 | Overall Description

2.1 Product Perspective

2.1.1 Class Diagram

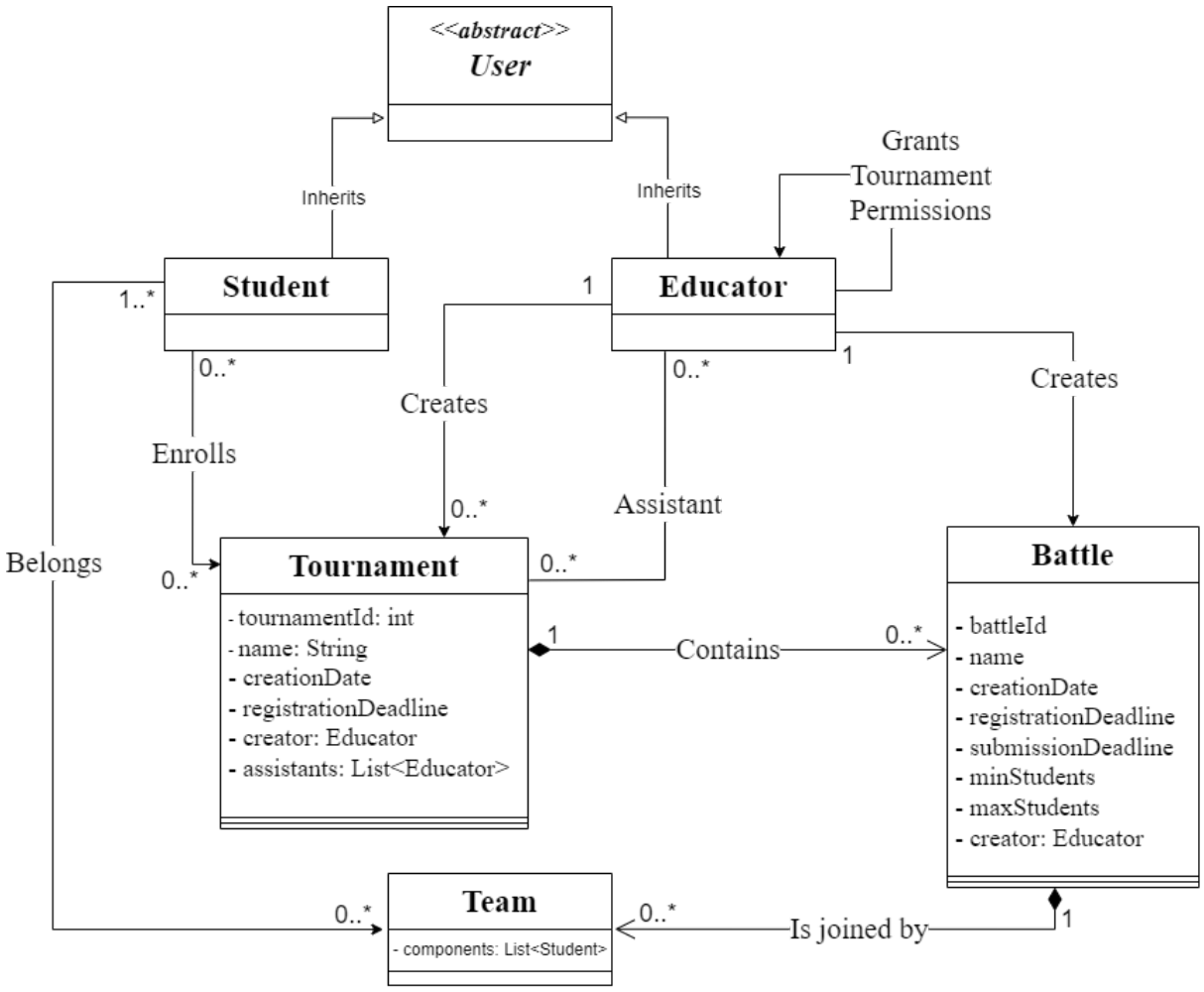


Figure 2.1: Class diagram

2.1.2 State Diagrams

The following state diagrams illustrate the main state of the major elements of the system: tournaments and battles.

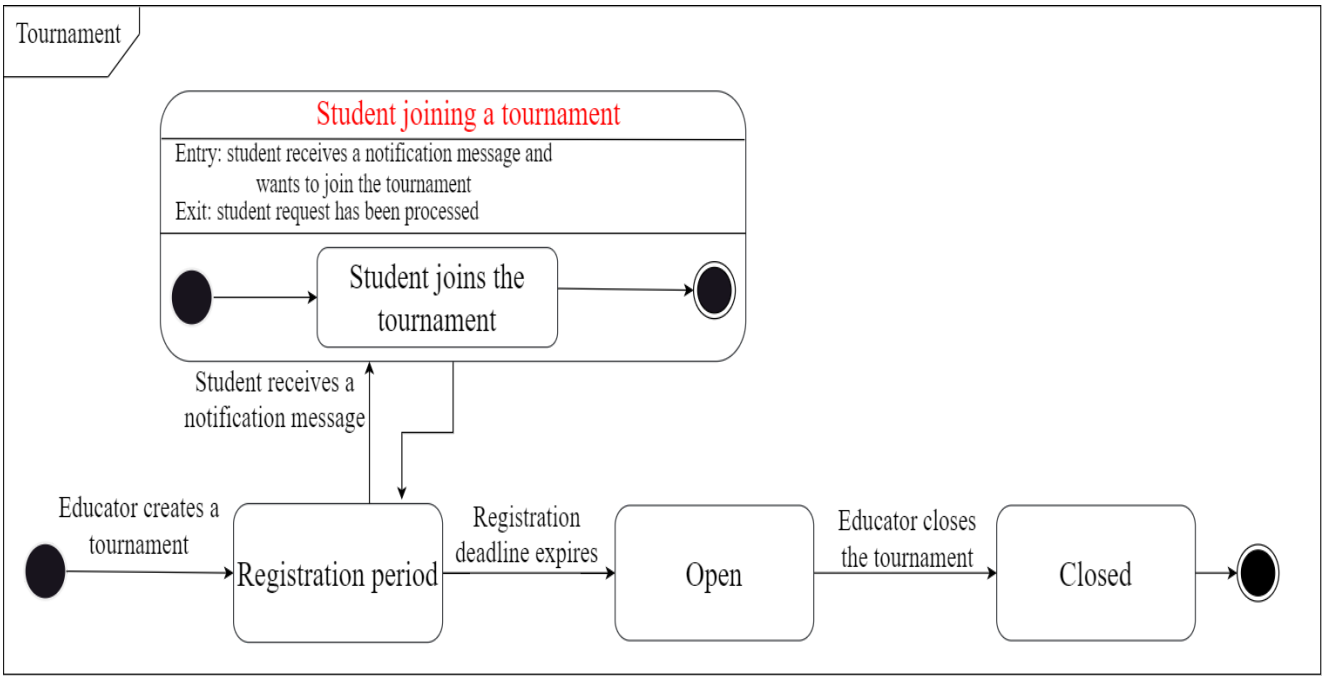


Figure 2.2: Tournament state diagram

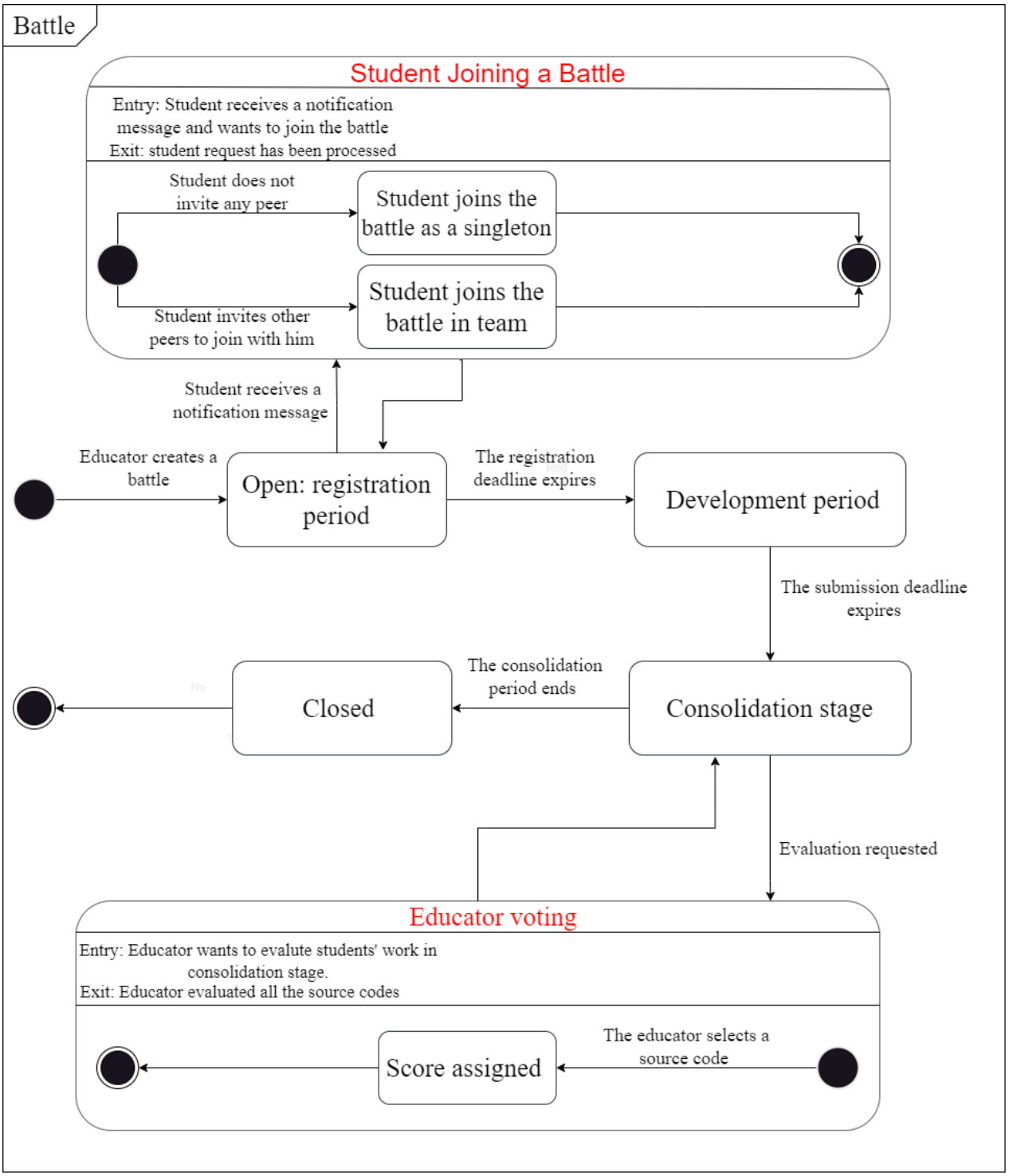


Figure 2.3: Battle state diagram

2.1.3 Scenarios

Here are several different scenarios:

- **Scenario 1: A student discovers the CKB platform and decides to try it.**
  Tom is a computer engineering student and he is currently attending his second year. Like all his colleagues, he is attending one of the most important and difficult courses of the bachelor level: Algorithms and data structures. Tom has several difficulties at grasping the concepts explained during the lecture since he finds them a little bit abstract due to the lack of real code and coding exercises, which prevents him from making real practice with all the data structures he is studying. Talking with his university friends, he discovers a new university platform where all his colleagues are practising the different concepts learned during the course: CodeKataBattle. Tom immediately decides to register for it, so he heads to the sign up page which, after selecting "CKB login - Student", leads him to the university login services. Here he inserts his university credentials, email and password, and, after that, the system creates his student account.

- **Scenario 2: A non-student discovers the CKB platform and decides to try it.**
  James is a 30 years old man who is not happy with his job. For this reason, every evening he studies computer science related concepts, like algorithms, software engineering principles and many more, to try to completely change his career. He realises he needs to make some practice on a few topics he just studied and he discovers CodeKataBattle. He decides to try it and heads to the sign up page. The system leads him to the university login services and James, not understanding that, inserts his personal email and password. Due to the fact that he is not a university student, James' request to sign up gets refused, so he is forced to look for other platforms.

- **Scenario 3: A student registers for a tournament.**
  Anne is a young student who just graduated at the bachelor level and started her master degree. While studying, she would like to obtain an internship as a software engineer and she is aware of the coding interviews that need to be passed to achieve that. Anne decides to make some practice on the coding concepts she is less familiar with and, to do so, she decides to use the CKB platform. After logging in, she heads up to the section where all the

tournaments are listed and, after browsing them, she decides to enroll in one of them dealing with binary search trees.

- **Scenario 4: A student enrolled in a tournament joins one of its battles.**
  Joe is a student who has been using the CKB platform for several months. During one of his afternoons of study, he receives an email that notifies him of a new battle which has just been created in a tournament he had previously enrolled in. Joe logs in into the platform, opens that specific tournament and he decides to take part in the battle. Since the registration deadline has not expired yet, Joe registers for the battle and, given that the minimum number of students per group set for it is exactly one, he joins it as a singleton. As soon as the registration deadline expires, Joe receives a new email including the link to the github repository containing the *code kata* and at this point, he starts to work on the coding exercise.

- **Scenario 5: A group of peers enrolled in a tournament joins one of its battles**
  Sam, Katie and Henry are a group of friends who just started to use the CKB platform and are not really familiar with all the aspects of it. After one of their university lessons, they decide to stay to make some coding exercise together, so they all log in into the platform. Sam opens a tournament they all had previously enrolled in and notices a new battle which has been created. Since the registration deadline has not expired yet, Sam joins the battle and, at registration time, she invites Katie, Henry and William, a friend of hers who currently is not at the university but will take part in the battle with them. As soon as she tries to send the invites, the system notifies her that the maximum number of students per group set for that battle is 3, so she can invite up to two other peers. Sam didn't notice that and so she decides to invite only Katie and Henry and to join the battle as a team with them.

- **Scenario 6: An educator login for the first time into the CKB platform**
  John, a university professor of the department of Information Technologies, decides to employ a hands-on approach for teaching his programming classes. Hoping to engage his students more effectively, he decides to try out a coding platform recommended by one of his colleagues, which is becoming very popular in the university: CodeKataBattle.
  He accesses the login page of the platform, chooses the option "CKB login - Educator", once redirected to the university login page he provides his institutional account's credentials (email and password) and there he is, in his platform's home page.

- **Scenario 7: An educator creates a tournament**
  John is logged into the CKB platform as an educator and wants to create the first tournament for one of his classes called “Programming with Python”.
  Once he chooses the option “*create new tournament*”, the platform asks him for a registration deadline and also if he wants to grant permission to create battles inside this new tournament to other colleagues, but he postpones this decision by pressing the button “*maybe later*”.
  The platform will now send a notification, in the form of a platform message and an email, to all the students who are registered on the platform.
  All he has to do now is to wait for the students to enrol in the tournament.

- **Scenario 8: An educator manages a battle (from the creation to the closing)**
  Professor John, after creating a tournament and after its registration deadline has expired, would like to create a battle. To do so he needs to provide to the platform a list of information and material: the *code kata*, a minimum and maximum number of students per group, the registration deadline, the submission deadline and additional configurations for the scoring.
  Once he’s provided all this information the platform will proceed to send an email and notify through the platform itself all the students who are enrolled in the tournament containing the new battle.
  As the teams participating in the battle submit their solutions, John can inspect the updating score of each of them through a ranking made available by the platform.
  The day of the submission deadline has come and John now proceeds with his personal evaluation of the work last submitted by each group, manually entering the scores.
  This is an optional part of the “Consolidation stage”, during which the score of each team is confirmed or merged with the personal evaluation of the professor.
  Once the “Consolidation stage” finishes, the final battle ranking becomes available and the platform notifies all the students with a platform notification message.
  The system finally updates the overall tournament’s ranking too.
  One battle is over, but the tournament is still open. John should start thinking about the next battle he will devise to challenge his students.

2.2 Product Functions

Here is the description of the high level functions the system has. In particular, three main major functions have been identified:

- **Sign up and Login.** These functions will be available to all users of the CKB platform and allow each of them to create and access into their personal account.
  Since the platform is thought for university students and educators, the authentication process is made through the university login services, hence the university email and password are required.

- **Join tournaments and battles.** These functions will be available to all the students registered on the CKB platform. The system will show to each student the list of all available tournaments he/she can enroll in. After the registration to one of them, the list of all battles of that same tournament are shown to the student who can join one of them or more. A battle can be joined by a student both as a singleton and as in group: it will be indeed possible to invite other peers to join the same battle as a team.

- **Create and Manage tournaments and battles.** These functions will be available to all the educators registered on the CKB platform. The system will allow each educator to create tournaments, battles for each of them and also to grant other educators the permission to create battles within the context of a specific tournament. After the end of each battle he/she has created, an educator can also optionally assign to each team a personal score. Lastly, an educator can close a tournament.

2.3 User Characteristics

There are mainly two kinds of users that interact with the system:

- **Student.** It indicates a university student who wants to improve his/her coding and development skills and, to achieve so, he/she decides to use the CKB platform to remain in the context of his university

- **Educator.** An educator represents, for example, a professor who wants to create tournaments and battles to challenge his students.

2.4 Assumptions, Dependencies and Constraints

Here are the main domain assumptions:

ID	Description
D1	The university login service correctly checks if the credentials provided by the user are valid
D2	The credentials provided during the registration phase to the university login services are sufficient to identify the user in the context of the university
D3	Every user has a github account
D4	Every user has a valid institutional mail account
D5	The static analysis tool provides a correct evaluation of the students' work

Table 2.1: Domain assumptions

3 | Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The CKB platform will be available as a web app and a desktop application and both can be used by both students and educators: the first ones will use them to join tournaments and battles, while the second ones to create, manage and close them. Since students and educators have different functions available, two different views are required.
The following mockups represent a basic idea of what the web app and the desktop application will look like.

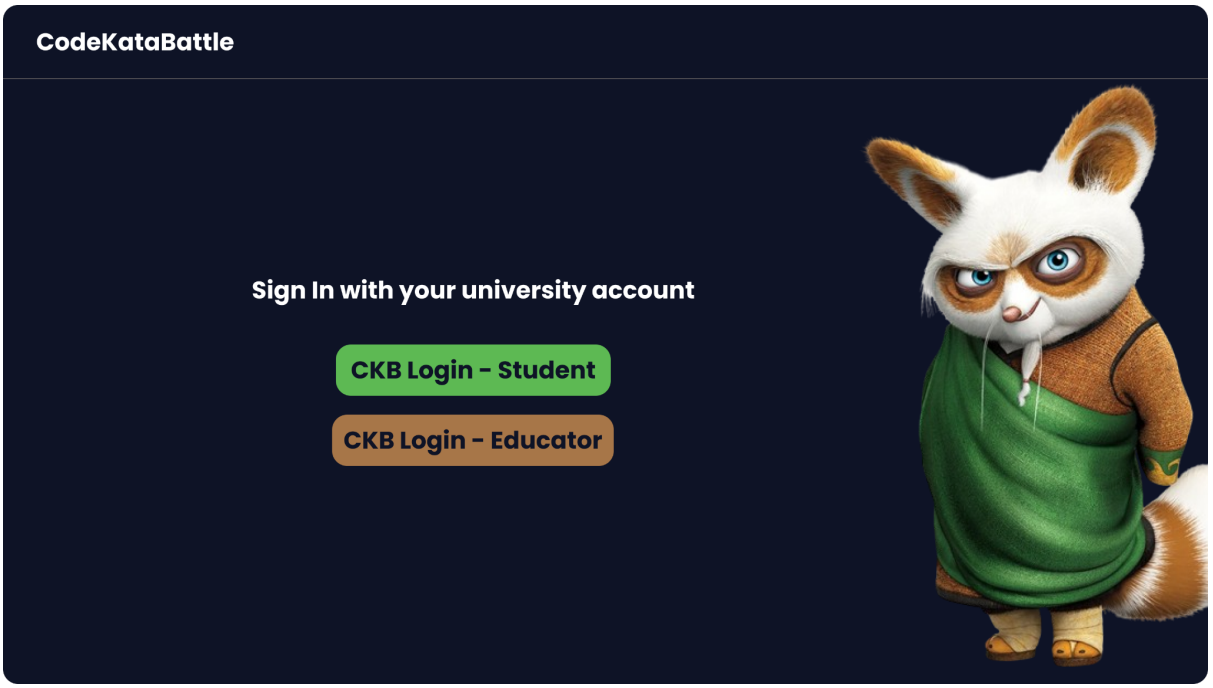


Figure 3.1: CKB login page

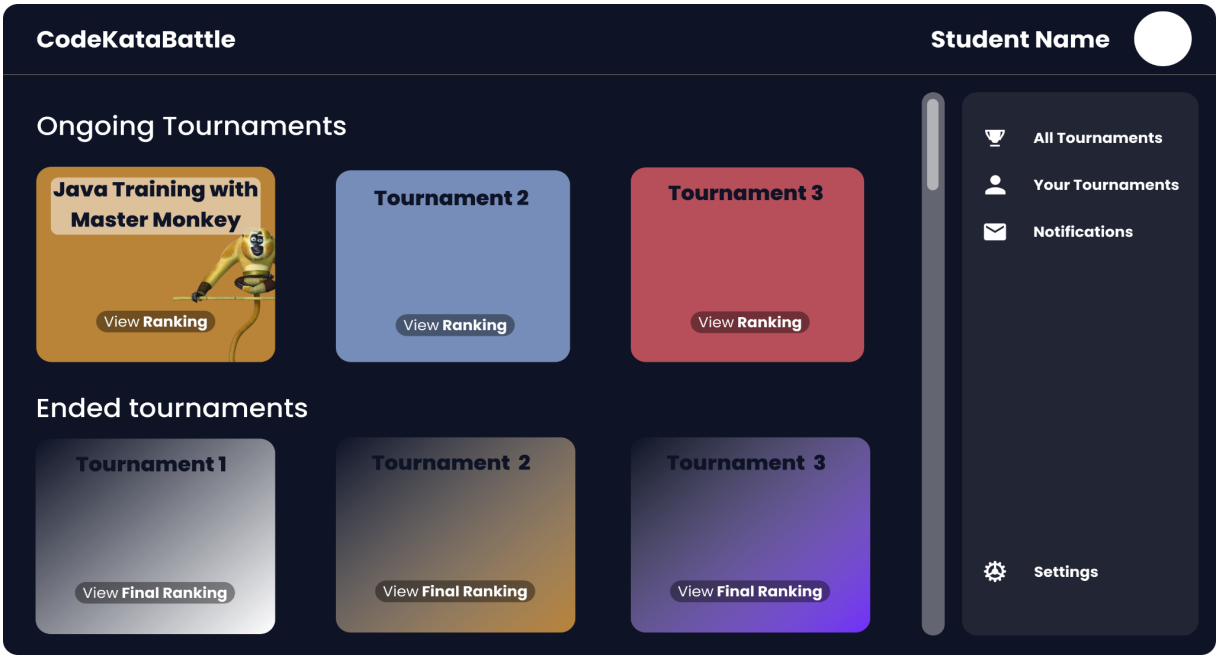


Figure 3.2: Student personal page

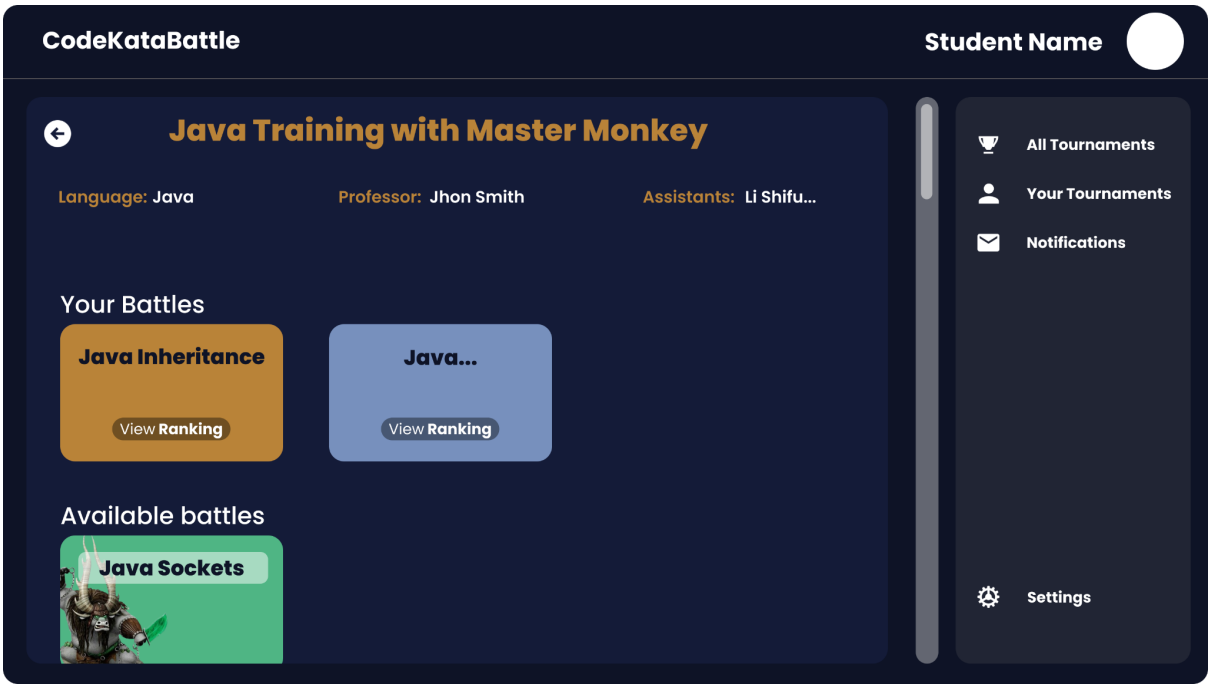


Figure 3.3: Student OnGoing tournament page[1]

Figure 3.4: Student OnGoing tournament page[2]

Figure 3.5: Student battle join page

Figure 3.6: Student all available tournaments page

Figure 3.7: Student tournament join page

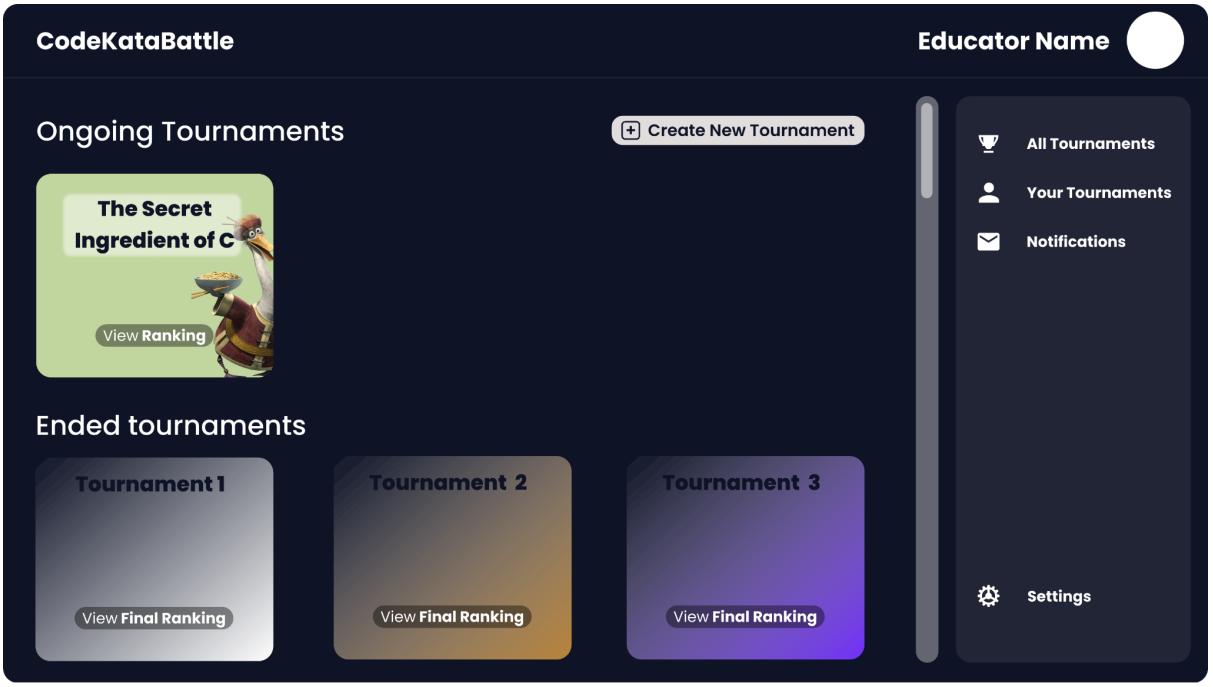


Figure 3.8: Educator personal page

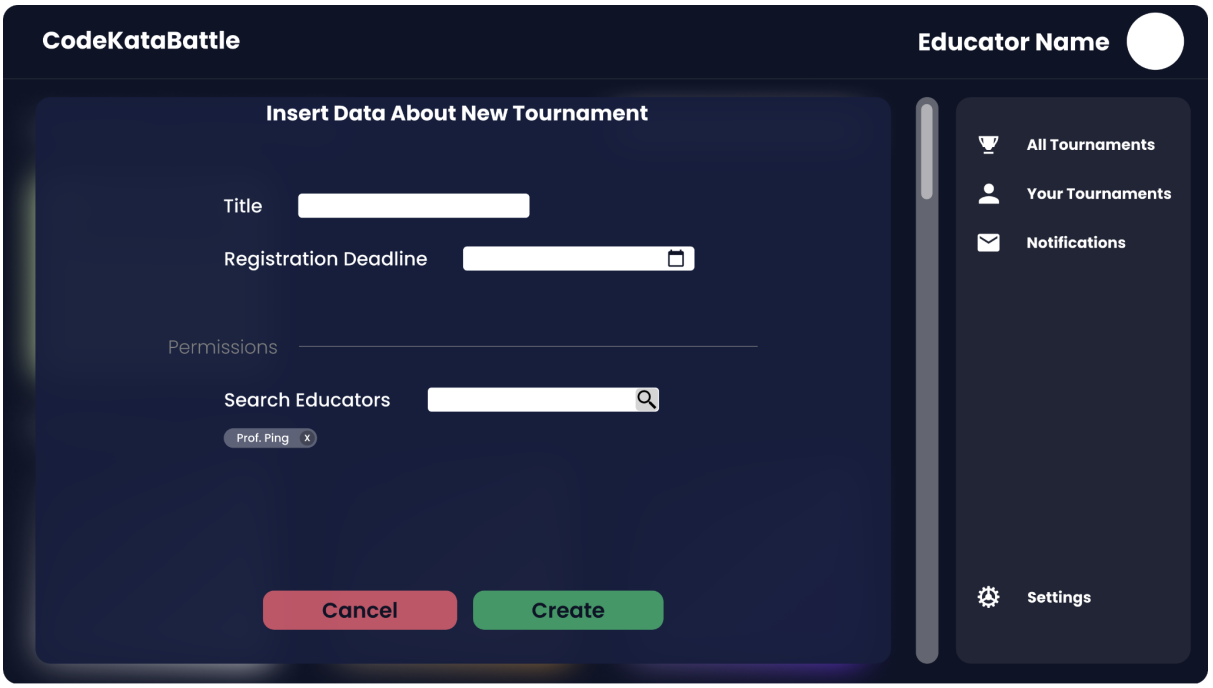


Figure 3.9: Educator create tournament page

Figure 3.10: Educator OnGoing tournament page

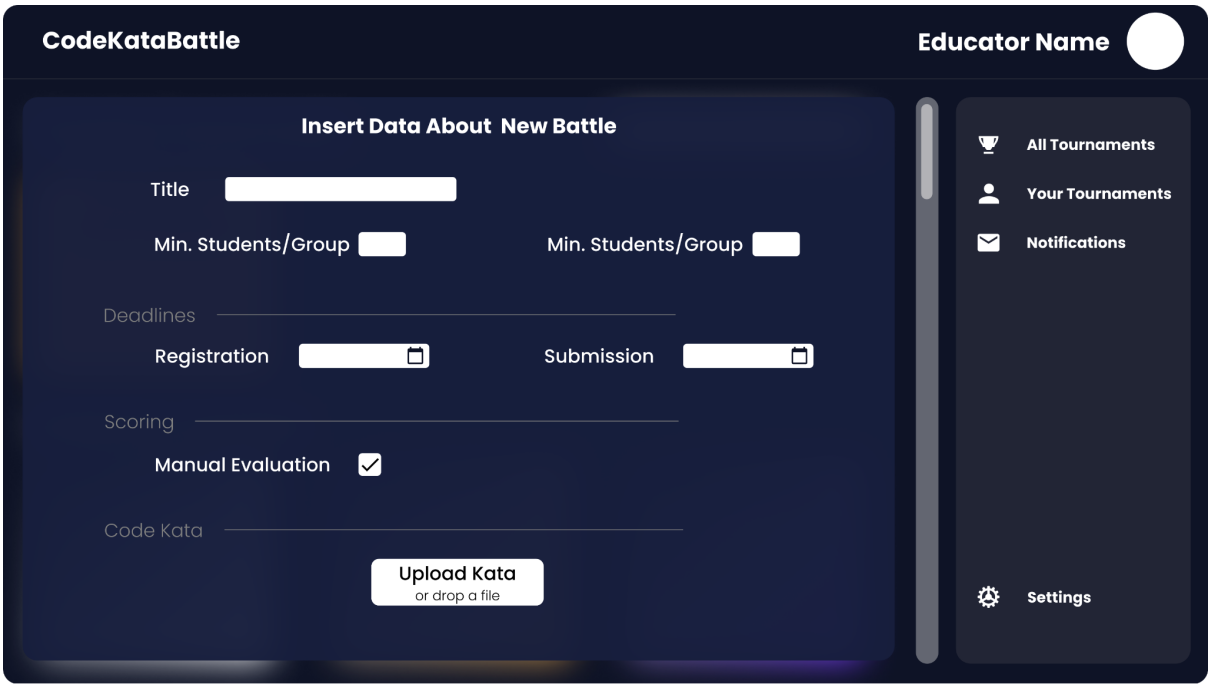


Figure 3.11: Educator create battle page[1]

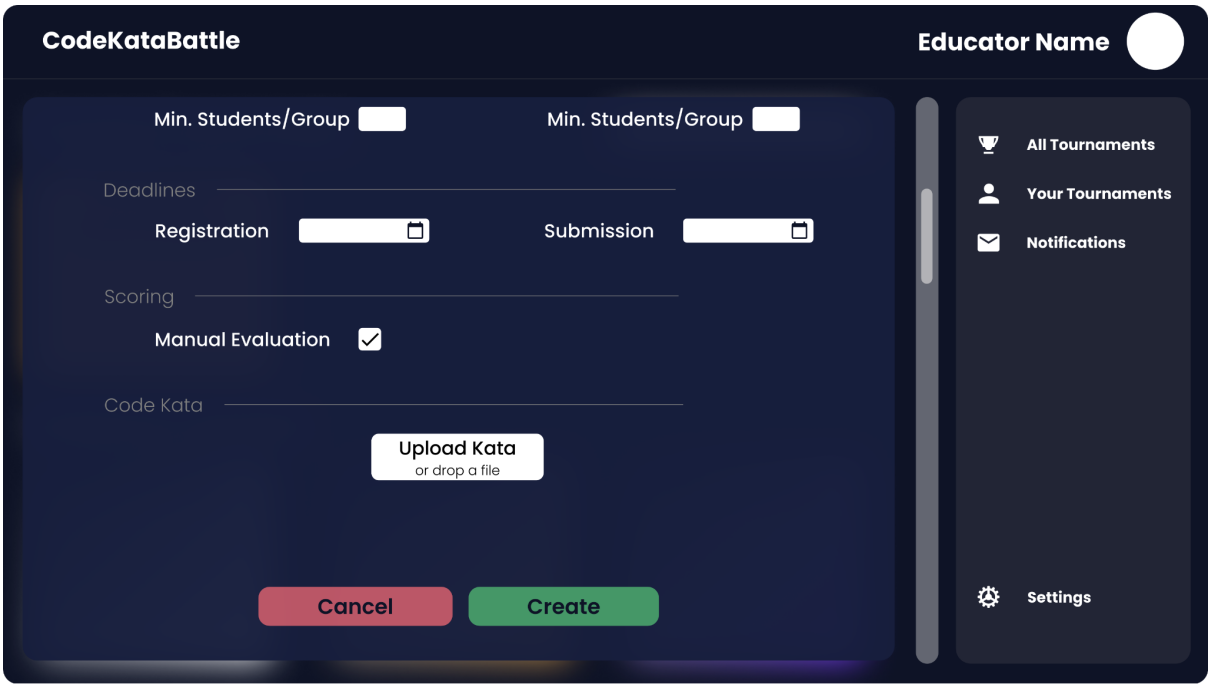


Figure 3.12: Educator create battle page[2]

3.1.2 Hardware Interfaces

Since the system won't interact with any particular external hardware, no remarkable constraints and hardware interfaces need to be declared.

3.1.3 Software Interfaces

The system will use the following external software interfaces:

- **University login services**. The system will rely on the university login services to identify each user and establish whether a user is a student or an educator, as explained in the domain assumptions section.

- **Github API$_S$.** The system will interact with GitHub, through proper API$_S$ calls, to be informed when a student pushes a new commit into the main branch of his/her repository.

- **Static analysis tool API$_S$**. The system will interact with a static analysis tool to automatically evaluate each student's submission.

- **Email Server**. The system will rely on an email server to correctly deliver every email notification to each student.

3.1.4 Communication Interfaces

The desktop application and the web app will communicate with CKB via an internet connection. The communication protocols that the system will use are:

- **HyperText Transfer Protocol over Secure Socket Layer (HTTPS).** The protocol is used every time data are exchanged with the external world.

3.2 Functional Requirements

3.2.1 Requirements & Goals - Requirements Mapping

The system provides various functionalities for both students and educators. In the following section, we've outlined all the identified requirements that the system must adhere to in order to ensure the achievement of its goals.

G1. All students and educators can register and then log in to the CKB platform for, respectively, participating and creating tournaments and battles.

ID	Description
R1	The system allows students to create and access into their personal account using their university credentials (email, password)
R2	The system allows educators to create and access into their personal account using their university credentials (email, password)

Table 3.1: Goal 1 requirements

G2. Students can improve their software development skills by joining the coding battles available for each tournament.

ID	Description
R3	When a new tournament is created, the system notifies the students registered on the platform through both a notification message on the platform and an email to their institutional email address
R4	The system allows every registered student to review information about all the available tournaments on the platform (i.e. whose registration period has not expired yet) and those he/she registered for, such as the title/name, the registration deadline and the name of the professor who created it
R5	The system allows a student to review information about every battle inside a tournament he/she registered for, such as the title/name, the registration and submission deadlines, the minimum and maximum number of students per group and the name of the educator who created it
R6	The system allows every registered student to enroll in a tournament for

	which the registration period has not expired yet
R7	Through both a notification message on the platform and an email to their institutional address, the system notifies all the students registered for a tournament when a new battle is created within that same tournament
R8	Until the registration deadline expires, the system allows a student to join as a singleton a battle of a tournament he/she has enrolled in, if it respects the minimum number of students per group set for that battle
R9	Until the registration deadline expires the system allows a student to join as a team a battle of a tournament he/she has enrolled in, inviting the other peers at registration time without exceeding the minimum and maximum number of students per group set for that battle
R10	The system notifies a student, through a platform notification, when he/she has been invited by a peer to join a team for taking part in a battle
R11	The system allows a student registered for a tournament to accept an invitation, in the form of a notification message on the platform, in order to join a battle within that same tournament
R12	As soon as the registration deadline for a battle expires, the system emails each participating student the GitHub repository containing the code kata
R13	The system notifies a student when a battle's final ranking is available with a message on the platform, if he/she is registered for that battle
R14	The system allows a student to view the updated ranking of a battle he/she is taking or has taken part in, even if the battle has finished or the tournament including that battle has been closed
R15	The system allows a student to view the updated score, a natural number between 0 and 100, of a battle he/she is taking or has taken part in, even if the battle has finished or the tournament including that battle has been closed
R16	Through a platform notification message, the system notifies each student enrolled in a tournament when the final ranking of that same tournament is available
R17	The system allows a student to view the updated ranking of a tournament he/she is taking or has taken part in, even if the tournament has been closed
R18	At the end of each battle, the system updates the score of the tournament containing that same battle and allows a student to view the updated score

Table 3.2: Goal 2 requirements

G3. Educators can create and manage tournaments, battles within each of them and evaluate the code submitted by the students.

ID	Description
R19	The system allows an educator to create a new tournament, specifying a name and a registration deadline
R20	The system allows an educator to create a new battle within a tournament for which he/she has the permissions to create one, specifying: ● A title/name for the tournament ● *Code Kata* ● Minimum and maximum number of students per group ● Registration deadline ● Final submission deadline ● Additional configuration for scoring
R21	At any point in time from the creation to the closure of a tournament, the system allows an educator to grant permissions to create battles in a tournament he/she has created to some of his/her colleagues
R22	The system notifies an educator when he/she has been granted permissions to create battles within a specific tournament, through a platform notification
R23	The system allows an educator to review information about all the tournaments available on the platform (i.e. whose registration deadline has not expired yet) and those for which he/she has permissions to create battles
R24	The system allows an educator to review information about all the battles created within a tournament for which he/she has the necessary permissions (title/name, registration and submission deadlines, minimum and maximum number of students per group and name of the educator who created it)
R25	The system allows an educator to view the ranking of each battle within a tournament for which he/she has the permissions to create a battle
R26	The system allows an educator to evaluate each team's source code assigning it a score
R27	The system allows an educator to view all tournaments' ranking he/she has created or for which has been given permissions to create battles
R28	The system allows an educator to close a tournament he/she has previously created

Table 3.3: Goal 3 requirements

3.2.2 Use Case Diagrams

In this section, we present some use cases of the CKB system. Use case diagrams are first illustrated to give a general and abstract view of the main actors and use cases associated with them. After that, all use cases illustrated in the diagram are described in their details in the following section.

For enhanced readability, the diagram is segmented into two parts, each specifically dedicated to a distinct actor.

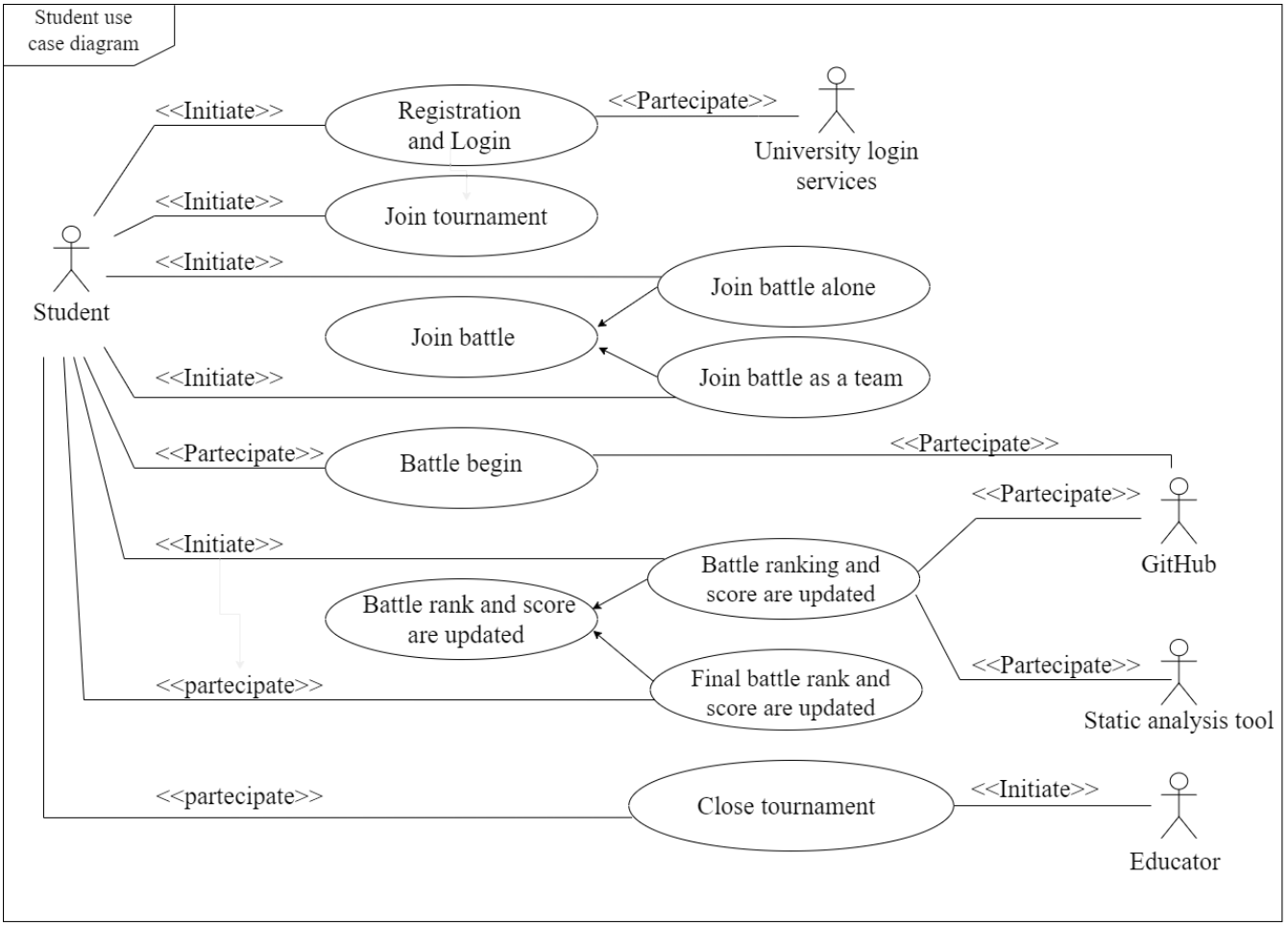


Figure 3.13: Student use case diagram

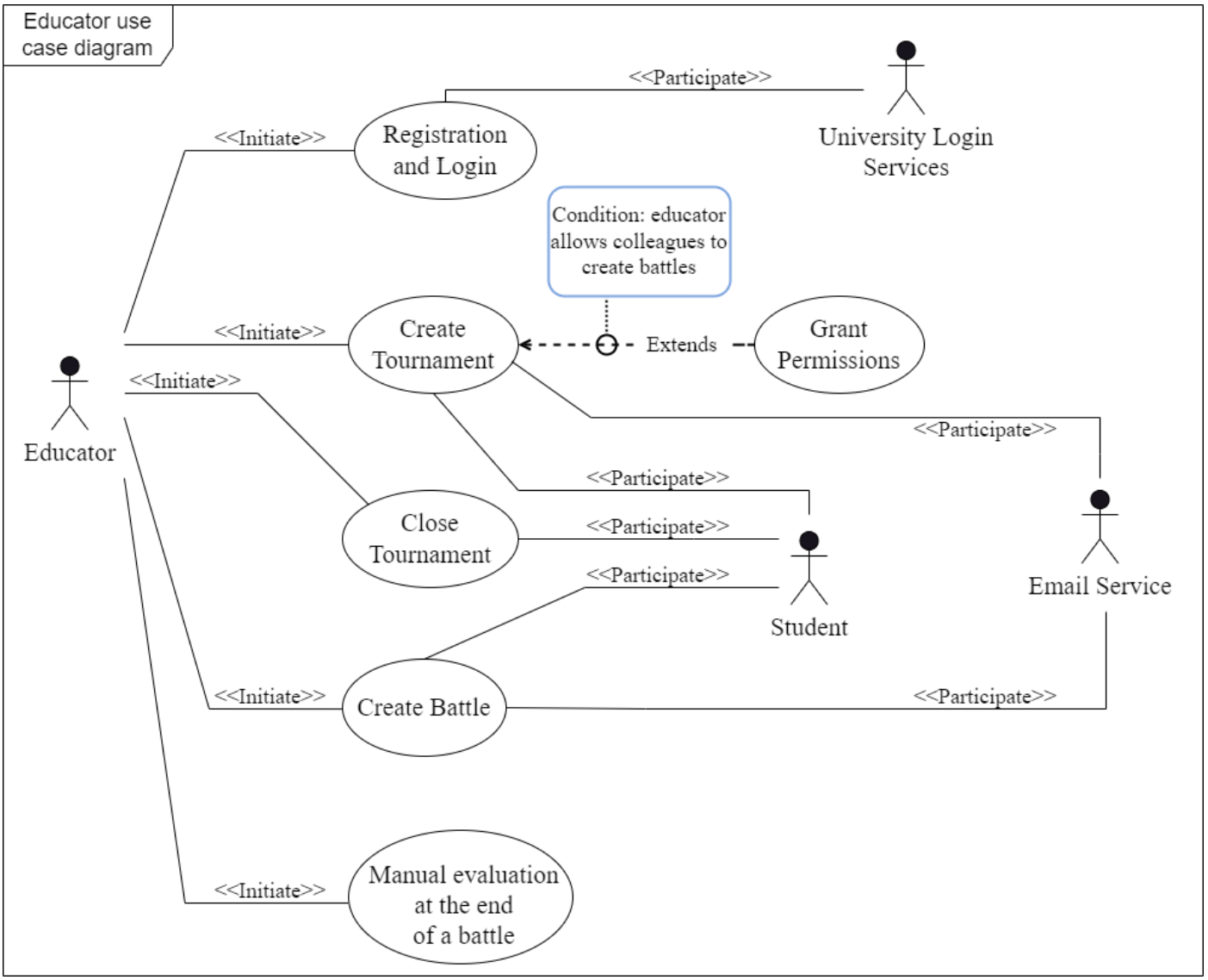


Figure 3.14: Educator use case diagram

3.2.3 Use Cases & Sequence Diagrams

UC1. Registration and login to the CKB platform as a student

Name	Registration and login to the CKB platform as a student
Actors	Student, University login services
Entry condition	• The student hasn't created an account yet
Flow of events	• The student opens the CKB platform • The student clicks on the "CKB login - Student"

	button ● The system redirects the student to the university login services ● The student fills the mandatory fields inserting his university mail and password in the university login page ● The system receives the university's approval along with some student's information, like name and email address ● The system creates a new account for the student, where it stores the user's name and email address, only if there wasn't one corresponding to that information
Exit condition	● The student is redirected to his/her personal home page
Exceptions	● The student inserts invalid university credentials An error is shown and the student is asked to insert his credentials again
Special Requirements	None

Table 3.4: Student registration and login use case

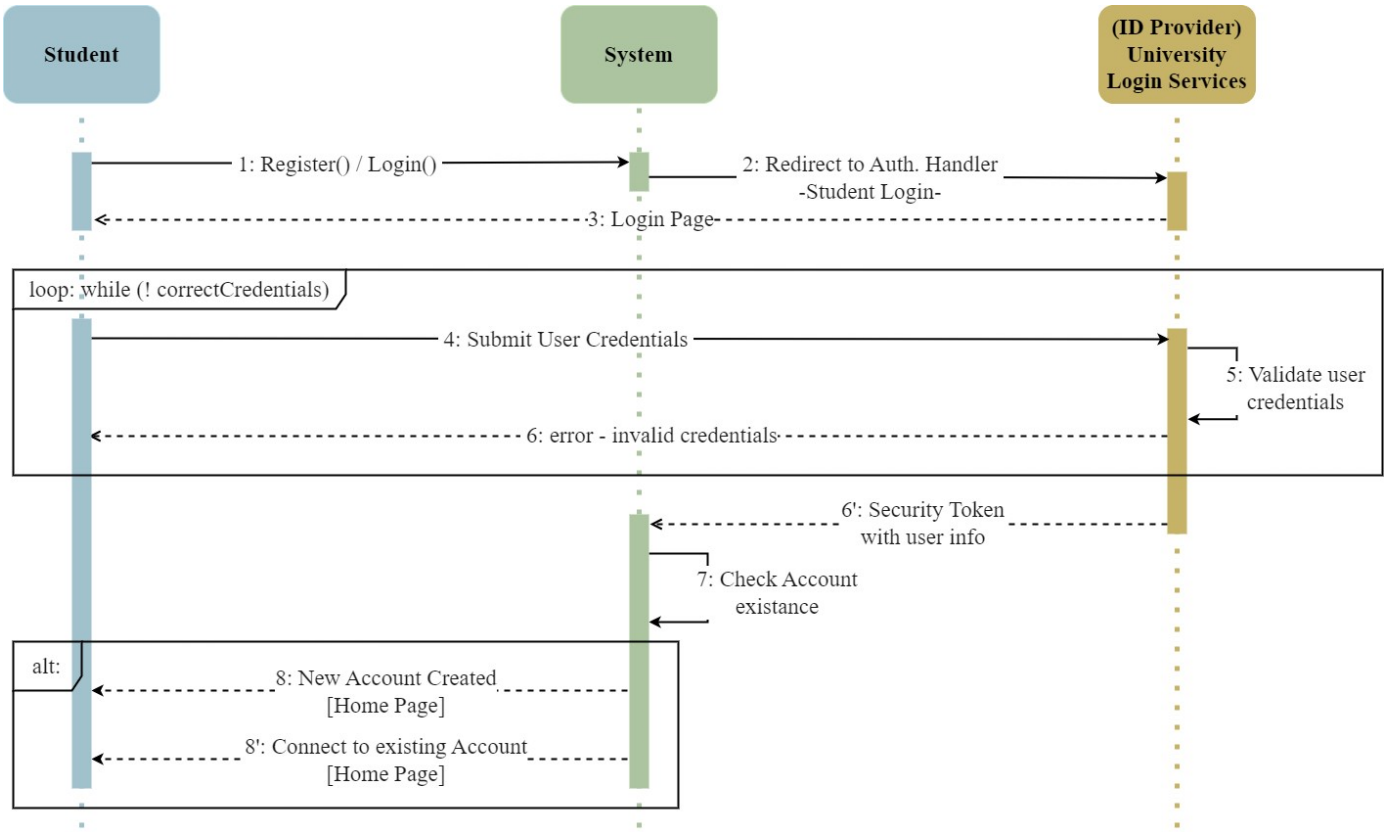


Figure 3.15: Student registration and login sequence diagram

UC2. Student registers for a tournament

Name	Student registers for a tournament
Actors	Student
Entry condition	• The student is logged into the CKB platform • The student has received both a platform and an email notification regarding the creation of the tournament • The registration period of the tournament has not expired yet
Flow of events	• The student accesses the section of the platform where all tournaments are shown and he/she finds one he/she wants to register to

	• The student opens the tournament and clicks on "join tournament" • The system receives the request, elaborates it and confirms to the student the registration
Exit condition	• The student is registered for the tournament and will be able to view the future battles created within it
Exceptions	None
Special Requirements	None

Table 3.5: Student tournament registration use case

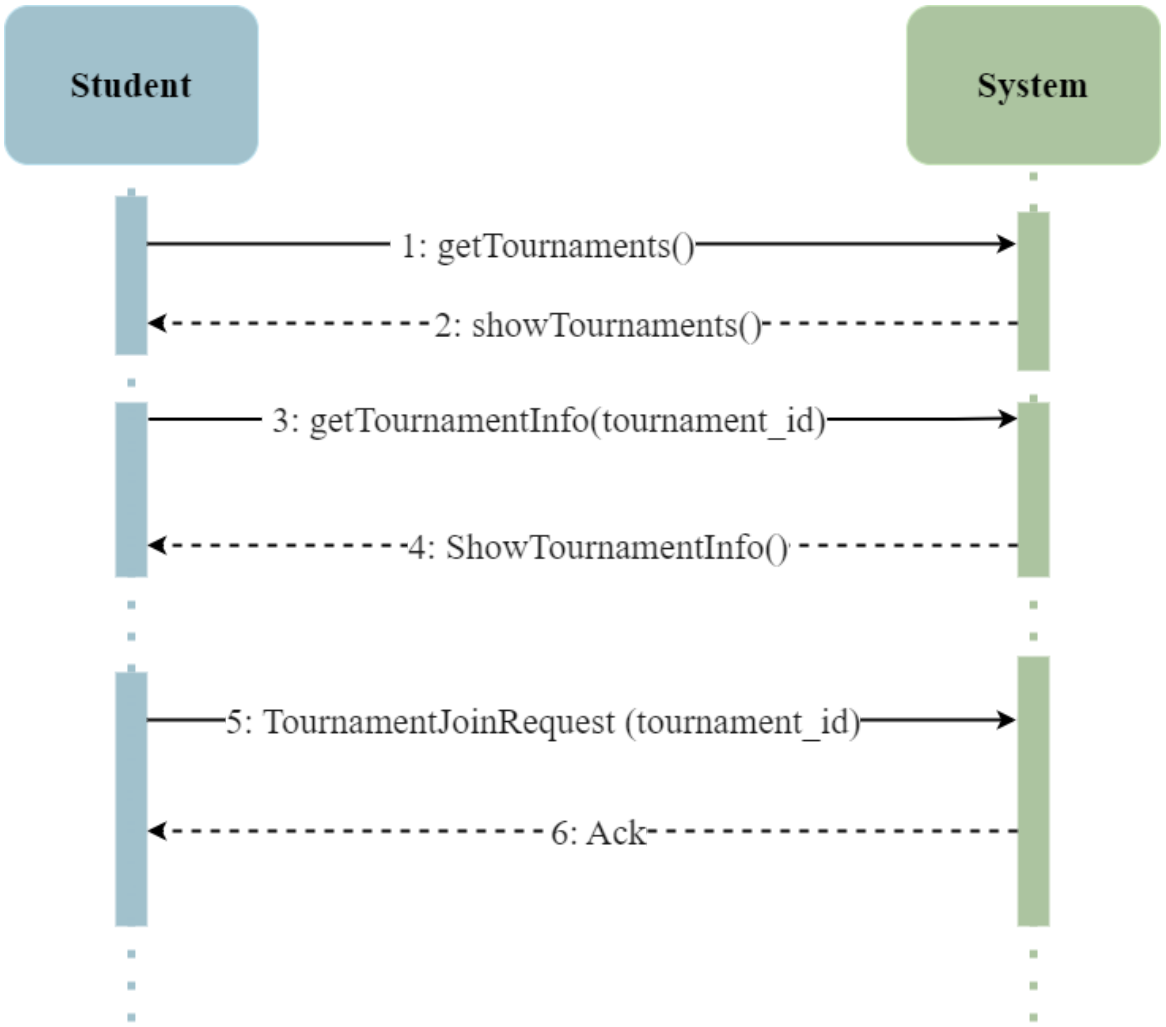


Figure 3.16: Student tournament registration sequence diagram

UC3. Student registers for a battle as a singleton

Name	Student registers for a battle as a singleton
Actors	Student
Entry condition	• The student is logged into the CKB platform • The student has received both a platform and an email notification regarding the creation of the battle in a tournament he/she is enrolled in • The registration period of the battle has not expired yet
Flow of events	• The student accesses the section of the platform where all the tournaments he/she is enrolled in are shown and he/she opens the tournament home page, where all its battles are listed • The student opens the battle and clicks on "join battle alone" without inviting any peer • The system receives the request, elaborates it and confirms to the student his/her registration
Exit condition	• The student is registered for the battle
Exceptions	• The minimum number of students per group set for that battle is greater than one In that case, the system rejects the request for the registration and notifies the student, who is asked to properly register for the battle
Special Requirements	None

Table 3.6: Student battle registration as a singleton use case

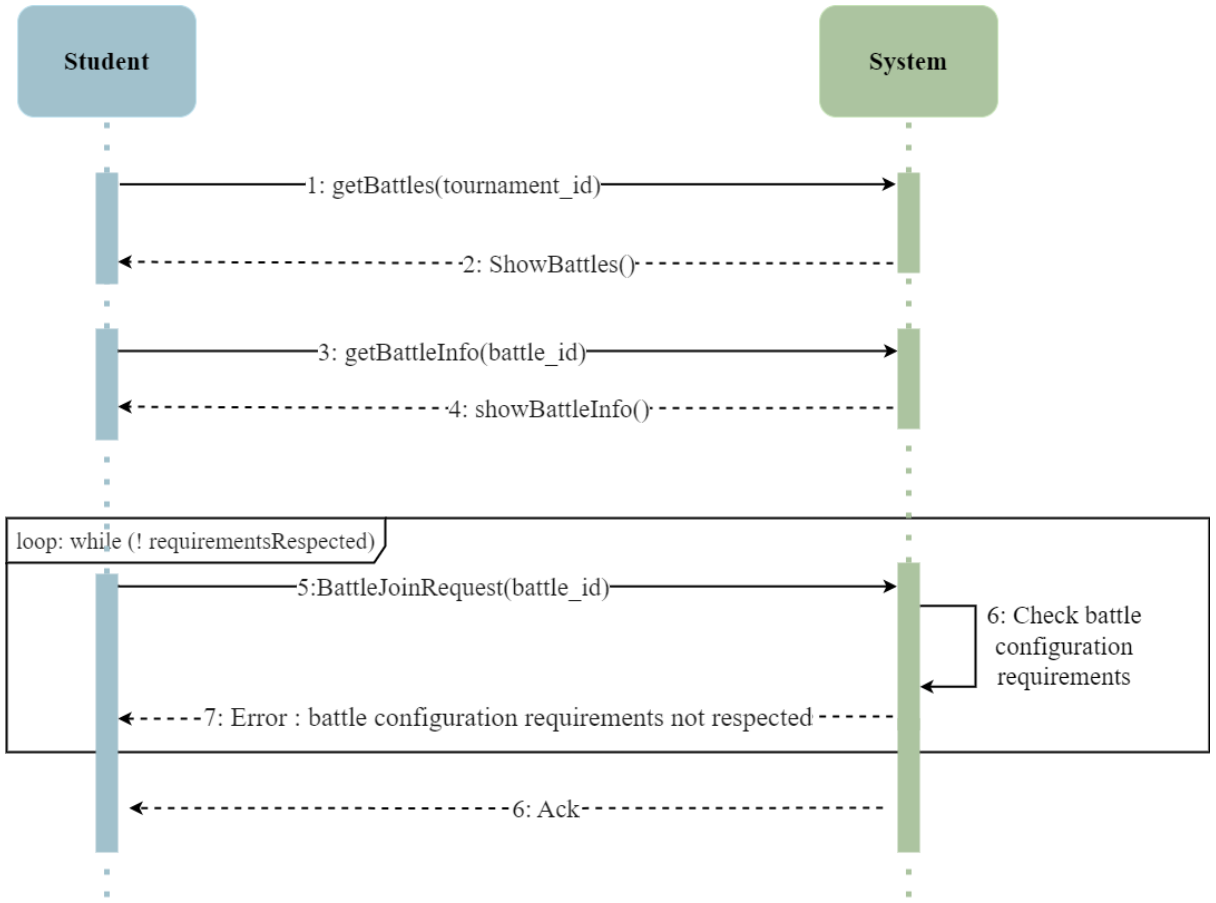


Figure 3.17: Student battle registration as a singleton sequence diagram

UC4. Student registers for a battle as a team

Name	Student registers for a battle as a team
Actors	Student
Entry condition	• The student is logged into the CKB platform • The student has received both a platform and an email notification regarding the creation of the battle in a tournament he/she is enrolled in • The registration period of the battle has not expired yet
Flow of events	• The student accesses the section of the platform where all the tournaments he/she is enrolled in are shown and he/she opens the tournament home page,

	where all its battles are listed • The student opens the battle and clicks on “join battle with peers” • The system receives the request and shows the student the invitation form • The student fills up the form with the email of the peers he wants to invite • The system receives the request, elaborates it and confirms to the student the registration of the team
Exit condition	• The student and the other peers are registered for the battle
Exceptions	• The minimum or maximum number of students per group set for that battle is not respected In that case, the system rejects the request for the registration and notifies the student, who is asked to properly register for the battle
Special Requirements	None

Table 3.7: Student battle registration as a team use case

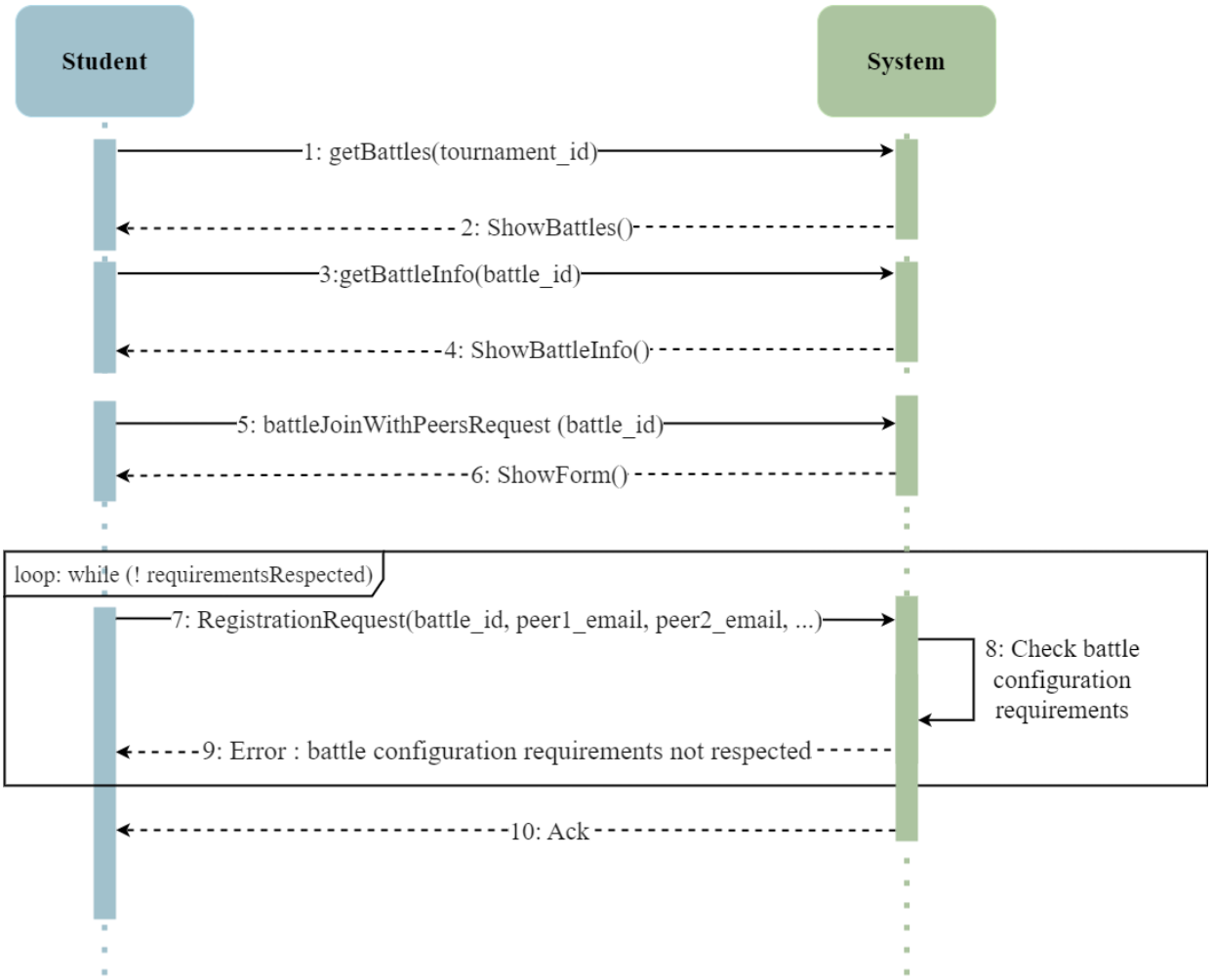


Figure 3.18: Student battle registration as a team sequence diagram

UC5. Battle begins

Name	Battle begins
Actors	Student, GitHub, Email service
Entry condition	• The student is registered for the battle • The battle registration deadline has expired
Flow of events	• The system creates a GitHub repository containing the code kata • The system emails each participating student the GitHub repository containing the code kata
Exit condition	• Each student registered for the battle has received

	the notification and can start working on the challenge
Exceptions	None
Special Requirements	None

Table 3.8: Battle begins use case

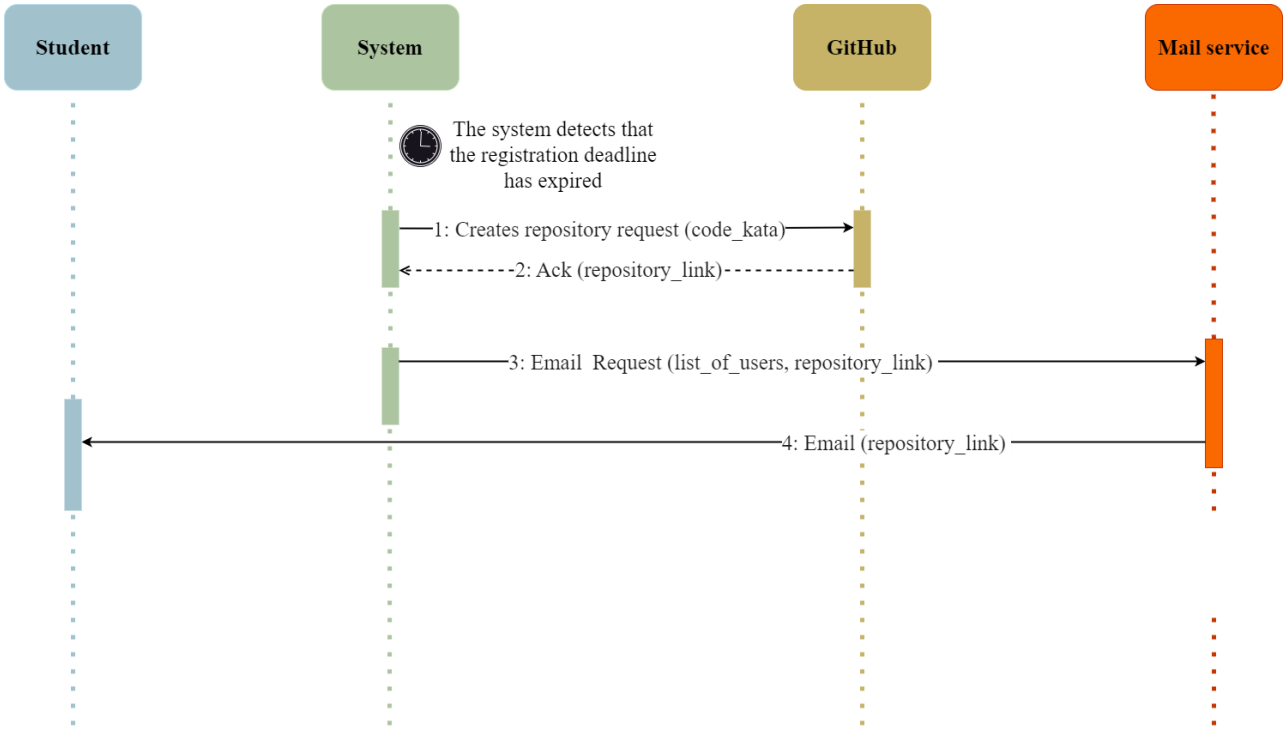


Figure 3.19: Battle begins sequence diagram

UC6. Battle ranking and score are updated

Name	Battle ranking and score are updated
Actors	Student, GitHub, Static analysis tool
Entry condition	• The student is enrolled in the battle • The student had previously correctly forked the GitHub repository containing the code kata • The student has just pushed a new commit into the main branch of his/her repository
Flow of events	• GitHub informs, through proper api calls, the

	system that a new push has just been performed • The system pulls the latest sources • The system evaluates the latest sources with the use of a static analysis tool • If the score of the team is greater than the previous one, the system updates the battle score of the team and the battle ranking
Exit condition	• The battle score of the team and the battle ranking are updated and each team member can view them
Exceptions	None
Special Requirements	None

Table 3.9: Battle ranking and score are updated use case

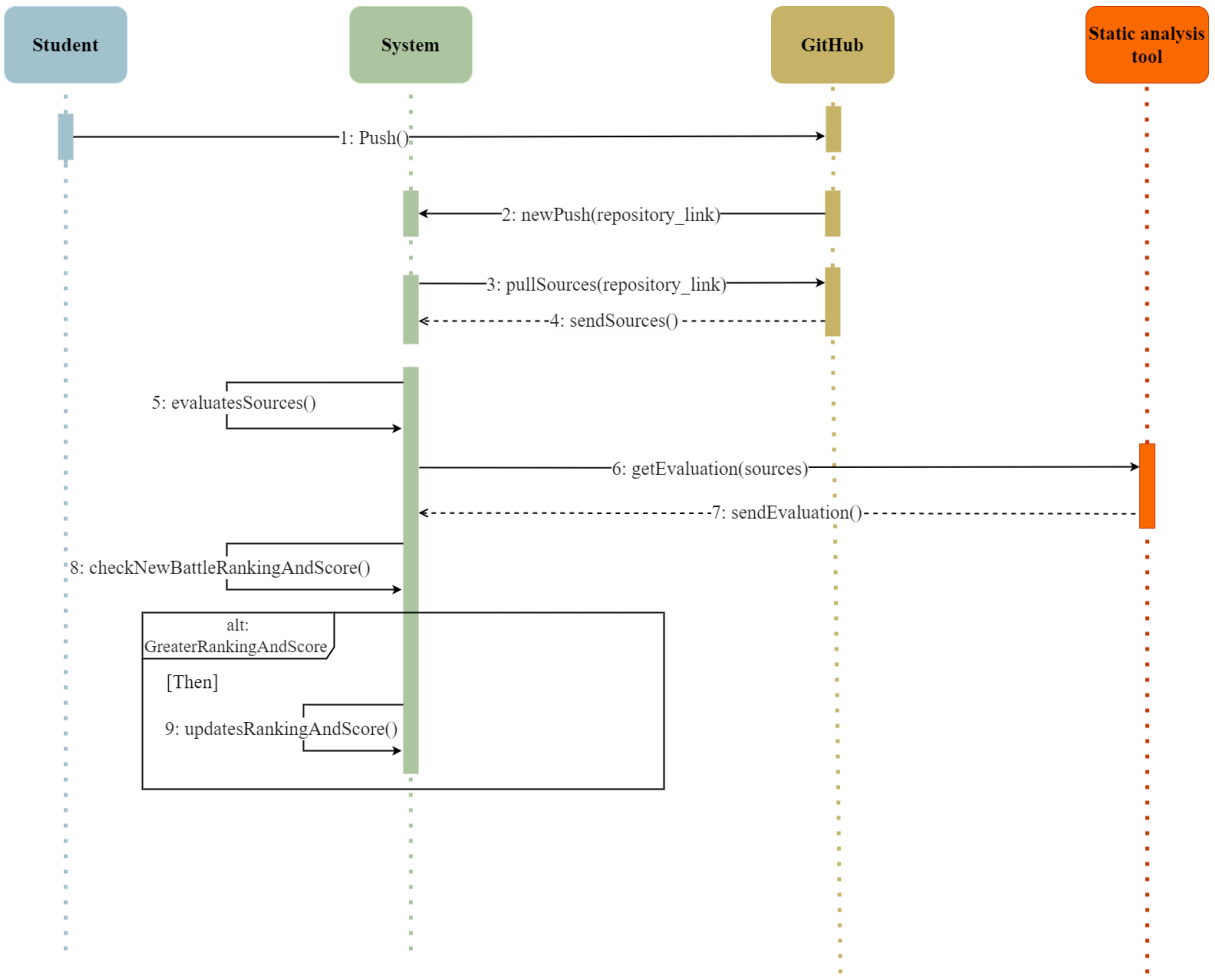


Figure 3.20: Battle ranking and score are updated sequence diagram

UC7. Final battle ranking and score are updated

Name	Final battle ranking and score are updated
Actors	Student
Entry condition	• The student is enrolled in the battle • The submission deadline of the battle has expired
Flow of events	• Consolidation stage: the system computes the final ranking of the battle adding, if needed, the scores given by the educator during the manual evaluation • The system notifies all the students enrolled in that battle that the final ranking is available through a platform notification
Exit condition	• The final battle score of the team and battle ranking are updated and each team member can view them
Exceptions	None
Special Requirements	None

Table 3.10: Final battle ranking and score are updated use case

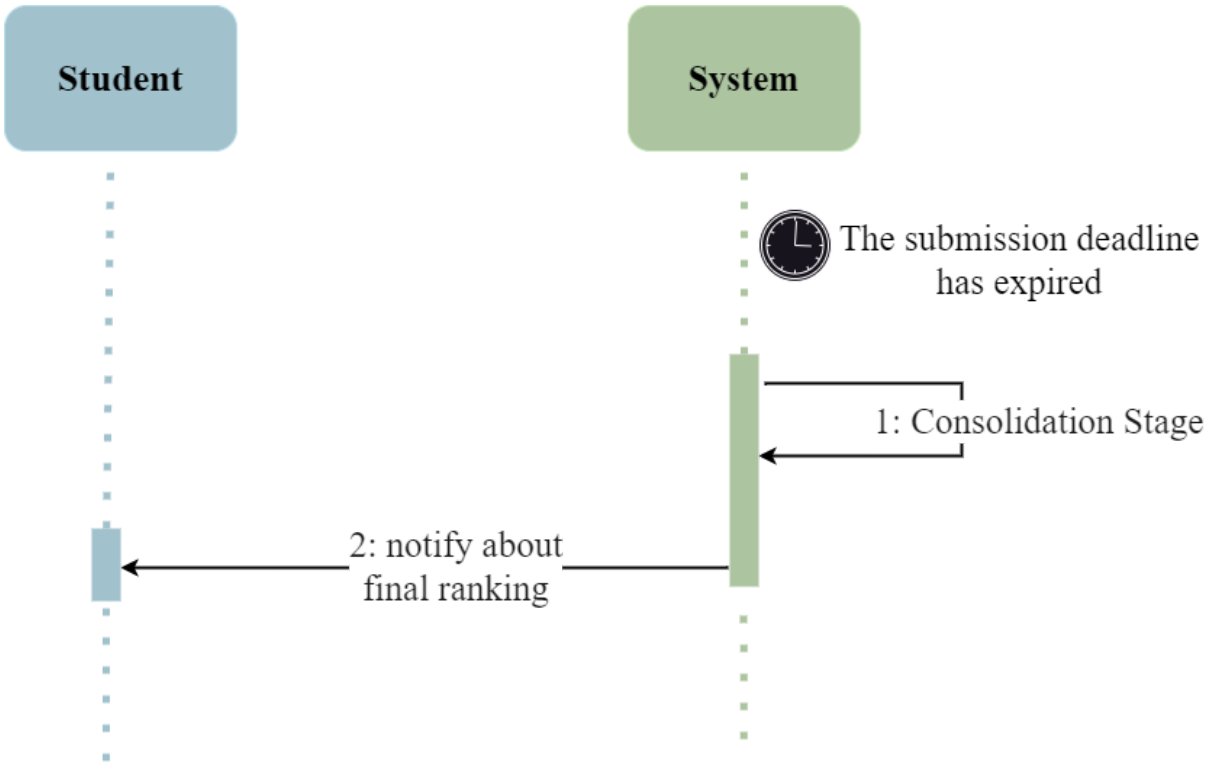


Figure 3.21: Final battle ranking and score are updated sequence diagram

UC8. Registration and login to the CKB platform as an educator

Name	Registration and login to the CKB platform as an educator
Actors	Educator, University login services
Entry condition	• The educator hasn't created an account yet
Flow of events	• The educator opens the CKB platform. • The educator clicks on the "CKB login - Educator" button. • The system redirects the educator to the university login services. • The educator fills the mandatory fields inserting his university mail and password in the university login page. • The system receives the university's approval along with some educator's information, like name and email address. • The system creates a new account for the educator, where it stores the user's name and email address, only if there wasn't one corresponding to that information.
Exit condition	• The educator is redirected to his/her personal home page.
Exceptions	• The educator inserts invalid university credentials. An error is shown and the educator is asked to insert his credentials again.
Special Requirements	None

Table 3.11: Educator registration and login use case

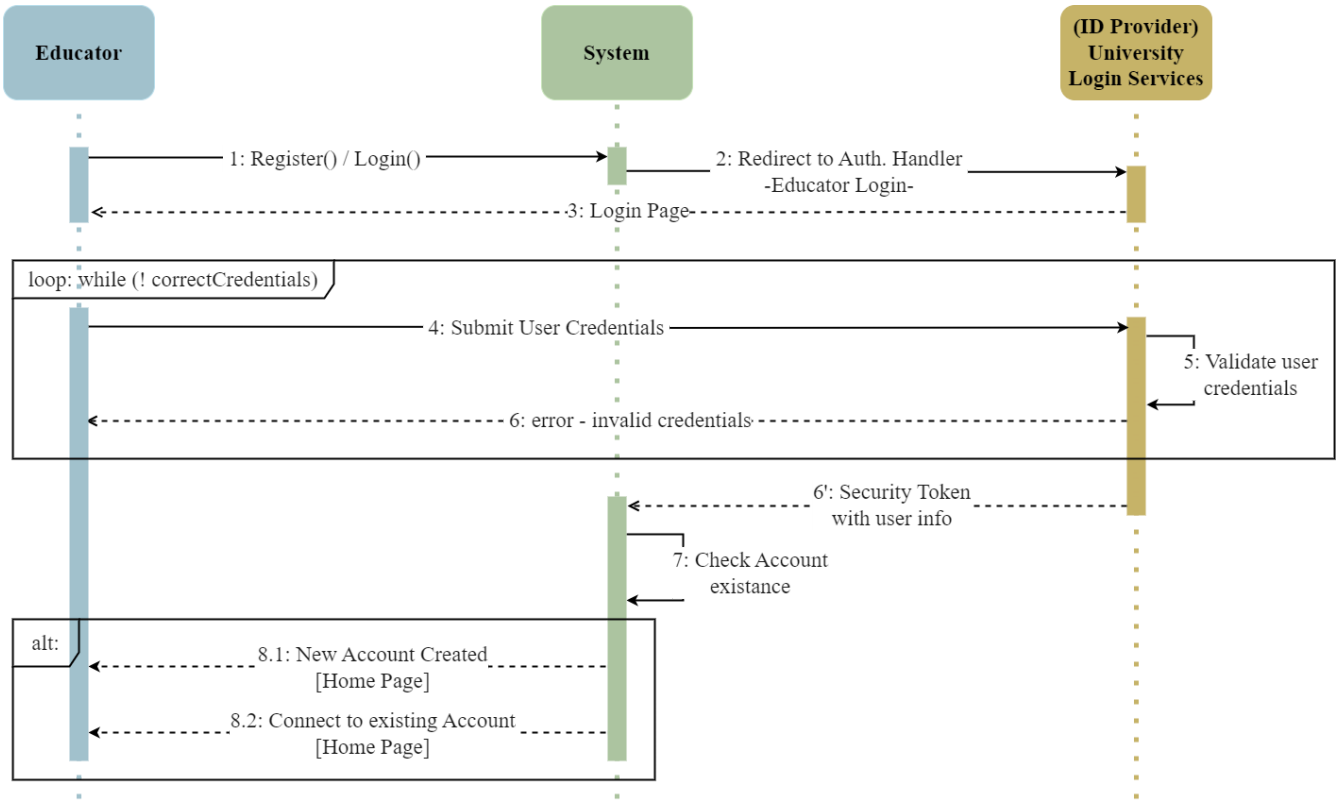


Figure 3.22: Educator registration and login sequence diagram

UC9. Educator creates a tournament

Name	Educator creates a tournament
Actors	Educator, Student, Email Service
Entry condition	• The educator is logged into the CKB platform • The educator clicks on the “create new tournament” button

Flow of events	• The system asks the educator for title/name for the tournament and a registration deadline
	• The system gives the educator the possibility to grant permissions to create battles to other colleagues
	  ○ “maybe later” button
	  ○ Choose educators to be granted permissions
	• The tournament is created and made available on the platform

	• The system sends a notification message and an email to all students who are registered to the CKB platform
Exit condition	• All the students who have a CKB account can now register for the newly created tournament
Exceptions	• The educator inserts erroneous data The system will prevent the educator from creating the tournament
Special Requirements	None

Table 3.12: Creation of a tournament use case

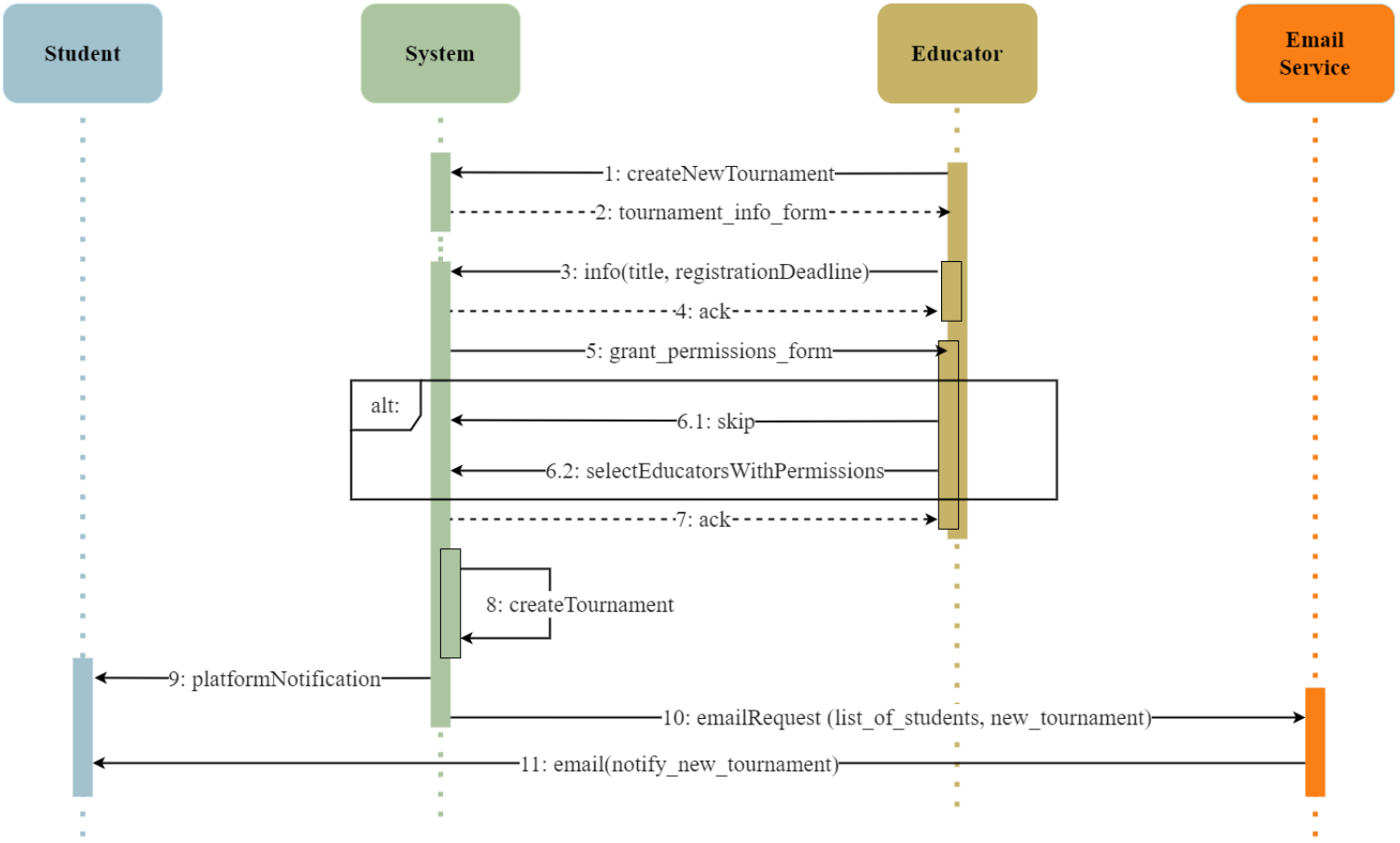


Figure 3.23: Creation of a tournament sequence diagram

UC10. Educator grants colleagues the permissions to create battles

Name	Educator grants colleagues the permissions to create battles
Actors	Educator
Entry condition	• The educator is logged into the CKB platform • The educator selects a tournament he/she created • The educator clicks on the “Grant Permissions” button
Flow of events	• The system opens a tab with a list of educators and a search bar • The educator can select the colleagues from the list or look for them by inserting their name or email address • The system returns the account of the required colleague if it exists • The educator can flag the returned account and after repeating the process for all the colleagues he/she wants to grant permissions to, submits the choices • The system processes the informations and notify all the selected colleagues through a platform notification message
Exit condition	• The selected colleagues can now create battles inside the tournament and they have all been notified about this
Exceptions	• The colleague the educator is looking for does not have an account on the CKB platform The system will show an error message telling that there is no user with that name or email address.
Special Requirements	None

Table 3.13: Educator grants permissions to create battles use case

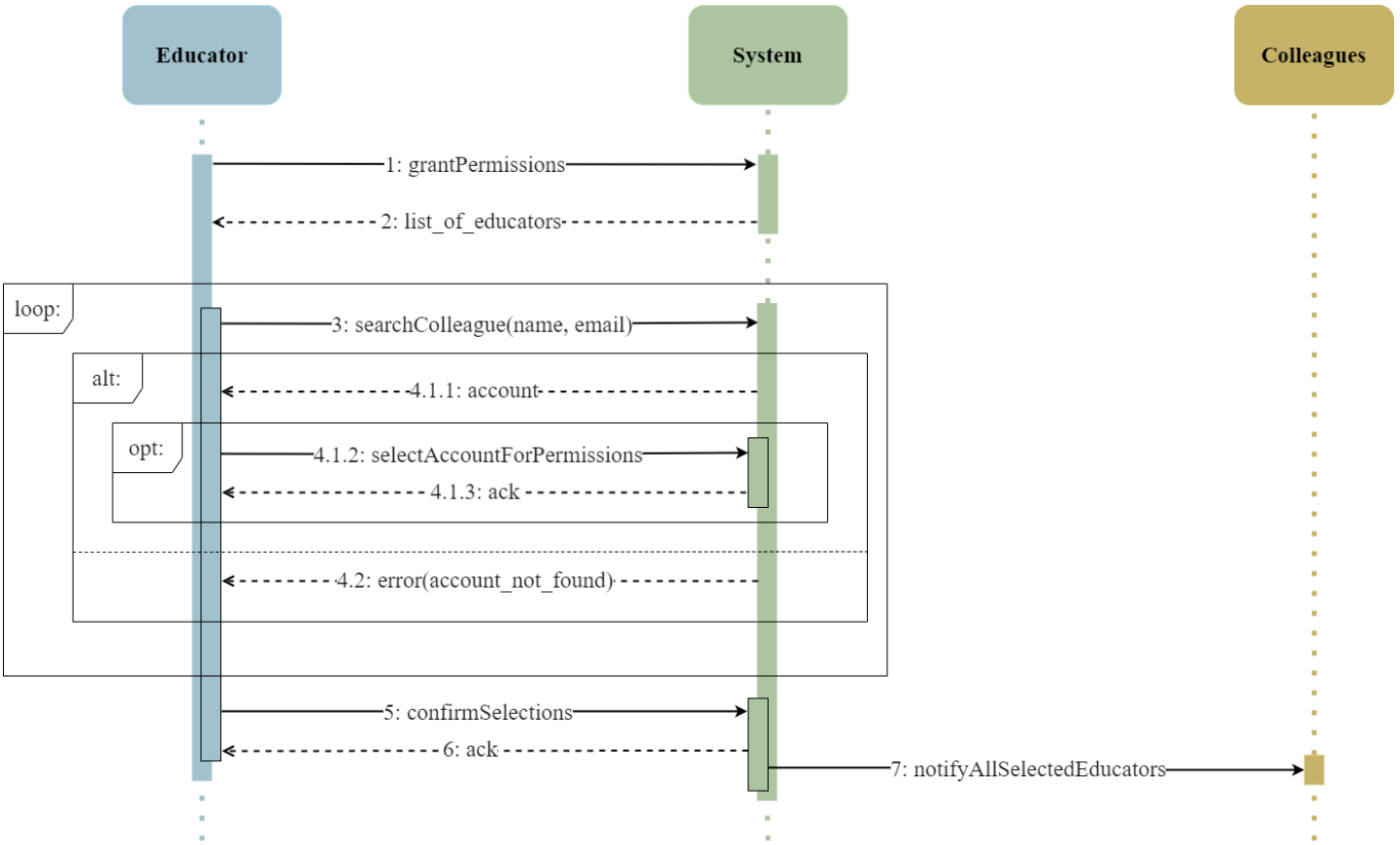


Figure 3.24: Educator grants permissions to create battles sequence diagram

UC11. Educator creates a battle

Name	Educator creates a battle
Actors	Educator, Student, Email Service
Entry condition	• The educator is logged into the CKB platform • The educator selects one of his/her tournaments • The registration deadline of the tournament has expired • The educator clicks on the button “create new battle”

Flow of events	• The system asks the educator to provide all the necessary information and material:
	  ○ A title/name for the battle
	  ○ The *Code Kata*

	○ The minimum and maximum number of students a team can consist of ○ The registration deadline ○ The submission deadline ○ Additional configuration for scoring • The system creates the new battle with all the specified information • The system sends a notification via both email and platform message to all the students registered for the tournament
Exit condition	• All the students registered for the tournament can now enrol in the battle
Exceptions	• The educator inserts erroneous data for the battle (e.g. does not provide the *code kata*) The system will prevent the creation of the new battle notifying the educator about the error and wait for the insertion of the correct information
Special Requirements	None

Table 3.14: Creation of a battle use case

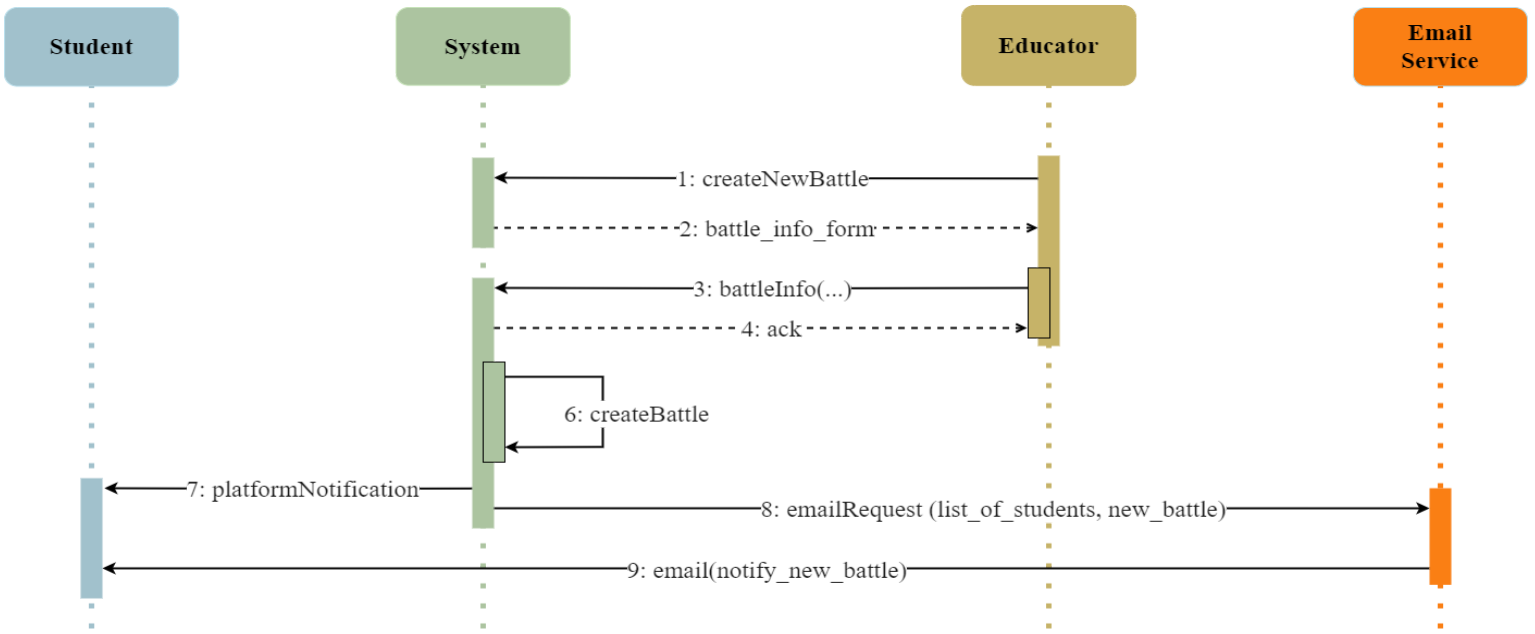


Figure 3.25: Creation of a battle sequence diagram

UC12. Manual Evaluation at the end of a battle

Name	Manual Evaluation at the end of a battle
Actors	Educator, Student
Entry condition	• The educator is logged into the CKB platform • The educator selects one of his/her tournament • The eductor chooses a battle whose submission deadline has expired • At the creation of the chosen battle the educator must have added the manual evaluation to the scoring configurations
Flow of events	• The educator clicks on “evaluate” • The system shows to the educator the list of the teams who participated to the battle and for each of them the last submitted source and an input field for the educator to enter the personal evaluation • The educator enters the scores (between 0 and 100) • The system merge the educator’s evaluation with the scores coming from the automatic analysis of the sources • The system notifies all the students who took part in the battle that the final battle ranking is available through a platform notification message
Exit condition	• The final battle ranking is available • Each student who took part to the battle has been notified
Exceptions	None
Special Requirements	None

Table 3.15: Manual evaluation of battle use case

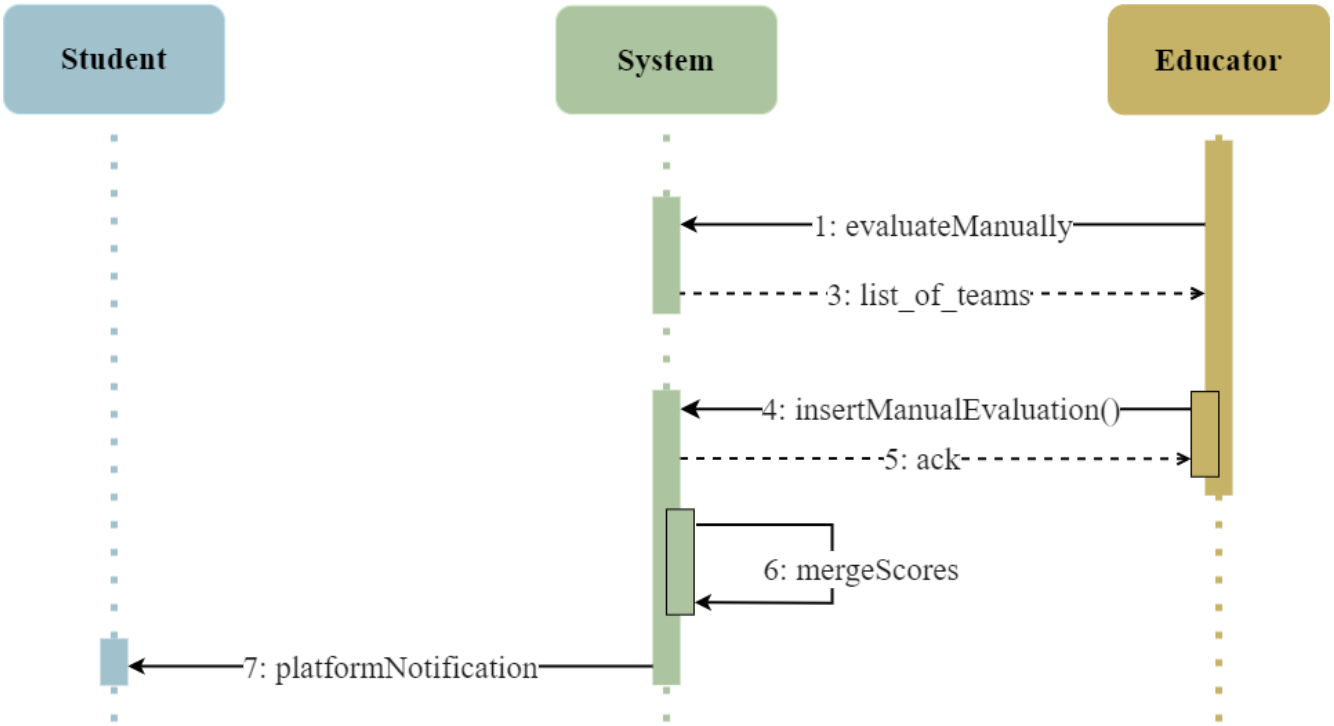


Figure 3.26: Creation of a battle sequence diagram

UC13. End of tournament

Name	End of tournament
Actors	Educator, Student
Entry condition	• The student is enrolled in the tournament • The educator is logged in • The educator is the creator of the tournament
Flow of events	• The educator closes the tournament by clicking on the “close tournament” button • The system computes the final ranking of the tournament • The system notifies all the students enrolled in that tournament that the final ranking is available through a platform notification

Exit condition	• The final tournament scores and ranking are updated and each student can view them
Exceptions	None
Special Requirements	None

Table 3.16: End of tournament use case

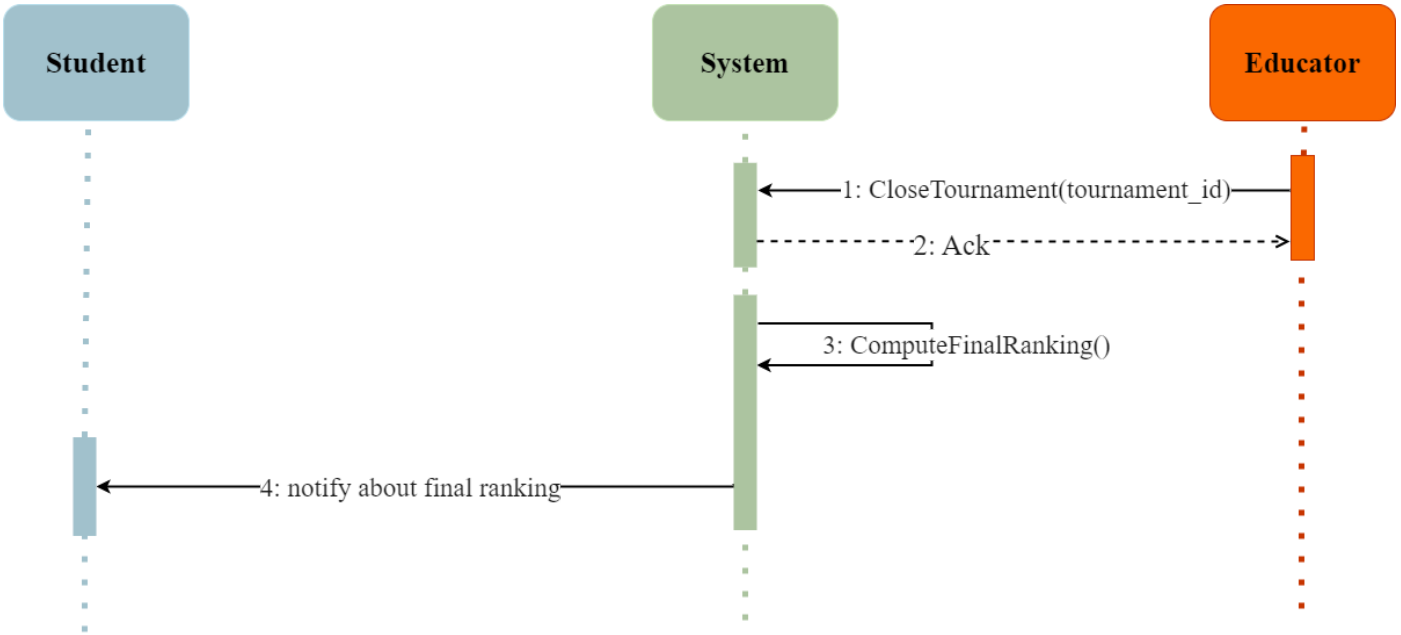


Figure 3.27: End of tournament sequence diagram

3.3 Performance Requirements

The system has to be able to serve a great number of users simultaneously with a low response time. Here we present the constraints for the main key aspects of the CKB platform:

- The system should be able to retrieve and present to a student the list of the tournaments he is enrolled in within 1s
- The system should be able to retrieve and present to a student the list of all available tournaments within 3s

- The system should confirm to a student both the registration for a tournament and for a battle within 1s
- At the end of each battle, the system should update and present the tournament ranking and score of each team within 5s
- The system should be able to create and post a new tournament within 5s
- The system should be able to create and post a new battle within 10s

3.4 Design Constraints

3.4.1 Standards compliance

There are no specific measures to adopt, the CKB project is subject to the General Data Protection Regulation (GDPR), a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA).

3.4.2 Hardware Limitations

The only requirements to access and use the CKB platform are:

- A smartphone and/or a computer (of all operating systems, e.g. Windows, Mac Os, Linux, Android) with an updated web browser to access the web app and/or to download the desktop application
- An internet connection to access the browser and/or to download the desktop application

3.4.3 Any Other Constraint

There are no other particular constraints.

3.5 Software System Attributes

3.5.1 Reliability

The system must be able to run without any interruptions for long periods. To achieve so and in order to be fault tolerant, the system backend deployment must use some sort of replication and redundancy.

3.5.2 Availability

Given that CKB is not an emergency service but it mainly represents a platform for students to practise and that it relies on the university login services to authenticate users, the system must provide an availability of 99%, which means that the average downtime should around 3.65 days/year and it could be contemporary with the downtime period of the university login services.

3.5.3 Security

The system deals with sensitive information, so the security aspect is an important one. Since the authentication process heavily relies on the university login services, the CKB system will store the minimum amount of possible information for each user and, among this information, there won't be any password, hence it's not required to encrypt it in the database. On the other hand, it is of paramount importance that, to communicate over the internet, CKB must use some sort of encryption to avoid traffic sniffing and spoofing to guarantee the users' privacy.

3.5.4 Maintainability

The system must guarantee a high level of maintainability: it should be designed in such a way that future additions require the minimum effort, hence appropriate design patterns should be used along with good standards. In addition, every implemented functionality has to be well documented.

3.5.5 Portability

The system will be available as a web app accessible both from the chrome browser for Android devices and the Safari browser for iOS devices. Furthermore, the desktop application must run on every OS (e.g. Windows, Mac OS, Linux)

4 | Formal Analysis Using Alloy

In this section we include the alloy model of CKB underlining the following dynamic aspects which represent, in our opinion, the core points and features of the CKB platform:

- Addition of an assistant to a tournament
- Student registering for a tournament during its registration period
- Student joining a team to take part in a battle
- Battle added to a tournament

4.1 Signatures & Functions

```
enum TournamentState {OpenRegistration, OnGoing, Closed}
enum BattleState {Registration, Active, Consolidation, Ended}

abstract sig User{}

sig Student extends User{}

/*All Educators could be owners of a tournament and colleague with permissions in another tournament*/
sig Educator extends User{}

sig Tournament{
	var state: one TournamentState,
	creator: one Educator,
	var assistants: set (Educator - creator), /*The colleagues with permissions to create a battle*/
	var battles: set Battle,
	var registered: set Student
}

sig Battle{
	var state: one BattleState,
	var teams: set Team,
	creator: one Educator
}

sig Team{
	var components: disj some Student,
	score: disj one Score
}

sig Score {
	var value: disj one Int
}{
	value >= 0 and value <= 100
}

/************** FUNCTIONS ************/

/* This function returns the state of a tournament */
fun tournamentState [t: Tournament] : one TournamentState {
	t.state
}

/* This function returns the state of a battle */
fun battleState [b: Battle]: one BattleState {
	b.state
}
```

4.2 Facts

```
/************** FACTS  ****************/
/* An educator can't be both a creator and an assistant for a Tournament, and reinforces
   what the 'assistants' field in the Tournament signature already states
*/
fact NoCreatorIsAssistant {
        all t: Tournament | t.creator not in t.assistants
}

/* A tournament has no battle during the registration period*/
fact tournamentHasNoBattleInRegistrationPeriod {
        all t: Tournament | tournamentState[t] = OpenRegistration
        implies
        no t.battles
}

/*A Closed tournament can only contain Ended battles*/
fact closedTournamentMeansEndedBattles{
        all t: Tournament | tournamentState[t] = Closed
        implies
                all b: Battle | b in t.battles
                implies
                battleState[b] = Ended
}

/* The creator of a battle is either the creator of the tournament
   containing the battle or an educator who has been granted the permissions to create one*/
fact creatorOfBattle{
        all b: Battle, t:Tournament | b in t.battles
        implies
        (b.creator in (t.creator + t.assistants))
}

/*The students in the teams participating to a battle must also be registered to the Tournament
containing that battle*/
fact correctRegistration{
        all b: Battle, t: Team, s: Student | s in t.components and t in b.teams
        implies
        one t: Tournament | b in t.battles and s in t.registered
}

/*A team exist only in the context of a battle:
  there can't be a team that does not participate to any battle*/
fact noTeamsOutsideBattles{
        all t: Team | one b: Battle | t in b.teams
}
```

```
/*A battle exists only in the context of a tournament:
   there can't be a battle which is not contained into a tournament*/
fact noBattleOutsideTournaments{
        all b: Battle | one t: Tournament | b in t.battles
}

/*A score exists only in the context of a Team:
  there can't be a score that is not relted to any team*/
fact noScoreWithoutTeam{
        all sc: Score | one t: Team | sc in t.score
}

/*If an educator is creator of a tournament he will always be*/
fact{
        all t: Tournament, c: Educator| c = t.creator
        implies
        after always t.creator = c
}

/*If an educator is assistant of a tournament he will always be*/
fact{
        all t: Tournament, a: Educator| a in t.assistants
        implies
        after always a in t.assistants
}

/*If a battle belongs to a turnament it will always do*/
fact{
        all b: Battle, t: Tournament | b in t.battles
        implies
        after always b in t.battles
}

/*If a student is part of a team he will always be*/
fact{
        all s: Student, t: Team | s in t.components
        implies
        after always s in t.components
}

/*If a student is registered for a tournament, he will always be*/
fact{
        all s: Student, t: Tournament | s in t.registered
        implies
        after always s in t.registered
}
```

```
/*If a team is registered for a battle, it will always be */
fact{
        all t: Team, b: Battle | t in b.teams
        implies
        after always t in b.teams
}

//Life cycle of a tournament
/* OpenRegistration → OnGoing ...*/
fact{
        all t: Tournament |
                always ( tournamentState[t] = OpenRegistration
                        implies
                        historically tournamentState[t] = OpenRegistration and
                        eventually tournamentState[t] = OnGoing)
}
// ...OnGoing → Closed
fact{
        all t: Tournament |
                always (tournamentState[t] = Closed
                        implies
                        once tournamentState[t] = OnGoing and
                        after always tournamentState[t] = Closed)
}

// Life cycle of a battle..
// Registration → Active
fact{
        all b: Battle |
                always (battleState[b] = Registration
                        implies
                        historically battleState[b] = Registration and
                        eventually battleState[b] = Active)
}
// Active → Consolidation
fact{
        all b: Battle |
                always (battleState[b] = Active
                        implies
                        eventually battleState[b] = Consolidation)
}
//Consolidation → Ended
fact{
        all b: Battle |
                always (battleState[b] = Consolidation
                        implies
                        eventually battleState[b] = Ended)
}
// Ended
fact{
        all b: Battle |
                always (battleState[b] = Ended
                        implies
                        once battleState[b] = Consolidation and
                        after always battleState[b] = Ended)
}
```

4.3 Assertions

```
/************** ASSERTIONS ****************/

/*This assert checks that there is no battle that has as its creator an educator who is not the creator
  of the tournament or does not have permissions for the tournament to which that battle belongs*/
assert noCreationWithoutPermissions {
        all b: Battle, t: Tournament | b in t.battles
        implies
        b.creator in ( t.creator + t.assistants)
}

/*This assert checks that there aren't students registered to a battle without being registered
   for the tournament containing the battle*/
assert noWrongRegistrations {
        all b: Battle, t: Team, s: Student | s in t.components and t in b.teams
        implies
        one t: Tournament | b in t.battles and s in t.registered
}
```

4.4 Predicates

```
/********* PREDICATES ***************/

/* Assistant added to a tournament */
pred addAssistantToTournament [t:Tournament, e: Educator]{
        e not in t.assistants
        after (e in t.assistants and tournamentState[t] != Closed)
}

/* Student registers to a tournament whose state is OpenRegistration */
pred registerStudentToTournament[t: Tournament, s: Student] {
        s not in t.registered
        after (s in t.registered and tournamentState[t] = OpenRegistration)
}

pred studentJoinTeam[t: Team, s: Student, b: Battle]{
        s not in t.components and t in b.teams
        after (s in t.components)
}

pred addBattleToTournament[t: Tournament, b: Battle] {
        tournamentState[t] = OnGoing and b not in t.battles
        after ( tournamentState[t] = OnGoing and b in t.battles)
}

pred world {
        #Team = 3
        #Student > 3
        #Tournament = 2
        #Battle > 2
}
```

4.5 Results

```
run world for 7 but 8 Int

run addAssistantToTournament for 3 but 1 Tournament

run registerStudentToTournament for 3 but 1 Tournament

run studentJoinTeam for 8 Int

run addBattleToTournament for 3 but 2 Tournament, 8 Int


check  noCreationWithoutPermissions

check noWrongRegistrations
```

```
7 commands were executed. The results are:
  #1: Instance found. world is consistent.
  #2: Instance found. addAssistantToTournament is consistent.
  #3: Instance found. registerStudentToTournament is consistent.
  #4: Instance found. studentJoinTeam is consistent.
  #5: Instance found. addBattleToTournament is consistent.
  #6: No counterexample found. noCreationWithoutPermissions may be valid.
  #7: No counterexample found. noWrongRegistrations may be valid.
```

Figure 4.1: Formal analysis results

4.5.1 Generated World

Here we include a few different images showing the generated worlds.

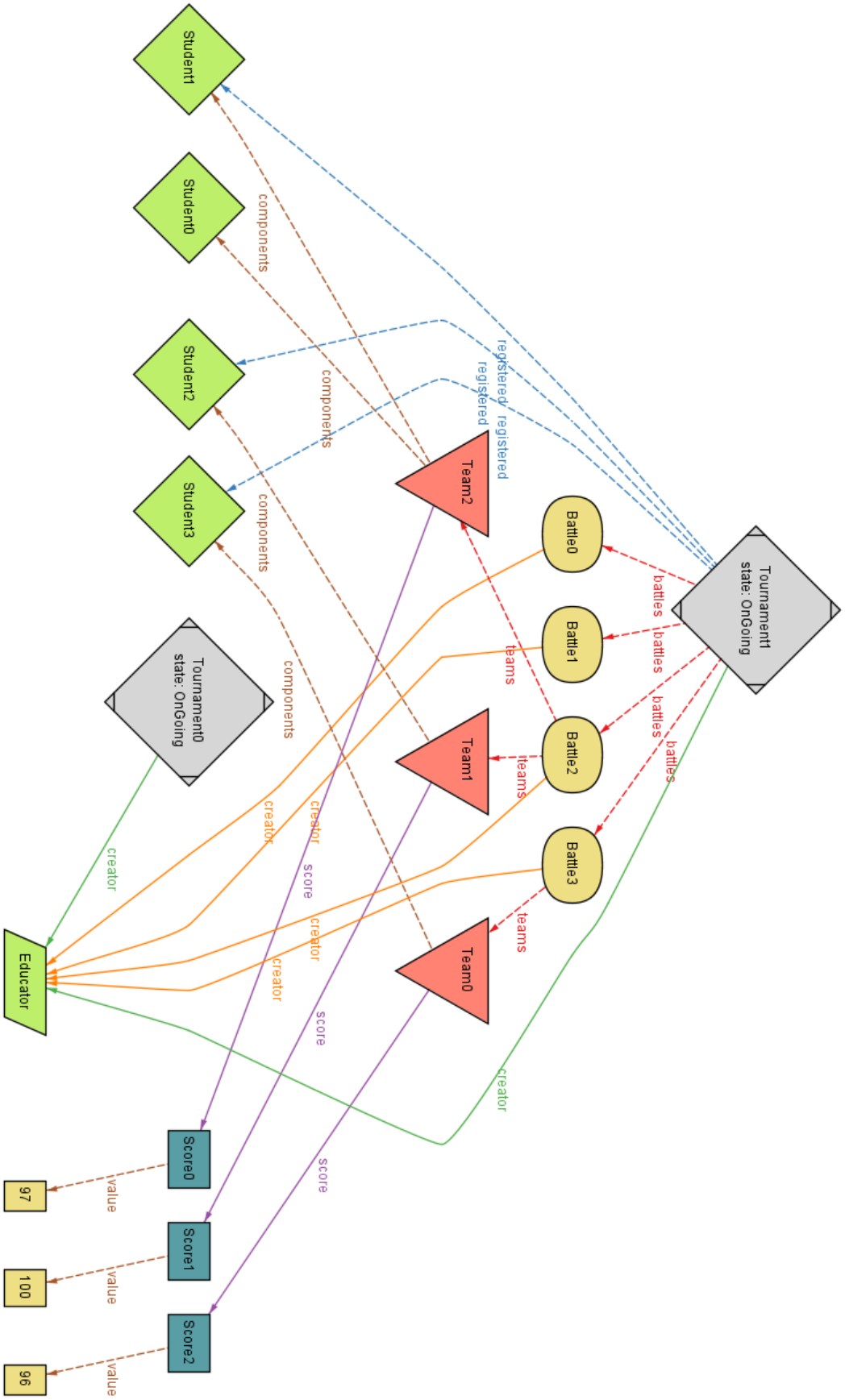


Figure 4.2: First model

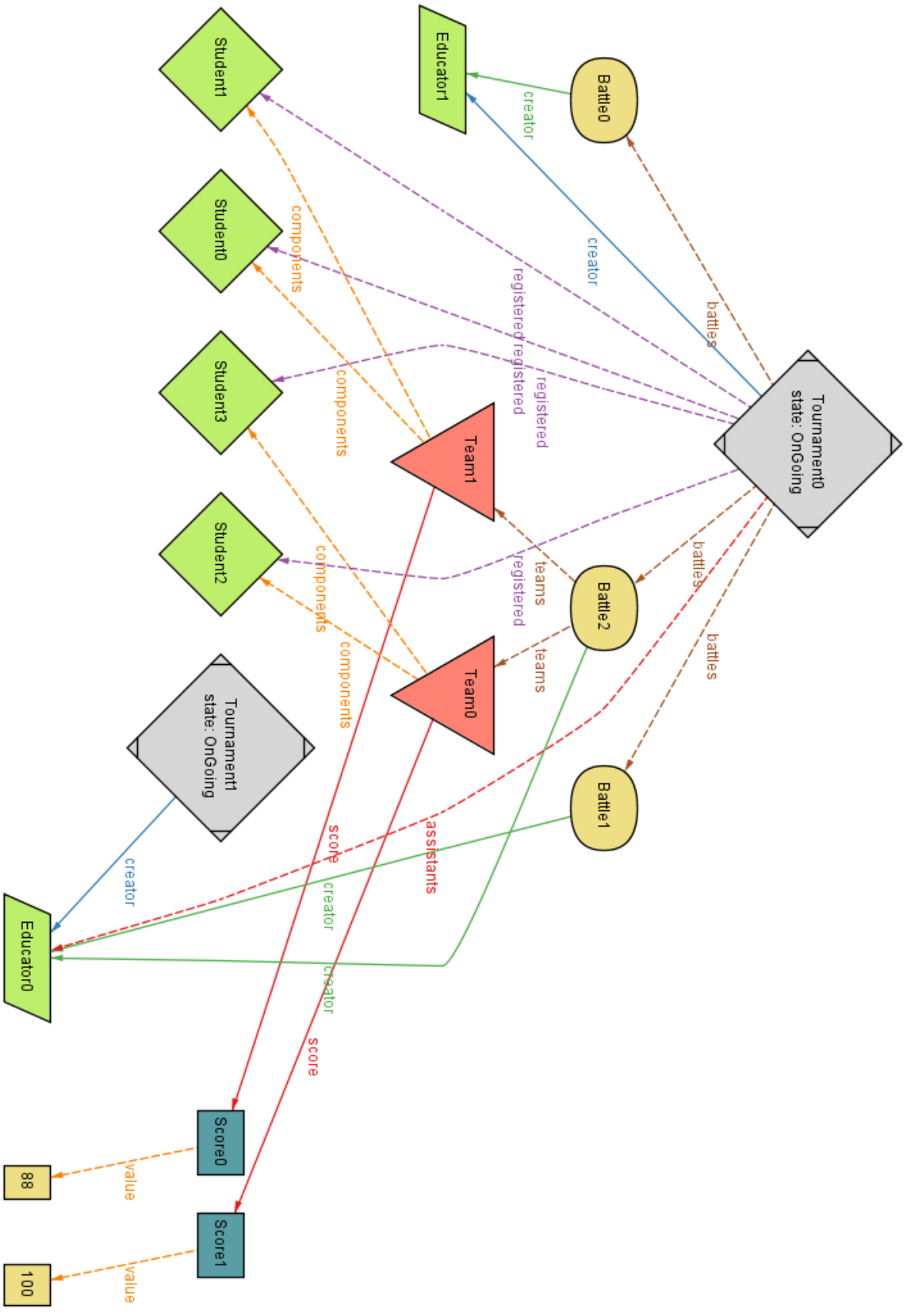


Figure 4.3: Second model

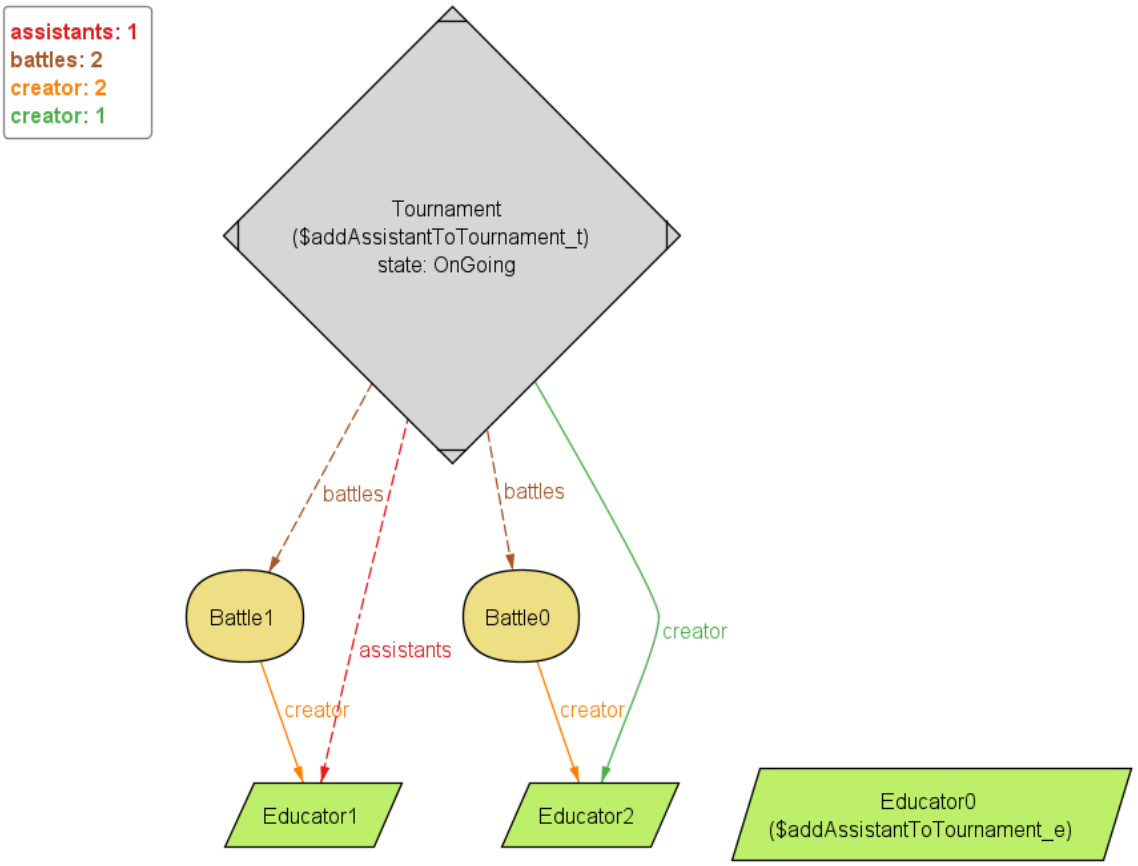
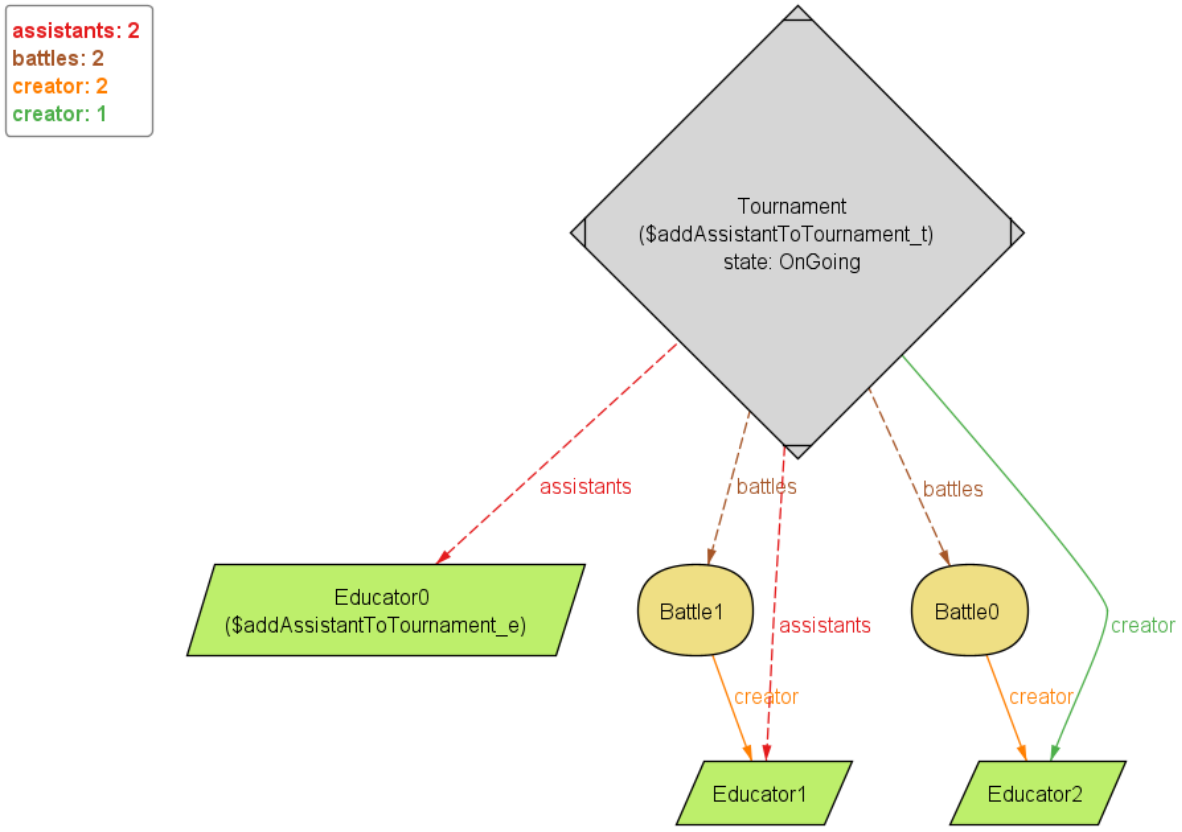


Figure 4.4: Add assistant to tournament



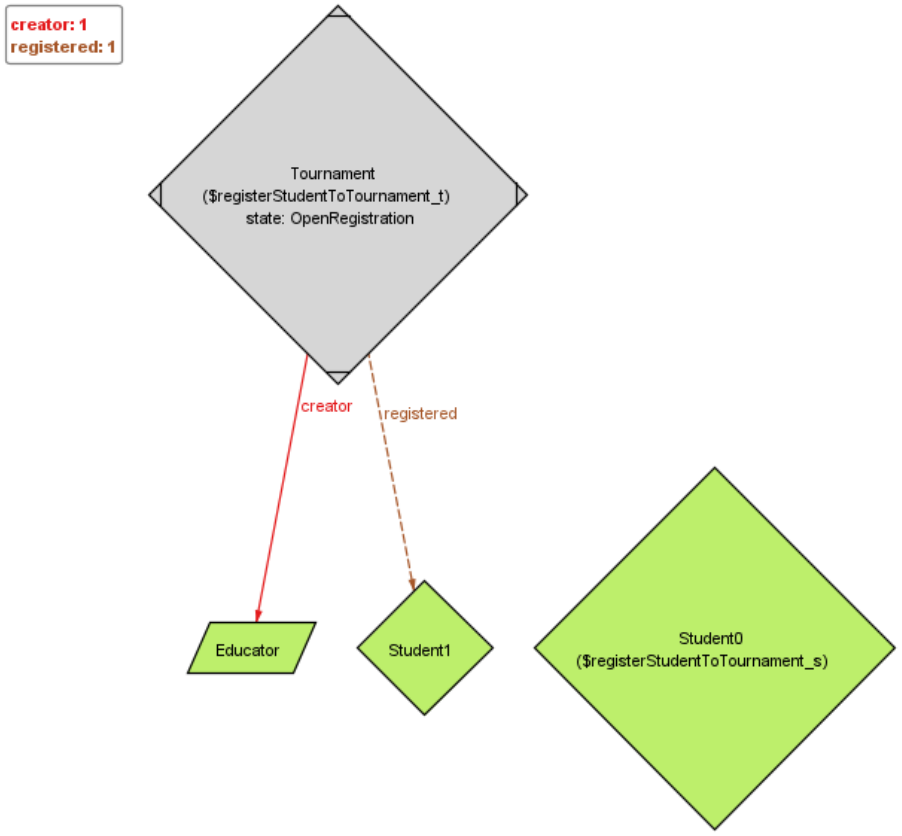
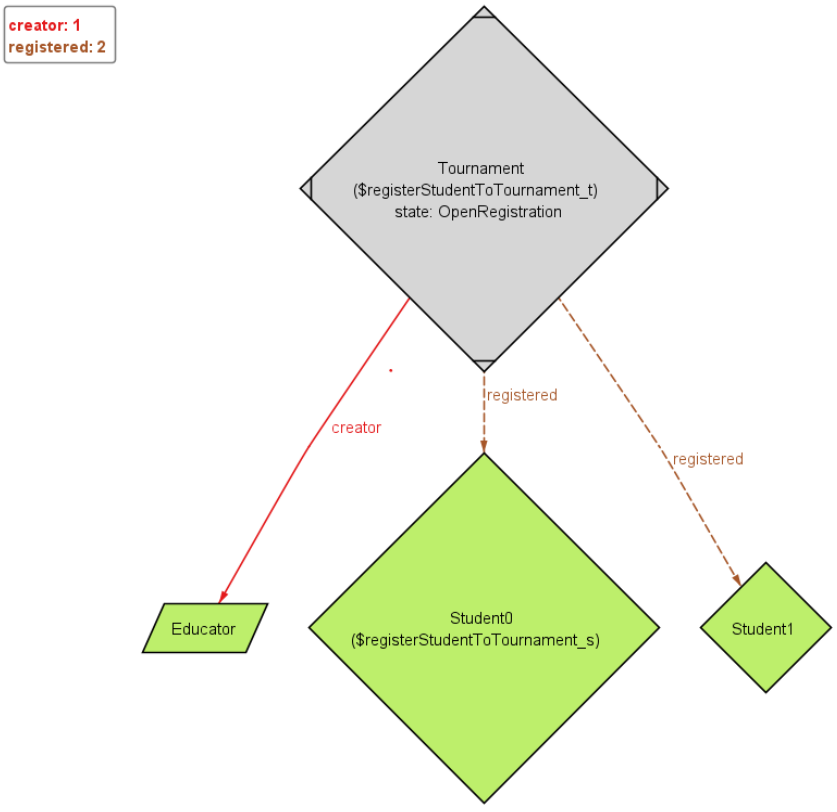


Figure 4.5: Student registers for a tournament



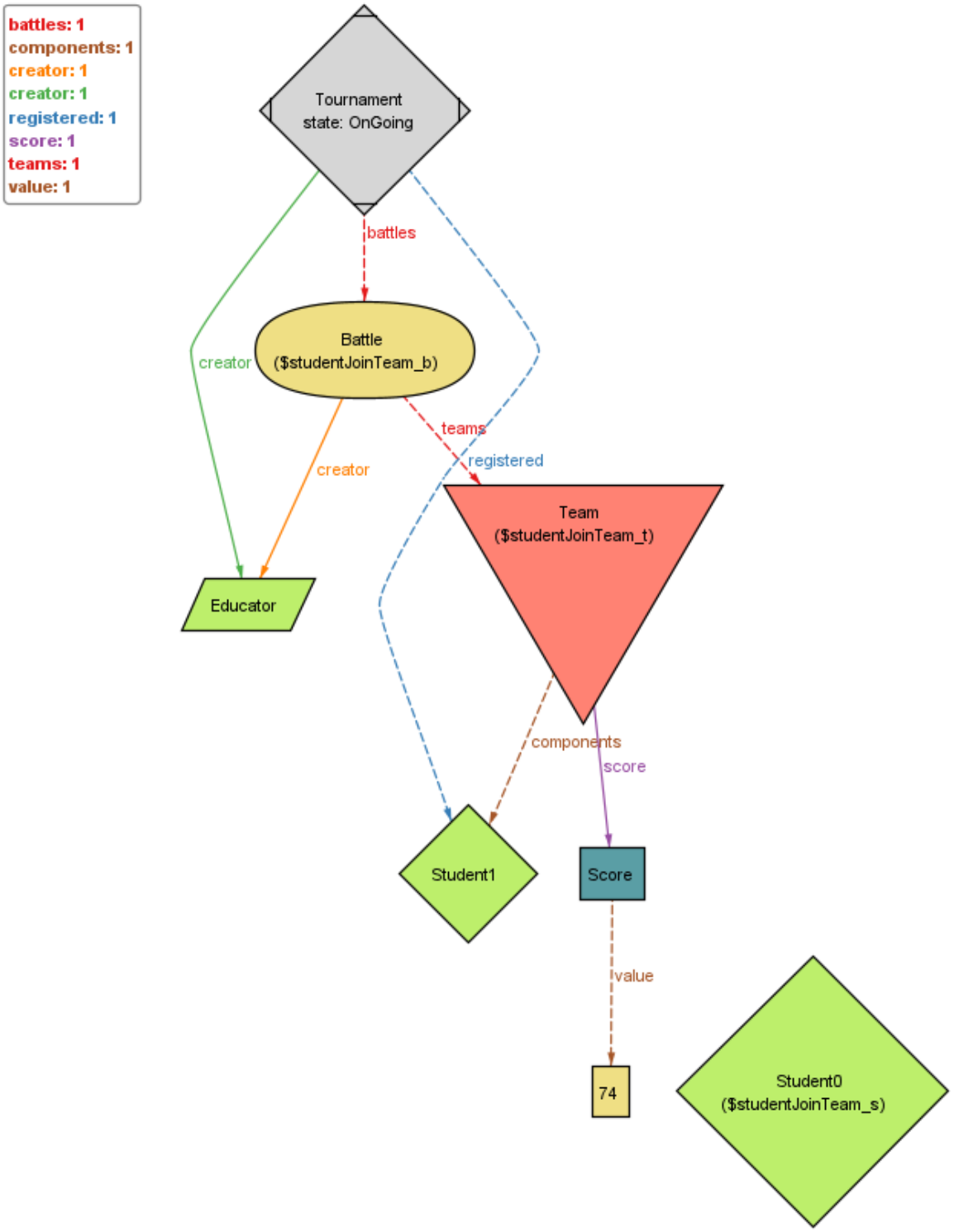
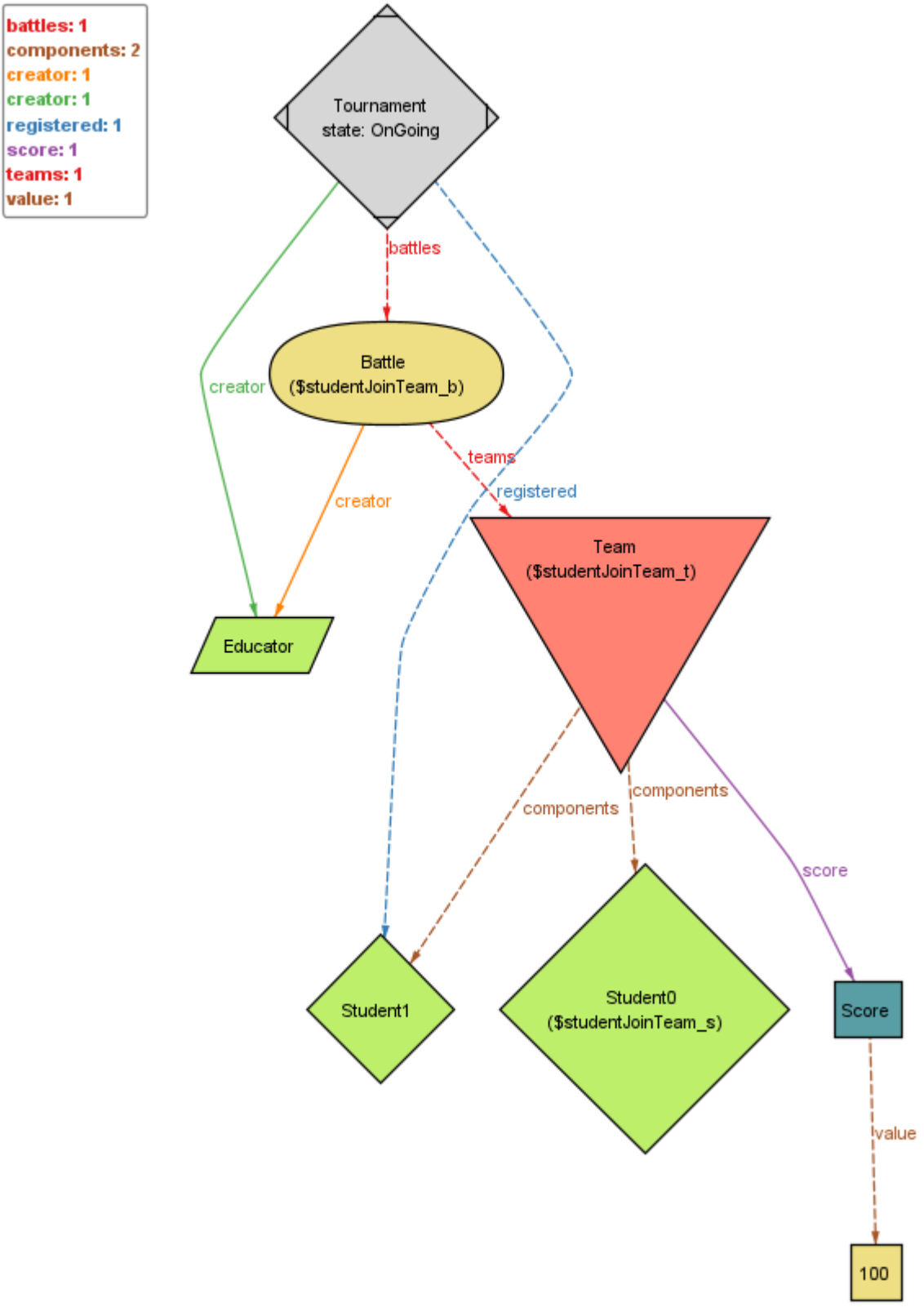


Figure 4.6: Student joins a team


battles: 1
components: 2
creator: 1
creator: 1
registered: 1
score: 1
teams: 1
value: 1
Tournament
state: OnGoing
battles
Battle
($studentJoinTeam_b)
creator
creator
teams
registered
Team
($studentJoinTeam_t)
Educator
components
components
score
Student1
Student0
($studentJoinTeam_s)
Score
value
100

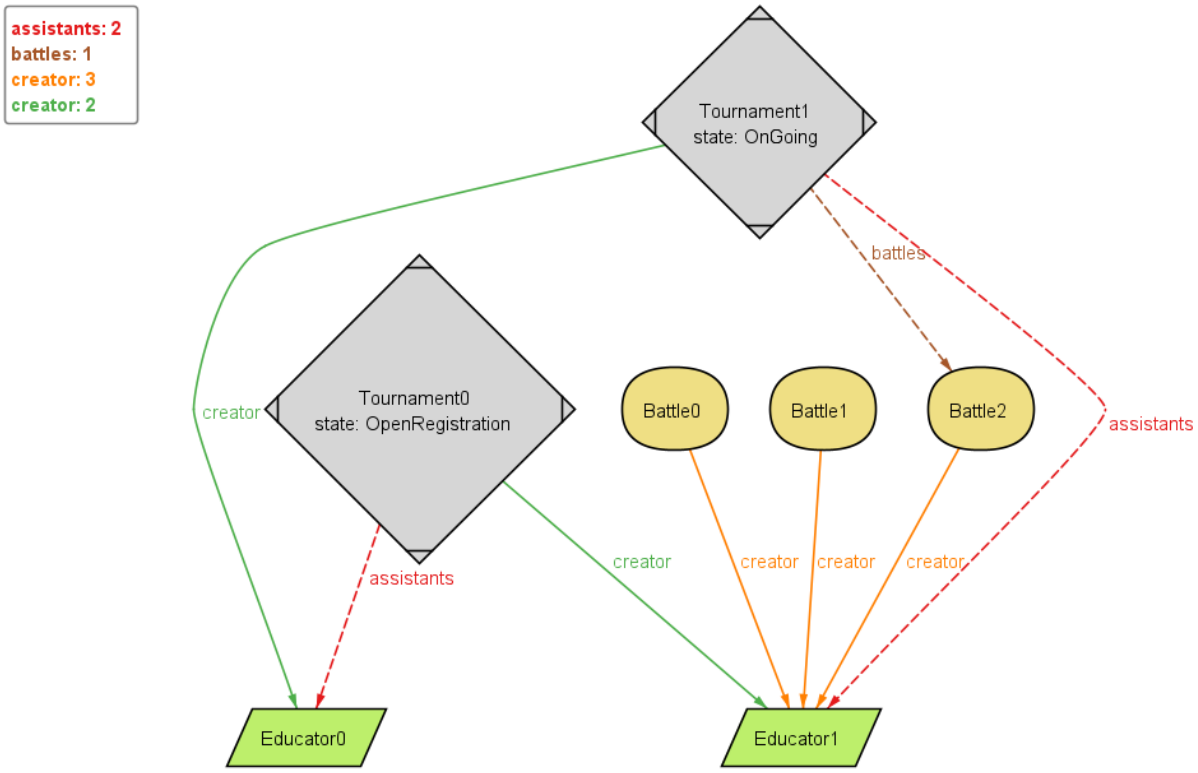
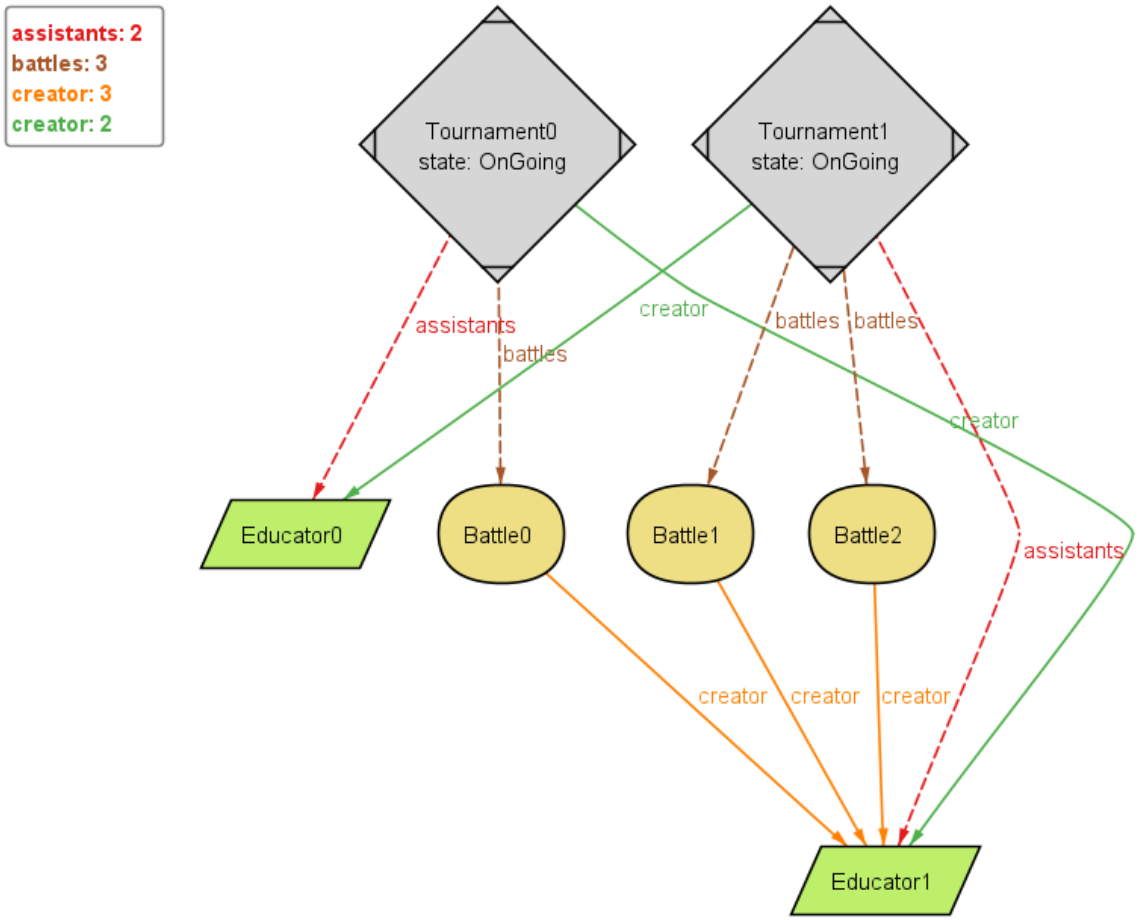


Figure 4.7: Battle added to a tournament



5 | Effort Spent

Alessandro Griffanti	Introduction Overall Description Specific Requirement User Interface Formal Analysis Reasoning	6*h* 8*h* 19*h* 2*h* 15*h* 9*h*
Luca Masiero	Introduction Overall Description Specific Requirement User Interface Formal Analysis Reasoning	1*h* 5*h* 20*h* 9*h* 15*h* 9*h*

Table 4.1: Effort spent

6 | References & Used Tools

6.1 Paper References

- Specification document: “Assignment RDD AY 2023-2024”
- Lecture slides from the Software Engineering 2 course

6.2 Used Tools

- Writing and drafting the document: Google Docs
- All diagrams made with: Draw.io
- Mockups made with: Figma
- Alloy models runned and tested with: Alloytools
- Versioning: GitHub
- Reasoning and notes: Notion

List of Figures

List of Tables