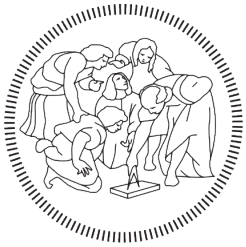
Software Engineering 2 - Prof. Di Nitto Elisabetta
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

CodeKataBattle

RASD
Requirement Analysis and Specification Document

November 1, 2023

Marco Cerino 244780
Mattia De Bartolomeis 245022



POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

In the context of education and development in the field of programming, students often face a series of challenges. The process of improving software development skills, both for beginners and more experienced students, requires a rigorous and structured approach.

The traditional method of learning based on theoretical lessons and assigned tasks can sometimes be limited in its effectiveness, as students may not have the opportunity to concretely apply what they have learned. Theory and practice must be integrated synergistically to ensure significant growth in software development skills.

CodeKataBattle (CKB) is an innovative response to these challenges in software learning and development. The CKB platform represents a revolutionary solution for students eager to enhance their programming skills. CKB is designed to transform the learning process into a collaborative and practical experience.

Thanks to CKB, students have the opportunity to engage in real code battles, solving programming exercises and overcoming a series of specific tests. These battles allow students to apply the theoretical knowledge they have acquired, putting into practice what they have learned through a series of specific challenges.

This document represents the RASD for the CodeKataBattle (CKB) system, providing a description focused on the system's requirements and specifications. It illustrates scenarios and use cases to detail the system's features, interactions with interested actors, and the limitations it is subject to.

1.1.1 Goal

The system is characterized by the following goals:

G1 Educators can manage coding tournaments and battles

G2 Students can form teams for coding battles

G3 Students can participate in coding battles

G4 Educators are able to evaluate manually the projects of the students

G5 Projects are evaluated in an automated way

G6 Educators and students can see the rank of the battles and tournaments

---

1.2 Scope

The main human actors in this system are students and educators. Educators use this platform to challenge students by creating competitions where groups of students compete against each other, demonstrating and improving their skills. The challenges consist of a programming exercise in a chosen language (such as Java or Python). Students must follow a "test-first" [1]approach, writing code to pass provided tests.
There is also a non-human actor that plays a crucial role in the platform : Github. GitHub plays a central role in the CodeKataBattle (CKB) for hosting battles and facilitating collaboration among students. It

[1]Test-first: The "test-first" approach involves writing tests before implementing the code, promoting test-driven development.

enables automated evaluations through GitHub Actions, tracking teams' progress and updating battle scores in real-time.
Here's how the system works: a teacher creates a "battle" following specific steps. They upload the problem description and the project to CKB, set the minimum and maximum number of students per group, define deadlines for registration and project submission, and set evaluation criteria. Once enrolled in a "battle," students form teams and start working on the project. The platform integrates GitHub, a source code management service, to facilitate collaboration.
Whenever students upload a new version of their code, the platform automatically calculates the team's score, considering aspects such as the number of passed tests, timeliness of submissions, and code quality. The automated assessment also includes static code analysis to evaluate aspects like security, reliability, and maintainability. Teachers can assign personal scores based on students' performance.
At the end of each "battle," the platform updates scores and displays the updated ranking. Students and teachers can monitor progress during the challenge. After the final submission deadline, a consolidation phase takes place, which may involve manual assessment by teachers. Once this phase is complete, all involved students are notified of the final "battle" ranking.
Scores obtained in each "battle" contribute to the overall tournament score for each student. These scores are used to create an overall tournament ranking, showcasing students' performance compared to their peers.

1.2.1 World phenomena

WP1	Educators evaluate the projects
WP2	Students solve the challenges

1.2.2 Shared phenomena

ID		Controlled by
SP1	Students invite other students to join their group	world
SP2	Students push a new commit into the main branch of their repository	world
SP3	Students can view the battle rank	world
SP4	Students can view the tournament rank	world
SP5	The educator uploads the battle	world
SP6	Students register for the battle	world
SP7	Students fork GitHub repository	world
SP8	Educator close the tournament	world
SP9	The professor sets the minimum and maximum number of students per group	world
SP10	The professor sets a registration deadline	world
SP11	The professor sets a final submission deadline	world
SP12	The professor configures additional scoring parameters	world
SP13	Educator use the CKB platform to see the sources produced by each team	world
SP14	Students receive the challenge	machine
SP15	System notify user about tournament creation	machine
SP16	System notify user about the final battle rank	machine
SP17	System notify user about the creation of a new battle	machine

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Student - An individual enrolled in an educational institution or self-learner who uses the CKB platform to enhance their software development skills.
- Educator - A teacher or education professional who uses the CKB platform to create and manage code kata battles and tournaments for students.
- Educator with permission - Educator who has the authority to create battles within a tournament , and also has the ability to close that same tournament.
- Tournament - A series of code kata battles, created and organized by an educator, where teams of

students compete to achieve the highest score in each battle and in the overall tournament.

- Battle - A programming exercise that consists of a textual description and a software project with build automation scripts and a set of test cases to be passed.
- Project - The solution proposed by the students for the exercise during the battle.
- Battle Score - A natural number between 0 and 100 assigned to each team in a battle, based on mandatory factors evaluated automatically and optional factors evaluated manually by educators
- Battle ranking - A ranking that lists students in order of performance. This ranking is determined by the battle score obtained by each student in the specific battle.
- Tournament Ranking - A ranking that lists students in order of performance within a single tournament on the CKB platform. This ranking is determined by summing the scores obtained by each student in all the code kata battles that make up the tournament they participated in

1.3.2 Acronyms

- CKB - Code Kada Battle.
- RASD - Requirement Analysis and Specification Document.
- UI - User interface.
- UML - Unified Modelling Language.

1.3.3 Abbreviations

- WP - World Phenomena.
- SP - Shared Phenomena.
- G - Goal
- R- Requirement

1.4 Revision history

- New version of 4/01/2024: Modification of requirements "The system must create and update the battle ranking in real time." in " The system must create the battle ranking when a new battle is created."
- New version of 13/02/2024: Refinement of the state chart diagram description.

1.5 Reference Documents

The creation of this document is based on :

- Slides of Software Engineering 2 course
- The specification of the RASD and DD assignment of the Software Engineering II course, held by professor Elisabetta Di Nitto at the Politecnico di Milano, A.Y 2023/2024;

1.6 Document Structure

This document is composed of six sections:

1. **Introduction**: In this section, we present an overview of the document's content and structure. We outline the problem at hand and articulate the goals that the system aims to achieve. The subsection on scope delves into a comprehensive description of various world and shared phenomena relevant to our subject matter. Additionally, essential information is provided to facilitate proper understanding, including definitions and abbreviations.
2. **Overall Description**: This section offers a holistic description of the system. We provide insights into the system's architecture and functionality, accompanied by an outline of the users and their primary functionalities. Domain diagrams are introduced to illustrate key components, and various scenarios are detailed to enhance comprehension. Finally, we present the fundamental assumptions within the system's domain.

3. **Specific Requirements**: In this section, we delve into the specific requirements of the system. Both functional and non-functional requirements are elucidated to provide a comprehensive understanding. Use case diagrams are included to visually represent system interactions, accompanied by detailed descriptions of each use case and related sequence diagrams. The section concludes with a mapping of requirements onto both overarching goals and individual use cases.
4. **Formal Analysis Using Alloy**: Here, we conduct a formal analysis of the system using Alloy. This includes a rigorous examination of the system's specifications and constraints to ensure coherence and reliability.
5. **Effort Spent**: In this section, we provide an estimate of the effort invested by each member of the group. This allows for a transparent understanding of the distribution of work within the team.
6. **References**: The document concludes with a comprehensive list of references used during the research and development phases. This section serves as a valuable resource for readers seeking additional information or verification of the document's content.

2 Overall Description

2.1 Product perspective

2.1.1 Scenarios

A. Educator Registration
Marco, an educator at the University of Naples, recently discovered the CKB platform platform thanks to a colleague. Interested in its potential to improve students' software development skills, Marco decided to software development skills of students, Marco decided to register as an educator. Visiting the home page of CKB home page, he selected the "Register" option and entered his email address, chose a password and provided his first and last name. After entering this information, Marco was asked to verify his email by clicking on a link sent to him to ensure the security of his account. Once Once this verification was completed, he proceeded with the registration by selecting the option "Sign up as educator" option. With the registration successfully completed, Marco was redirected to his personal dashboard within CKB, where he had access to an overview of the functionalities for educators, including the possibility to create new tournaments and battles.

B. Student Registration
Andrea, a student at the University of Bologna, discovered the CKB application thanks to the advice of his professor. Fascinated by the opportunity to hone his software development skills through code kata challenges, Andrea decided to subscribe to the platform. Once on the home page of CKB, he selected the "Register" option and provided his email address, creating a password. She also entered her first and last name, selecting the "Register as a student" option to complete the registration. However, before finalising the process, the system prompted Andrea to verify his email address by clicking on a link sent to him, to ensure the authenticity of his account. of her account. During this step, Andrea received an error message: the address was already present in the CKB system. He promptly corrected the email address, using a valid account. a valid account. Once he had successfully verified his email and completed the registration process, Andrea was redirected to his personal dashboard on CKB. Here, he was able to explore the various features available to students, such as active tournaments, upcoming competitions upcoming competitions and information about her progress on the platform.

C. Tournament Creation
Daniele, a professor of computer science at the Politecnico di Milano, would like to create a tournament on the CKB platform to allow his students to develop and improve their programming skills through participation in code kata battles. Daniele accesses the platform using his login credentials as a professor and navigates to the section dedicated to the creation of tournaments on the CKB platform. He then starts the process of creating a new tournament by selecting the appro- priate option from the main dashboard. selected from the main dashboard. The system prompts Daniel to enter information about the tournament, such as the name, a deadline for students to register for the tournament and the list of colleagues who can create battles within that tournament. The teacher completes the form and, after entering the required information, confirms the creation of the tournament. The system verifies the validity of the information provided and, if the name of the tournament If the name of the tournament already exists, the system sends the following warning: "The name of the tournament already exists". Otherwise, in In the event of positive verification, the system registers the new tournament within the system. All students registered on the platform receive notifications about the creation of the new tournament.

D. Battle Creation
John, a professor with battle creation privileges within the 'Algorithms and data structures' on the CKB platform, decides to create a new battle. To this end, after logging in, he navigates to the battle creation section and starts the process of creating a new battle. of creating a new battle. The system prompts Daniel to enter essential information about the battle, including an battle, including a short textual description, the software project with the scripts for automation of the build, the minimum and maximum number of students per group, the expiry date for registration and delivery of the project. In addition, Daniele sets additional information that will affect the scoring, such as security, maintenance and affid- ability. Finally, the system integrates the new battle into the platform. Students registered for the tournament relevant tournament receive notifications about the next battle.

E. Tournament registration
Sarah, a computer engineering student, finds herself eager for a new challenge to raise her programming skills. She then decides to log onto the CKB platform. After logging in, she explores the section on available tournaments and spots one in particular - "Python Challenge". Sarah clicks on the tournament and reads its description and notices that the deadline for registration has not expired. Without hesitation, she decides to participate in the Python Challenge. By clicking on 'Join the tournament' Sarah receives a confirmation message and is officially part of the tournament. She can now view all scheduled battles within the tournament and will also be notified when future battles are created.

F. Battle registration
Leonardo, a student enrolled in the "Super Tournament" on CKB, receives a notification that catches his attention: "Call to Battle - Python Coding Challenge". Keen to put his programming skills to the test, Leonardo, after logging in, enters the section dedicated to the battle, carefully reads the details provided and decides to sign up for the battle by clicking on "Join the battle". Subsequently, the platform allows Leonardo to choose whether to participate individually or invite other colleagues to form a team for the battle, respecting the minimum and maximum number of participants allowed. The platform only allows Leonardo to invite students who do not already have a team for the respective battle. After the registration phase has expired, the CKB platform automatically generates a dedicated repository for the 'Python Coding Challenge' project. Subsequently, the direct link to the repository is delivered to team members and made available in the "Your Battles" section of the application.

G. Face the battle
The 'CodeCrafters' team, consisting of enthusiastic students eager to immerse themselves in the battle, receives by e-mail the link to the repository on GitHub dedicated to the 'Advanced Algorithms' challenge on CKB. With a simple click, they proceed to fork that repository, containing the kata code, thus creating their own virtual workspace. To ensure continuous and accurate evaluation of their work, CodeCrafters configure GitHub Actions, which ensures timely communication with the platform. Immersed in the exciting challenge of 'Advanced Algorithms', CodeCrafters start the development process by adopting the test-first approach. Creatively, they create the first implementations, test them and commit them to the main repository, thus tracking every step of their iterative journey. Each push before the battle deadline activates the platform, which automatically evaluates the functional aspects and timeliness of the CodeCrafters' work. Subsequently, the platform calculates and updates their battle score, offering real-time feedback on the quality of their valuable contribution.

H. Battle elimination
Luca, an educator in computer science at 'Politecnico di Milano', decides to delete a battle in a tournament for which he has permissions, due to the low number of registered teams. After logging into the system, he navigates to the battles section of a specific tournament and selects the battle he wishes to remove. If the entry date for that battle has not yet expired or the battle has ended, he is given the option to click on "Remove Battle". By proceeding with 'Confirm' on a confirmation message, Luke finalises his decision. Once the action has been confirmed, the system automatically sends an e-mail to all students who are part of the teams registered for that battle, informing them that the battle has been eliminated. This process ensures that students are promptly informed of any changes to their activities in the tournament by e-mail.

I. Ranking display
Matthew, while participating in a battle on CKB, can monitor his team's position in real time through a dynamic ranking by logging into the tournament and battle section of interest. At the end of the battle, he is faced with a consolidation phase, during which the educator can decide whether to assign a personal score to the project or rely solely on the automatic scoring. At the conclusion of this phase, Matthew can consult the final ranking by going to the section dedicated to the battle that has just ended, thus obtaining a complete overview of his team's performance.

The score obtained in this specific battle is added to those accumulated in previous battles, contributing to his overall score in the tournament. Matthew and all users have the possibility to explore the ranking in the section dedicated to the tournament itself, accessible to all interested users.

Finally, once the educator definitively closes the tournament, Matteo and the other CKB members

can consult the final ranking of the tournament in the section relating to that tournament.

J. Evaluates project
Once the submission phase of a battle of the "Advanced Algorithms" tournament on CKB has expired, Professor Bianchi, the creator of the challenge, prepares to perform a manual evaluation of the projects submitted by the participating teams.He logs onto the platform, selects the tournament and the battle that has just ended. Within the interface, he finds a complete list of student projects if manual evaluation was added during the creation of the battle. Start by carefully examining the source code produced by each team, analysing the aspects that cannot be fully evaluated automatically. During this phase, Professor Bianchi assigns a personal score to each team based on his experience and in-depth knowledge, in particular rewarding the creativity, ingenuity and strategic approach of the participating teams. Once the evaluation is complete, the final score for each team is determined by averaging the score assigned manually by Professor Bianchi and the score generated automatically by the platform.

K. Tournament Closure
Luisa, a lecturer in computer science at the University of Turin, has successfully run the "Logic Masters" tournament on the CKB platform.After the last battle, she decides to close the tournament.Luisa logs in to CKB with her credentials and navigates to the "Logic Masters" tournament section.In the tournament dashboard, she finds the option "Close the Tournament". After clicking on this option, a warning message appears: "Are you sure you want to close the tournament?". Luisa must choose between "Confirm" to proceed with the closing or "Cancel" to abort the operation. Upon confirmation of closure, the system closes the tournament if there are still no active battles.

L. Tournament Elimination
Luigi, a professor registered on the CKB platform, decides to remove a tournament he had previously created. After completing all the planned battles within the tournament, Luigi proceeds with the removal. He navigates to the 'Tournament' section of the platform and accesses the page of the specific tournament. There, he finds and presses the 'Delete Tournament' button. A confirmation message appears, asking if Luigi is sure he wants to proceed with the deletion. Upon answering 'Yes', the system starts the process of deleting the tournament. Before permanently removing the tournament, the platform performs a check to ensure that there are no battles still in progress. If it finds an active battle, the system interrupts the elimination process and displays an 'elimination error' message indicating that the tournament cannot be eliminated at that time due to the presence of ongoing activity.

2.1.2 User interface

The system needs to interact with users via devices that require an internet connection. To access the system, all users must connect through a web interface hosted on an existing domain, such as www.codekatabattle.com.

2.1.3 Software interface

The system will use some external interfaces in order to accomplish its functionalities.

1. The system is required to retrieve data from database, necessitating the implementation of interfaces to ensure accurate data retrieval.
2. The system interacts with external APIs to offer email notification services.
3. The system interacts with GitHub to create and manage repositories for code kata exercises. It sets up automated workflows using GitHub Actions to track code commits, trigger updates, and calculate battle scores.

2.1.4 Interfaccia Hardware

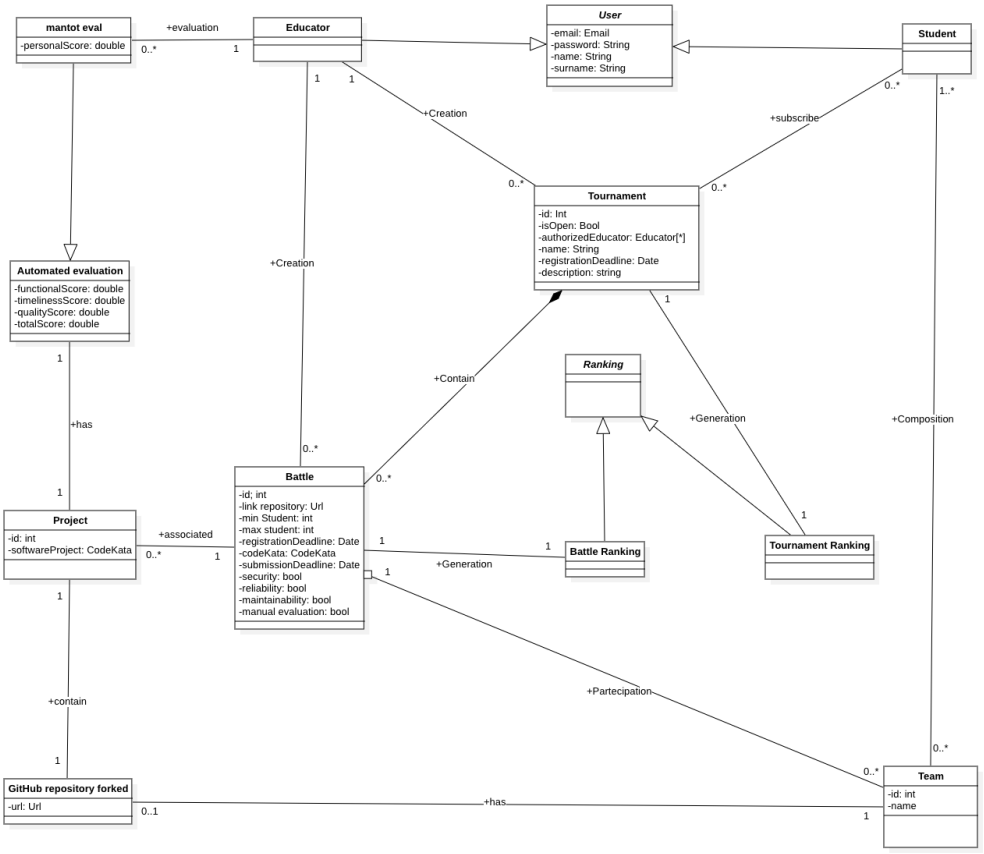
CKB è accessibile da una varietà di dispositivi hardware, garantendo un'esperienza utente ottimale su diverse piattaforme. Di seguito sono elencati i principali dispositivi hardware supportati:

- Computer Desktop e Laptop
- Smartphone e Tablet

2.2 Domain class diagram

In the figure is represented the domain class diagram related to CKB. This diagram is used to represent the key concept of our system and the interaction between them. The main elements of the diagram are:

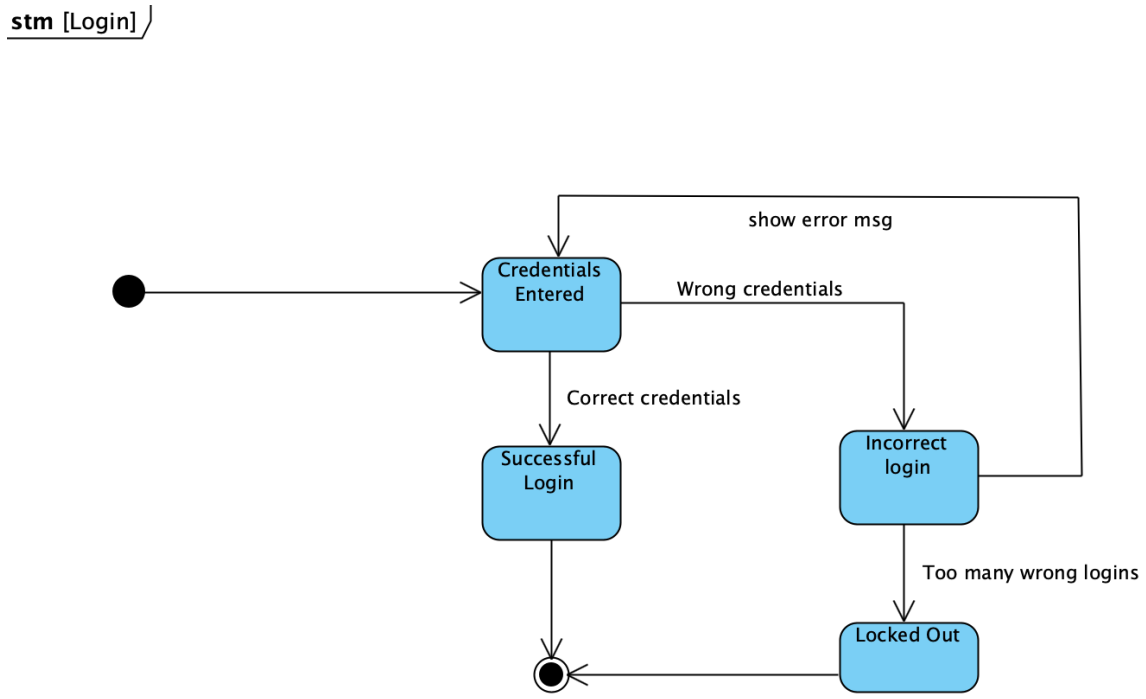
- The 'User' class is responsible for representing users within the system. It is divided into two specialized subclasses: 'Educator' and 'Student.' This division allows for the more effective management of specific functionalities and behaviors for each type of user. Educators have the ability to create tournaments and battles within the system. Students have the opportunity to enroll in tournaments created by educators and participate in battles.
- The 'Tournament' class represents tournaments within the system. It possesses a crucial attribute known as 'Educator Permissions.' This attribute serves the purpose of indicating which educators have the authorization,given by the tournament's creator, to create battles within that specific tournament.
- The 'Automated Evaluation' class represents an evaluation process that is automated and consistently carried out. It encompasses predefined parameters and criteria to assess specific elements automatically. Furthermore, 'Manual Evaluation' is a specialized form of evaluation within the system, and it extends from the 'Automated Evaluation' class. It shares the same predefined parameters and criteria as automated evaluation, but with the added capability of manual assessment by educators. Unlike automated evaluations, manual evaluations are not performed automatically but are initiated at the discretion of educators. Educators have the flexibility to decide whether to conduct a manual evaluation or not, enhancing the evaluation process when educator intervention is desired



2.3 StateChart Diagrams

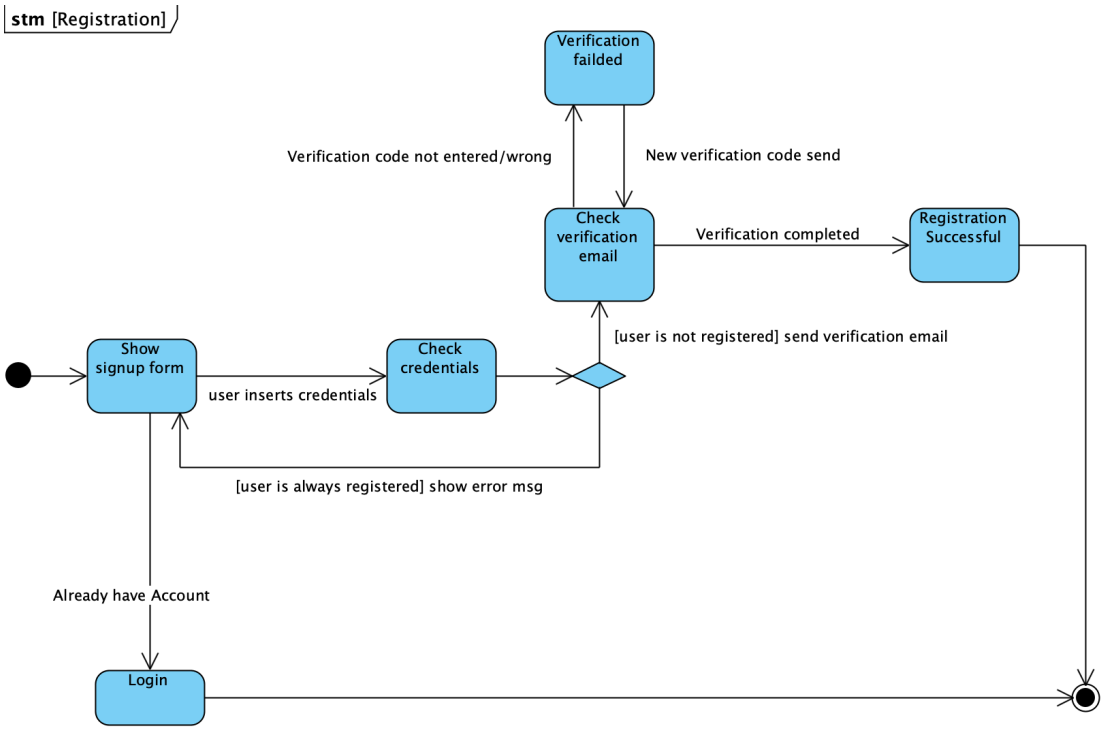
State diagrams describe the behaviour of the system while considering all possible states the objects can have when an event occurs.Di seguito sono riportati alcuni StateChart rappresentativi del sistema.

Login process



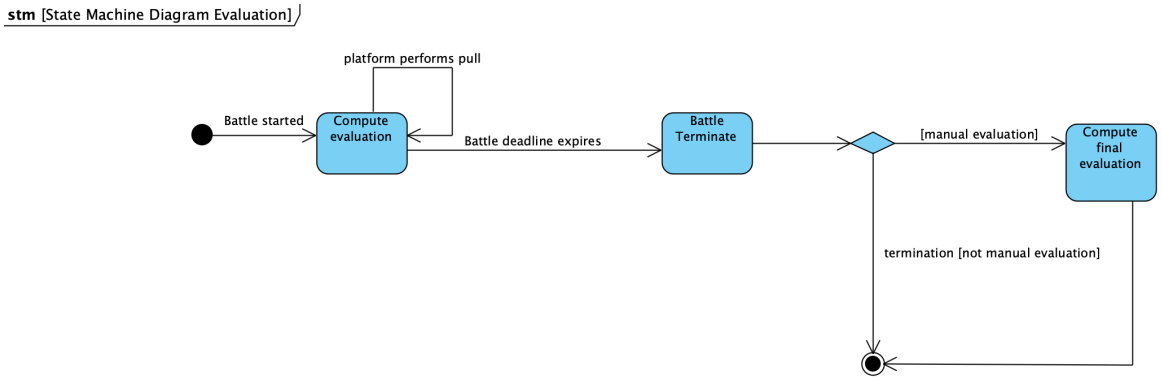
The displayed statechart describes the authentication process for a user, whether it be an educator or a student. The diagram begins with the state "Credentials Entered," which branches into two possible paths depending on the outcome of the credential verification. In the case where the credentials are recognized as correct, the process continues into the "Successful Login" state, concluding successfully and allowing the user to access protected resources. Conversely, if the credentials are incorrect, the control flow is directed to the "Incorrect login" state. Here, the user is presented with an error message, indicating the need to retry entering the data. In case of repeated authentication failures, indicated by the transition "Too many wrong logins," the system moves the user to the "Locked Out" state, where they will remain until the lock is lifted after a certain interval of time.

Registration process



This statechart describes the registration process for a user, whether it be an educator or a student. The initial state "Show signup form" represents the display of the registration form for the user. After the user enters their credentials, the process continues with the verification of these credentials in the "Check credentials" state. If the user is already registered, an error message is shown, and the user is redirected to the login process via the "Login" state. If the user is not registered, the system proceeds with sending a verification email, as indicated by the transition "Check verification email." If the user does not enter the verification code or enters an incorrect code, the process moves to the "Verification failed" state, and a new verification code is sent. When the user successfully completes the verification by entering the correct code, the registration process is successfully completed in the "Registration Successful" state.

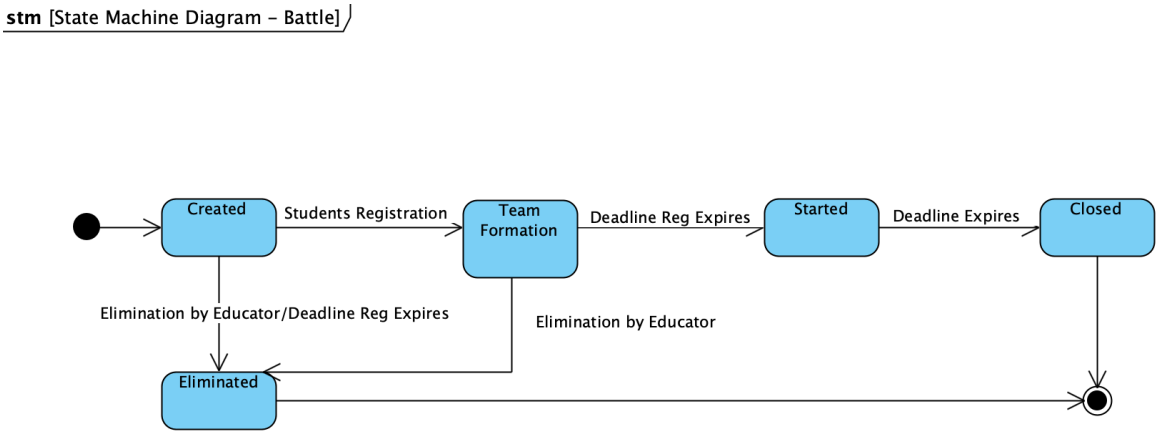
Evaluation process



This statechart represents the project evaluation process in a battle. In the initial state, "Compute evaluation," the evaluation of a team's project is automatically performed every time the system executes a pull operation to retrieve the data necessary for evaluation. Subsequently, the flow moves to the "Battle

terminated" state if the "battle" has ended, meaning when the deadline for the battle submission is reached. At this point, if the educator has set a manual evaluation at the time of creating the battle, the process proceeds to the "Compute final evaluation" state, where the educator assigns a subjective score to each project. Then, the process concludes. Otherwise, the process concludes directly.

Battle process



The statechart shown in the figure represents the process of evolution of a battle from its creation to its closure/deletion. The process begins with the state "Created," indicating that the battle was created by an educator. After the event creation, three situations may occur:

- If no student attempts to register for the battle and the registration deadline expires, then the system transitions to the state "Eliminated" and the battle is automatically deleted from the system.
- If no student has yet registered for the battle and the creating educator decides to delete it, then the system transitions to the state "Eliminated" and the battle is deleted from the system.
- If a student wants to register for the battle, the state transitions to "TeamFormation." In this state, the student invites other students to form a team for the battle. Subsequently, when the registration deadline expires, the state transitions to "Started" where the battle officially begins. When the submission deadline expires, the battle closes.

2.4 Product Functions

This section provides a summary of the major functions that the software will perform.

- A general user can:
  - Register
  - Login
- A student can:
  - Subscribe to a tournament
  - Subscribe to a battle
  - Form a team for the battle
  - Explore the ranking of battles/tournaments
- An educator can:
  - Create a tournament
  - Create a battle
  - Give permission to create battle within his tournament to other educators

– Manually evaluate student's projects.

2.5 User characteristics

As we have seen in the class diagram, there are two type of users: student and educator.

2.5.1 Student

A student is defined as someone eager to learn and enhance their programming skills. Students can navigate through dedicated pages on the CKB site to select the tournaments they are most interested in. They have the opportunity to participate in individual challenges or form teams, depending on the specific rules of each battle. In doing so, they compete with other students registered on the platform, testing their abilities and knowledge.

2.5.2 Educator

Anyone who wishes to create tournaments and battles, incorporating engaging and challenging tasks, is considered an educator. An educator has the freedom to upload various types of software projects within a battle, thus allowing anyone who registers to participate and compete. Moreover, educators have the ability to grant permissions to other educators to create battles within their tournaments. They also have the option to personally evaluate the tournament results or rely on the platform's default automated evaluation system.

2.6 Assumptions, dependencies and constraints

2.6.1 Assumptions

These assumptions represent properties and/or conditions that the system takes for granted, primarily because they are beyond the control of the system itself. It is necessary to verify them to ensure the correct behavior of CKB.

ID	Description
DA1	Both educators and students must have an e-mail.
DA2	Students must have a Github account.
DA3	Users who register on the CKB platform in the role of 'educator' are skilled in designing meaningful and challenging code katas and also in evaluating them.
DA4	Users have consistent and reliable access to the internet and the necessary technology (computers, software development tools) to participate in coding battles and tournaments.
DA5	The integration with GitHub and GitHub Actions functions correctly, allowing for seamless repository management, code submission, and automated workflow for the students' projects.
DA6	Users who register on the CKB platform in the role of 'student' have familiarity with programming languages, GitHub, and test-driven development methodologies.

Table 1: Domain assumptions.

2.6.2 Hardware Constraints

The system has to run under the following worst-case conditions:

- Internet Connection: Minimum 5 Mb/s.
- Screen Resolution: 1024x768.
- Modern Web like

2.6.3 Regulatory policy

The CKB application requires users to provide personal details, including their first name, last name, and email address. We assure users that their email addresses will not be utilized for any commercial or marketing purposes. All personal data collected will be handled and processed in strict adherence to the General Data Protection Regulation (GDPR), ensuring the highest standards of privacy.

3 Specific Requirement

In this section, it is given a complete description of the functional requirements of the system.

3.1 Requirements

R1 The system must allow users to register on the platform as educator or student.

R2 The system must allow users to login.

R3 The system must enable educators who have created a tournament to grant permissions to other educator to create battles within that specific tournament.

R4 The system must allow the educator to see which tournaments they have permission for.

R5 The system must allow educator to insert information about a tournament.

R6 The system must allow educator to insert information about a battle for the tournament in which he has the permissions.

R7 The system must notify via mail subscribed students to the platform upon the creation of new tournaments.

R8 The system must allow students to subscribe to tournament on the CKB platform before a specific deadline.

R9 The system must notify via mail all students subscribed in the tournament whenever a new battle is created within that tournament.

R10 The system must allow students to subscribe in the battle.

R11 The system should enable students to invite other students who do not yet have a team to create a team.

R12 The system must notify via mail all students who are invited to form a team.

R13 The system must allow students to accept an invitation via email to join in a team.

R14 The system must allow student to decline the invitation via email to join in a team

R15 The system must notify via mail the team's creator when an invited student declines his invitation.

R16 The system must notify via mail all members of a team when a new student joins their team.

R17 The system must allow user to see the list of available tournaments.

R18 The system must allow user to see the list of available battles.

R19 The system must integrate with GitHub for repository management and automate the process of code submission and evaluation.

R20 The system must allow educators who have set manually evaluation to see the submitted projects.

R21 The system must allow educators who have set manually evaluation to upload a manually score for each project.

R22 The system must automatically create a GitHub repository containing the code kata upon the expiration of the battle's registration deadline.

R23 The system must notify via email the link to the repository containing the code kata to all students who are members of a team registered for the battle.

R24 The system must automatically pull and analyze the latest sources from student repositories upon each commit.

R25 The system must support automated test execution and static analysis of submitted code.

R26 The system must calculate battle scores based on functional aspects, timeliness, and quality level of sources.

R27 The system must update battle scores in real-time as students push new commits.

R28 The system must consolidate and finalize battle scores after the submission deadline and provide a final battle rank.

R29 The system must notify via mail all students involved in a battle when final battle rank becomes available.

R30 The system must calculate and update tournament score at the end of each battle.

R31 The system must create and update the tournament rankings.

R32 The system must create the battle rankings when a new battle is created.

R33 The system must allow educators with permission on the tournament to close it.

R34 The system must ensure that the final score of each battle for a team falls within the range of 0 to 100.

R35 The system must allow educators with the necessary permissions to delete a tournament in which no battles are currently ongoing.

R36 The system must allow educators with necessary permissions to delete a battle that is not currently ongoing.

R37 The system automatically deletes a battle when no teams have subscribed to it by the registration deadline

3.1.1 Mapping between Goal,Domain Assumption and requirements

This paragraph will show the mapping between goals described in section 1.1.1 and requirements and domain assumptions.

G1 - Educators can manage coding tournaments and battles.

*R*01 The system must allow users to register on the platform as educator or student.

*R*02 The system must allow users to login.

*R*03 The system must enable educators who have created a tournament to grant permissions to other educator to create battles within that specific tournament.

*R*04 The system must allow the educator to see which tournaments they have permission for.

*R*05 The system must allow educator to insert information about a tournament.

*R*06 The system must allow educator to insert information about a battle for the tournament in which he has the permissions.

*R*17 The system must allow user to see the list of available tournaments.

*R*33 The system must allow educators with permission on the tournament to close it.

*R*35 The system must allow educators with the necessary permissions to delete a tournament in which no battles are currently ongoing.

*R*36 The system must allow educators with necessary permissions to delete a battle that is not currently ongoing.

*R*37 The system automatically deletes a battle when no teams have subscribed to it by the registration deadline.

*DA*1 Both educators and students must have an e-mail.

*DA*3 Users who register on the CKB platform in the role of 'educator' are skilled in designing meaningful and challenging code katas and also in evaluating them.

*DA*4 Users have consistent and reliable access to the internet and the necessary technology to participate in coding battles and tournaments.

G2 - Students can form teams for coding battles.

*R*01 The system must allow users to register on the platform as educator or student.

*R*02 The system must allow users to login.

*R*11 The system should enable students to invite other students who do not yet have a team to create a team.

*R*12 The system must notify via mail all students who are invited to form a team.

*R*13 The system must allow students to accept an invitation via email to join in a team.

*R*14 The system must allow student to decline the invitation via email to join in a team.

*R*15 The system must notify via mail the team's creator when an invited student declines his invitation.

*R*16 The system must notify via mail all members of a team when a new student joins their team.

$DA1$ Both educators and students must have an e-mail.

$DA4$ Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

G3 - Students can participate in coding battles

$R01$ The system must allow users to register on the platform as educator or student.

$R02$ The system must allow users to login.

$R07$ The system must notify via mail subscribed students to the platform upon the creation of new tournaments.

$R08$ The system must allow students to subscribe to tournament on the CKB platform before a specific deadline.

$R09$ The system must notify via mail all students subscribed in the tournament whenever a new battle is created within that tournament.

$R10$ The system must allow students to subscribe in the battle.

$R18$ The system must allow user to see the list of available battles.

$R19$ The system must integrate with GitHub for repository management and automate the process of code submission and evaluation.

$R22$ The system must automatically create a GitHub repository containing the code kata upon the expiration of the battle's registration deadline

$R23$ The system must notify via email the link to the repository containing the code kata to all students who are members of a team registered for the battle.

$DA1$ Both educators and students must have an e-mail.

$DA2$ Students must have a Github account.

$DA4$ Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

$DA5$ The integration with GitHub and GitHub Actions functions correctly, allowing for seamless repository management, code submission, and automated workflow for the students' projects

$DA6$ Users who register on the CKB platform in the role of 'student' have familiarity with pro- gramming languages, GitHub, and test-driven development methodologies.

G4 - Educators are able to evaluate manually the projects of the students

$R01$ The system must allow users to register on the platform as educator or student.

$R02$ The system must allow users to login.

$R04$ The system must allow the educator to see which tournaments they have permission for.

$R18$ The system must allow user to see the list of available battles.

$R20$ The system must allow educators who have set manually evaluation to see the submitted projects.

$R21$ The system must allow educators who have set manually evaluation to upload a manually score for each project.

$R34$ The system must ensure that the final score of each battle for a team falls within the range of 0 to 100.

$DA1$ Both educators and students must have an e-mail.

$DA3$ Users who register on the CKB platform in the role of 'educator' are skilled in designing meaningful and challenging code katas and also in evaluating them.

$DA4$ Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

G5 - Projects are evaluated in an automated way

$R24$ The system must automatically pull and analyze the latest sources from student repositories upon each commit

$R25$ The system must support automated test execution and static analysis of submitted code.

$R26$ The system must calculate battle scores based on functional aspects, timeliness, and quality level

of source

$R34$ The system must ensure that the final score of each battle for a team falls within the range of 0 to 100.

$DA5$ The integration with GitHub and GitHub Actions functions correctly, allowing for seamless repository management, code submission, and automated workflow for the students' projects.

G6 - Educators and students can see the rank of the battles and tournaments

$R01$ The system must allow users to register on the platform as educator or student.

$R02$ The system must allow users to login.

$R17$ The system must allow user to see the list of available tournaments.

$R18$ The system must allow user to see the list of available battles.

$R26$ The system must calculate battle scores based on functional aspects, timeliness, and quality level of sources.

$R27$ The system must update battle scores in real-time as students push new commits.

$R28$ The system must consolidate and finalize battle scores after the submission deadline and provide a final battle rank.

$R29$ The system must notify via mail all students involved in a battle when final battle rank becomes available.

$R30$ The system must calculate and update tournament score at the end of each battle.

$R31$ The system must create and update the tournament rankings

$R32$ The system must create and update the battle rankings in real time.

$DA1$ Both educators and students must have an e-mail.

$DA4$ Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

$DA5$ The integration with GitHub and GitHub Actions functions correctly, allowing for seamless repository management, code submission, and automated workflow for the students' projects

3.2 External interface requirements

3.2.1 User interface

Within this section of the document, we showcase a portion of the CKB web application's user interface (UI). Specifically, we present three interfaces that pertain to the functionalities of Login, Registration, and the Homepage.

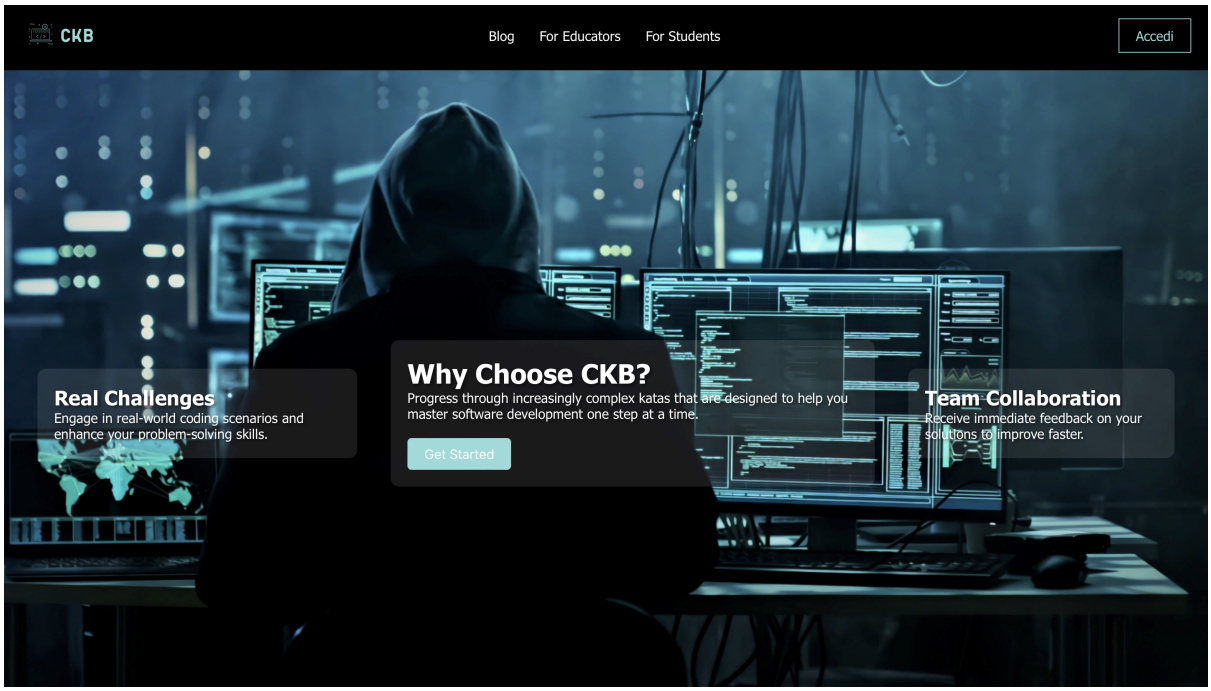


Figure 1: Home Page

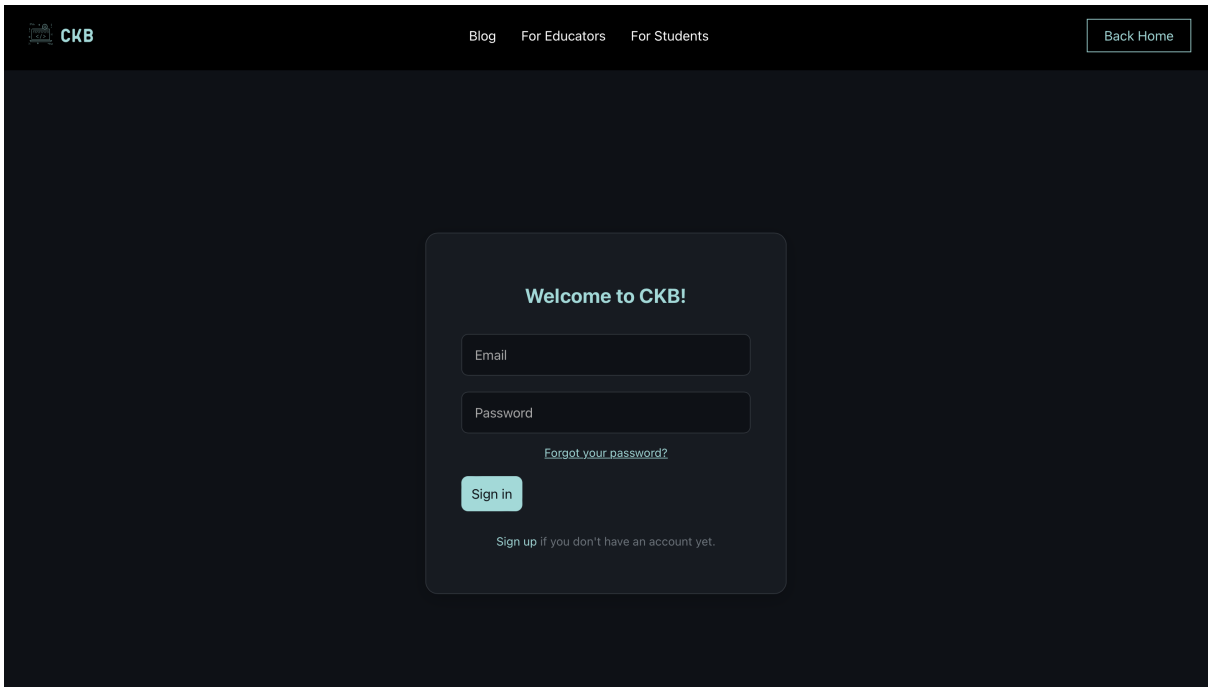


Figure 2: Login Page

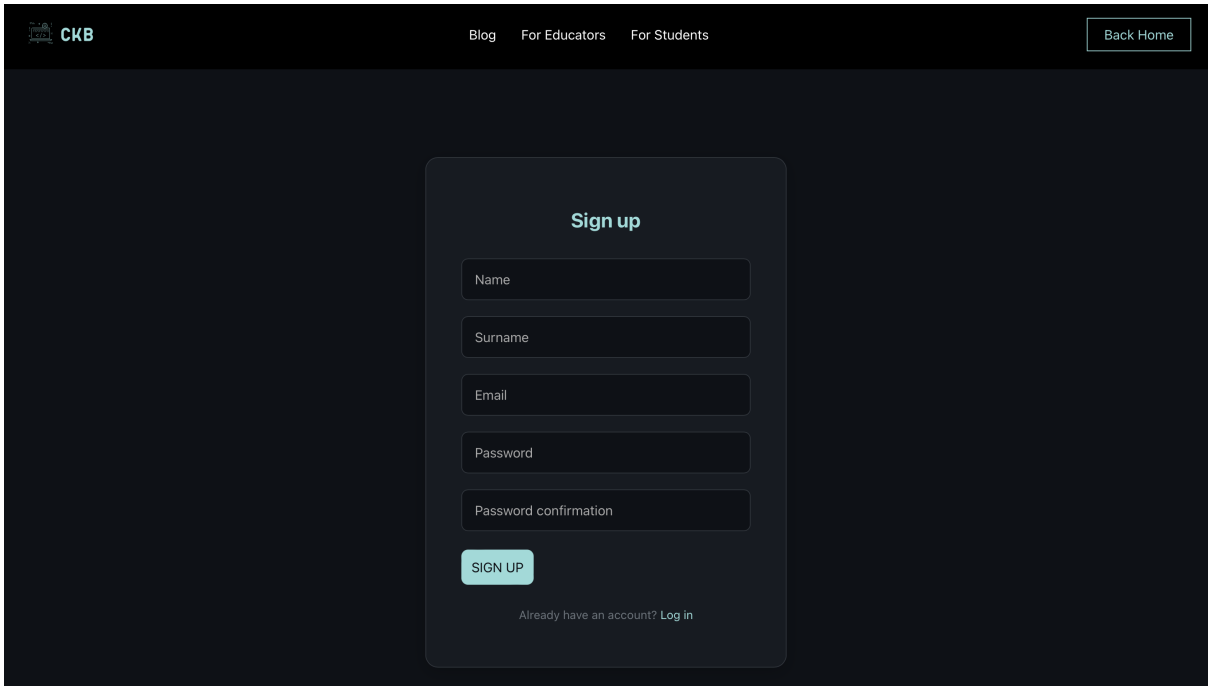


Figure 3: Registration Page

3.3 Use case diagrams

The following use case diagrams have been generated based on the scenarios described in paragraph 2.1.1. These diagrams assist in recognizing the individuals or entities engaging with the system and elucidate their responsibilities within each specific use case.

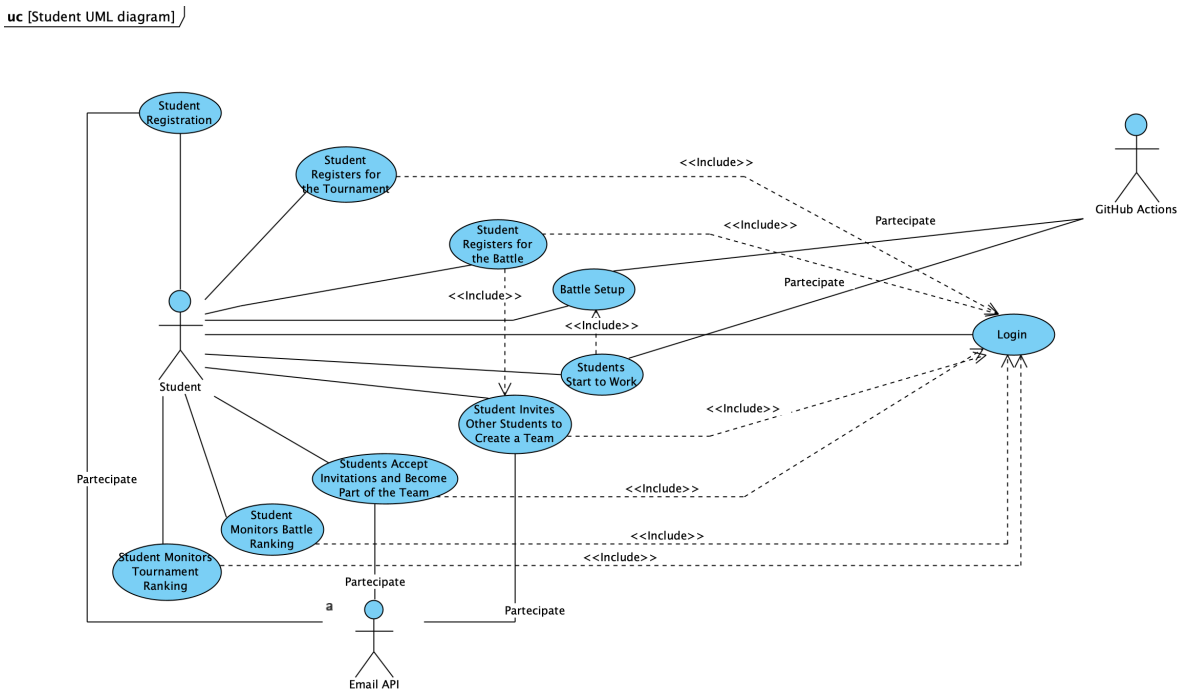


Figure 4: Student use case diagram

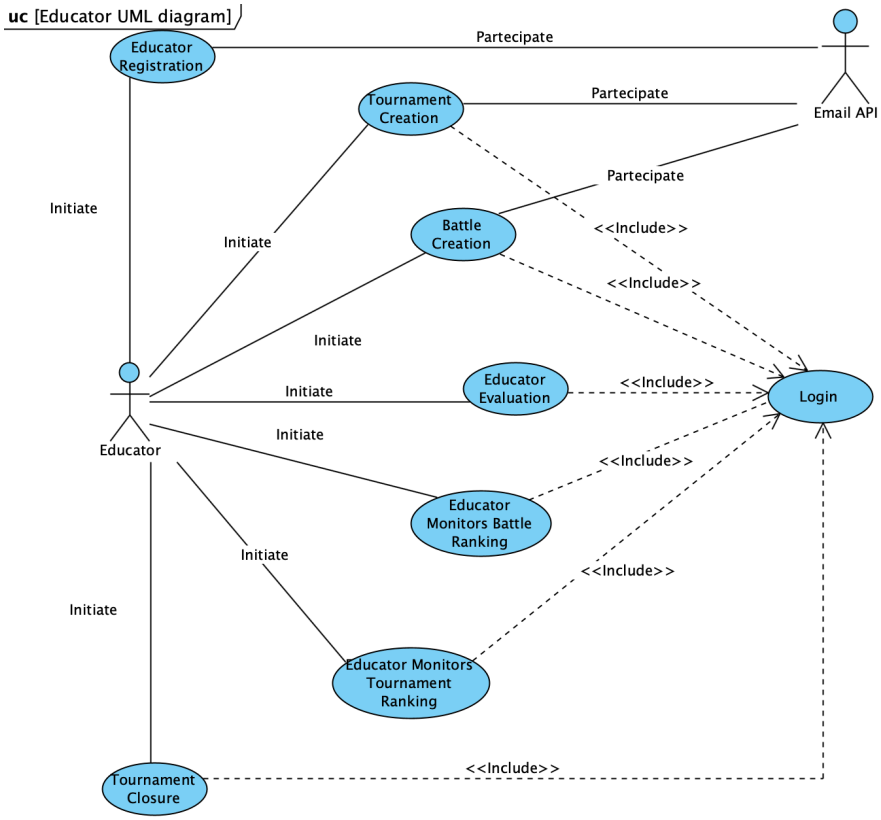


Figure 5: Educator use case diagram

3.4 User's use case

3.4.1 Educator registration

ID	1
Name	Educator Registration
Actor	Educator, email API
Entry Conditions	The educator has opened the web page on his computer
Events Flow	1. On the homepage, the educator clicks on the "Register/Login" section 2. The system shows two options: • Register • Login 3. Educator selects "Registration" 4. The system shows a list of fields that the educator needs to enter: • Name • Surname • Email • Password • A checkbox to select only if you are an educator 5. Educator enters the data, selects the checkbox and accepts the "Terms of Service" 6. Educator clicks on the "Confirm" button 7. The system shows the acceptance of the registration and invites the educator to go to his mailbox to confirm the registration 8. The educator opens his mailbox, searches for the email sent by the platform and clicks on the "accept registration" link
Exit Conditions	• Registration successfully completed: now the educator's data has been entered into the database. • The educator clicks on "You already have an account: is redirected to the Login section"

Exceptions	• The educator enters an email that is already present in the database. So, after the user clicks on the "Confirm" button, the platform shows the same page with an error message inviting the user to change the email since it is already stored in the database • Educator enters an incorrect email. So, after the user clicks on the "Confirm" button, the platform shows the same page with an error message inviting the user to modify the email since it is incorrect. • The educator does not receive the registration confirmation message in his mailbox. So, he clicks on "Send a new verification email" and the system sends a new registration confirmation link to the educator's email.

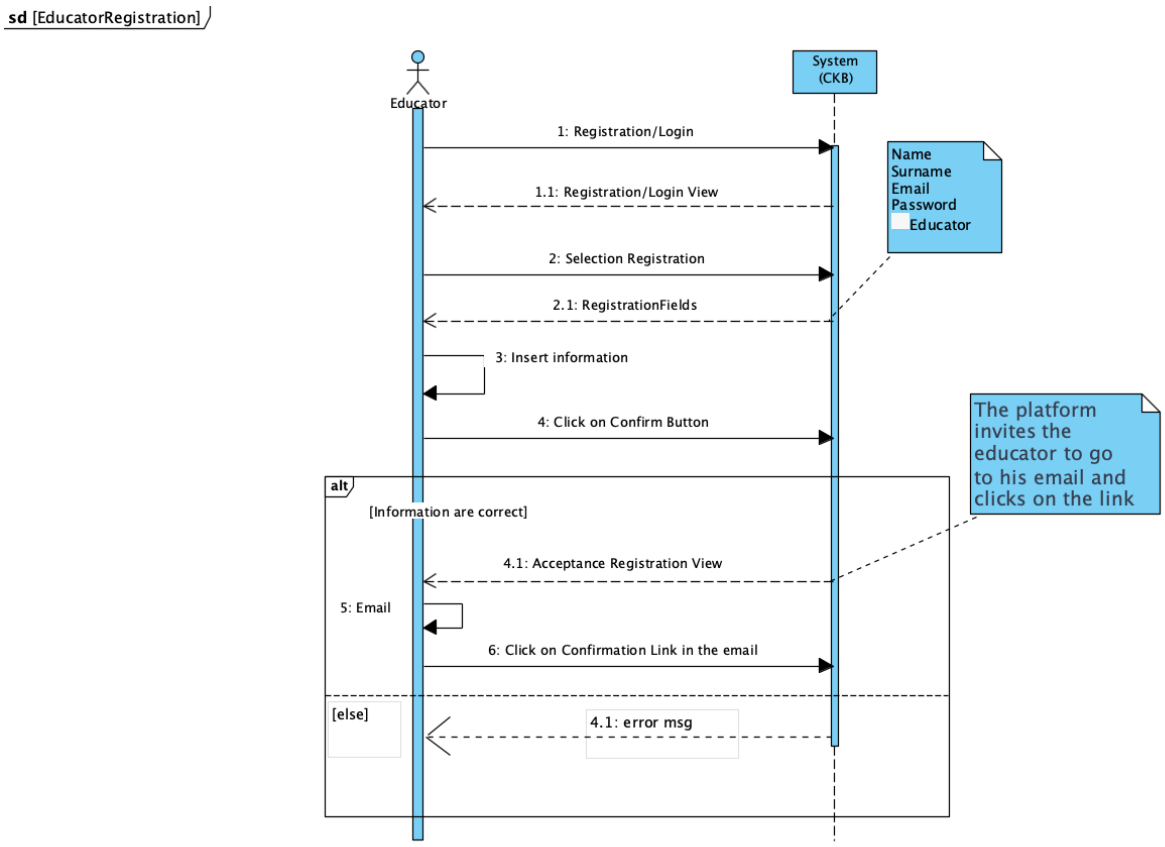


Figure 6: Educator registration

3.4.2 Student registration

ID	2
Name	Student Registration

Actor	Student, email API
Entry Conditions	The student has opened the web page on his computer
Events Flow	• On the homepage, the student clicks on the "Register/Login" section • The system shows two options: – Register – Login • Student selects "Registration" • The system shows a list of fields that the student needs to enter: – Name – Surname – Email – Password – A checkbox to select only if you are an educator • Student enters the data,does not select the checkbox and accepts the "Terms of Service" • Student clicks on the "Confirm" button • The system shows the acceptance of the registration and invites the student to go to their mailbox to confirm the registration • The student opens their mailbox, searches for the email sent by the platform, and clicks on the "accept registration" link
Exit Conditions	• Registration successfully completed: now the student's data has been entered into the database. • The student clicks on "You already have an account: is redirected to the Login section"
Exceptions	• The student enters an email that is already present in the database. So, after the user clicks on the "Confirm" button, the platform shows the same page with an error message inviting the user to change the email since it is already stored in the database • Student enters an incorrect email. So, after the user clicks on the "Confirm" button, the platform shows the same page with an error message inviting the user to modify the email since it is incorrect. • The student does not receive the registration confirmation message in his mailbox. So, he clicks on "Send a new verification email", and the system sends a new registration confirmation link to the student's email.

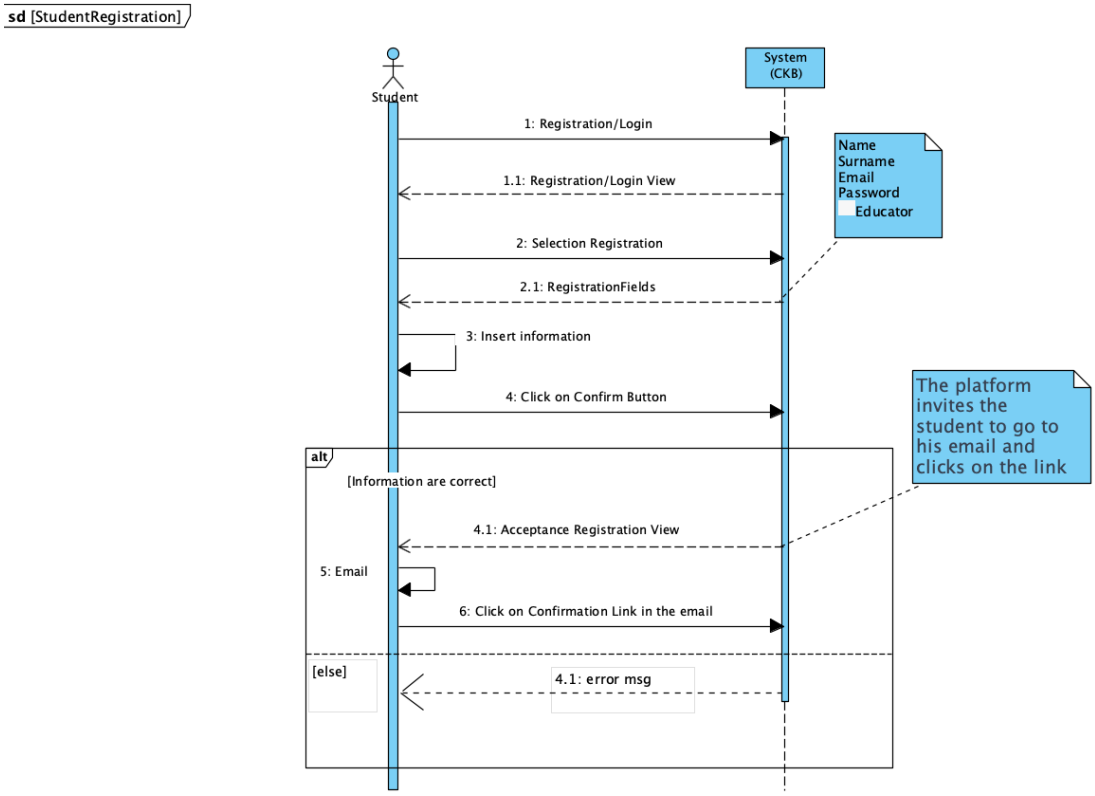


Figure 7: Student registration

3.4.3 User Login

ID	3
Name	User Login
Actor	Educator or Student
Entry Conditions	The user has opened the web page on his computer
Events Flow	1. On the homepage, the user clicks on the "Register/Login" section. 2. The system shows two options: • Register • Login 3. User selects Login. 4. The system shows a list of fields that the user needs to enter: • Email • Password 5. The user clicks the "Confirm" button. 6. The system verifies the credentials.
Exit Conditions	• Login is successfully completed, and the platform displays the user's profile dashboard.
Exceptions	• The user enters incorrect credentials. So, after the user clicks the "Confirm" button, the platform displays the same page with an error message explaining that the credentials are incorrect. • If the user makes too many login attempts, the platform redirects them to the initial homepage with an error message specifying that they have made too many attempts and blocks the login for 30 minutes.

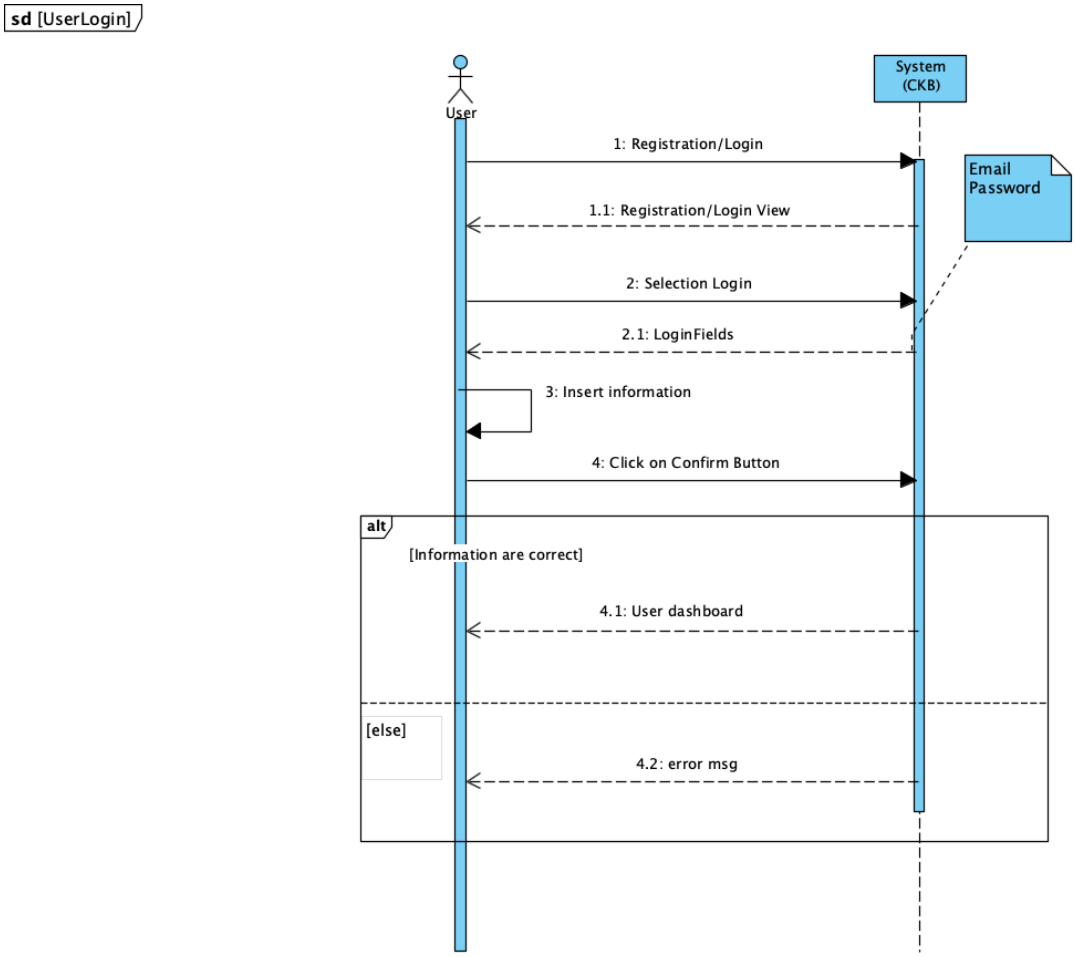


Figure 8: User Login

3.4.4 Tournament creation

ID	4
Name	Tournament creation
Actor	Educator, email API
Entry Conditions	The educator has logged into the system.

Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on "Create tournament." 4. The system shows a list of fields that the educator needs to enter: • Tournament name • Tournament description • Registration Deadline • List of email addresses of educators who have permission to create battles within the tournament 5. The educator fills in the various fields and clicks the "Confirm" button. 6. The system checks that the email addresses entered by the educator are valid and exist in the database. 7. The tournament creation is successfully completed: the platform stores the tournament information in the database and displays the following message to the user: "Tournament created successfully."
Exit Conditions	• The tournament is added to the educator's tournaments section dashboard. • The system notified via email all students subscribed to the platform.
Exceptions	• The tournament name entered by the educator is already assigned to another tournament. Therefore, once the educator clicks the "Confirm" button, the platform displays the same page with an error message instructing the educator to change the name as it already exists. • One of the email addresses entered by the educator is incorrect. Therefore, the system displays the same page to the educator but with an error message explaining that at least one of the entered emails is incorrect. • One of the email addresses entered by the educator is not present in the database. Therefore, the system displays the same page to the educator but with an error message explaining that at least one of the educators is not registered on the platform.

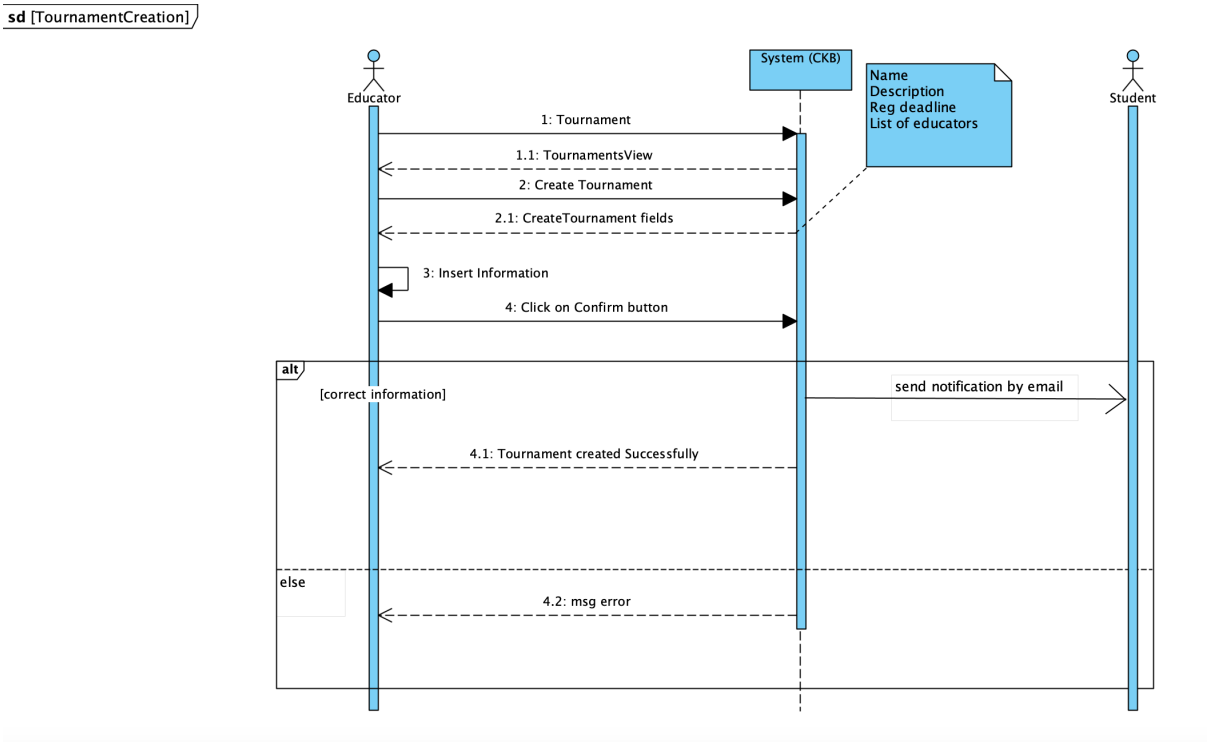


Figure 9: Tournament creation

3.4.5 Battle creation

ID	5
Name	Battle creation
Actor	Educator, email API
Entry Conditions	The educator has logged into the system.

Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on a specific tournament (for which they have permission to create battles). 4. The system displays the dashboard containing battles for the selected tournament to the educator. 5. The educator clicks on "Create a new battle" 6. The system shows a list of fields that the educator needs to enter: • Upload Code Kata • Minimum number of students to form a team • Maximum number of students to form a team • A checkbox to enable the manual evaluation of the educator on the projects • A checkbox to select only if security considerations are required in the static analysis of the project submitted by the students. • A checkbox to select only if reliability considerations are required in the static analysis of the project submitted by the students. • A checkbox to select only if maintainability considerations are required in the static analysis of the project submitted by the students. 7. The educator fills in the various fields, selects the checkboxes of interest and clicks the "Confirm" button. 8. The system checks that the battle name is not already in use. 9. The battle creation is successfully completed: the platform stores the battle information in the database and displays the following message to the user: "Battle created successfully."
Exit Conditions	• The battle is added to the dashboard of the specific tournament. • All students registered for the specific tournament selected by the educator are notified of the battle creation via email.
Exceptions	The battle name entered by the educator is already assigned to another battle in the same tournament. Therefore, when the educator clicks the "Confirm" button, the platform displays the same page with an error message instructing the educator to change the name as it already exists.

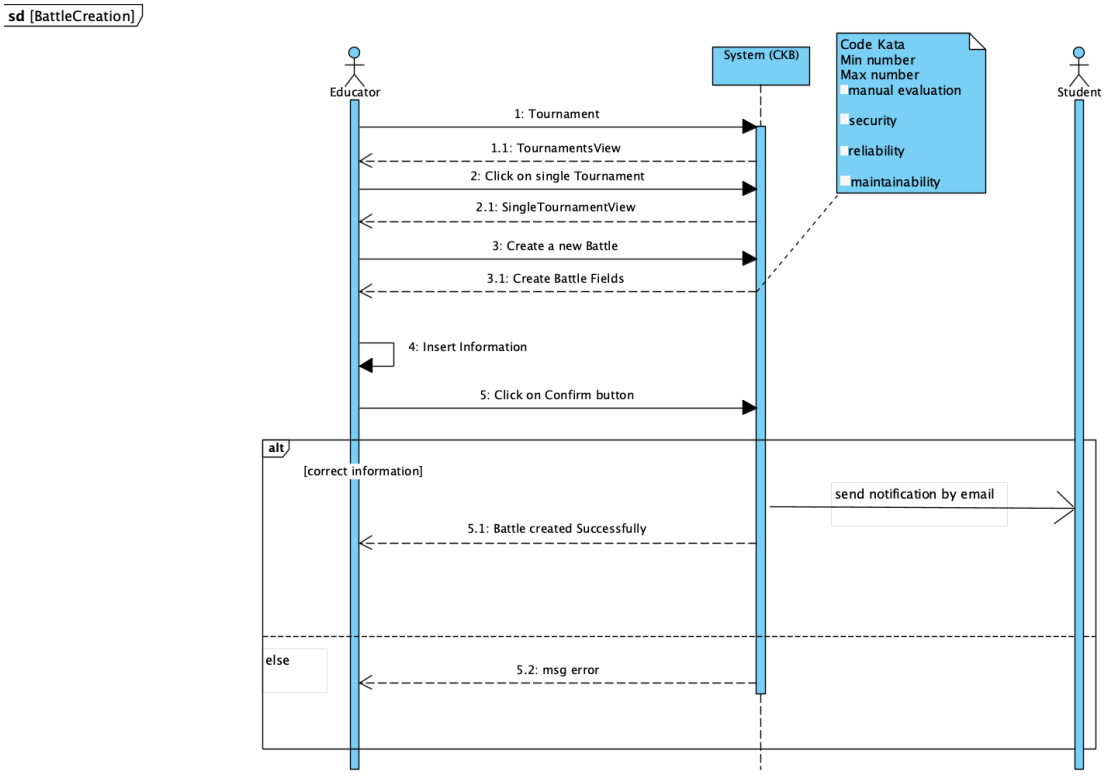


Figure 10: Battle creation

3.4.6 Student registers for the tournament

ID	6
Name	Student registers for the tournament
Actor	Student
Entry Conditions	The student has logged in the system.
Events Flow	1. The student clicks on the "Tournaments" section of the dashboard 2. The system displays the page containing the list of available tournaments 3. The student browses through the list and selects one he is interested in. 4. The system displays the specific tournament page. 5. The student clicks on the "Register" button to begin the registration process. 6. The student successfully registers for a tournament.
Exit Conditions	The student receives a confirmation message, confirming their participation in the tournament and the system updates the database, recording the student's participation in the tournament.
Exceptions	If the registration for the tournament is closed or the tournament is closed , the student will simply not be allowed to register.

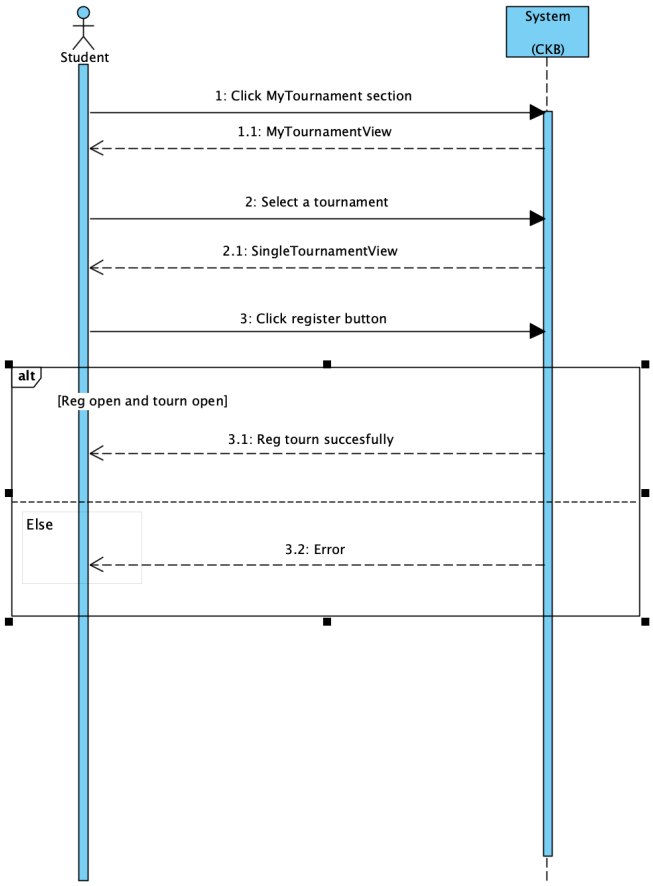


Figure 11: Student registers for the tournament

3.4.7 Student registers for the battle

ID	7
Name	Student registers for the battle
Actor	Student
Entry Conditions	The student has logged in the system.
Events Flow	1. The student clicks on the "My tournament" section to view all the tournaments in which he is registered. 2. The system displays the page containing the list of the tournament in which the student is subscribed. 3. The student selects a tournament from within the "My tournament" section. 4. The system displays the specific tournament page. 5. Within the tournament, the student reviews available battles and selects the one they wish to participate in. 6. The system displays the specific battle section. 7. The student reads through the battle details, the code kata description, deadlines, and specific rules or requirements. 8. The student clicks on the "Register for Battle" button. 9. The system shows a section who allows student to invite other students to join his team and set information about the team. 10. The student forms a team by inviting his friends.
Exit Conditions	The student successfully registers for the selected battle, and the system updates the database, recording the team's participation in the battle
Exceptions	• If a student invites one or more students who are already participating in a team within that battle, the system displays an error message. • If the registration for the battle is closed, the student will simply not be allowed to register. • If the student tries to register for a battle in a tournament they are not enrolled in, the platform displays an error message explaining that they are not registered for the tournament.

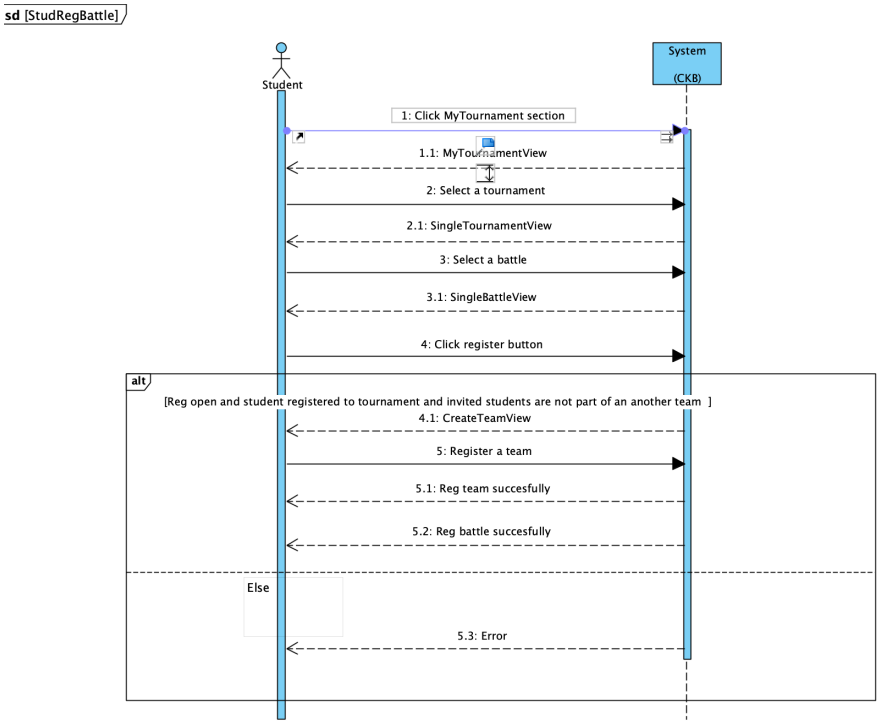


Figure 12: Student registers for the battle

3.4.8 Student invites other students to create a team

ID	8
Name	Student invite other student to create a team
Actor	Student, email API
Entry Conditions	The student has clicked "Register" for a specific battle on the platform and is at the section for forming a team.
Events Flow	1. The student inserts a team name. 2. The student select the students to form the team 3. The student click on "Create team"
Exit Conditions	The system creates team and notify all the students invited via email .

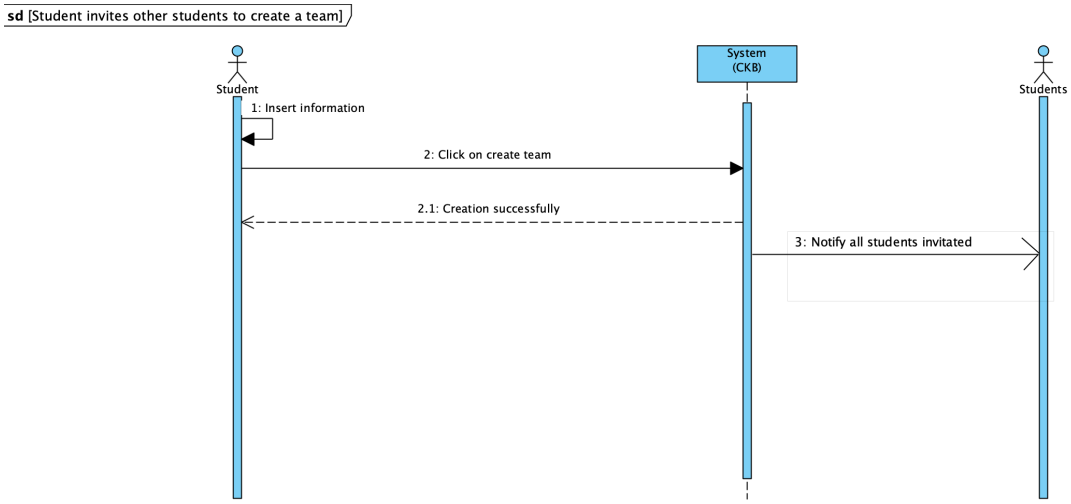


Figure 13: Student invites other students to create a team

3.4.9 Students accept invitations and become part of the team

ID	9
Name	Students accept invitations and become part of the team
Actor	Student, email API
Entry Conditions	The student has logged in the system.
Events Flow	1. The students receive a link via email from the system. 2. The students click on the link and is reported to CKB home page. 3. The system displays a confirmation message, in which the student can choose "Yes" to accept the invitation. 4. Once the students have accepted the invitation, they are ufficialy part of the team. 5. System send a confirmation notifications via email to all team members.
Exit Conditions	• Students join the team and system updates the database accordingly.

Exceptions	• The students select "No" to decline the invitation. In this case, the system will notify the team's creator about this event. • If the student tries to join a team for a battle in a tournament they are not enrolled in, the system displays an error message explaining that the user must first register for the tournament.

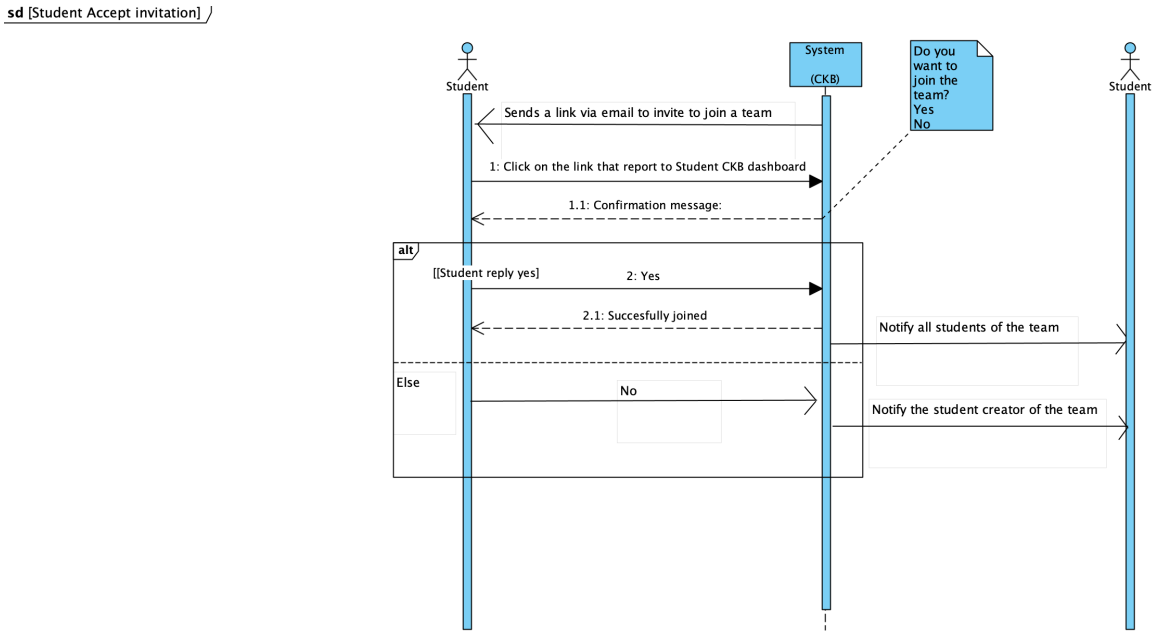


Figure 14: Students accept invitations and become part of the team

3.4.10 Battle Setup

ID	10
Name	Battle Setup
Actor	Student, GitHub Actions
Entry Conditions	The registration deadline for the battle expires
Events Flow	1. The platform creates a GitHub repository containing the code kata(the project) 2. The system emails all students who have registered for the battle the link to the GitHub repository and instructions for setting up an automated workflow through GitHub Actions. 3. Students fork the repository they've been sent 4. Students set up an automated workflow through GitHub Actions
Exit Conditions	The battle environment is successfully setted up and students start coding.

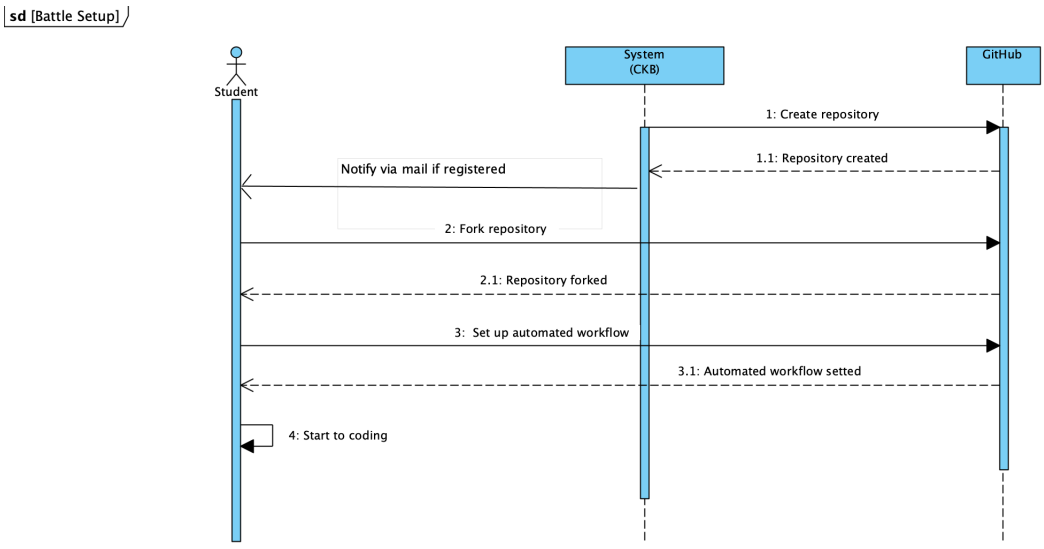


Figure 15: Battle Setup

3.4.11 Students start to work

ID	11
Name	Students start to work
Actor	Student, GitHub Actions
Entry Conditions	The battle environment has been successfully set up.
Events Flow	1. Students start coding the project 2. Students commit to GitHub every time they want to update a change. 3. GitHub Actions notifies the CKB platform (via appropriate API calls) immediately when students push a new commit. 4. The CKB platform pull the latest source code. 5. The CKB platform calculates the team's score using automated tools configured during the battle creation 6. The CKB platform updates the score of the team 7. Upon the submission deadline, the platform stops monitoring for further pushes 8. The system close the battle
Exit Conditions	The battle has been successfully concluded, and a consolidation process starts to finalize the scores.

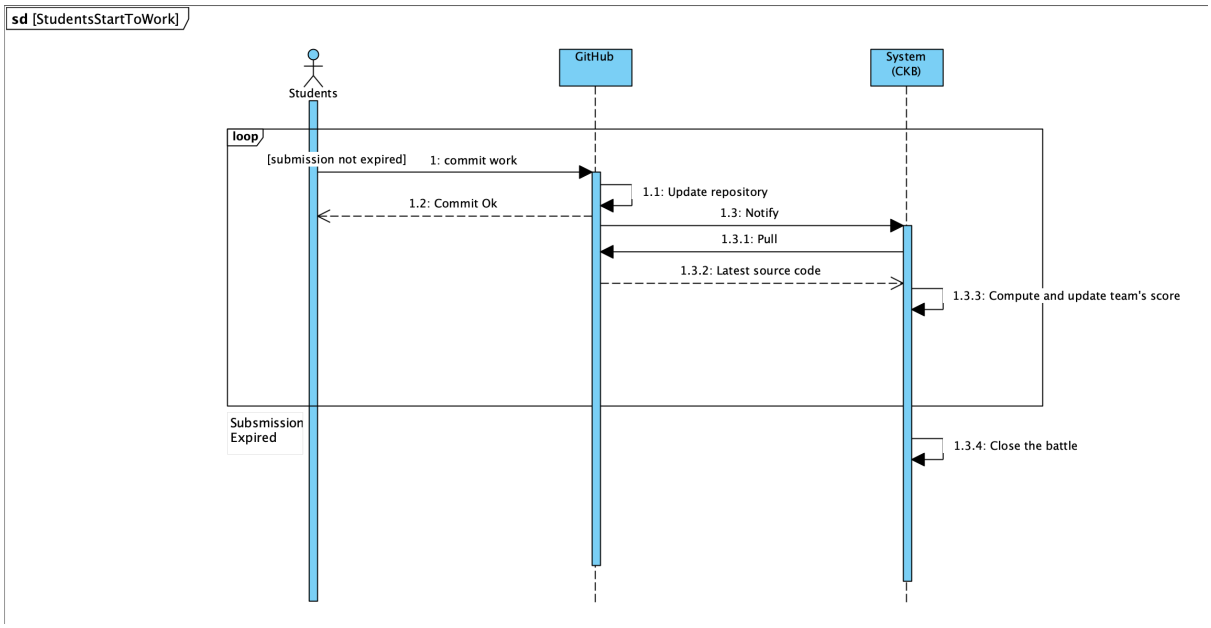


Figure 16: Students start to work

3.4.12 Educator evaluation during the consolidation process

ID	12
Name	Educator evaluation during the consolidation process
Actor	Educator
Entry Conditions	The Code Kata Battle has reached its submission deadline
Events Flow	1. The system updates all project files in the 'Final Submission' section of the respective Battle section. 2. Educator clicks on the "Final Submission" section 3. The system displays "Final Submission" section 4. Educator clicks on a project 5. The system open a new page referred to the project 6. The educator download the project by clicking on "Download". 7. The educator reviews the project and assigns a personal score (ranging from 0 to 100) to it. 8. Educator,after completing the review, click on "Submit evaluation". 9. The educator repeats the process for all submitted projects. 10. The platform calculates the final rankings by averaging the evaluations provided by educators and those generated through automated processes.
Exit Conditions	The platform notify via email all students registered for the battle, informing them about the available rankings.
Exceptions	If manual evaluation is disabled, the platform will not upload the project to the 'Final Submission' section; instead, it will calculate the final rankings directly based on its automated evaluation.

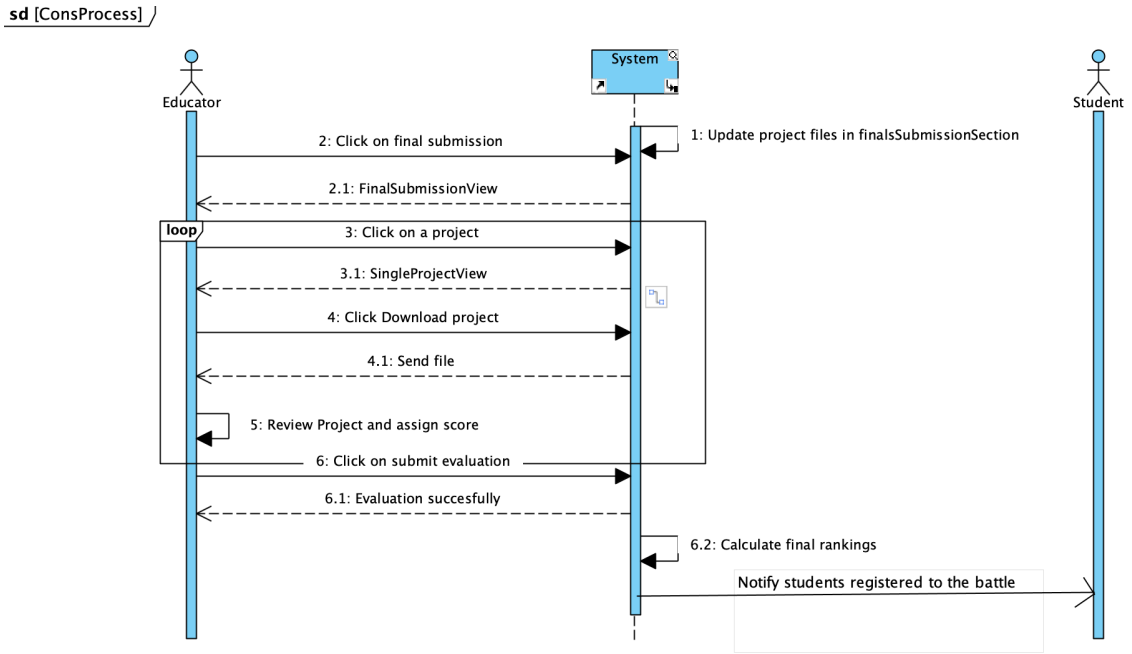


Figure 17: Educator evaluation during the consolidation process

3.4.13 Educator monitors battle ranking

ID	13
Name	Educator monitors battle ranking
Actor	Educator
Entry Conditions	The educator has logged into the system.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on "All Tournaments". 4. The system displays a page that includes all tournaments of the platform, even those for which the educator does not have permission to create battles. 5. The educator clicks on a specific tournament. 6. The system displays the dashboard containing battles for the selected tournament to the educator. 7. The educator selects the battle they want to view the ranking of.
Exit Conditions	The system displays the battle ranking to the educator, containing the following fields: • Team ID • Team name • Team Score calculated in real-time by the system

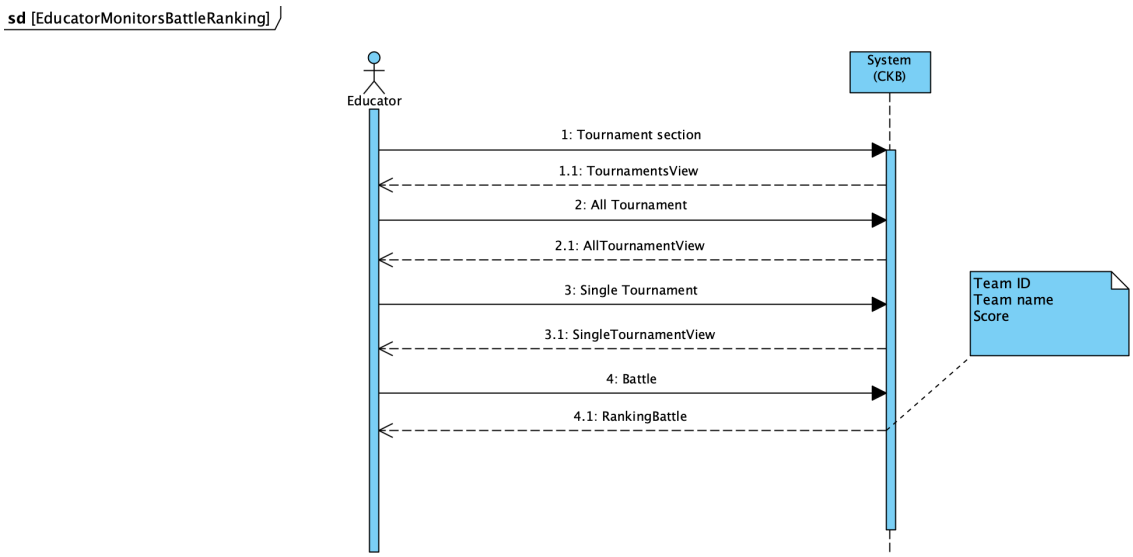


Figure 18: Educator monitors battle ranking

3.4.14 Student monitors battle ranking

ID	14
Name	Student monitors battle ranking
Actor	Student
Entry Conditions	The student has logged into the system.
Events Flow	1. The student, through the homepage, clicks on the "Tournament" section. 2. The system presents a list of all the tournaments available on the platform. 3. The student clicks on a specific tournament. 4. The system displays the dashboard containing battles for the selected tournament to the student. 5. The student selects the battle they want to view the ranking of.
Exit Conditions	The system displays the battle ranking to the student, containing the following fields: • Team ID • Team name • Team Score calculated in real-time by the system

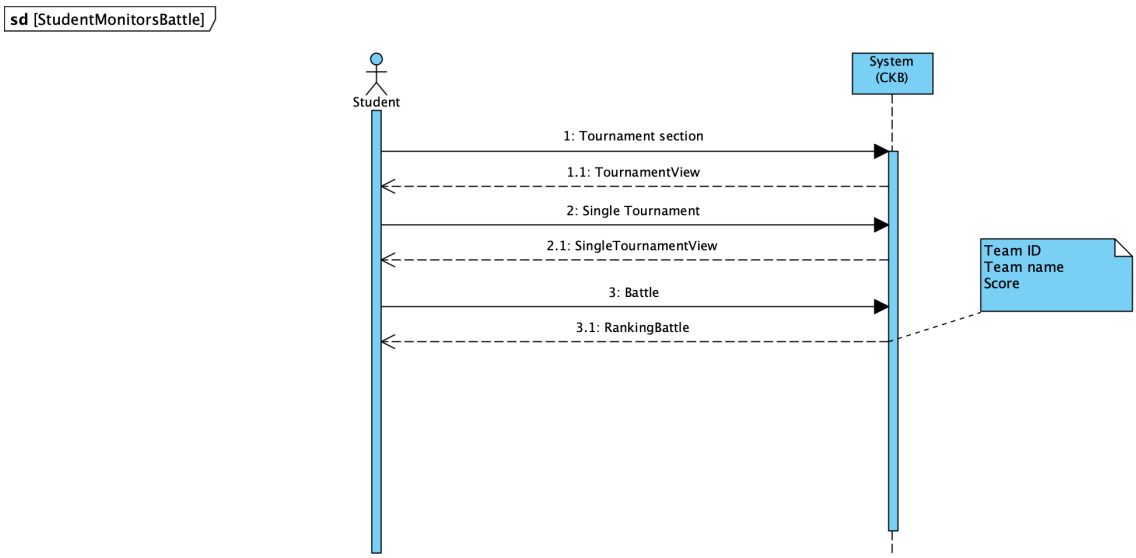


Figure 19: Student monitors battle ranking

3.4.15 Educator monitors tournament ranking

ID	15
Name	Educator monitors tournament ranking
Actor	Educator
Entry Conditions	• The educator has logged into the system. • The tournament of interest has been created, and the educator has the necessary permissions.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on "All Tournaments". 4. The system displays a page that includes all tournaments of the platform, even those for which the educator does not have permission to create battles. 5. The educator clicks on a specific tournament. 6. The system displays the dashboard containing battles for the selected tournament to the educator and the "Tournament ranking" button. 7. The educator clicks on "Tournament ranking".
Exit Conditions	The system displays the tournament ranking containing various fields: • Student ID • Student Name • Student Surname • Student Score

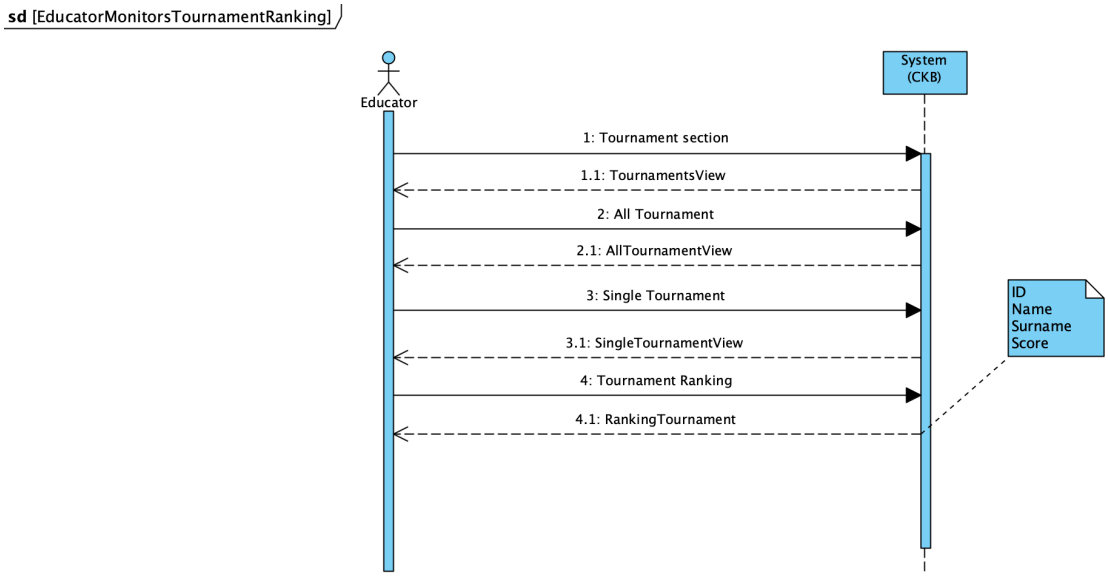


Figure 20: Educator monitors tournament ranking

3.4.16 Student monitors tournament ranking

ID	16
Name	Student monitors tournament ranking
Actor	Student
Entry Conditions	The student has logged into the system.
Events Flow	1. The student, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays all the tournaments available on the platform. 3. The student clicks on a specific tournament. 4. The system displays the dashboard containing battles for the selected tournament to the student and the "Tournament ranking" button. 5. The student clicks on "Tournament ranking".
Exit Conditions	The system displays the tournament ranking containing various fields: • Student ID • Student Name • Student Surname • Student Score

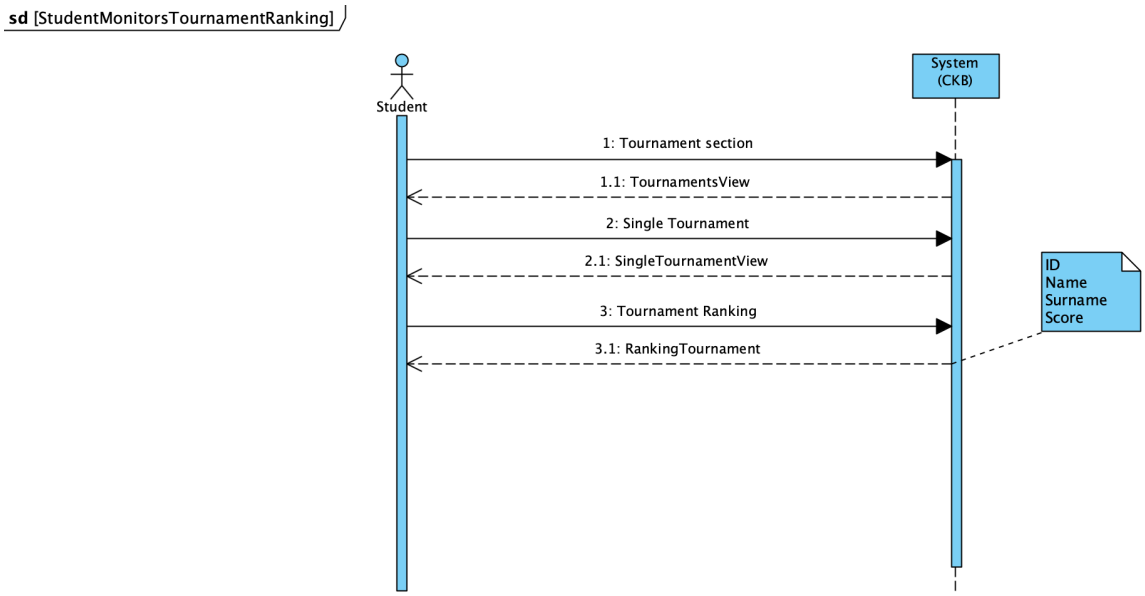


Figure 21: Student monitors tournament ranking

3.4.17 Tournament Closure

ID	17
Name	Tournament Closure
Actor	Educator
Entry Conditions	• The educator is logged into the system. • The educator has created the tournament of interest.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on a specific tournament. 4. The system displays the dashboard containing battles for the selected tournament to the educator and the "Closure the tournament" button. 5. The educator selects "Close the tournament." 6. The system displays a warning message to confirm if the educator really intends to close the tournament. 7. The educator confirms the closure of the tournament, and the system checks if there are still open battles within the tournament. 8. The system changes the status of the same tournament from "Open" to "Closed."
Exit Conditions	The system updates the database and displays to the educator the message "Closure successfully completed.".
Exceptions	• The educator responds to the warning sent by the platform about closing the tournament by clicking on "Cancel." Then, the platform displays the same page with a message explaining to the educator that the tournament closure operation has been canceled. • The educator decides to close the tournament when there are still ongoing battles. So, the platform displays the same page with a message explaining to the educator that the tournament cannot be closed due to battles still in progress.

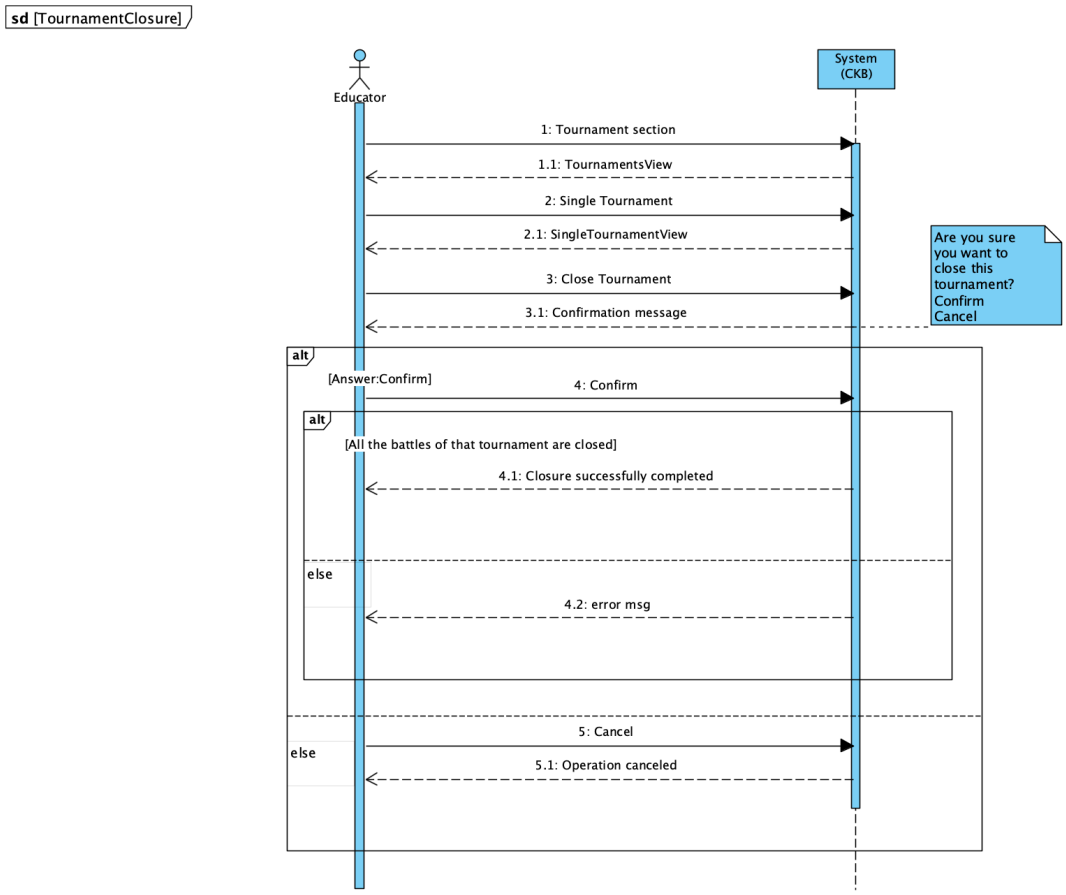


Figure 22: Tournament Closure

3.4.18 Battle elimination

ID	18
Name	Battle elimination
Actor	Educator
Entry Conditions	• The educator is logged into the system. • The educator has created the tournament of interest.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on a specific tournament. 4. The system displays the dashboard containing battles for the selected tournament to the educator and the "Closure the tournament" button. 5. The educator selects the battle of interest. 6. The system displays the BattleView. 7. The educator, before the registration deadline or after the termination of the battle, selects "Eliminate the battle". 8. The system displays a warning message to confirm if the educator really intends to eliminate the battle. 9. The educator confirms the elimination of the battle.
Exit Conditions	The system updates the database and displays to the educator the message "Closure successfully completed.".
Exceptions	The educator responds to the warning sent by the platform about the elimination of the tournament by clicking on "Cancel." Then, the platform displays the same page with a message explaining to the educator that the tournament closure operation has been canceled.

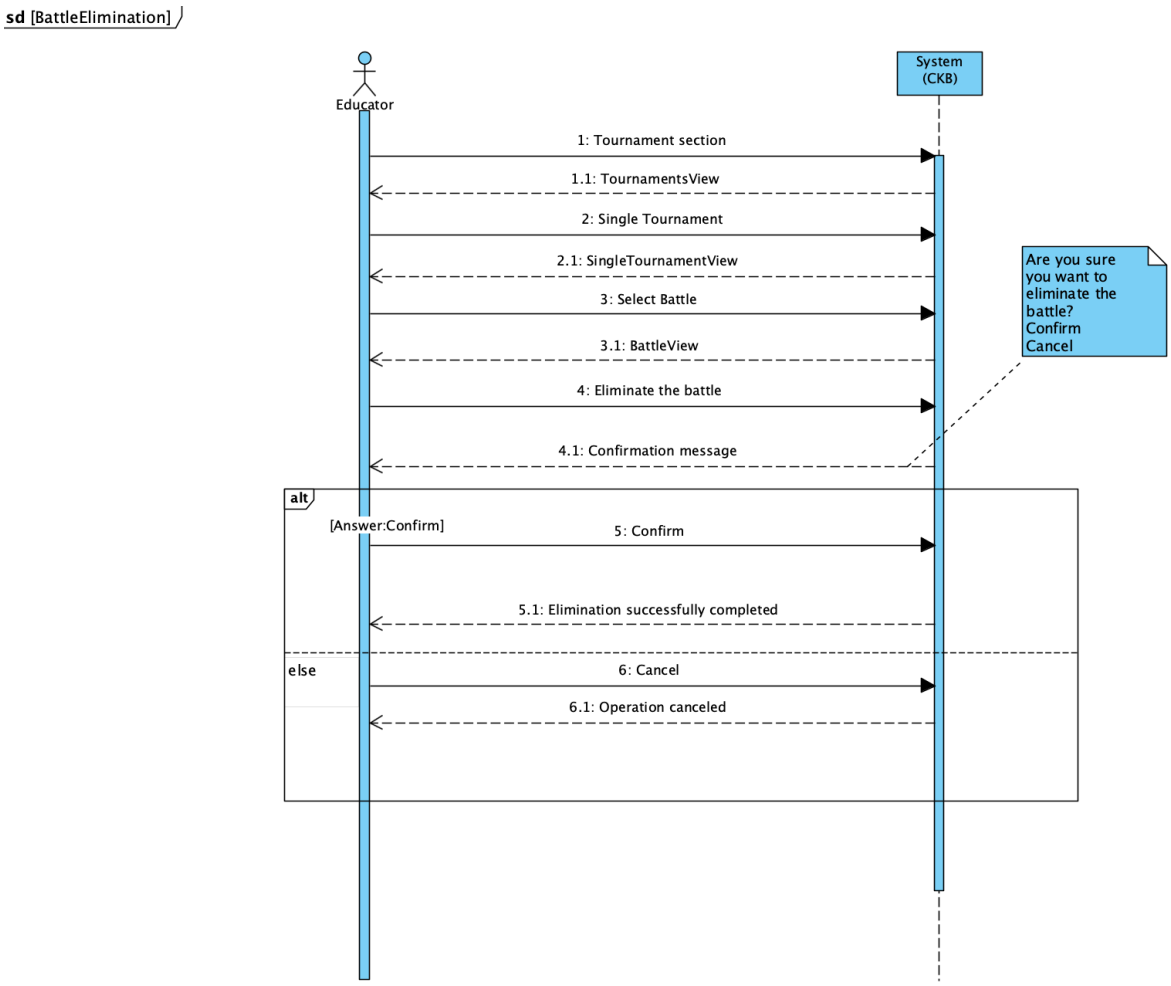


Figure 23: Battle elimination

3.4.19 Tournament Elimination

ID	19
Name	Tournament Elimination
Actor	Educator
Entry Conditions	• The educator is logged into the system. • The educator has either created the tournament of interest or possesses the necessary permissions to manage it.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on a specific tournament. 4. The system displays the dashboard containing battles for the selected tournament to the educator and also the "Delete tournament" button. 5. The educator selects "Delete tournament." 6. The system displays a warning message to confirm if the educator really intends to delete the tournament. 7. The educator confirms the elimination of the tournament, and the system checks if there are still open battles within the tournament. 8. The system delete the tournament.
Exit Conditions	The system updates the database and displays to the educator the message "Elimination successfully completed.".
Exceptions	• The educator responds to the warning sent by the platform about eliminating the tournament by clicking on "Cancel." Then, the platform displays the same page with a message explaining to the educator that the tournament elimination operation has been canceled. • The educator decides to delete the tournament when there are still ongoing battles. So, the platform displays the same page with a message explaining to the educator that the tournament cannot be deleted due to battles still in progress.

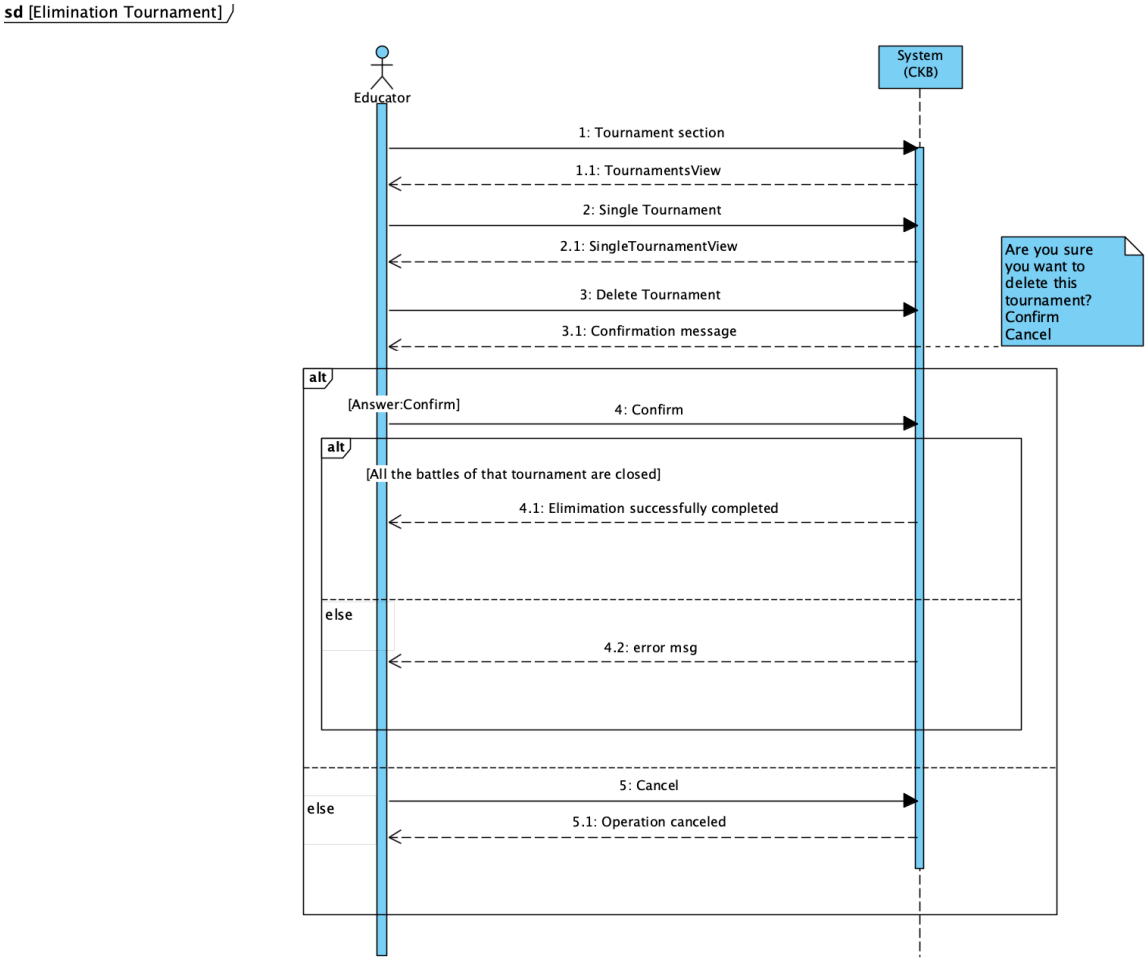


Figure 24: Tournament Elimination

4 Alloy

4.1 Alloy Model

In this section will be provided a formal model of the problem achieved using Alloy.

```
open util/boolean

sig Text{}
sig Day{}
sig Month{}
sig Year{}
sig Hour{}
sig Minute{}
sig Second{}
sig Date{
  day: one Day,
  month: one Month,
  year: one Year,
  hour: one Hour,
  minute: one Minute,
  second: one Second
}
sig Score{}

abstract sig User {
  email: one Text,
  password: one Text,
  name: one Text,
  surname: one Text
}

sig Student extends User {
  // A student can subscribe to zero or more tournaments
  subscribedTournaments: set Tournament,
  hasTeam: set Team
}

sig Educator extends User {
  personalEvaluation: set MantotEvaluation,
  // An educator creates one or more tournaments
  createdTournaments: set Tournament,
  // An educator creates one or more battle
  createdBattle: set Battle
}



sig AutomatedEvaluation {
  functionalScore: one Score,
  timelinessScore: one Score,
  qualityScore: one Score,
  totalScore: one Score,
  // An automated evaluation is related to one project
  project: one Project
}

sig MantotEvaluation extends AutomatedEvaluation{
  personalScore: one Score,
       doneBy: one Educator
}

sig Project {
  //A project is associated to one battle
```

```
  associatedBattle: one Battle,
  // A project has a GitHub repository
  repository: one GithubRepository,
  // A project has one automated evaluation
  automatedEvaluation: one AutomatedEvaluation
}

sig GithubRepository {
  // A GitHub repository can be forked one project
  forkedProjects: one Project,
  // Each GithubRepository is associated to one team
  hasTeam: one Team
}

sig Tournament {
  isOpen: one Bool,
  //only educators that are authorized by the creator
  authorizedEducator: set Educator,
  registrationDeadline: one Date,
  // A tournament has one ranking
  tournamentRanking: one TournamentRanking,
  // A tournament is composed of zero or more battles
  battles: set Battle,
  // A tournament can have zero or more participating students
  participatingStudents: set Student,
  // A tournament is created by one educator
  creator: one Educator
}

abstract sig Ranking {
  // This will contain common fields or relations for all types of rankings
}

sig BattleRanking extends Ranking {
  // A battle ranking is generated from one battle
  generatedFrom: one Battle
}

sig TournamentRanking extends Ranking {
  // A tournament ranking is generated from one tournament
  generatedFrom: one Tournament
}

sig Battle {
  minStud: one Int,
  maxStud: one Int,
  registrationDeadline: one Date,
  submissionDeadline: one Date,

  // A battle contains one or more projects
  projects: set Project,
  // A battle generates one battle ranking
  battleRanking: one BattleRanking,
  // A battle has zero or more participating teams
  participatingTeams: set Team,
  // A battle is created by one educator
  creator: one Educator,
  // A battle is associated to one tournament
  associatedTournament: one Tournament

}{minStud <= maxStud && minStud >= 1}

sig Team {
```

```
122   // A team has one or more students
123   members: some Student,
124   // A team participates in a battle
125   participate: one Battle,
126   //Each team can have a gitHub repository
127   associatedRepository: lone GithubRepository
128 }
129   fact {
130 // Two user can  t  have the same email address
131   no disj u1 , u2: User | u1 . email = u2 . email
132 }
133
134 // A student can be enrolled in a team only if they are enrolled in at least
      one tournament.
135 fact {
136   all s: Student | (some t: Team | s in t.members) => (some t: Tournament | s
      in t.participatingStudents)
137 }
138
139 // if an educator is a creator of the tournament, he is in the list of
      authorized educators.
140 fact {
141   all t: Tournament | t.creator in t.authorizedEducator
142 }
143
144 //An Educator can create battles only if they have tournament permissions
145 fact {
146   all e: Educator, b: Battle | e in b.creator => e in b.associatedTournament.
      authorizedEducator
147 }
148
149
150
151
152 //For every team participating in a battle, each member of that team must also
      be a
153 //registered participant in the tournament associated with the battle.
154 fact {
155   all t: Team, b: Battle |
156    t in b.participatingTeams =>
157      all s: t.members | s in b.associatedTournament.participatingStudents
158 }
159
160
161
162
163 //Every team associated with a GitHub repository used in a project linked to a
      competition (battle) is listed
164 //as a participant in that competition.
165 fact {
166   all t: Team, r: GithubRepository, p: Project, b: Battle |
167     r.hasTeam = t and p.repository = r and p.associatedBattle = b => t in b.
      participatingTeams
168 }
169
170
171
172 //if a battle is associated to a tournament, then that tournament contains that
       battle
173 fact {
174  all b: Battle, t: Tournament | b in t.battles <=> t in b.associatedTournament
175 }
176
```

```
177 // If a Github repository is associated to a Team, then the Team has that
        Github repository
178 fact{
179   all r: GithubRepository, t: Team| r in t.associatedRepository <=> t in r.
        hasTeam
180 }
181
182 // if a github repository is associated to a project, then the project is
        associated to that repository
183 fact{
184   all r: GithubRepository, p: Project | r in p.repository <=> p in r.
        forkedProjects
185 }
186
187 //if  battle generates a battle ranking, then the battle ranking is associated
        to that battle
188 fact {
189   all b: Battle , r: BattleRanking | b in r.generatedFrom <=> r in b.
        battleRanking
190 }
191
192 // if a tournament generates a tournament ranking, then the tournament ranking
        is associated to that tournament
193 fact {
194   all t: Tournament , r: TournamentRanking | t in r.generatedFrom <=> r in t.
        tournamentRanking
195 }
196
197 // if an educator does a man tot Evaluation, then the man tot evaluation is
        done by Educator
198 fact{
199   all m: MantotEvaluation, e: Educator | e in m.doneBy <=> m in e.
        personalEvaluation
200 }
201
202 // if a project has an automated evaluation, then the automated evaluation is
        associated to that project
203 fact{
204   all a: AutomatedEvaluation, p: Project | p in a.project <=> a in p.
        automatedEvaluation
205 }
206
207 // if an educator has created the tournament, then the creator of the
        tournament is that educator
208 fact {
209   all t: Tournament, e: Educator | e in t.creator <=> t in e.createdTournaments
210 }
211
212 // if an educator has created the battle, then the creator of the battle is
        that educator
213 fact {
214   all b: Battle, e: Educator | e in b.creator <=> b in e.createdBattle
215 }
216
217 // if the battle has an associated project, then the associatedBattle of the
        project is that battle
218 fact {
219   all b: Battle, p: Project | p in b.projects <=> b = p.associatedBattle
220 }
221
222 // if the battle has a team as a participant, then that battle is the one the
        team participates in
223 fact {
```

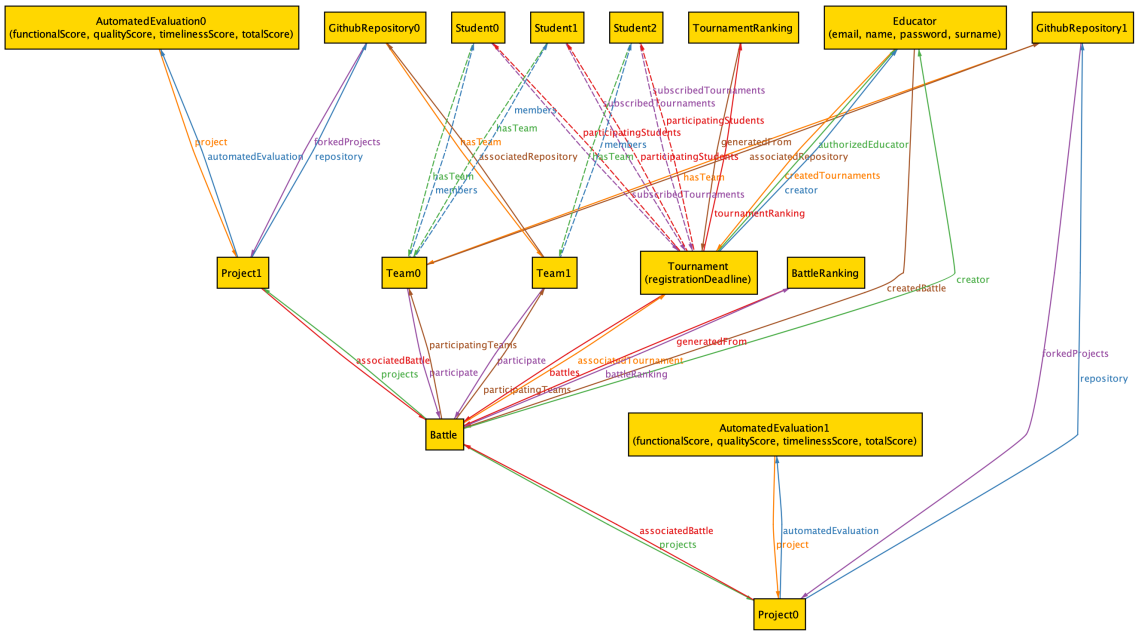
```
  all b: Battle, t: Team | t in b.participatingTeams <=> b in t.participate
}

// if the student partecipates in a tournament, then that student is in the students partecipating to the tournament

fact {
  all s: Student, t: Tournament | s in t.participatingStudents <=> t in s.subscribedTournaments
}

// if the student participates in a team, then that student is one of the team's members
fact {
  all s: Student, t: Team | s in t.members <=> t in s.hasTeam
}

// if a student is part of a team of a battle then he can't be part of other teams participating to the same battle
fact StudentInOnlyOneTeamPerBattle {
  all b: Battle |
    no disj t1, t2: b.participatingTeams |
    some t1.members & t2.members
}


// Ensure that every battle created by an educator is part of a tournament that the educator is authorized to manage
assert EducatorAuthorizedForTheirBattles {
  all e: Educator, b: e.createdBattle |
    e in b.associatedTournament.authorizedEducator
}

//check EducatorAuthorizedForTheirBattles


// Ensure that every project in a battle is associated with a team participating in that battle
assert ProjectTeamBattleConsistency {
  all b: Battle, p: b.projects |
    some t: Team | p.repository.hasTeam = t and t in b.participatingTeams
}


//check ProjectTeamBattleConsistency for 6


// world 1 emphasizes the role of the student.
pred world1 {
  #Student =3
  #Educator = 1
  #AutomatedEvaluation = 2
  #MantotEvaluation = 0
  #Tournament = 1
  #GithubRepository = 2
  #TournamentRanking = 1
  #BattleRanking = 1
  #Team = 2
  #Battle = 1
  #Project = 2
}

//run world1  for 5
```

```
281
282
283 //world 2 emphasizes the role of the educator and the various methods of
        evaluation and ranking
284 pred world2 {
285    #Student =2
286   #Educator = 2
287   #AutomatedEvaluation = 2
288   #MantotEvaluation = 1
289   #Tournament = 2
290   #GithubRepository = 2
291   #TournamentRanking = 2
292   #BattleRanking = 2
293   #Team = 2
294   #Battle = 2
295   #Project = 2
296 }
297 //run world2 for 5
```

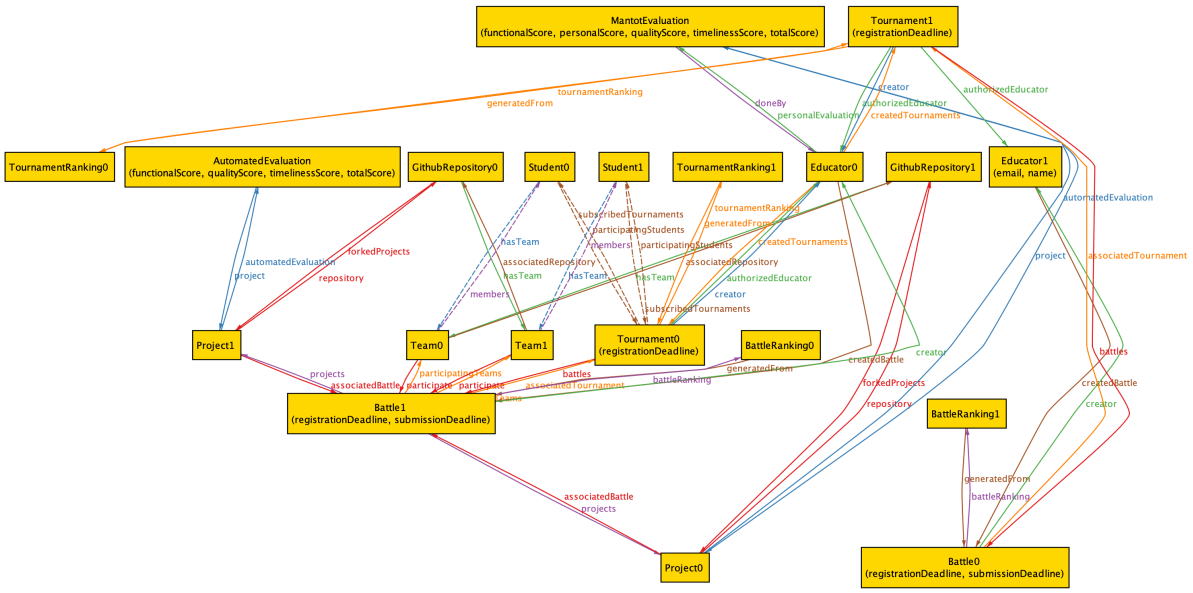
4.2 World

4.2.1 First World

The 'First World' highlights the role of the students, showcasing their main interactions and relationships within the educational system.



4.2.2 Second World



The 'Second World' emphasizes the role of educators and the methods of evaluation and ranking.

4.3 Example of dynamic model

By making the isOpen relationship mutable for the tournament , it is possible to describe the situation in which the tournament is closed by an educator. From that point on, by imposing this fact, it becomes possible to establish that the tournament cannot be reopened. In this Alloy model, we incorporate only the components that differ from the previous model.

```
sig Tournament {
  //The tournament can be closed by the educator
  var isOpen: one Bool,
  //only educators that are authorized by the creator
  authorizedEducator: set Educator,
  registrationDeadline: one Date,
  // A tournament has one ranking
  tournamentRanking: one TournamentRanking,
  // A tournament is composed of zero or more battles
  battles: set Battle,
  // A tournament can have zero or more participating students
  participatingStudents: set Student,
  // A tournament is created by one educator
  creator: one Educator
}



fact StatusTournament { all t: Tournament|
always ( t.isOpen = False implies
after always  t.isOpen= False )
}

// Predicato per chiudere un torneo
pred closeTournament[t: Tournament] {
  t.isOpen = True
  implies
  t.isOpen' = False
}
```

```
30
31
32 assert CheckStatusTournament { all t: Tournament |
33   always closeTournament[t]
34    implies
35    always after t.isOpen = False
36 }
37 check CheckStatusTournament for 3 steps
```

5 Time Spent

The time tables written below represent just an approximation of the time spent for the writing and discussions the team had for each specific chapter of this document. It is important to note that both team members worked together for the entire duration of each task.

Task	**Hours**
Chapter 1	20-20
Chapter 2	30-30
Chapter 3	30-30
Chapter 4	25-25

6 References

- Diagrams made with: StarUML, Visual Paradigm
- Alloy models made with: Alloy Analyzer
- Alloy models runned and checked with: Alloy Analyzer