POLITECNICO
MILANO 1863

SOFTWARE ENGINEERING II

CKB – CodeKataBattle for All
Requirements Analysis and Specification Document

Version 1.0

Polito Attilio

Rigione Pisone Raimondo

Soricelli Francesco

December 22, 2023

CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

1.1 PURPOSE

The aim of the CodeKataBattle (CKB) platform is to provide a dynamic and engaging environment for students to improve their software development skills through collaborative learning. The project aims to foster a culture of continuous improvement using coding challenges and competitions in a structured educational environment. The main goals of the project are:

2 Skill Development: CodeKataBattle promotes a test-first approach to make students improve their programming skills by challenging them with diverse programming exercises.
3 Collaborative learning: Encourage teamwork and collaboration between students by forming teams and allowing them to participate in coding battles. This collaborative approach not only strengthens technical capabilities, but also promotes effective communication and problem-solving skills.
4 Define Educators' role: Provide educators with a platform to create, manage, and evaluate coding battles. Educators can create tournaments, give permissions to other educators to manage them, design badges for the gamification aspect, and guide students to hone their coding skills through their Manual Evaluations relative to students performances.
5 Automated Evaluation: Implement a robust automated evaluation system to evaluate code quality, functional aspects, and timeliness. Integration with static analysis tools ensures comprehensive evaluation and timely feedback, motivating students to iterate and refine their solutions.
6 Competition and Ranking: Introduce competitive elements to stimulate students' motivation. By assigning scores during battles and creating real-time ranks, the platform promotes a sense of accomplishment and healthy competition among participants.
7 Gamification: Use gamification elements, such as badges, to enhance the learning experience. These badges provide additional motivation for students to excel in various aspects of coding challenges, to recognize and celebrate individual achievements.
8 Tournament Structure: Organize coding battles into tournaments, allowing students to accumulate scores in multiple challenges and visualize tournament informations.
9 Notification system: Implement a notification system that facilitates effective communication with the platform. Students and Educators will receive timely updates on upcoming battles, deadlines and tournament results to provide information and maintain engagement.

Through these goals, CodeKataBattle platform not only improves technical skills, but also provides continuous learning and improvement for the students in the field of software development.

1.1.1 GOALS

These are the goals we want to achieve by implementing our software:

Goals	Description
G1	Students improve their software development skills
G1.1	Allow students to easily find and explore upcoming coding challenges, including details on the challenge description, requirements, and associated deadlines.
G1.2	Allow students to join a battle.
G2	Implement a competitive learning environment.
G2.1	Allow educators to create tournaments.
G2.1.1	Allow educators to delegate tournament management to their peers after the tournament has been created.
G2.2	Allow educators to initiate a battle relative to a tournament, providing options to specify the minimum and maximum number of participants, evaluation criteria, registration cutoff, final submission deadline, and the chosen code kata.
G2.3	Allow students to submit their coding solutions, triggering the automated evaluation process and updating team scores accordingly.
G2.4	Allow users to access battle and tournament scores through a real-time leaderboard. Extend this feature even after the conclusion of the competitions for ongoing visibility.
G2.5	Allow users to see the list of ongoing tournaments with the corresponding tournament rank for each student.
G2.6	Allow students to form coding teams through an invitation system, permitting them to efficiently join and collaborate on CKB challenges.
G3	Develop a comprehensive notification system to facilitate timely communication, ensuring both students and educators stay informed about upcoming tournaments, ongoing battles, battle conclusions, and tournament results.
G4	Provide educators with a platform to create, manage, and assess coding battles and to create students' achievements though badges.
G4.1	Allow educators to manually assess students' work by assigning a numerical score at the end of a battle, during the consolidation phase (if it is present).
G4.2	Allow educators to create variables in order to make rules for badges' assignment.
G4.3	Allow educators to make rules for badges' assignment.
G4.4	Allow educators to create new badges.

TABLE 1: TABLE OF GOALS

1.2 SCOPE

The CodeKataBattle (CKB) project demands innovative solutions to effectively train students, preparing them for the challenges of the professional world, with the increasing importance of skills in software development and the increasing complexity of the industry providing students with an interactive and collaborative environment to improve their abilities. CKB provides a platform that allows students to refine their skills through practical and exciting programming challenges.

The platform has 2 main actors - students and educators - who are actively involved, each playing a unique role. Students who make up the active components of the system participate in the Battle by registering individually or forming teams based on specific participation requirements. When a student joins the battle, after the battle starts, he will receive a notification with a link to the assigned code kata GitHub repository for his Team. Here, follow the test-driven development (TDD) approach, he'll complete the exercise and submit the solution by the specified deadline.

Educators, on the other hand, act as architects of competitions. They have a role in creating tournaments and defining combat specifications, including details such as code type descriptions, programming languages, tests to pass, registration and submission deadlines. An Educator can also set mandatory evaluation parameters, such as test code quality, and optional evaluation parameters, such as the manual evaluation feature. After an Educator has created a battle, the platform manages the student enrollment process, distributes GitHub repositories, and automates evaluation through GitHub Actions.

During the competition, the system dynamically assigns scores to the team based on automatic evaluation (test success, timeliness of submission, quality of code, etc.) and manual evaluation by educators. The scores accumulated in each battle contribute to each student's overall tournament score. To further enrich the experience, the platform has introduced a gamification system through the awarding of badges. Educators define rules and variables associated with badges that are assigned to students based on their performance during the tournament. These gamification badges will be displayed in the student's profile.

Joining CodeKataBattle not only improves technical skills, but also promotes collaboration, problem solving, and compliance with best practices in software development. This platform prepares students for the challenges they face in their professional careers. In addition, the gamification element adds an exciting layer to the learning process and motivates students to strive for excellence.

Going forward, the CKB team expects continuous improvement and expansion of the platform. Future plans include considering additional features, improving the evaluation process, and incorporating feedback from both students and educators to ensure an optimal learning experience. The goal is a major platform to hone the skills of the next generation of software developers evolving with the ever-changing landscape of software development education.

1.2.1 World Phenomena

World Phenomena	Description
WP1	The student navigates CKB platform to improve his software development skills.
WP2	The student sees the notification relative to the creation of a new tournament.
WP3	The student sees the notification relative to the creation of a new battle.
WP4	The student choses a tournament to participate.
WP5	The student choses a coding battle to participate.
WP6	The students communicate with other students to form coding teams to participate to a battle.
WP7	The students adopt Test-Driven Development (TDD) practices as guided by coding challenges on the platform.
WP8	The educator checks students' code during the consolidation phase.
WP9	The user sees the notification of the availability of tournament leaderboard.
WP10	The user sees the notification of the availability of battle leaderboard.
WP11	The student utilizes GitHub to fork a coding challenge repository and sets up an automated workflow through GitHub Actions.

Table 2: Table of World Phenomena

1.2.2 Shared Phenomena

Shared Phenomena	Description	Control
SP1	The user registers into the CKB platform.	World Controlled
SP2	The user logs into the platform.	World Controlled
SP3	The users explore comprehensive profiles of subscribed students, viewing their badges.	World Controlled
SP5	The educator creates a coding tournament.	World Controlled
SP6	The educators close a coding tournament.	
SP7	The educator delegates the management of a coding tournament to a colleague after its creation.	World Controlled
SP8	The educator initiates a coding battle within a tournament, defining key parameters such as participants, the code kata, evaluation criteria, and deadlines.	World Controlled

SP9	The educator introduces gamification elements, creating variables, rules, and badges that recognize and reward student achievements.	World Controlled
SP10	The educator assesses students' work manually by assigning numerical scores during the consolidation phase.	World Controlled
SP11	The student joins a tournament.	World Controlled
SP12	The student joins a battle.	World Controlled
SP13	GitHub notifies the platform after every student commit	World Controlled
SP14	Students efficiently collaborate by forming coding teams through the platform's invitation system.	World Controlled
SP15	The student submits his coding solution, triggering an automated evaluation process that updates team scores.	World Controlled
SP16	The users access real-time battle and tournament's scores through an interactive leaderboard.	Machine Controlled
SP17	The platform notifies all the tournament enrolled students when a battle is created within the tournament.	Machine Controlled
SP18	The platform's notification system automatically sends alerts and updates to users about ongoing tournaments and ranks.	Machine Controlled
SP19	The platform notifies all the students subscribed to a battle with the link of the automatically created GitHub repository when the registration deadline expires.	Machine Controlled
SP20	The platform automatically creates a GitHub repository when the registration deadline for a battle expires.	Machine Controlled
SP21	The platform runs an automated evaluation processes in response to student submissions, updating team scores based on predefined criteria.	

TABLE 3: TABLE OF SHARED PHENOMENA

1.2.3 MACHINE PHENOMENA

Machine Phenomena	Description
MP2	The platform's system automatically updates real-time battle and tournament ranking for users.
MP3	The platform awards badges to students who successfully meet the specified objectives outlined in the corresponding badge rules.

TABLE 4: TABLE OF MACHINE PHENOMENA

The platform runs an automated evaluation processes in response to student submissions, updating team scores based on predefined criteria.

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

1.3.1 DEFINITIONS

Definition	Description
CodeKataBattle (CKB)	The online platform designed to hold coding competitions and challenges to improve students' software development abilities.
Code Kata	The exercise that includes a brief textual description and a software project with build automation scripts that contains a set of test cases.
Test-first or test-driven development (TDD) approach	A software development model that necessitates writing automatic tests before writing the software that needs to be tested. The development of the application software is solely focused on passing the previously written automatic tests.
Tournament	Organization on CKB created by an educator, consisting of a series of challenges.
Battle	Challenge or competition that is developed by an educator and held during a tournament.
Team	A group of students coming together to take part in a particular CKB battle. Students create teams on the platform before starting a battle.
GitHub repository	A distributed version control system and digital storage area powered by Git. In this instance, the "code kata" refers to the CKB platform programming project linked to a particular battle, and it can be found in the GitHub repository. For the battle, every team has a dedicated GitHub repository where students can work together to track changes, solve problems, and version code.
GitHub Action	GitHub provides an automation service that lets developers automate software workflows. In this case, GitHub Actions are useful to set up an automated workflow between GitHub and the Platform. This workflow is used in the competition to send an API notification to the CKB platform after every commit, enabling the team's score to be automatically updated and computed based on the most recent changes made to the code.
Score	A natural numerical value between 0 and 100 that is established by accounting for some required factors that are assessed entirely automatically and other optional factors that are assessed manually by educators.
Manual evaluation	A subjective evaluation component made by educators who look over and assess students' work during the consolidation phase. This stage provides a chance to assess factors that might not be picked up by an automated evaluation and that might be more specialized or unique.
Notification system	The automated system that alerts teachers and students to impending fights, deadlines, and tournament outcomes.
Gamification badges	Virtual awards that educators define as a representation of each student's achievements during a competition.
Valid credentials	The term valid credentials is referred to the email and the password inserted by a user during the login process; they are said to be valid

	if they correspond to the one inserted by such user during the successful registration process.
Conflicting data	The email address inserted during the registration process by a user is defined as a conflicting data if there are already others users of the CKB application that have registered with the same address
Tournament Management Permissions	An educator can manage a tournament (creating battles, closing the tournament…) if he has created the tournament or another educator has given him the permissions though the "granting permission" function.

TABLE 5: TABLE OF DEFINITIONS

1.3.2 ACRONYMS

Acronym	Description
RASD	Requirement Analysis and Specification Document
API	Application Programming Interface
CKB	CodeKataBattle
TDD	Test-Driven Development

TABLE 6: TABLE OF ACRONYMS

1.3.3 ABBREVIATIONS

Abbreviation	Description
Gn	n-th goal
WPn	n-th world phenomenon
SPn	n-th shared phenomenon
MPn	n-th machine phenomenon
Rn	n-th functional requirement

TABLE 7: TABLE OF ABBREVIATION

1.4 REVISION HISTORY

Version 1.0 - 22/12/2023

Version 2.0 – 04/02/2024

Version	Date	Description
1.0	23/12/2023	First version
2.0	04/02/2024	Second version. • Domain Assumption • World Phenomena • Shared Phenomena • Machine Phenomena • Class Diagram These changes provide a solid basis for understanding the operational context,

		interactions and internal structure. Deepening these areas contributes to an increase in the robustness of the system, promotes effective communication between components, facilitates the development process and encourages adaptability to future developments, improving the quality of the system overall.

1.5 REFERENCE DOCUMENTS

1- Specification document: R & DD Assignment A.Y. 2022-2023;
2- Course slides;
3- RASDs of the previous academic years;
4- Given links.

1.6 DOCUMENT STRUCTURE

Section 1: In this section it is presented an introduction of the project. Initially, it is discussed the purpose of the system and the goals that it aims to reach. Then, it is described its scope. Moreover, the problem it has been examined by means of a list of world, shared and machine phenomena that represent the system. Finally, it is specified a list of definitions, acronyms and abbreviations that are used in the document.

Section 2: Here it is presented an overall description of the project, analyzing multiple scenarios and presenting the UML class diagram that describes the different entities of the system. Furthermore, the use of some statecharts helps to understand what are the actions that can be made through the system and how they happen in a sequence. Finally, it is described more in detail which are the functionalities of the system and the characteristics of the users that interact with it.

Section 3: This section contains the specification of the interface requirements, i.e. the definition of the user, hardware, software and communication interfaces needed for the system. Moreover, the functional requirements necessary for the system to work as desired and to be consistent are defined. It is also specified which of these requirements are needed for each goal to be reached. Then, the requirements are analyzed through the use of use cases and their respective diagrams that represent all the possible scenarios described in the previous section.

Section 4: This section includes the alloy model and the discussion of its purpose. The results obtained by running the model are properly justified, presenting screens of the generated worlds with the respective comments.

Section 5: This section indicates the effort spent by each group member while working on the project.

Section 6: this section includes the references.

2 OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

2.1.1 SCENARIOS

1 - User Registration on CKB

Sarah, a budding developer, discovers CodeKataBattle (CKB) and decides to join the platform. She initiates the user registration process by providing her name, email address, and creating a password. As part of the registration, Sarah receives a verification email from the system to confirm her account. Once verified, she gains access to the CKB platform and its features, including exploring challenges, participating in tournaments, and joining battles.

2 - Exploring and Joining Coding Tournaments

Mark, a student, logs into the CKB platform. He navigates to the upcoming Tournament section to explore details on challenge descriptions, requirements, and associated deadlines and he joins into a tournament.

3 - Joining a Battle

Lisa, a student, decides to participate in a coding battle on the CKB platform. She selects a battle of her choice, follows the registration process, and joins the battle individually. The platform notifies her about successful enrollment and provides details about the upcoming battle.

4 - Creating a Tournament

Professor Smith, an educator, logs into the CKB platform. He creates a new tournament by specifying details such as code type descriptions, programming languages, tests to pass, registration and submission deadlines. He creates new badges with specific rules and variables associated with them. After the tournament concludes, badges are assigned to students based on their performance. Students like Emily can view the badges they have achieved when exploring their profiles on the platform.

5 - Initiating a Battle

Jones, an educator, with management permissions for a tournament given by Professor Smith, initiates a coding battle within the tournament. Jones configures the battle by setting the minimum and maximum number of participants, evaluation criteria, registration cutoff, final submission deadline, and selects a code kata. The platform then manages the student enrollment process and distributes GitHub repositories.

8 - Submitting Coding Solutions

A team of students, including Mark, Lisa, and John, work collaboratively on the coding challenge. They submit their solutions, triggering the automated evaluation process. The platform updates team scores based on test success, timeliness of submission, and code quality.

9 - Real-time Leaderboard Access

Emily accesses the real-time leaderboard to monitor the ongoing competition. She can see the list of ongoing tournaments; she selects one of them and she can access to the corresponding tournament rank for each student. Now she navigates into the battle status section of a battle of the selected tournament, and she can see the real-time Team and Student leaderboard for that battle.

10 - Forming Coding Teams

Mark, Lisa, and John decide to form a coding team for an upcoming battle. They use the invitation system to efficiently join and collaborate on the challenge. The platform ensures that the team size aligns with the specified minimum and maximum number of participants.

11 - Manual Assessment by Educator

Professor Brown, an educator, accesses the platform after the submission deadline. He manually assesses students' work by assigning numerical scores during the consolidation phase. This assessment complements the automated evaluation and provides a complete view of students' performances.

12 - Closing a Tournament

Smith, an educator, state that all the battle relative to a tournament are ended and decides to close the tournament. As soon as the final tournament rank becomes available, the platform notifies all students involved in the tournament.

2.1.2 CLASS DIAGRAM

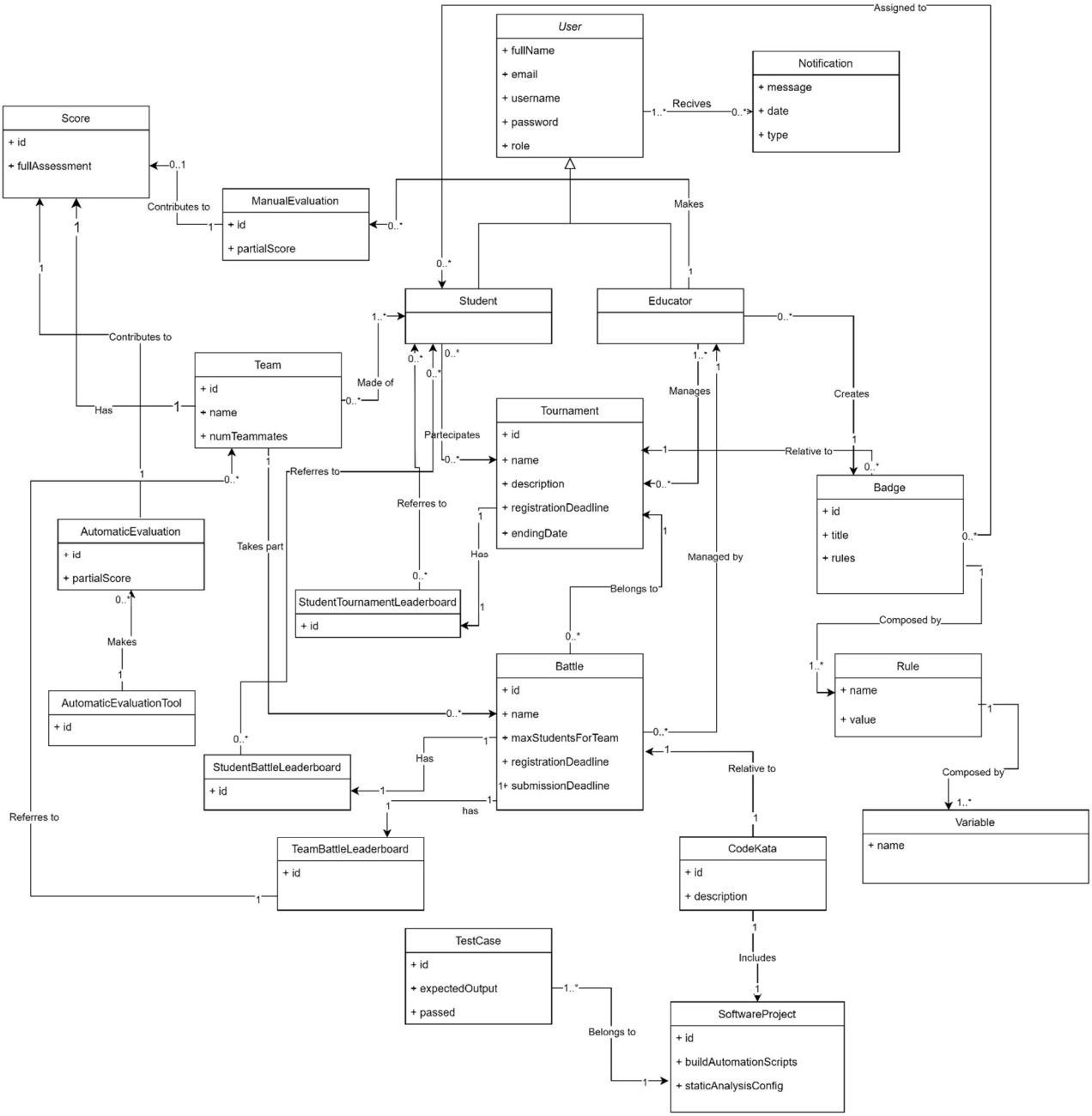


FIGURE 1: CLASS DIAGRAM

In order to avoid any ambiguity into the class diagram, it can be useful do briefly describe such diagram and the choices made to create it.

1- The AutomaticEvaluationTool handles students submits on GitHub, providing the automatic evaluation function for the system.
2- Each battle belongs only to one tournament and can be managed only by one educator (the one who created the battle).
3- The Code Kata contains a textual description and includes a software project with build automation scripts that contains a set of test cases (one or more) that the program must pass.
4- Teams can be composed of one or more students so a student can join a battle alone or in team with other students according to battle parameters.

5- A Tournament is created by an Educator and can be managed by one or more Educators thanks to the “granting permissions” function of the system.

Other cardinalities or information are more intuitive, it is not needed to explain them in detail in order to avoid being unnecessarily verbose.

2.1.3 STATECHARTS

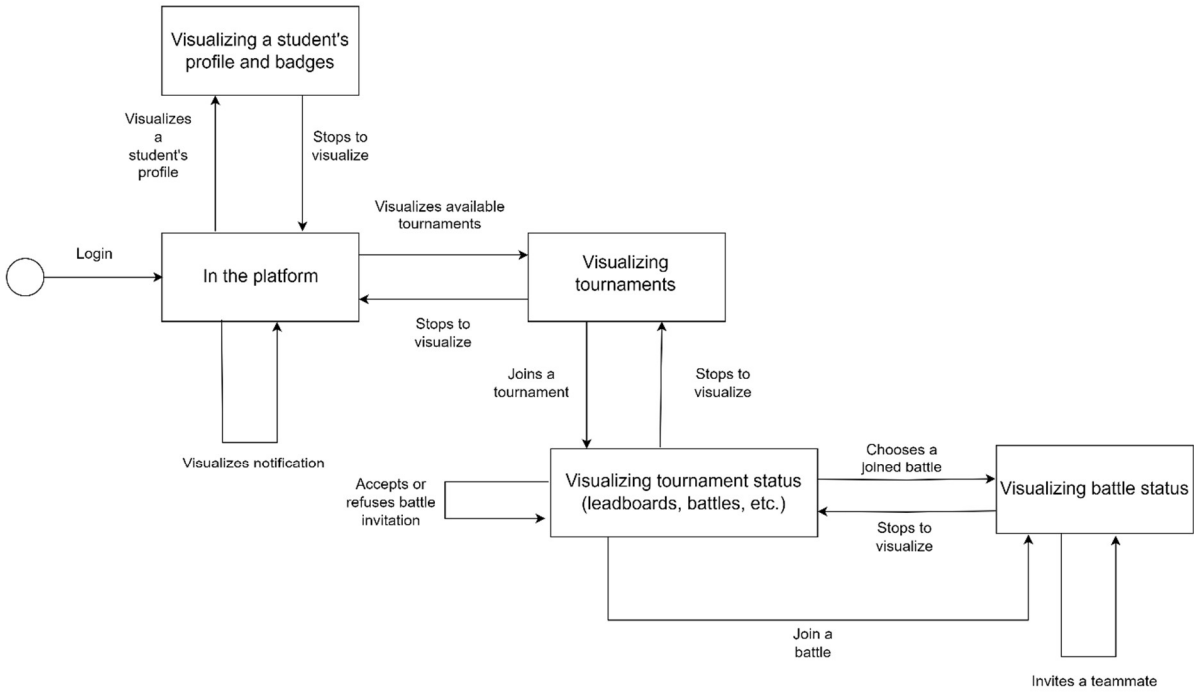


FIGURE 2: STATECHART THAT DESCRIBES STUDENTS' INTERACTION WITH TE SYSTEM

A Student can interact with the system through these actions; he can:

- Visualize profiles of other students.
- Visualize available tournaments and join them.
- Visualize Tournament Status and Leaderboards.
- Join a battle, alone or with a team.
- See Notification and Invitations.

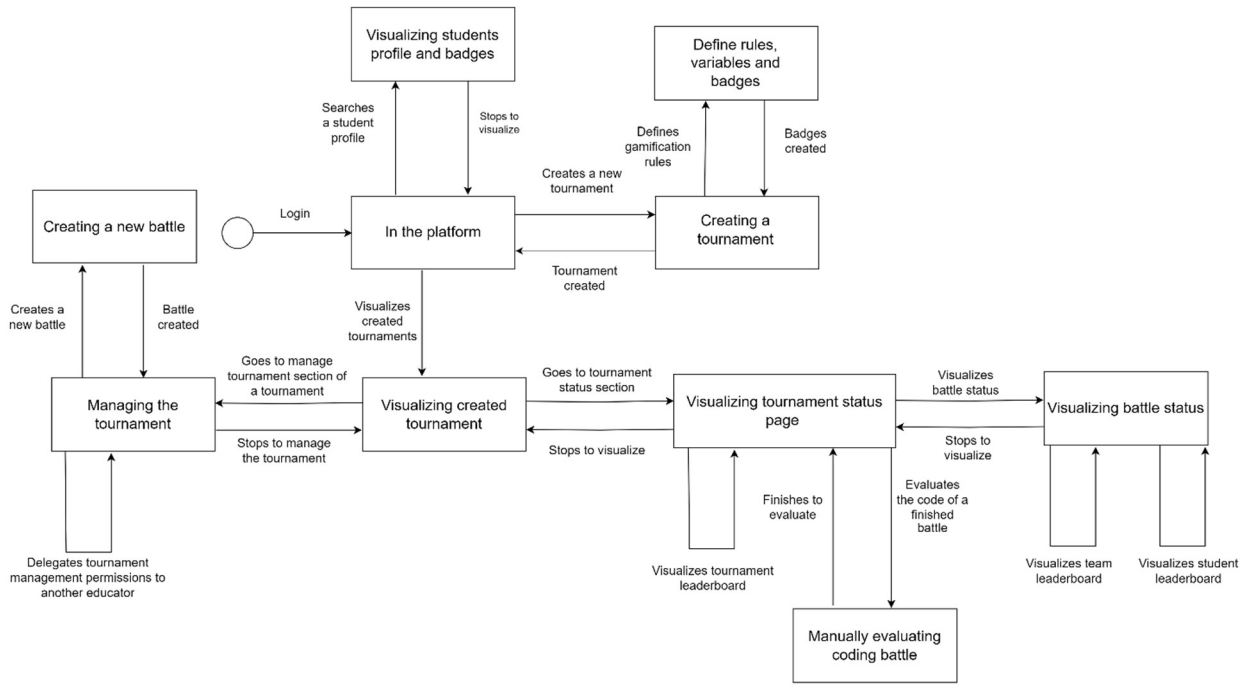


FIGURE 3: STATECHART THAT DESCRIBES EDUCATORS' INTERACTION

An Educator can interact with the system through these actions; he can:

- Visualize profiles of other students.
- Create and Manage Tournaments, creating battle and giving management permissions to other Educators.
- Visualize Tournaments and Battles' status.
- Manually Evaluate students submits.
- Handle students achievements through the gamification aspect.

2.2 PRODUCT FUNCTIONS

The system provides to the students the following functionalities:

Exploration of Coding Challenges

Students can easily find upcoming coding challenges, with a brief description of each challenge, as well as the requirements and deadlines. Challenges are categorized based on difficulty level, programming language, and other criteria.

Participation in Coding Battles

Students can participate in coding battles, competitive events where students compete to achieve the highest scores. Battles can be individual or team-based.

Real-time Competition and Scores

Students can monitor their scores in real-time during battles and tournaments. Scores are automatically updated as students submit their solutions.

Timely Notifications

Students receive notifications about deadlines, results, and new challenges.

Team Formation

Students can form teams through invitations. Teams can work together to solve challenges and compete in battles.

Detailed Student Profile

The detailed student profile displays achievements and badges obtained by students. The profile can be used by students to track their progress and by educators to assess student performance.

The system provides to the educators the following functionalities:

Creation and Management of Tournaments

Educators can create custom coding tournaments based on their needs. They can set the duration, evaluation criteria, included challenges, and other parameters.

Manual Evaluation

Battle scores can be manually evaluated by educators. Manual evaluation can be conducted by educators when the consolidation phase is required for a battle to provide more detailed feedback and evaluate parameters that cannot be automatically analyzed by the system.

Results Display

Educators can view final and real-time leaderboards of battles and tournaments. Results include scores, rankings, and other data.

Badge Creation and Gamification Rules

Educators can create badges and define gamification rules. Badges can be awarded to students based on their performance or the completion of specific objectives.

Delegation of Tournament Management

Educators can delegate tournament management to other educators. This feature can be useful for effectively distributing responsibilities through educators.

2.3 USER CHARACTERISTICS

The system involves the following actors:

Students: Students are the primary users of the system. They use the system to learn coding, enhance their skills, and compete with other students.

Educators: Educators are responsible for creating and managing battles and tournaments. They use the system to assess students' performance.

2.4 ASSUMPTIONS, DEPENDENCIES AND CONSTRAINTS

Domain assumption	Description
D1	The users have access to a device on which they can run the CKB system.
D2	Each student is responsible for maintaining the confidentiality of their account credentials to prevent misuse by others.
D3	The toggle button that users use to sign up can be toggled by every user, so it is assumed that, during the signing-up phase, an educator switches the toggle button on, and a student doesn't.
D4	The personal information inserted during the registration process by a user are assumed to be truthful.
D5	Educators possess all the knowledge to create engaging and challenging tournaments and battles that effectively contribute to students' skill development.
D6	Educators have all the knowledge necessary to evaluate students' work, following a precise methodology.
D7	The GitHub platform consistently triggers CKB updates and scoring calculations in response to student commits, maintaining the integrity of real-time rankings.
D8	Educators are qualified in defining rules, variables, and criteria for gamification badges, ensuring that badges are motivating, fair, and aligned with learning objectives.
D9	Students participating in a battle are expected to follow a test-first approach and develop a solution that passes the required tests.
D10	Students adopt the same username for both the CKB platform and the GitHub platform

TABLE 8: TABLE OF DOMAIN ASSUMPTION

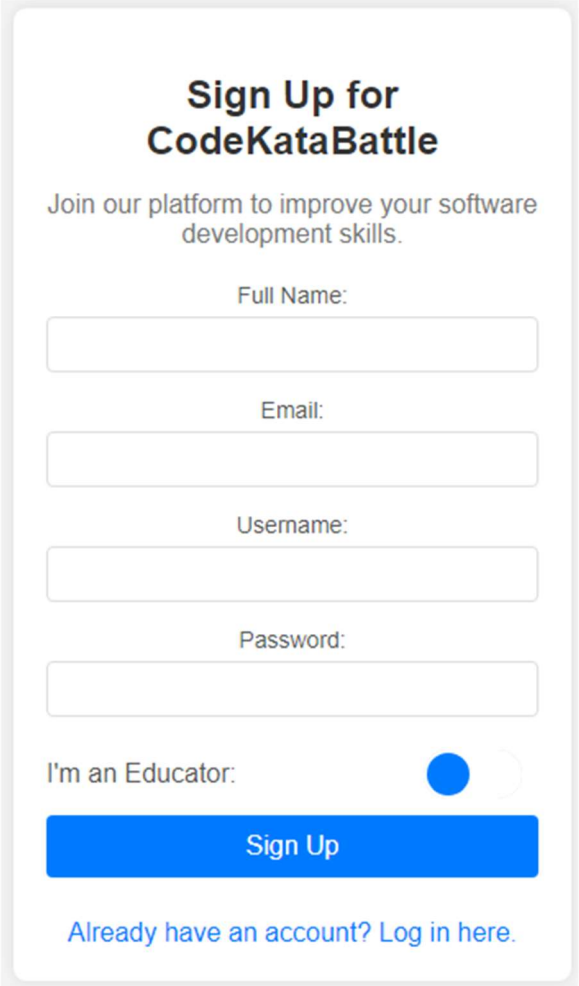
3 SPECIFIC REQUIREMENTS

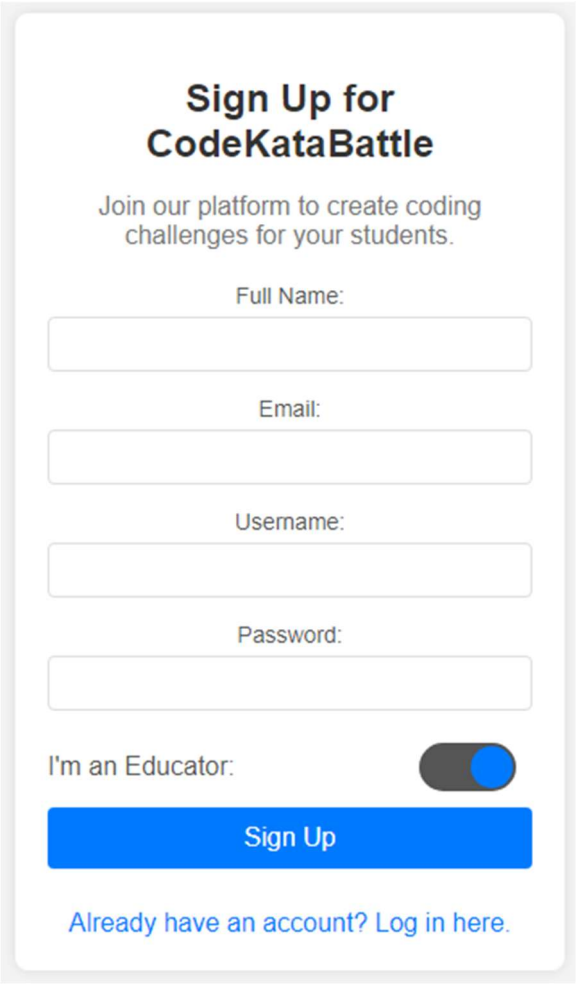
3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 User Interfaces

The user interfaces showed in this document are useful to have a first idea of the structure of the application, both from the side of an educator and from the side of a student. The mockups are realized for a web application. Furthermore, only log in, sign in and home interfaces will be showed; other more specific ones will be presented in the DD document.

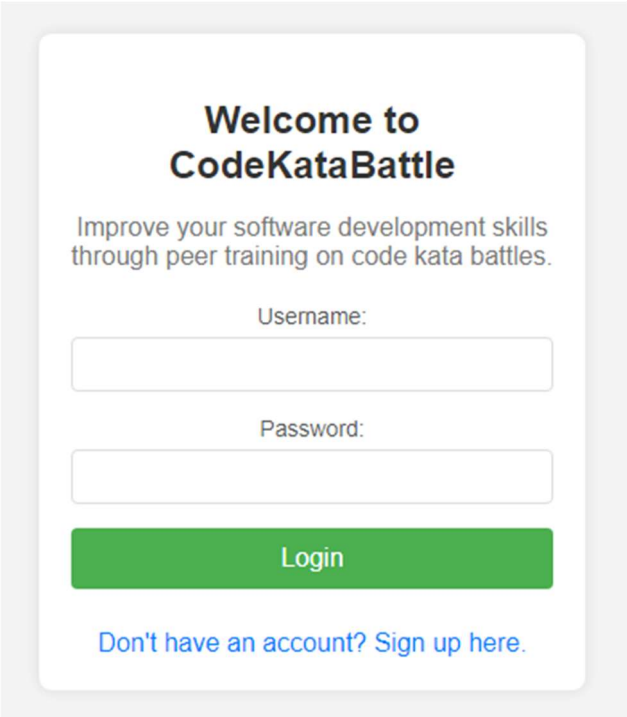
Sign up interfaces





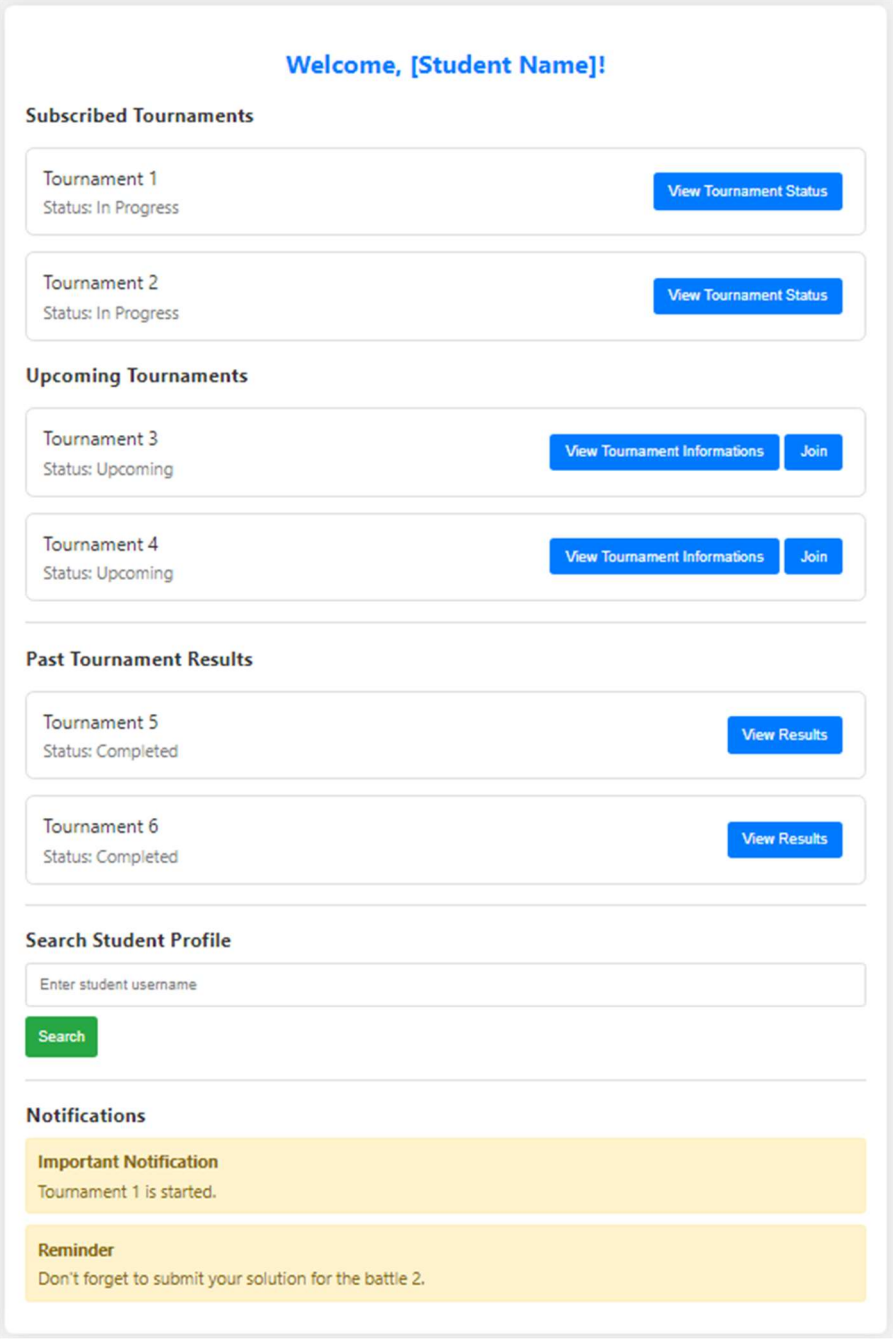
These screens represent the registration page. It is possible to register both as a new student and as an educator by filling in the fields with the required data, making sure to set the toggle button to on if you are an educator.

Login interface



That interface shows the very first screen of the application in which a student or an educator can sign into the application inserting his credentials. If a student or educator is not yet registered on the platform, they can proceed with registration by following the instructions on the web page.

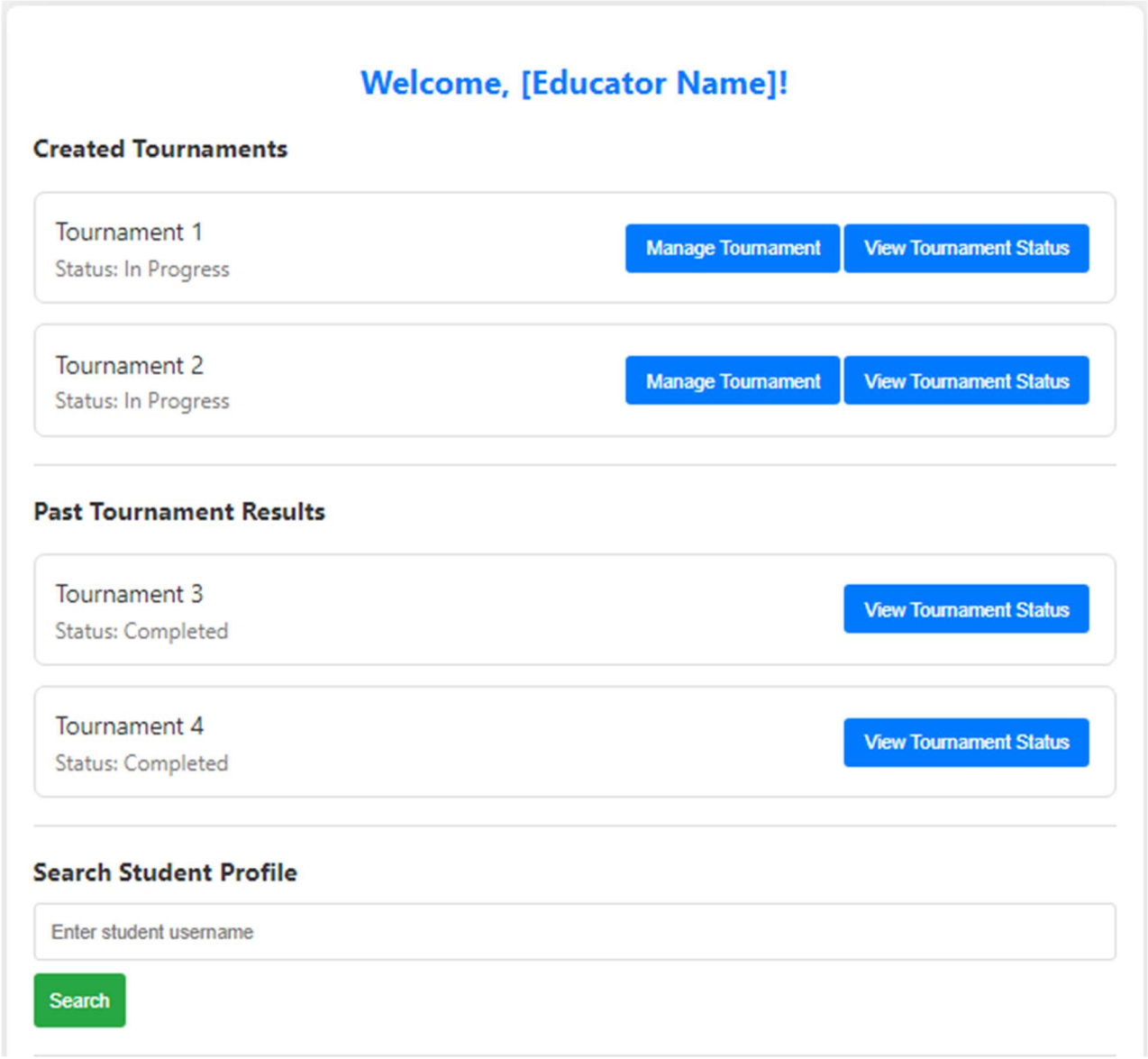
Students' interface



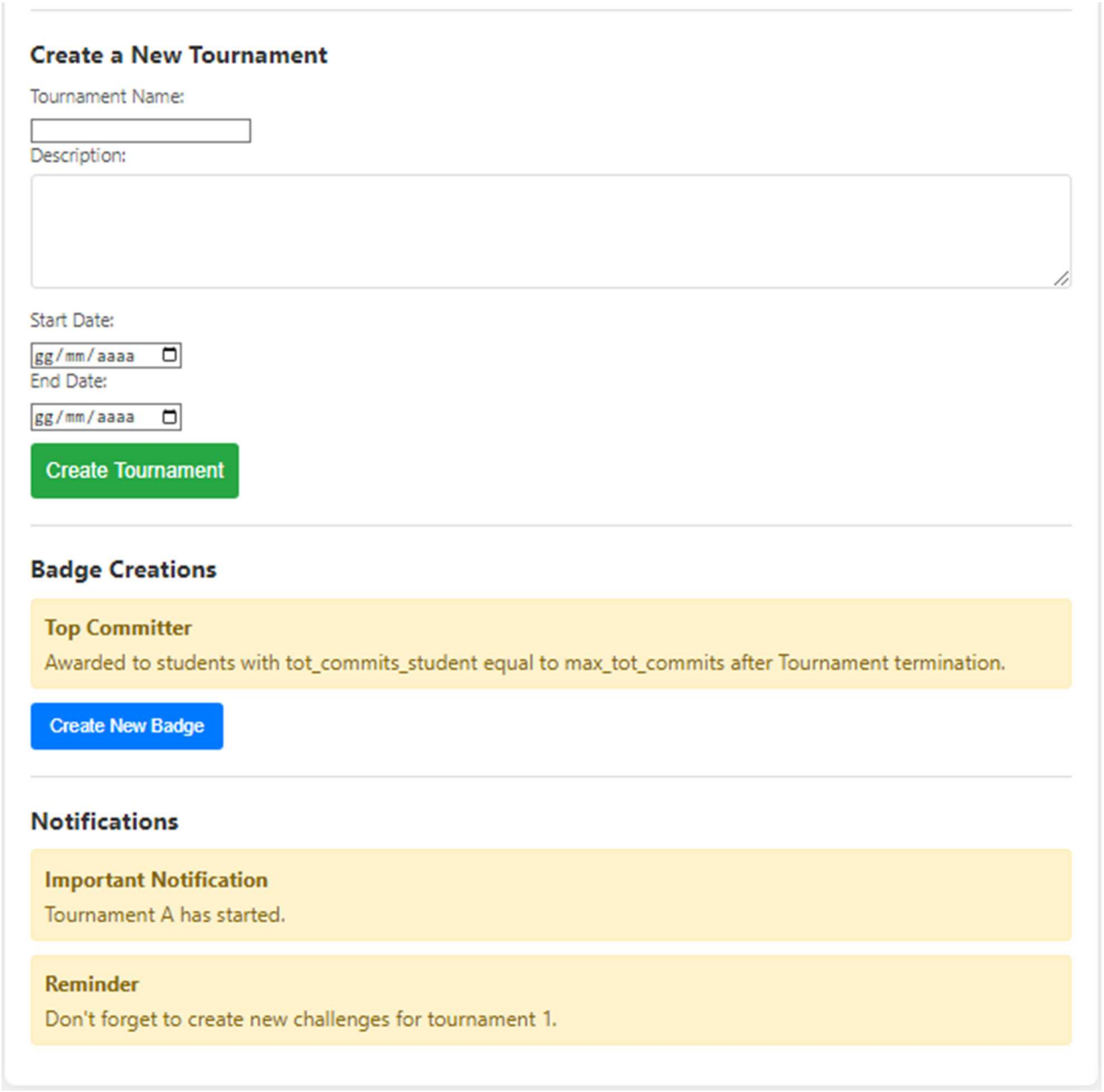
The "CodeKataBattle" student dashboard offers a view of the activities and opportunities present in the learning environment. The 'Subscribed tournaments' section allows to see the status of the tournament in which a student is enrolled. The "Upcoming Tournaments" section features upcoming tournaments, allowing students to easily identify upcoming events and quickly participate using the "Join" button. In parallel, the "Past Tournament Results" section provides access to the results of previous tournaments, giving students the opportunity to review their past performances. The student search function, in the "Search Student Profile" section, facilitates connection between participants, allowing a student to access specific information by entering a colleague's username. Finally, the

"Notifications" section provides a central hub for important communications. Here, students receive notifications regarding tournament status, final ranking updates, and other alerts.

Educators' interface



The "CodeKataBattle" educator dashboard offers a complete picture of the activities and responsibilities related to managing tournaments and interacting with students. The "Created Tournaments" section presents a list of tournaments created by the educator, highlighting the name of each tournament and its current status. Through the "Manage Tournament" button, the educator can access the management of tournament activities. The next section, "Past Tournament Results," gives the educator access to past tournament that need the manual evaluation to be concluded; with "Evaluate Results" button the educators are allowed to make evaluation of participants' performance. The student search function, in the "Search Student Profile" section, provides a means to locate student profiles by entering their username, making it easier to manage interactions with students, allowing educators to see student's achieved badges.



The “Create a New Tournament” section features an interactive module that allows educators to start new coding competitions. The form requires educators to enter details such as tournament name, description, starting date, and ending date. Once complete, educators can submit the form to create a new coding challenge. The "Badge Creations" section highlights the badges created by the educator, each with a title and a short description. The "Create new badge" and "Manage badge" buttons allow the educator to expand the collection of badges and manage them. Criteria for the badge are provided, specifying that it is awarded to students with a certain level of effort after the conclusion of a specific tournament. Finally, the "Notifications" section acts as a communications hub, providing the educator with instant notifications about events, such as the start of new tournaments, the need to grade student work or other important information.

3.1.2 Hardware Interfaces

No hardware interface needed.

3.1.3 Software Interfaces

The CKB system interacts with GitHub services in order to retrieve fundamental data for the scope of the application. To this end, it is required the usage of different APIs. First of all, it is needed an API to let the application communicate with its database. Then, it is also important the usage of the API for the start of the automatic evaluation phase; through these API calls (related to the set upped automated workflow with GitHub Actions), inform the CKB platform as soon as students push a new commit into the main branch of his repository. Furthermore, the registration and login processes can be managed by an external API, in order to avoid the implementation of encrypting of data.

3.1.4 Communication Interfaces

The web application is going to communicate with the server using the HTTPS protocol. In fact, in order to obtain security and reliability, with HTTPS it is used the Transport Layer Security (TLS) for the encryption.

3.2 FUNCTIONAL REQUIREMENTS

3.2.1 List of requirements

Requirement	Description
R1	The system shall allow an unregistered user to create an account.
R2	The system shall check if the personal data inserted by an unregistered user in the registration page are conflicting with respect to other existing accounts.
R3	The system shall send an email containing a link to the unregistered user once he has inserted his data.
R4	The system shall show a confirmation page to the user once he clicks on the link contained in the email.
R5	The system shall redirect the user, who is now registered, to the homepage.
R6	The system shall allow a registered user to login into the platform.
R7	The system shall check if username and password inserted by a user for login are valid.
R8	The system shall allow students to view the list of available tournaments.
R9	The system shall allow students to join a tournament.
R10	The system shall allow students to select a tournament to view its status.
R11	The system shall allow students to visualize information about upcoming tournaments.
R12	The system shall allow students to view past tournament results.

R13	The system shall allow students to participate in a coding battle if they are enrolled in the tournament of the battle.
R14	The system shall allow educators to create a new coding battle if they have tournament management permission.
R15	The system shall allow educators to insert the code kata for a coding battle he's creating.
R16	The system shall allow educators to specify battle parameters such us minimum and maximum number of team member for the battle.
R17	The system shall allow educators to specify the submission deadline of a coding battle he's creating.
R18	The system shall allow educators to specify the duration of a coding battle he's creating.
R19	The system shall allow educators to specify the evaluation criteria of a coding battle he's creating.
R20	The system shall allow users to monitor students' scores in real-time during a tournament.
R21	The system shall allow students to monitor their scores in real-time during a coding battle when they are enrolled in the battle.
R22	The system shall allow students to monitor teams' scores in real-time during a coding battle, when they are enrolled in the battle.
R23	The system shall allow educators to monitor students' scores in real-time during a coding battle, if they've created the battle.
R24	The system shall allow educators to monitor teams' scores in real-time during a coding battle, if they've created the battle.
R25	The system shall automatically update scores when students submit their solutions.
R26	The system shall send timely notifications to students when a new tournament is created by an educator.
R27	The system shall send timely notifications to users when tournament results are available.
R28	The system shall send timely notifications to users when battle results are available.
R29	The system shall send timely notifications to students when there is a new available battle for a subscribed tournament.
R30	The system shall create a GitHub repository for each team when a battle starts.
R31	The system shall send a notification to all the students with the link of the created GitHub repository.
R32	The system shall allow students to form teams through invitations to collaborate on participation in battles.
R33	The system shall allow users to see the profile of every student subscribed to the platform.
R34	The system shall allow educators to create a new tournament.
R35	The system shall allow educators to specify the duration of a tournament he's creating.
R36	The system shall allow educators to specify the name of a tournament he's creating.
R37	The system shall allow educators to make a description of a tournament he's creating.

R38	The system shall allow educators to establish the starting and ending date of a tournament he's creating.
R39	The system shall provide automatic evaluation of battle scores.
R40	The system shall allow to educators to manually evaluate students work relative to an ended battle that requires the consolidation phase.
R41	The system shall allow educators to view final results of a battle, including student and team battle leaderboard.
R42	The system shall allow educators to view final results of a tournament.
R43	The system shall allow students to view final results of a battle, including student and team battle leaderboard.
R44	The system shall allow students to view final results of a tournament.
R45	The system shall allow educators to define gamification rules for badge assignment during the tournament creation phase.
R46	The system shall allow educators to create badges.
R47	The system shall allow educators to delegate tournament management to other educators.
R48	The system shall allow students to view other students' profiles.
R49	The system shall allow educators to view profiles of all the students subscribed to the platform.
R50	The system shall display badges earned by a student in the student profile.

TABLE 9: TABLE OF REQUIREMENTS

3.2.2 MAPPING ON GOALS

Goals	Domain assumption	Requirements
G1.1	D1 D2 D3 D4	R1 R2 R3 R4 R5 R6 R8 R10 R11 R12
G1.2	D2 D9	R32
G2.1	D5 D6 D7	R14 R20 R34
G2.1.1	D6 D8	R8 R14 R47
G2.2	D5 D6 D8	R6 R7 R9 R13 R15 R16 R17 R18 R19 R30 R31 R34 R35 R36 R37 R38
G2.3	D7 D9	R7 R21 R22 R25 R39
G2.4	D7	R10 R12 R20 R23 R24 R27 R28 R41 R42 R43 R44
G2.5	D7	R8 R9 R10 R22 R33 R48 R49 R50
G2.6	D7 D9	R9 R22 R30 R31 R32
G3	D7	R26 R27 R28 R29
G4.1	D6 D8	R4 R25 R40
G4.2	D5 D6 D8	R45
G4.3	D6 D8	R45 R46
G4.4	D6 D8	R46

TABLE 10: TABLE OF MAPPING OF REQUIREMENTS AND DOMAIN ASSUMPTIONS ON GOALS

3.2.3 USE CASE DIAGRAMS

Use case diagrams are used in the document in order to give a graphical description of users' interaction with the system. In particular, it is provided a use case diagram for each possible user, one for the unregistered user, one for the student and one for the educator.

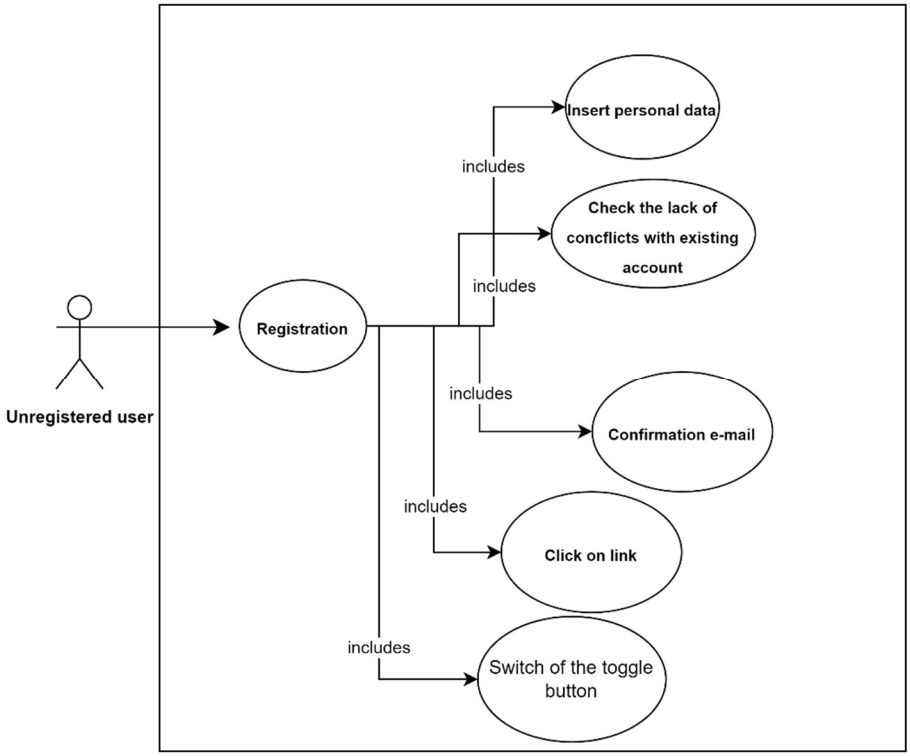


FIGURE 4: USE CASE DIAGRAM REFERRED TO THE UNREGISTERED USER

An unregistered user can only register himself into the system. The registration includes several phases, i.e. insertion of personal data by the user, checking the lack of conflicts with existing accounts by the system, sending of a confirmation email by the system and the clicking on the confirmation link by the usert. All these phases are mandatory.

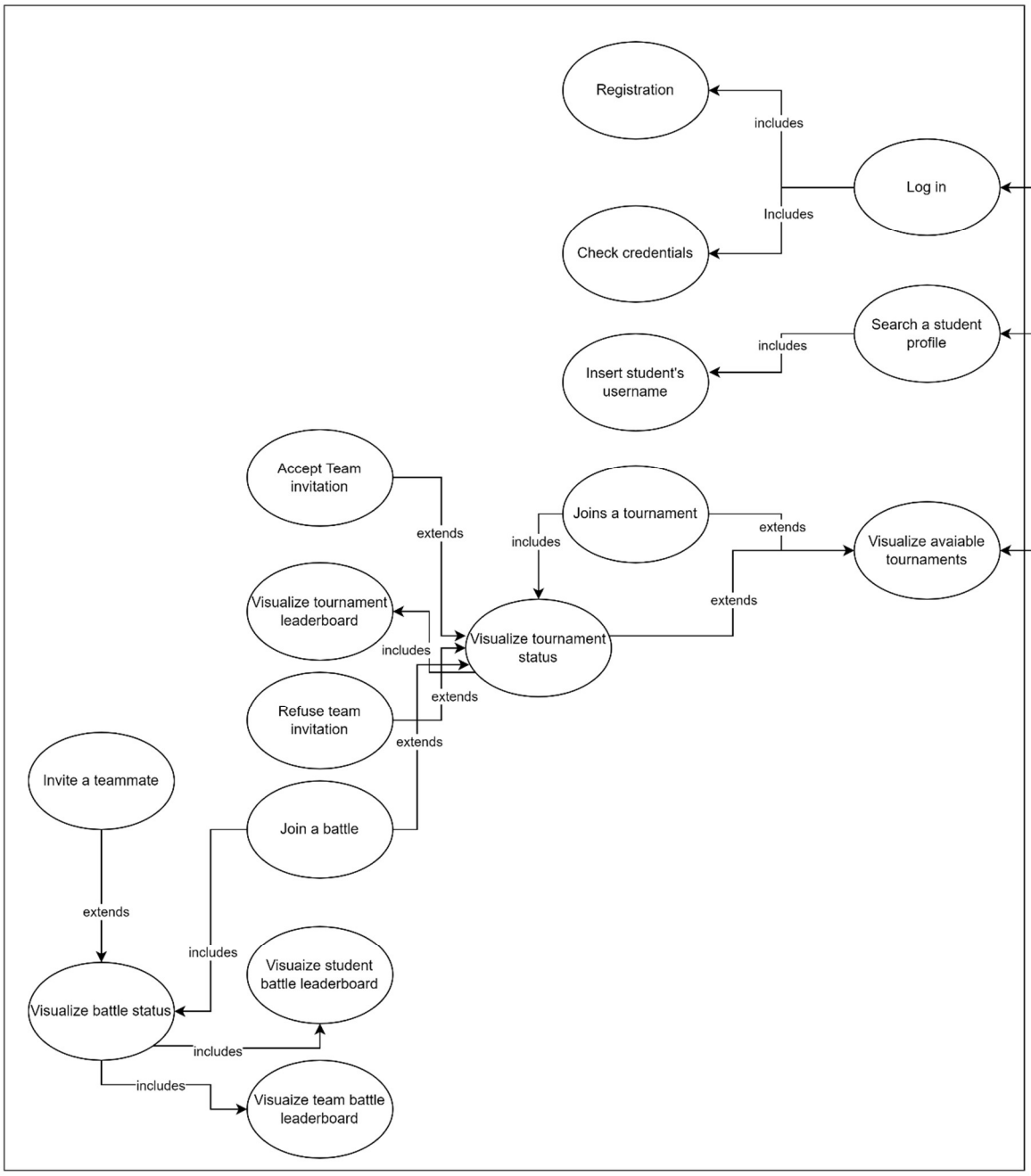


FIGURE 5: USE CASE DIAGRAM REFERRED TO THE STUDENT

A student can perform three main actions: logging in, searching for a student profile, visualizing the avaiable tournaments,

- For what concerns the process of logging in, it includes the check of the credentials' correctness by the system, and the registration to the platform, that are mandatory.
- The searching of a student profile includes the typing of the student's username in order to correctly display his profiles and badges,
- For what concerns the visualization of available tournaments, the student can visualize the status of a tournament, or he can join a tournament, this will let him into the tournament status state; from here the student will see the tournament leaderboard and he'll be able to accept/refuse the invitation for a team or join into a battle. Joining into a battle will let the student into the "visualize battle status" state, that includes the visualization of students and

team's leaderboards for that battles and gives the student the possibility to invite another student in his team.

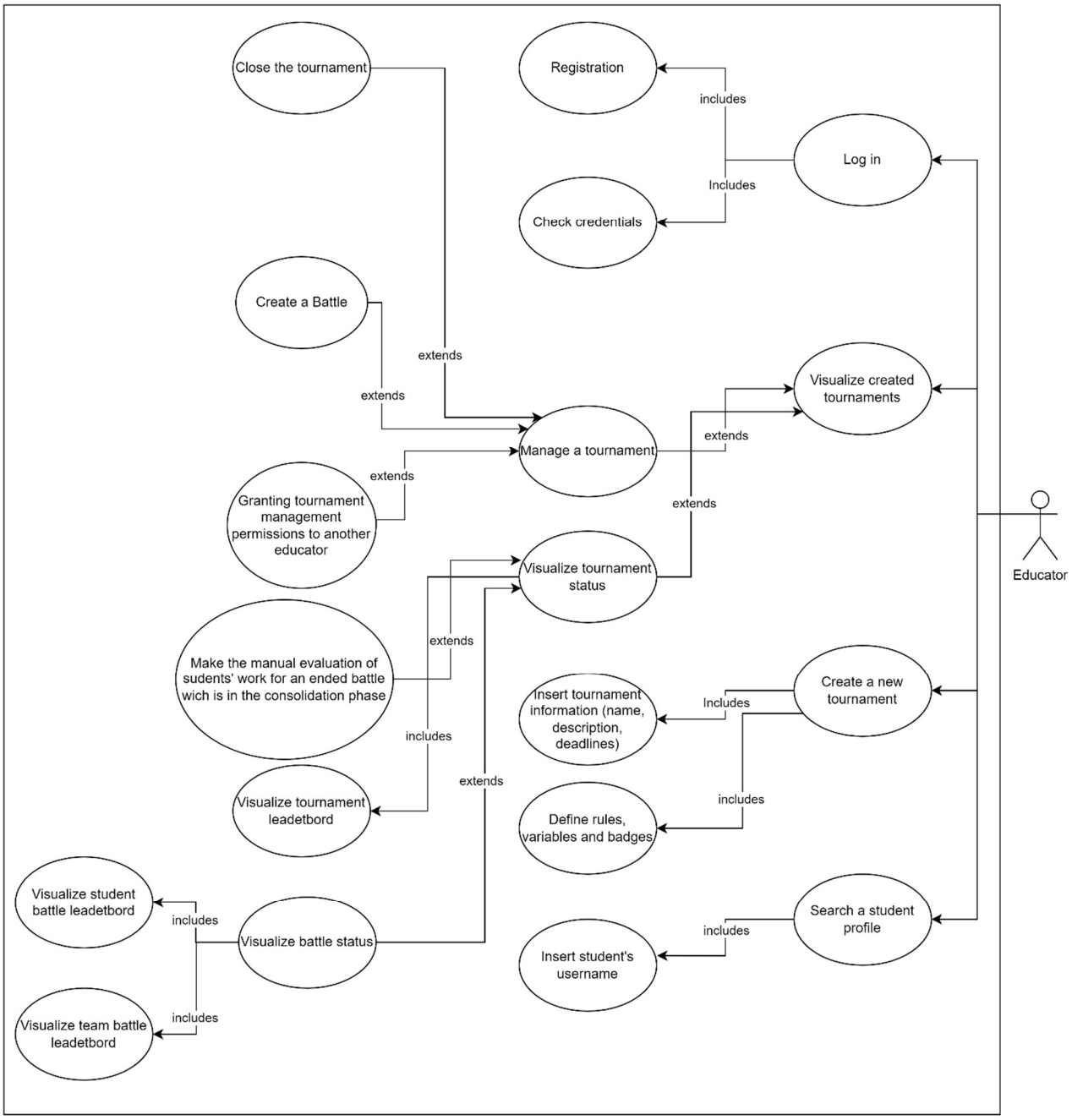


FIGURE 6: USE CASE DIAGRAM REFERRED TO THE EDUCATOR

An educator can perform several actions:

- For what concerns the process of logging in, it includes the check of the credentials' correctness by the system, and the registration to the platform, that are mandatory.
- The searching of a student profile includes the typing of the student's username in order to correctly display his profiles and badges,

- For what concerns the creation of a new tournament, it includes to insert tournament information, such as name, description etc., and to define rules, variables, and badges relative to the tournament, for the gamification aspect.
- For what concerns the "visualize created tournaments" state, it opens to the educator the possibility to manage a tournament (if he has the permission to do it), or to visualize the status of a tournament. The former will give the educator the possibility to create a battle for the tournament, or it gives him the possibility to grant management permissions for the tournament to other educators; the latter will let the educator to visualize the tournament leaderboard, it also gives him the possibility to make the manual evaluation for an ended battle (if it is in the consolidation phase), or to visualize the status of a battle. This last state will let the educator to see real-time students and teams' leaderboards for that battle.

3.2.4 USE CASES

Name	**Student Registration on CKB**
Actors	Unregistered User, Student
Entry condition	The user is not currently registered on the CKB platform. The user is on registration page of CKB platform.
Event flow	- The unregistered user chooses the option to sign up. - The system asks him to insert personal data (full name, email, username) and to create a password. - The unregistered user inserts all these information. - The unregistered user doesn't toggle the "I'm an educator" toggle button - The system sends a confirmation link to the unregistered user's email address to confirm it. - The unregistered user receives the verification email. - The unregistered user clicks on the verification link in the email. - The system verifies the unregistered user's account and marks it as verified. - The unregistered user (who is now a student) gains access to the CKB platform.
Exit conditions	- The user has successfully registered and now he's able to use the platform as a student. - The student visualizes student's homepage he was interested into.
Exceptions	- The user is already signed up. - The users' email is already used. These exceptions are handled by warning the user with an error message and redirecting him to the log in page. - The unregistered user does not fill one or more fields. This exception is handled by warning the user with an error message which asks to fill them. - The confirmation email does not arrive to the user. This exception is handled by giving the possibility to the unregistered user to click on a button in order to receive the email again.

TABLE 11: TABLE OF THE 1ST USE CASE

Name	Educator Registration on CKB
Actors	Unregistered User, Educator
Entry condition	The user is not currently registered on the CKB platform. The user is on registration page of CKB platform.
Event flow	- The unregistered user chooses the option to sign up. - The system asks him to insert personal data (full name, email, username) and to create a password. - The unregistered user inserts all these information. - The unregistered user toggles the “I’m an educator” toggle button - The system sends a confirmation link to the unregistered user’s email address to confirm it. - The unregistered user receives the verification email. - The unregistered user clicks on the verification link in the email. - The system verifies the Unregistered User's account and marks it as verified. - The unregistered user (who is now an educator) gains access to the CKB platform.
Exit conditions	- The user has successfully registered and now he’s able to use the platform as an educator. - The educator visualizes educator’s homepage he was interested into.
Exceptions	- The user is already signed up. - The user’s email is already used. These exceptions are handled by warning the user with an error message and redirecting him to the log in page. - The unregistered user does not fill one or more fields. This exception is handled by warning the user with an error message which asks to fill them. - The confirmation email does not arrive to the user. This exception is handled by giving the possibility to the unregistered user to click on a button in order to receive the email again.

TABLE 12: TABLE OF THE 2ND USE CASE

Name	Joining a Tournament
Actors	Student
Entry condition	The student is logged into the platform. The student is on his homepage.
Event flow	- The student navigates to the "Upcoming Tournaments" section. - The system displays a list of upcoming coding tournaments. - The student selects a specific tournament to view its details. - The system shows complete details of the tournament, including the description, requirements, and deadlines. - The system provides an option for the student to join the tournament. - The student clicks on the "Join" button. - The system promptly sends a notification confirming his successful registration for the coding tournament.
Exit conditions	- The student is now a tournament’s participant.

Exceptions	No Exception

TABLE 13: TABLE OF THE 3RD USE CASE

Name	**Joining a Battle**
Actors	Student
Entry condition	The student is logged into the platform. The student is already enrolled in the tournament containing the selected battle. The student is on the tournament status page
Event flow	- The student visualizes battles within the tournament. - The student selects a specific battle to view its details. - The system displays information about the selected battle. - The student completes the registration process for the battle clicking on the "Join" button. - The system registers the student for the battle and sends a confirmation notification.
Exit conditions	- The student has successfully joined the battle and is now registered for participation.
Exceptions	- The student is already registered for the selected battle - The registration process encounters an error.

TABLE 14: TABLE OF THE 4TH USE CASE

Name	**Creating a Tournament**
Actors	Educator, CKB platform
Entry condition	Educator is logged into the CKB platform.
Event flow	- The educator navigates to the "Create Tournament" section of their dashboard. - The system prompts the educator to enter details for the tournament creation. - The educator moves to the "Create Badge" section to create badges related to that tournament. - The educator clicks the "Create Badge" button. - The system guides the educator in defining variables, rules, and details specific to the badges. - The system shows the educator the created badge on the badge's section of the page. - The educator enters the required parameters for the tournament, such as title, description, requirements, and deadlines. - The educator clicks the "Create Tournament" button. - The system performs a check on the validity of the parameters entered by the educator to ensure they meet the necessary criteria. - The system sends a notification to the educator, confirming the successful creation of the tournament. - The system notifies all registered students on the platform about the availability of a new tournament.
Exit conditions	- The tournament is successfully created on the CKB platform.
Exceptions	- The educator attempts to create a tournament with incomplete or invalid information.

TABLE 15: TABLE OF THE 5TH USE CASE

Name	**Initiating a Battle**

Actors	Educator
Entry condition	An existing tournament is available on the CKB platform. The educator is logged into the CKB platform. The educator is on the tournament management page.
Event flow	- The educator selects the option to initiate a coding battle within the existing tournament. - The system prompts Educator to configure the battle by setting parameters such as the minimum and maximum number of participants, evaluation criteria, registration cutoff, final submission deadline, and selecting a code kata. - The educator fills in the required information and reviews the battle configuration. - The system sends a notification to the educator regarding the validity of the correct parameters. -The educator submits the battle initiation request clicking the "Create Battle" button. - The system processes the request and manages the student enrollment process. - The system promptly sends a notification to the educator confirming that the battle was successfully created. - The system notifies all students enrolled in that tournament about the availability of a new battle within that specific tournament.
Exit conditions	- The coding battle within the existing tournament is successfully initiated.
Exceptions	-The educator attempts to initiate a battle with invalid or incomplete information.

TABLE 16: TABLE OF THE 6TH USE CASE

Name	**Real-time Tournament Leaderboard Access**
Actors	User
Entry condition	The user is registered on the platform. There are ongoing tournaments on the CKB platform. The user is logged into the CKB platform. The user is on the platform's main dashboard.
Event flow	-The user selects a specific tournament clicking on the "View Tournament Status" button to view detailed rankings. -The system displays real-time rankings for the selected tournament. -The user can view rankings of all the students enrolled in the tournament.
Exit conditions	- The student has successfully accessed the real-time leaderboard and viewed the ongoing competition rankings.
Exceptions	No exceptions

TABLE 17: TABLE OF THE 7TH USE CASE

Name	**Real-time Team Leaderboard Access**
Actors	User
Entry condition	The user is logged into the CKB platform. There is an ongoing tournament on the CKB platform.

	There is an ongoing battle for that tournament. The user is the educator who created the battle or a student who is enrolled in the battle. The user is on the platform's main dashboard.
Event flow	-The user selects a specific tournament clicking on the "View Tournament Status" button to view tournament battles. -The user selects the battle for which he wants to see the ranking -The system displays the team leaderboard for the selected battle.
Exit conditions	- The user has successfully accessed the real-time team leaderboard for the selected battle.
Exceptions	No exceptions

TABLE 18: TABLE OF THE 8TH USE CASE

Name	**Real-time Battle Student Leaderboard Access**
Actors	User
Entry condition	The user is logged into the CKB platform. There is an ongoing tournament on the CKB platform. There is an ongoing battle for that tournament. The user is the educator who created the battle or a student who is enrolled in the battle. The user is on the platform's main dashboard.
Event flow	-The user selects a specific tournament clicking on the "View Tournament Status" button to view tournament battles. -The user selects the battle for which he wants to see the ranking -The system displays the student leaderboard for the selected battle.
Exit conditions	- The user has successfully accessed the real-time student leaderboard for the selected battle.
Exceptions	No exceptions

TABLE 19: TABLE OF THE 9TH USE CASE

Name	**Forming a Coding Team**
Actors	Students
Entry condition	The student is registered on the CKB platform. The student is logged into the CKB platform. The invited student is enrolled in the tournament of the considered battle. The student is already enrolled in the battle. The student is on the team formation section in the battle status page.
Event flow	- The system provides an option to create a new team or join an existing one. - A student creates a new team inviting his colleagues by providing their usernames or email addresses. - The colleagues receive invitations from the system and choose to accept. - The system verifies that the team size aligns with the specified minimum and maximum number of participants for the upcoming battle. - The system confirms the team formation. -The system sends a confirmation message to the student

	- The system sends a notification to the invited students to confirm their entry into the new team
Exit conditions	- The students have successfully formed a coding team for the upcoming battle.
Exceptions	-The student attempts to create a team with a size outside the specified minimum and maximum number of participants.

TABLE 20: TABLE OF THE 10TH USE CASE

Name	**Manual Assessment by an Educator**
Actors	Educator
Entry condition	The educator is authenticated on the platform. The educator has created the battle that wants to evaluate. The educator has access to the consolidation phase after the submission deadline of a battle. The educator is on the tournament status page.
Event flow	- The educator navigates to the consolidation phase section of an ended battle. - The system provides access to submitted solutions, automated evaluation results, and relevant information. - The educator manually assesses each student's work, assigning numerical scores based on specific criteria. - The educator submits the manual assessment scores to the CKB platform. - The system updates the overall student scores, considering both automated and manual assessments. - The system updates the team scores based on the changes made thanks to educator's evaluation. - The system sends a notification to the educator to inform him of the successful of the manual assignment. - The system notifies all students enrolled in the battle that the final ranking is available.
Exit conditions	- The educator has successfully completed the manual assessment for the selected challenge, contributing to the overall evaluation of student performance.
Exceptions	No Exception

TABLE 21: TABLE OF THE 11TH USE CASE

Name	**Viewing a Student Profile**
Actors	User
Entry condition	The user is logged into the platform. The user is on the platform's main dashboard.
Event flow	- The user navigates through the "Search Student Profile" section of the homepage. - The user inserts the username of the student whose profile he wants to view. -The user clicks on the "Search" button. -The system displays the searched student's profile with all his information and achieved badges.
Exit conditions	- A user can view the profile of any student subscribed to the platform.
Exceptions	- The searched student does not exist in the platform.

	The platform simply doesn’t display any profile.

TABLE 22: TABLE OF THE 12TH USE CASE

Name	Closing a Tournament
Actors	Student, Educator
Entry condition	The educator is authenticated on the CKB platform. The educator has the necessary permissions to close the tournament. All battles within the tournament have concluded, and their results are finalized. The educator is on the platform's main dashboard.
Event flow	- The educator navigates to the tournament page section on the platform’s homepage. - The educator selects the “Manage tournament” button of the tournament he wants to close. - The educator initiates the tournament closure process clicking the “Close Tournament” button. - The system notifies all students involved in the tournament about its closure. - The students receive notifications containing information about the closed tournament and their final ranks.
Exit conditions	- The tournament is successfully closed, and all involved students have received notifications regarding the closure and final ranks.
Exceptions	Some battles within the tournament haven’t concluded yet, or their results aren’t finalized.

TABLE 23: TABLE OF THE 13TH USE CASE

Name	Granting Permissions
Actors	Educator
Entry condition	The educator is logged into the platform. The educator has the permissions to manage the tournament in which he wants to grant permissions to another educator. The educator is on the platform's main dashboard.
Event flow	- The educator views the details of the tournament by clicking on the “View Tournament Status” button. - The system shows the tournament status’ page. - The educator provides the username or email of the educator to whom he wants to grant permissions into the “Grant permissions” section of the page. - The Educator clicks the "Grant Permissions" button. - The system processes the request providing management permissions to the provided educator. - The system displays a confirmation message.
Exit conditions	- The educator has successfully assigned permissions to another educator
Exceptions	The educator provides an incorrect or non-existent username or identifier of the educator to whom they want to grant permissions.

TABLE 24: TABLE OF THE 14TH USE CASE

3.2.5 SEQUENCE DIAGRAMS

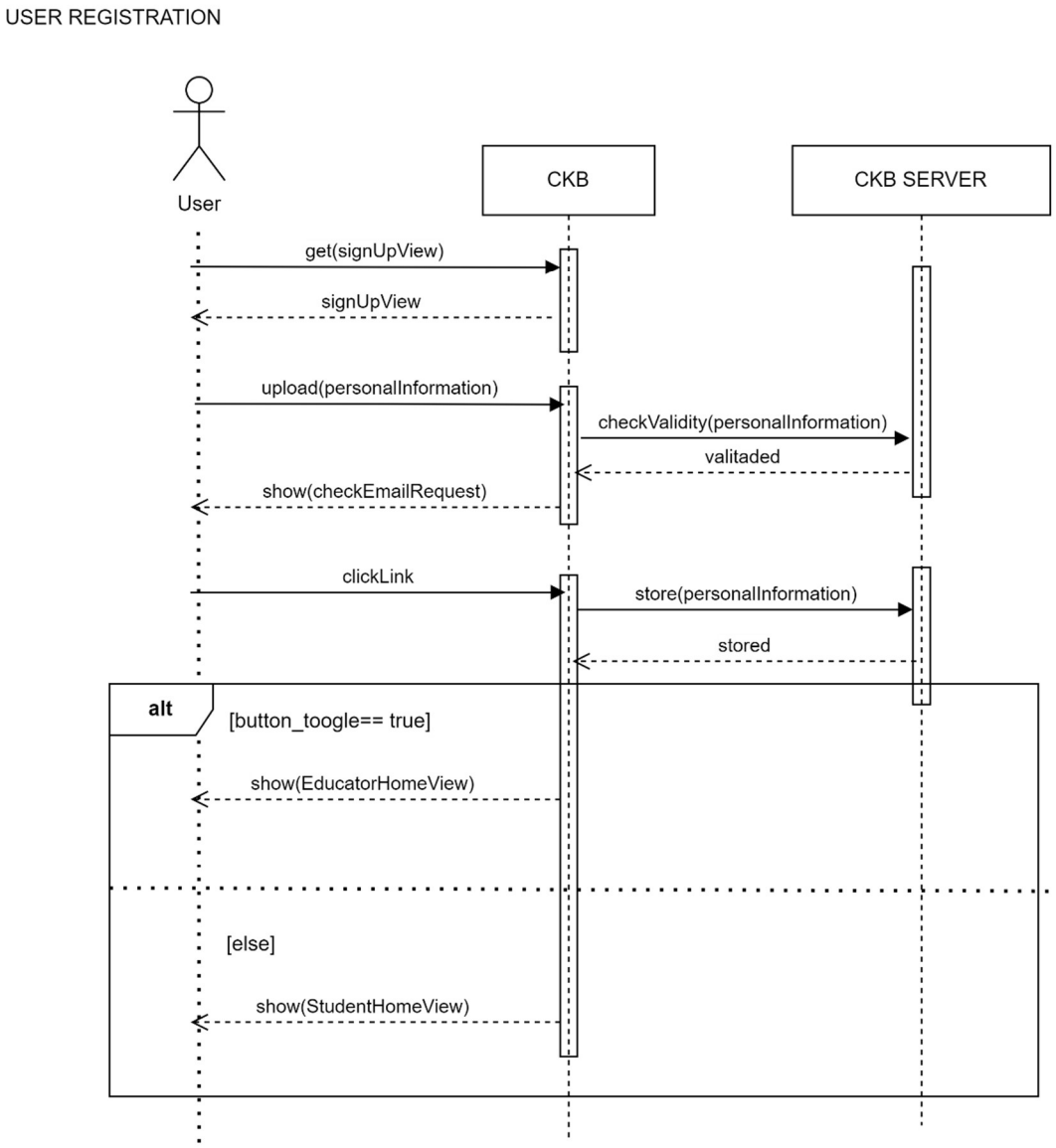


FIGURE 7: SEQUENCE DIAGRAM REFERRED TO THE 1ST USE CASE AND TO THE 2ND USE CASE (SIGN UP BY A STUDENT AND SIGN UP BY AN EDUCATOR)

JOIN A TOURNAMENT

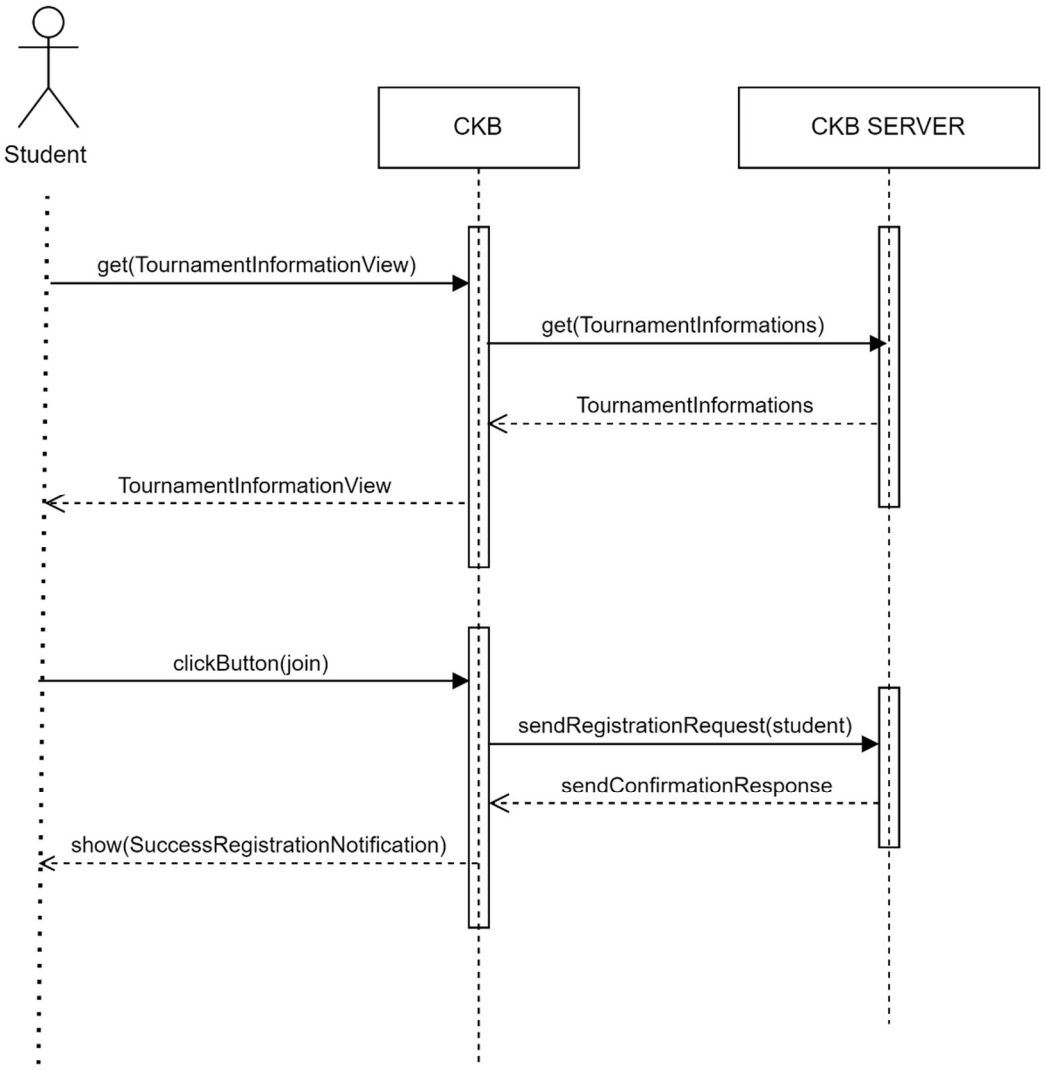


FIGURE 8: SEQUENCE DIAGRAM REFERRED TO THE 3RD USE CASE

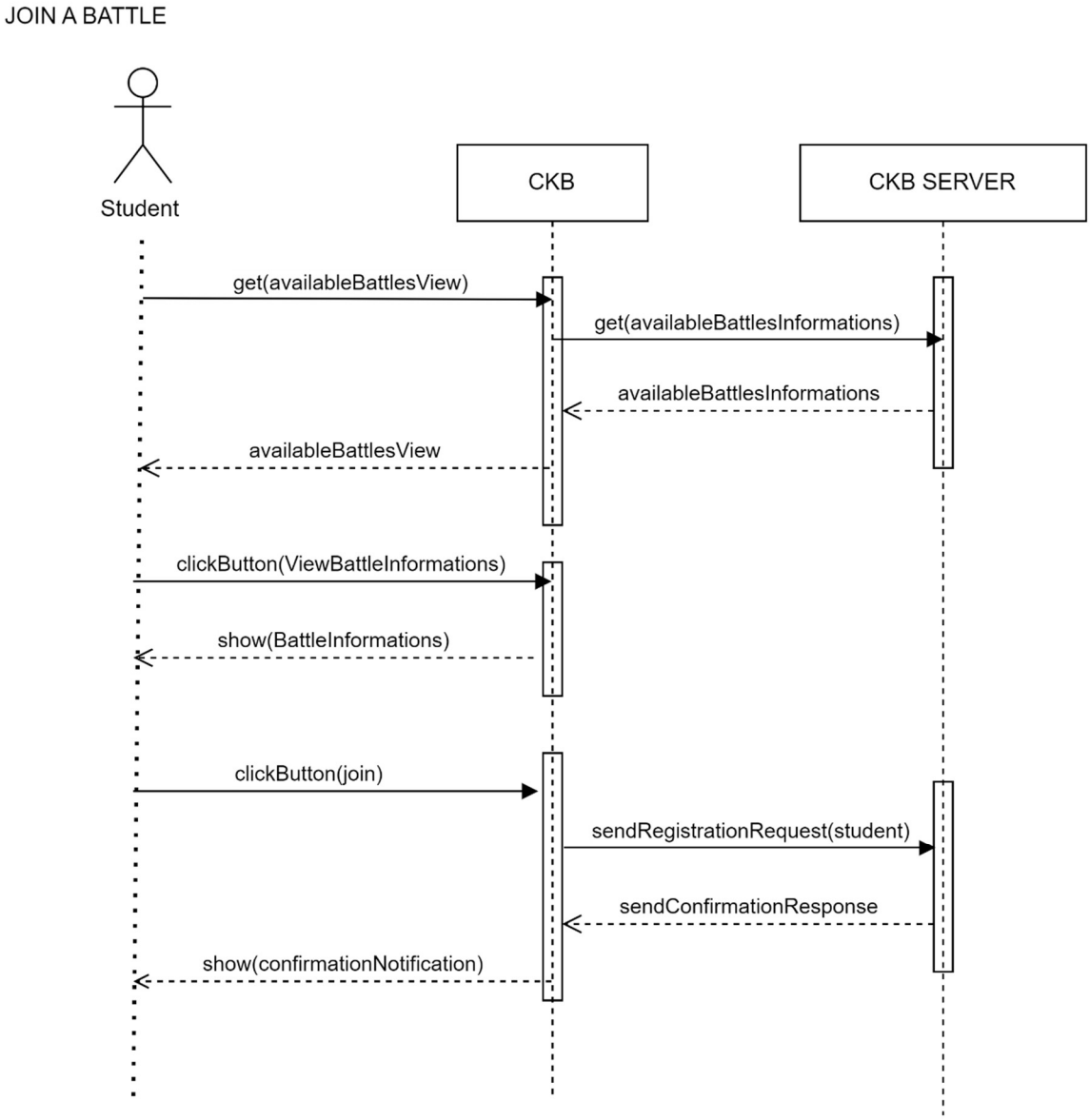


FIGURE 9: SEQUENCE DIAGRAM REFERRED TO THE 4TH USE CASE

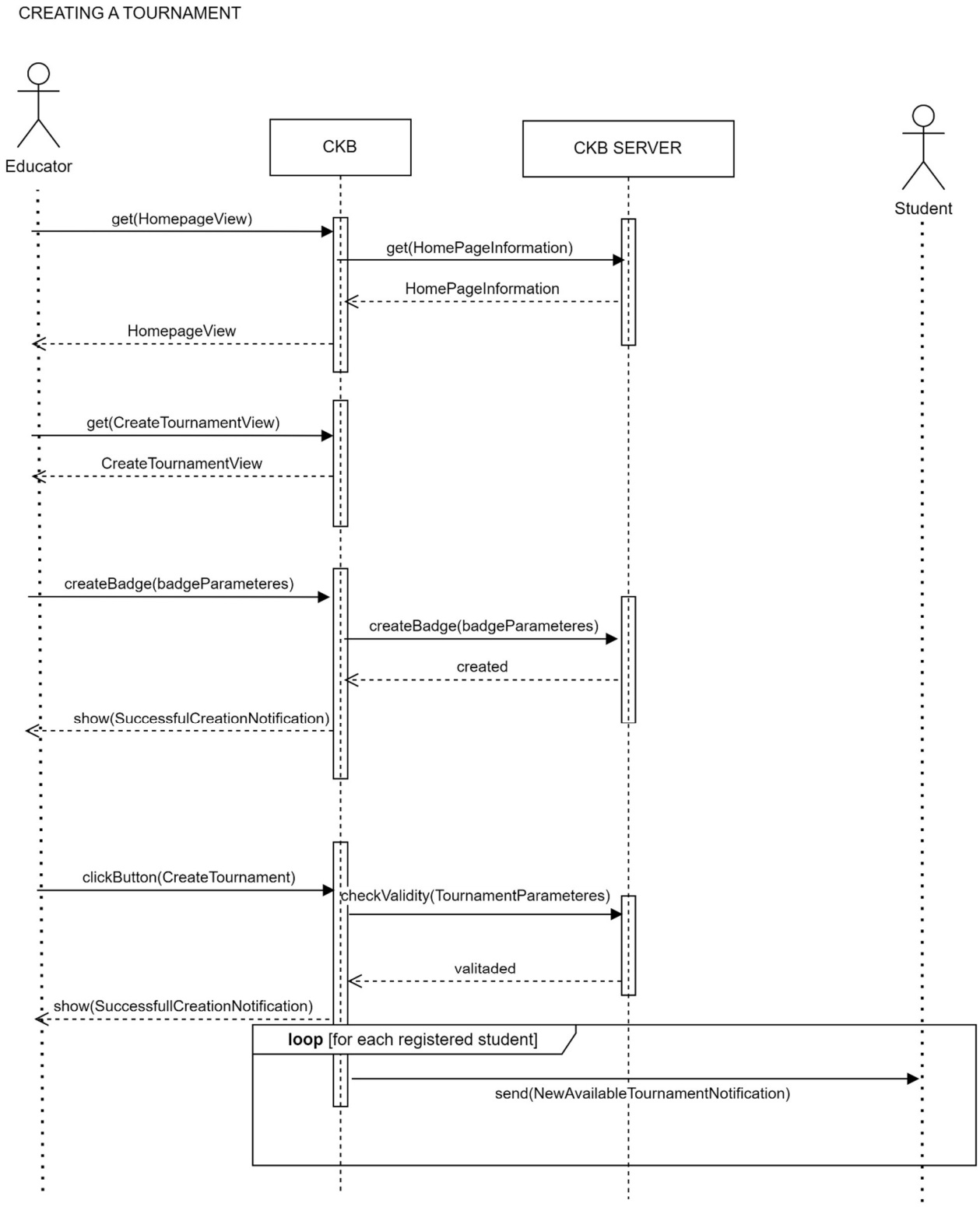


FIGURE 10: SEQUENCE DIAGRAM REFERRED TO THE 5TH USE CASE

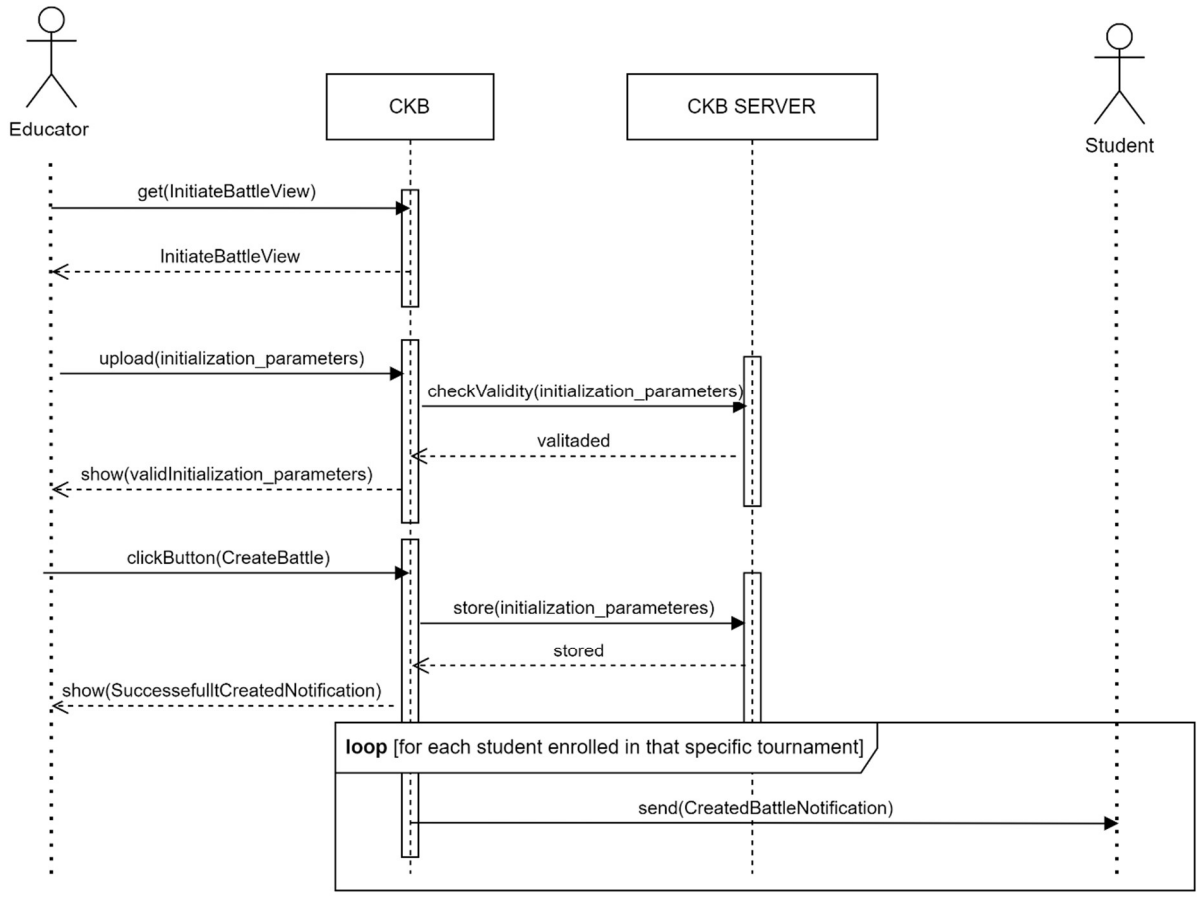


FIGURE 11: SEQUENCE DIAGRAM REFERRED TO THE 6TH USE CASE

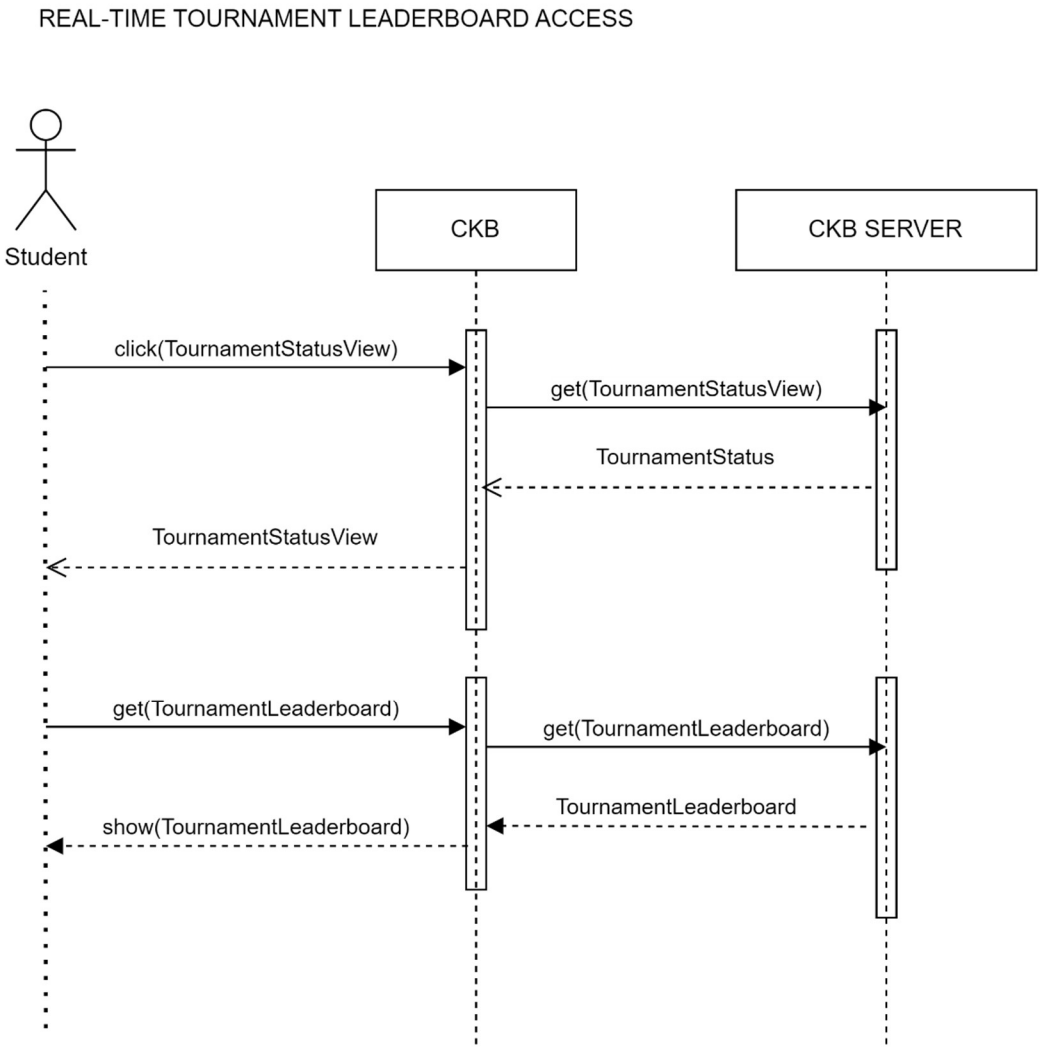


FIGURE 12: SEQUENCE DIAGRAM REFERRED TO THE 7TH USE CASE

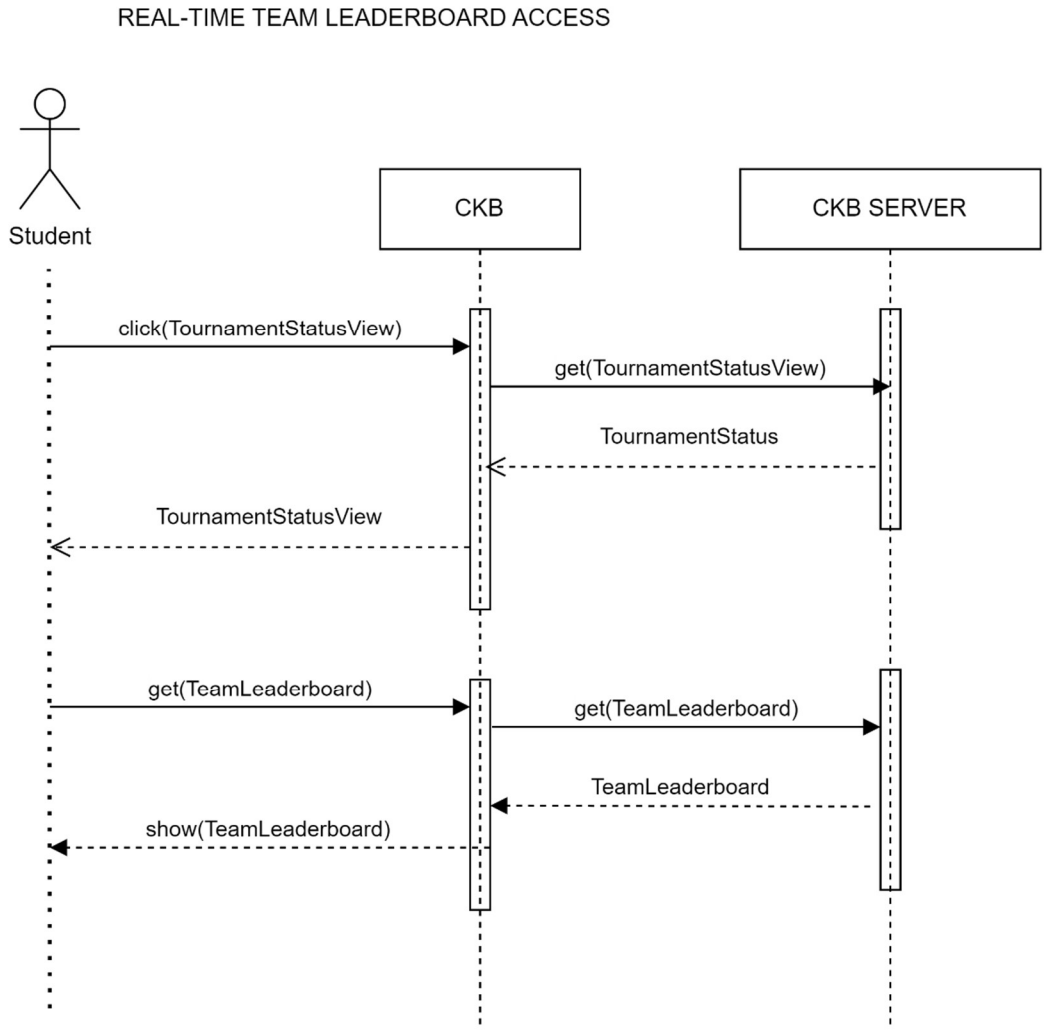


FIGURE 13: SEQUENCE DIAGRAM REFERRED TO THE 8TH USE CASE

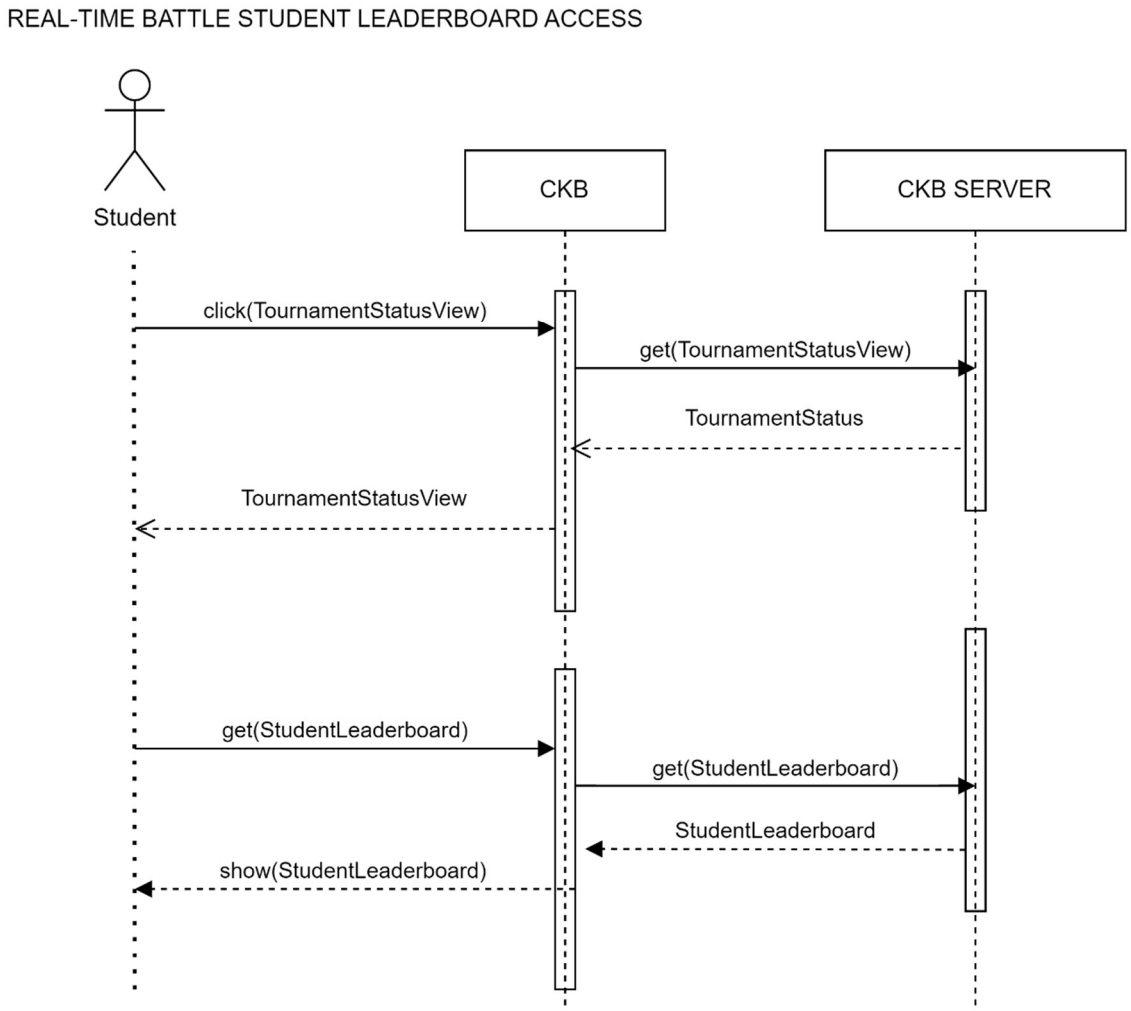


FIGURE 14: SEQUENCE DIAGRAM REFERRED TO THE 9TH USE CASE

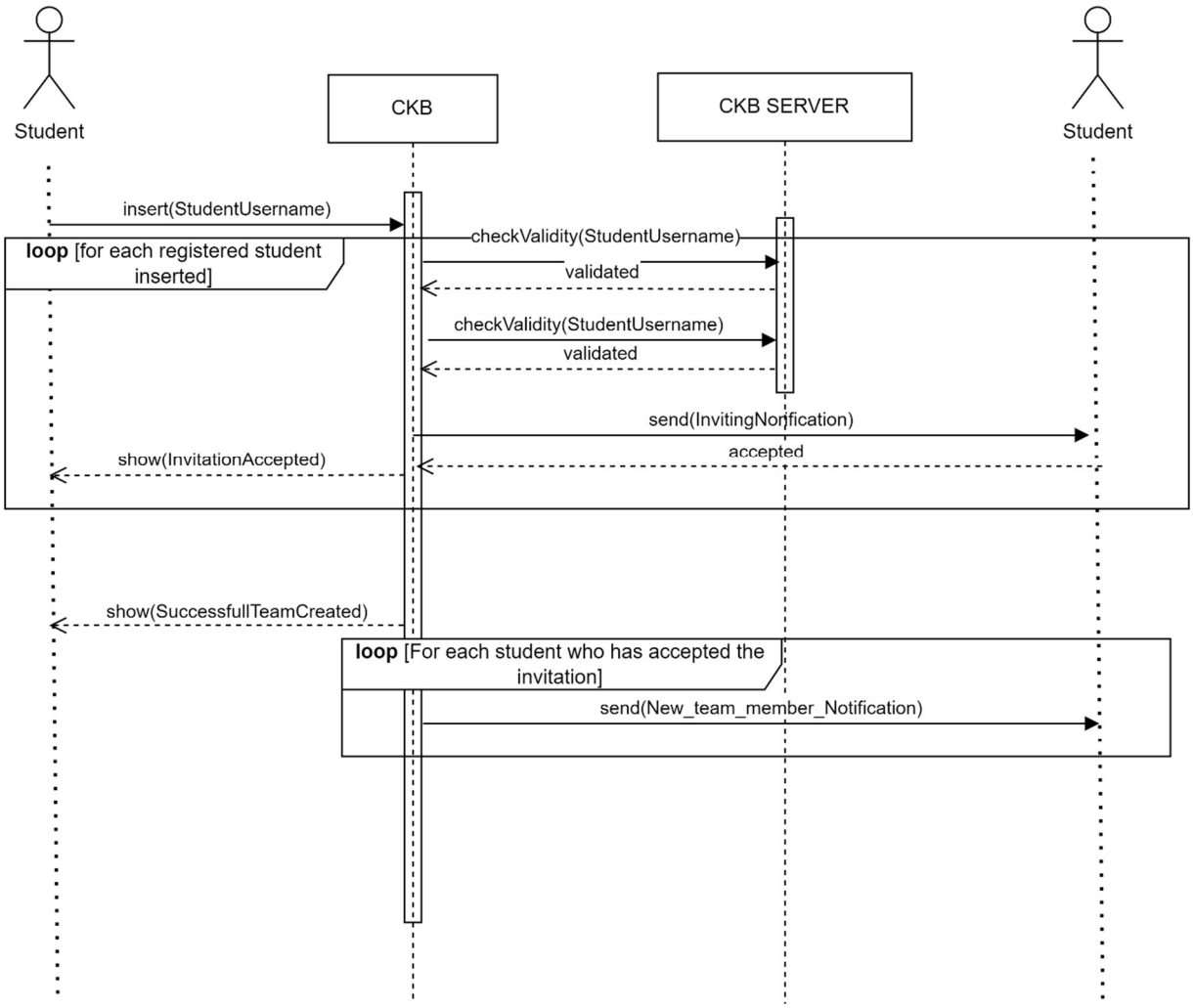


FIGURE 15: SEQUENCE DIAGRAM REFERRED TO THE 10TH USE CASE

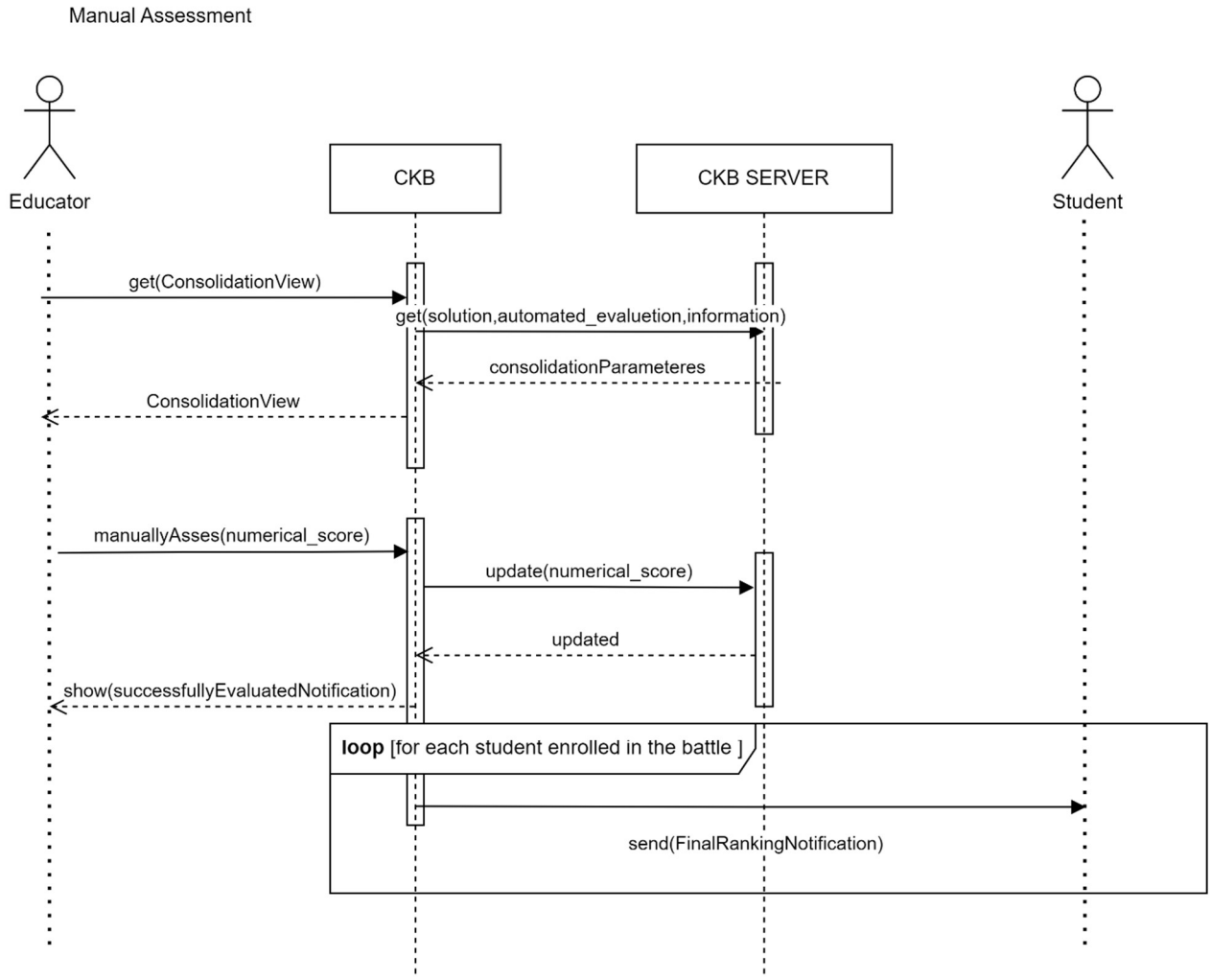


FIGURE 16: SEQUENCE DIAGRAM REFERRED TO THE 11TH USE CASE

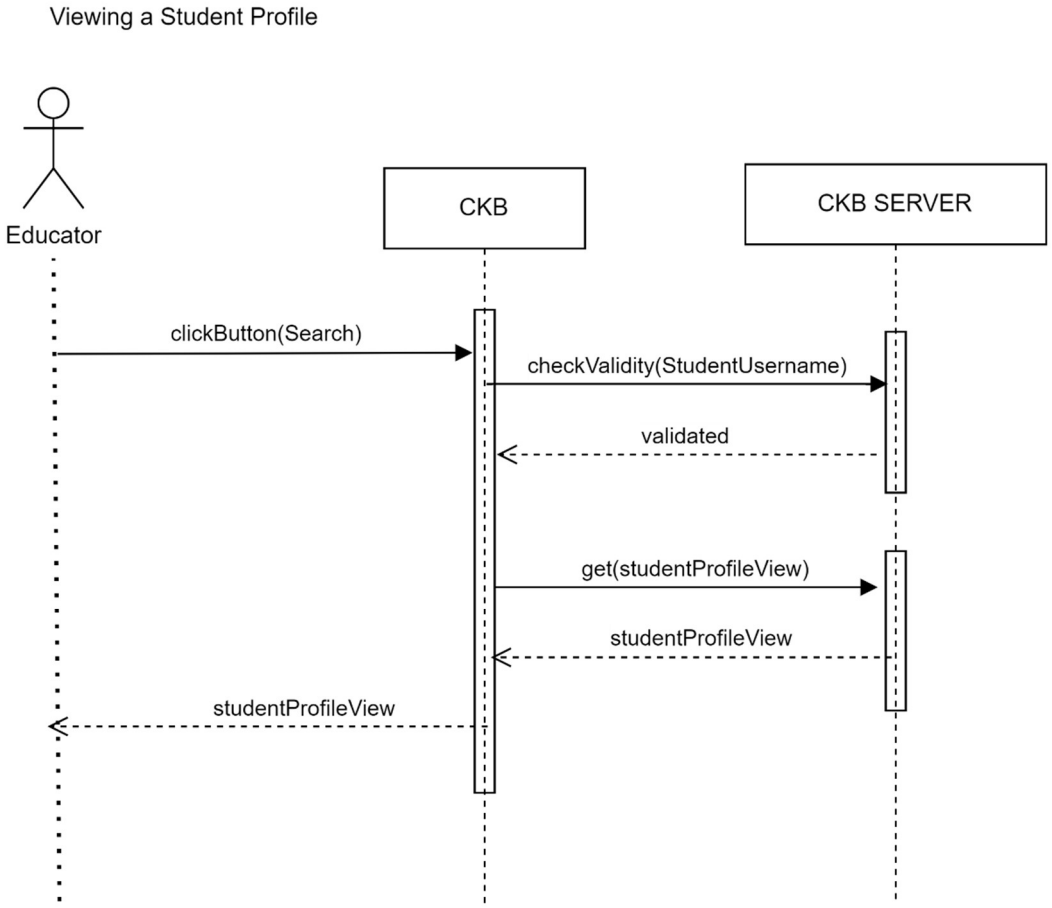


FIGURE 17: SEQUENCE DIAGRAM REFERRED TO THE 12TH USE CASE

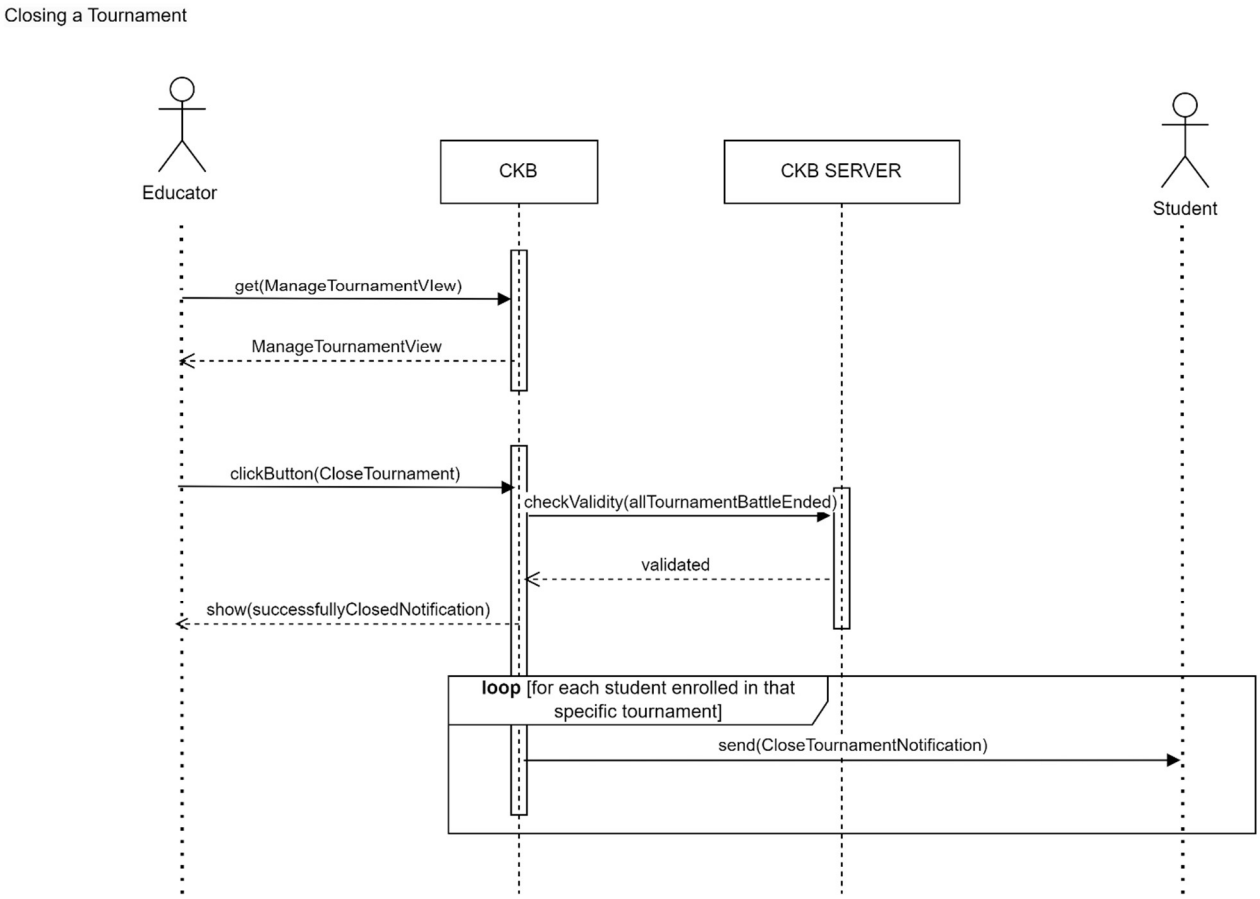


FIGURE 18: SEQUENCE DIAGRAM REFERRED TO THE 13TH USE CASE

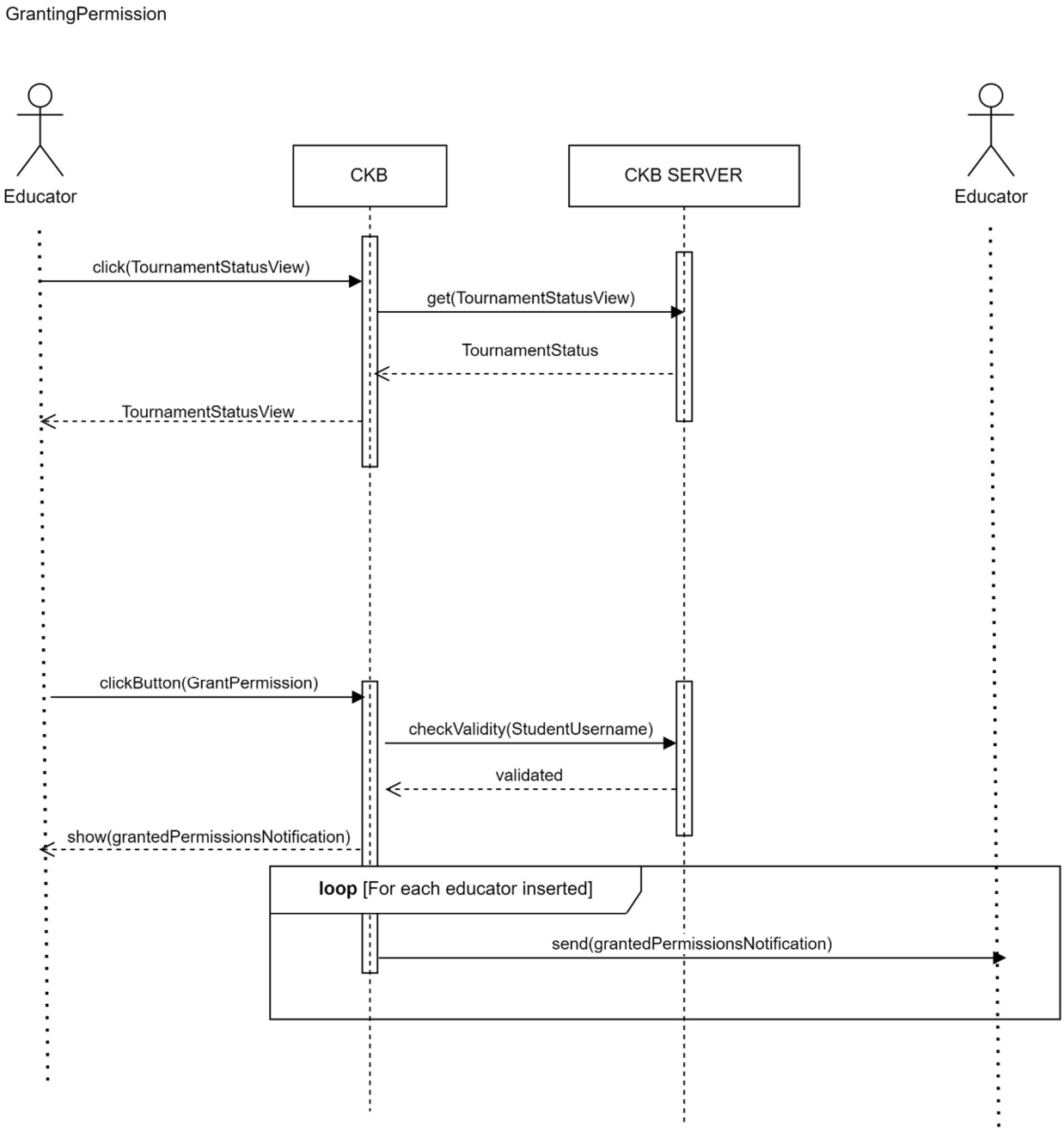


FIGURE 19: SEQUENCE DIAGRAM REFERRED TO THE 14TH USE CASE

3.3 PERFORMANCE REQUIREMENTS

The system needs to ensure that code is evaluated accurately and quickly and that competition results are updated instantly. The platform must guarantee prompt functional aspect evaluation, responsiveness to GitHub repository modifications, and effective consolidation phase management. By attending to these requirements, CKB can provide educators and students participating in cooperative coding challenges with a flexible and adaptable setting.

3.4 DESIGN CONSTRAINTS

3.4.1 STANDARDS COMPLIANCE

The CKB platform must adhere to privacy laws, be compliant with regulations like GDPR, and incorporate features like open-source software in accordance with the principles of a public digital good.

In compliance with GDPR regulations, users' consent is required before using their personal data for ranking analysis and data sharing. To ensure compliance with privacy regulations, the system must additionally obtain explicit user permission prior to accessing data from their GitHub accounts. This strategy emphasizes openness, compliance with the law, and user consent, bringing the platform into compliance with accepted guidelines for handling data responsibly, such as GDPR.

3.4.2 HARDWARE LIMITATIONS

The CKB system requires users to have access to a personal computer with a compatible web browser in order to be used to its full potential. In order for the features of the web application to function as intended, the computer must be online.

3.5 SOFTWARE SYSTEM ATTRIBUTES

3.5.1 RELIABILITY

In order to reduce the possibility of mistakes and malfunctions during coding competitions, the system must guarantee the precise and reliable execution of functionality. Score management and competition evaluations need to be trustworthy and dependable.

3.5.2 Availability

Throughout the competition time windows, the system must be consistently available and ensure dependable access for educators and students. It is necessary to put procedures in place to deal with service interruptions as soon as possible.

3.5.3 Security

The top priority is security. Students' and teachers' private information must be safeguarded by the system.

Data confidentiality during transmission and storage requires the implementation of secure authentication protocols and encryption techniques.

3.5.4 Maintainability

To make continuing maintenance easier, the source code for the system needs to be properly documented and organized. It is imperative to execute updates and corrections without jeopardizing the system's integrity.

3.5.5 Portability

The system needs to be built with ease of environment adaptation in mind. To ensure a positive user experience regardless of the operating system or device being used, the software must be able to run on a variety of platforms and devices.

4 FORMAL ANALYSIS USING ALLOY

In this section it is analyzed the Alloy model. The main aim of this analysis is to create a model of the system, defining all its entities with the respective properties and imposing the constraints that make the model consistent. In such way, it will be possible to create some instances of the model using the Alloy tool Analyzer and to visualize them through a graphical notation, which help to better understand if the imposed constraints are valid and consistent with the real-world domain associated with the system.

4.1 CODE

```
open util/integer

abstract sig User{
recivesNotification: set Notification
}

sig Student extends User{
participates: set Tournament,
recives: set Badge,
joins: set Team,
is_present: set StudentTournamentLeaderboard,
is_in: set StudentBattleLeaderboard
}

sig Educator extends User{
manages: set Tournament,
creates: set Badge,
makes: set ManualEvaluation,
managesBattle: set Battle
}

sig Tournament{
leaderboard: one StudentTournamentLeaderboard,
made_of: set Battle,
is_managed_by: some Educator,
enrolled_sudents: set Student,
has: set Badge
}


sig StudentTournamentLeaderboard{
referres_to: set Student,
relative_to: one Tournament
}
```

```
sig Battle{
studentLeaderboard: one StudentBattleLeaderboard,
teamLeaderboard: one TeamBattleLeaderboard,
managed_by: one Educator,
belongs_to: one Tournament,
teams: set Team,
relative_codeKata: one CodeKata
}

sig CodeKata{
battle: one Battle,
softwareProject: one SoftwareProject
}

sig SoftwareProject{
is_included: one CodeKata,
has: some TestCase
}

sig TestCase{
softwareProject: one SoftwareProject
}


sig StudentBattleLeaderboard{
referres_to: set Student,
relative_to: one Battle
}

sig TeamBattleLeaderboard{
referres_to: set Team,
relative_to: one Battle
}
```

```
sig Team{
battle : one Battle,
made_of: some Student,
score: one Score,
is_in: one TeamBattleLeaderboard
}

sig Score{
relative_to: one Team,
contribute1: lone ManualEvaluation,
contribute2: one AutomaticEvaluation
}


sig ManualEvaluation{
educator: one Educator,
score: one Score
}

sig AutomaticEvaluation{
score: one Score,
automaticEvaluationTool: one AutomaticEvaluationTool
}

sig Badge{
assigned_to: set Student,
tournament: one Tournament,
created_by: one Educator
}

sig AutomaticEvaluationTool{
makes: set AutomaticEvaluation
}

sig Notification{
is_recived_by: some User
}
```

```
//FACTS

//if an educator manages a tournament, then that educator should be included in the set of educators managing that particular tournament.
fact relEducatorTournament{
 all  e: Educator, t: Tournament | t in e.manages implies e in t.is_managed_by
}

// if a battle (b) is managed by an educator (b.managed_by = e), then that educator (e) should be included in the set of educators managing the tournament (e in t.is_managed_by) relative to the battle.
fact BattleIsCreatedOnlyFromEducatorWithTournamentManagmentPermissions{
	all b: Battle, e: Educator, t: Tournament | b.managed_by=e implies e in t.is_managed_by
}

// if a student (s) is enrolled in a tournament (s in t.enrolled_sudents), then that tournament (t) should be included in the set of tournaments in which the student participates (t in s.participates).
fact studentPartecipatesToMultipleTournaments{
 all s:Student, t:Tournament | s in t.enrolled_sudents implies t in s.participates
}

//if a student (s) is a member of a team (s in t.made_of), then that student should also be enrolled in the tournament to which the team's battle belongs (s in t.battle.belongs_to.enrolled_sudents).
fact studentsInATeamEntrolledInTheTournament{
	all s: Student, t: Team | s in t.made_of implies s in t.battle.belongs_to.enrolled_sudents
}

//if an automatic evaluation (ae) is made by an automatic evaluation tool (ae.automaticEvaluationTool = aet), then that automatic evaluation should be included in the set of evaluations
//made by that specific automatic evaluation tool (ae in aet.makes).
fact AutomaticEvaluationIsMadeByAET{
	all ae: AutomaticEvaluation, aet: AutomaticEvaluationTool | ae.automaticEvaluationTool=aet implies ae in aet.makes
}

//each automatic evaluation (ae) is associated with exactly one automatic evaluation tool, and no two distinct automatic evaluation tools share the same set of evaluations.
fact eachAutomaticEvaluationIsMadeByOneAutomaticEvaluationTool{
	all ae: AutomaticEvaluation, disj aet1, aet2: AutomaticEvaluationTool | ae in aet1.makes implies ae not in aet2.makes
}

//if a manual evaluation (me) is created by an educator (me in e.makes), then that educator (e) must be the one managing the battle to which the manual evaluation is associated (e = (me.score.relative_to.battle.managed_by)).
fact educatorEvaluateOnlyWithPermissions{
	all me: ManualEvaluation, e: Educator | me in e.makes implies (e=(me.score.relative_to.battle.managed_by))
}

//there is a one-to-one correspondence between scores (s) and manual evaluations (me) in the context of the "contribute1" relationship.
fact singleManualEvaluationForScore{
	all s: Score, me: ManualEvaluation | s.contribute1=me iff me.score=s
}

// there is a one-to-one correspondence between scores (s) and automatic evaluations (ae) in the context of the "contribute2" relationship.
fact singleAutomaticEvaluationForScore{
	all s: Score, ae: AutomaticEvaluation | s.contribute2=ae iff ae.score=s
}

//if a manual evaluation (me) is associated with an educator (me.educator = e), then that manual evaluation should be included in the set of manual evaluations created by that specific educator (me in e.makes).
fact RelManualEvaluationEducator{
	all me: ManualEvaluation, e: Educator | me.educator=e implies me in e. makes
}

//each manual evaluation (me) is associated with exactly one educator, and no two distinct educators share the same set of manual evaluations.
fact singleEducatorForManualEvaluation{
	all me: ManualEvaluation, disj e1,e2: Educator | me in e1.makes implies me not in e2.makes
}

//there is a one-to-one correspondence between student battle leaderboards (sbl) and battles (b) in the context of the relative_to relationship.
fact singleStudentBattleLeaderboardForBattle{
	all sbl: StudentBattleLeaderboard, b: Battle | sbl.relative_to=b iff b.studentLeaderboard=sbl
}

//there is a one-to-one correspondence between team battle leaderboards (tbl) and battles (b) in the context of the relative_to relationship.
fact singleTeamBattleLeaderboardForBattle{
	all tbl: TeamBattleLeaderboard, b: Battle | tbl.relative_to=b iff b.teamLeaderboard=tbl
}

// there is a one-to-one correspondence between student tournament leaderboards (stl) and tournaments (t) in the context of the "relative_to" relationship.
fact singleTournamentLeaderboardForTournament{
	all stl: StudentTournamentLeaderboard, t: Tournament | stl.relative_to=t iff t.leaderboard=stl
}
```

```
// that there is a one-to-one correspondence between CodeKatas (ck) and Battles (b) in the context of the "relative_codeKata" relationship.
fact singleCodeKataForBattle{
    all b: Battle, ck: CodeKata | b.relative_codeKata=ck iff ck.battle=b
}

//there is a one-to-one correspondence between SoftwareProjects (sp) and CodeKatas (ck) in the context of the "is_included" relationship.
fact singleSoftwareProjectForcodeKata{
    all sp: SoftwareProject, ck: CodeKata | sp.is_included=ck iff ck.softwareProject=sp
}


//if a TestCase (tc) is associated with a SoftwareProject (tc.softwareProject = sp), then that TestCase should be included in the set of TestCases belonging to that specific SoftwareProject (tc in sp.has).
fact singleSoftwareProjectForTestCase{
    all sp: SoftwareProject, tc: TestCase | tc.softwareProject=sp implies tc in sp.has
}

//each SoftwareProject (sp) has a distinct set of TestCases, meaning that if a TestCase (tc) is in the set of TestCases of one SoftwareProject (tc in spl.has), it should not be in the set of TestCases of another distinct SoftwareProject
//(tc not in sp2.has).
fact eachSPHasDifferentTestCases{
    all disj spl,sp2: SoftwareProject, tc: TestCase | tc in spl.has implies tc not in sp2.has
}

//if a team (t) is associated with a battle (t.battle = b), then that team should be included in the set of teams associated with that specific battle (t in b.teams).
fact teamRelativeToABattle{
    all t: Team, b:Battle | t.battle=b implies t in b.teams
}

//each team (t) is associated with exactly one battle, meaning that if a team is in the set of teams associated with one battle (t in bl.teams), it should not be in the set of teams associated with another distinct battle (t not in b2.teams).
fact singleBattleForTeam{
    all disj bl, b2: Battle, t: Team | t in bl.teams implies t not in b2.teams
}

// there is a one-to-one correspondence between scores (s) and teams (t) in the context of the relative_to relationship.
fact singleScoreForTeam{
    all s: Score, t:Team | s.relative_to=t iff t.score=s
}


//each educator creates a unique set of badges, meaning that the set of badges created by one educator is not the same as the set of badges created by another educator.
fact educatorsCreatesDifferentBadges{
    all disj e1, e2: Educator | e1.creates != e2.creates
}

//each badge (b) is associated with exactly one educator (e), meaning that if a badge is created by an educator (b.created_by = e), then that badge should be included in the set of badges created by that specific educator
//(b in e.creates).
fact EachBadgeIsRelativeToAnEducator{
    all b: Badge, e: Educator | b.created_by=e iff b in e.creates
}

// if a badge (b) is created by an educator (b.created_by = e), then that educator should be one of the educators managing the tournament in which he's creating the badge.
fact EachEducatorCreatesBadgeWithPermissions{
    all b: Badge, e: Educator, t: Tournament | b.created_by=e implies e in t.is_managed_by
}

//if a badge (b) is associated with a specific tournament (b.tournament = t), then that badge should be included in the set of badges associated with that specific tournament (b in t.has).
fact  badgeAssignedInASingleTournament{
     all b: Badge, t: Tournament | b.tournament=t implies b in t.has
}



pred createScenario{
    #Educator=2
    #Student=3
    #Tournament=1
    #Battle=2
    #TestCase=4
    #Team=2
    #ManualEvaluation=2
    #AutomaticEvaluationTool=1
    #Student=3
}

run createScenario for 5
```

4.2 RESULTS AND GENERATED INSTANCES

4.2.1 FIRST RUN

Initially, the predicate "createScenario" it has been run in order to visualize a significant representation of the world.

```
Executing "Run createScenario for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20 Mode=batch
  16367 vars. 1285 primary vars. 29777 clauses. 337ms.
  Instance found. Predicate is consistent. 122ms.
```

FIGURE 20: RESULT OBTAINED BY RUNNING THE ALLOY MODEL

Below is reported an instance of the world obtained with the above specified execution of the Alloy model. Subsequently, in order to better analyze the validity of the constraints imposed by the definition of facts, there are provided some relevant zoomed screens.

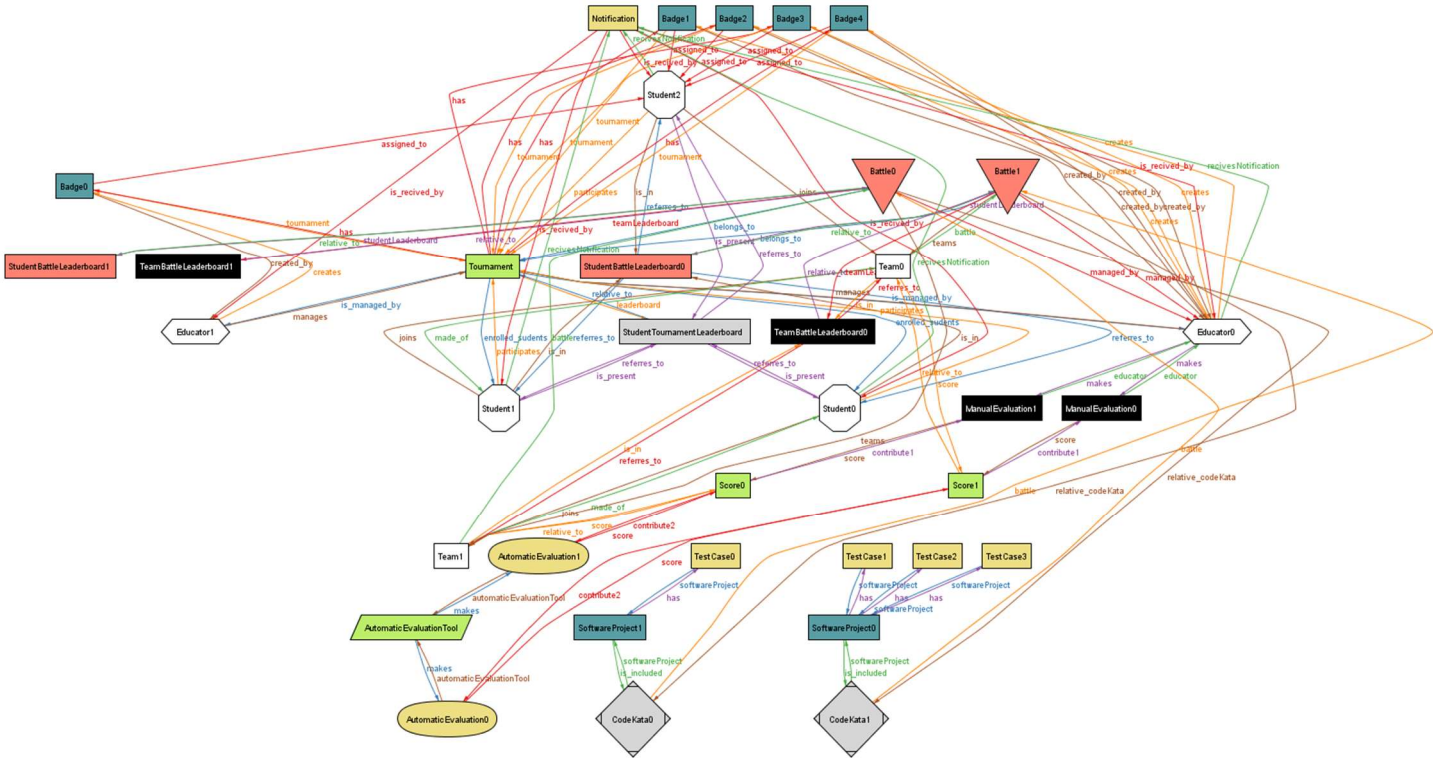


FIGURE 21: INSTANCE OF THE WORLD OBTAINED BY RUNNING THE ALLOY MODEL

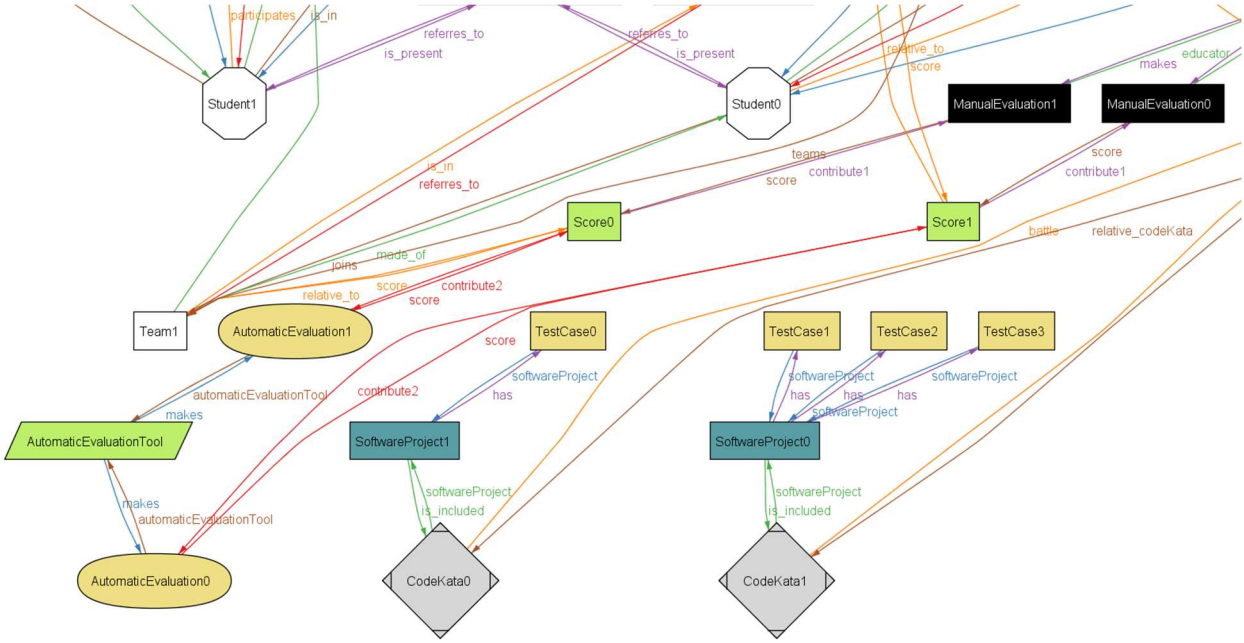


FIGURE 22: CODEKATA AND SCORE STRUCTURES

In this figure we can easily observe how the structure of The CodeKata and Score objects are perfectly described by the model: each score is made up of an automatic and a manual evaluation, while each CodeKata has a SoftwareProject including some TestCases in it.

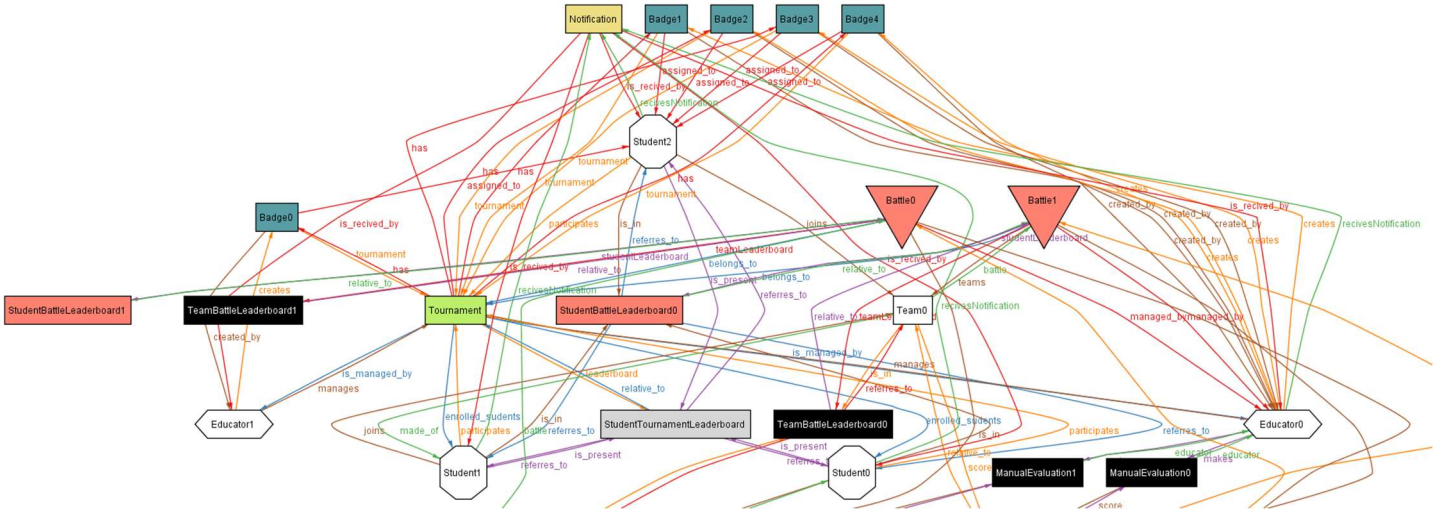


FIGURE 23: MAIN SECTION OF THE MODEL

This image perfectly shows that the Alloy model incorporates a set of constraints and facts that collectively define the relationships and behaviors within the specified domain. These constraints ensure the integrity and consistency of the model, reflecting the desired properties and interactions among different entities.

We can easily state by the image that:

- Educators are associated with tournaments they manage.
- Battles can only be created by educators with specific tournament management permissions.

- Students within a team are enrolled in the tournament relative to the battle in which the team participates.
- Each automatic evaluation is generated by a specific automatic evaluation tool.
- Educators can create manual evaluations only for battles they manage.
- Each manual evaluation is associated with only one educator.
- Each battle is associated with a unique CodeKata.
- Each team is associated with battles they participate in.
- Each team is uniquely associated with one battle.
- Each team is uniquely associated with one score.
- Each educator creates a unique set of badges.
- Educators creating badges only if they have permissions for tournament management.

In the following images we can have a more focused view of the model.

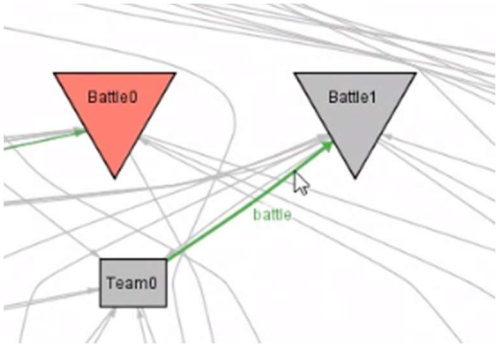


FIGURE 24: TEAM ENROLLED IN A BATTLE

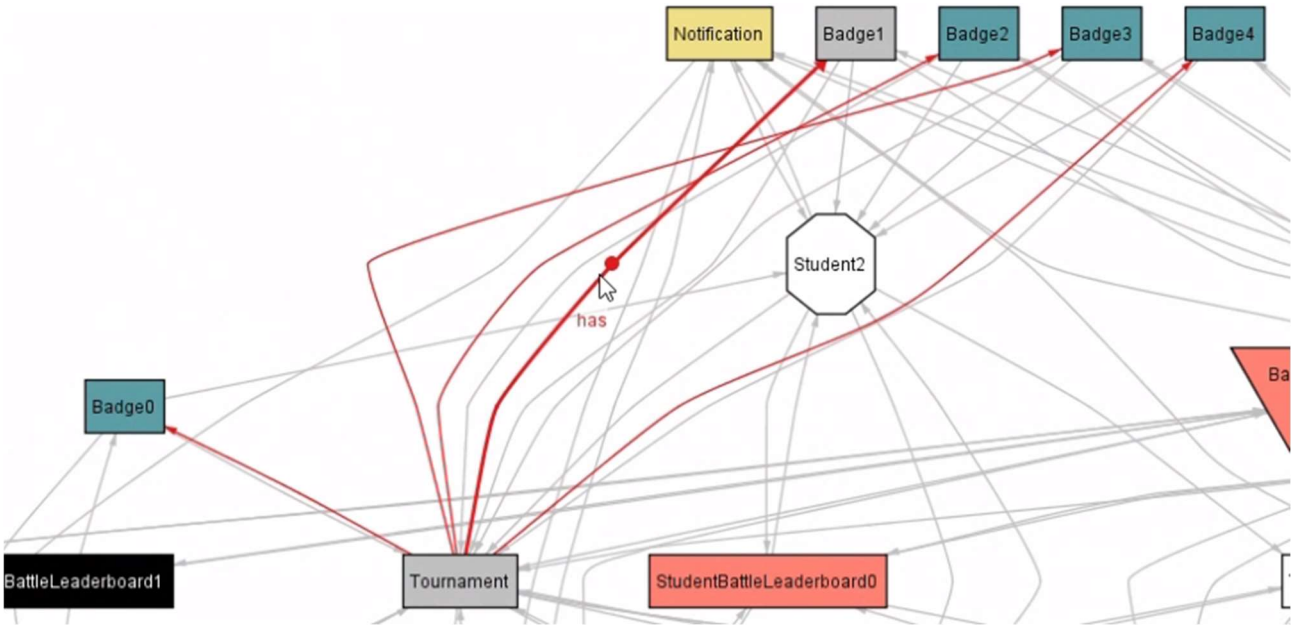


FIGURE 25: BADGES RELATIVE TO A TOURNAMENT

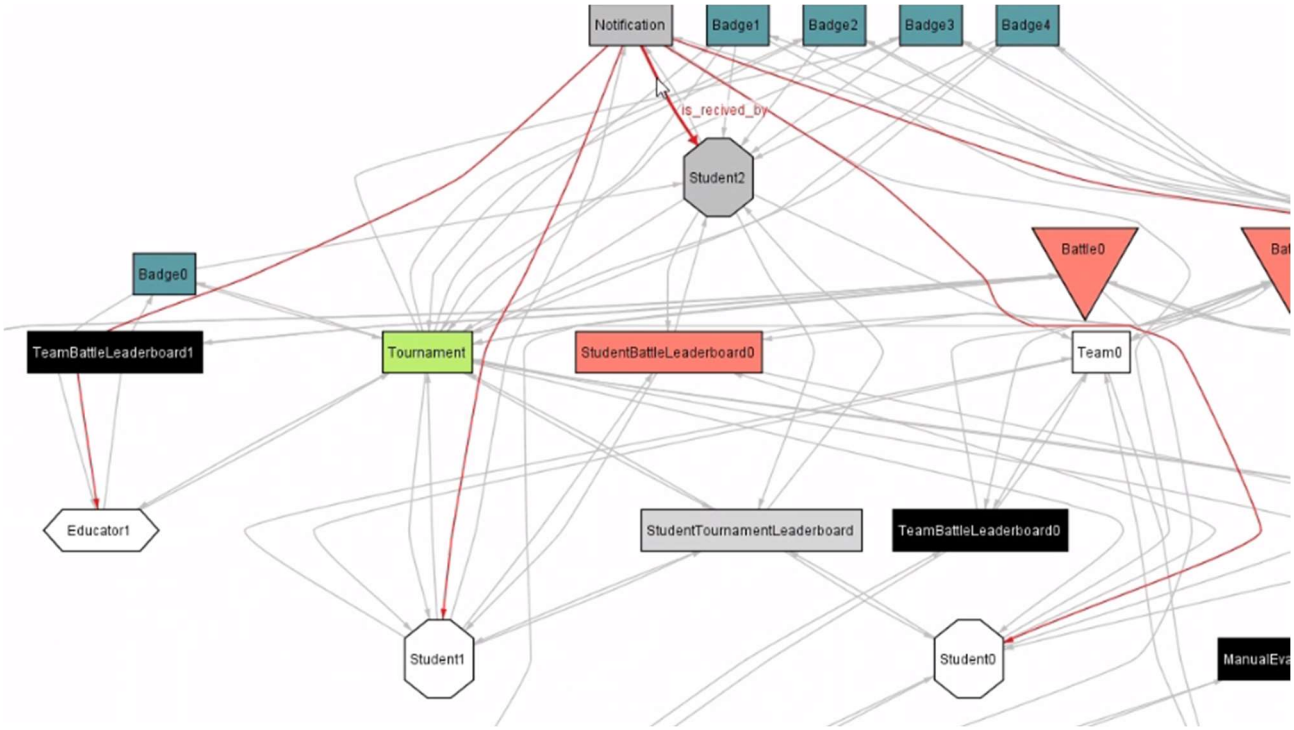


FIGURE 26: NOTIFICATION SYSTEM

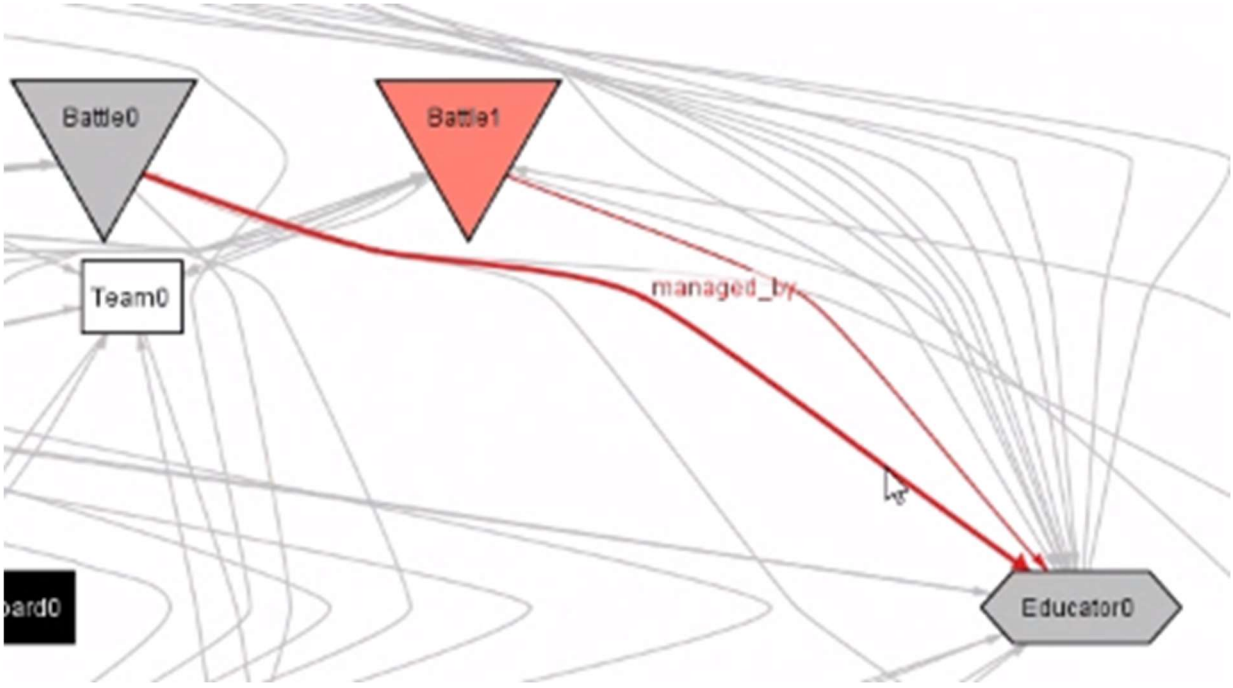


FIGURE 27: BATTLE MANAGMENT BY EDUCATOR

4.2.2 SECOND RUN

The figure below represents a second instance of the world, generated by the same code previously described. It can be easily observed that all the constraints discussed in the previous section continue to be valid.

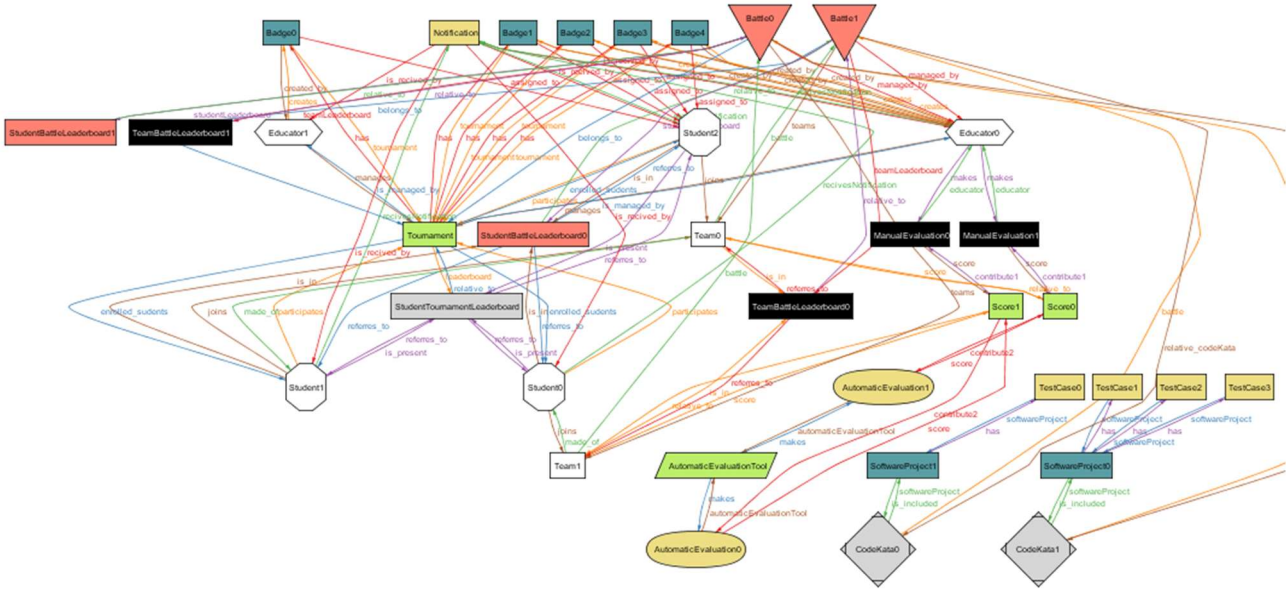


FIGURE 28: SECOND INSTANCE OF THE WORLD OBTAINED BY RUNNING THE ALLOY MODEL.

4.2.3 THIRD RUN

Below it is reported another instance generated by running the code. Also this case will not be analyzed in detail, but it can be observed that all the relations are consistent with the model for the reasons already specified.

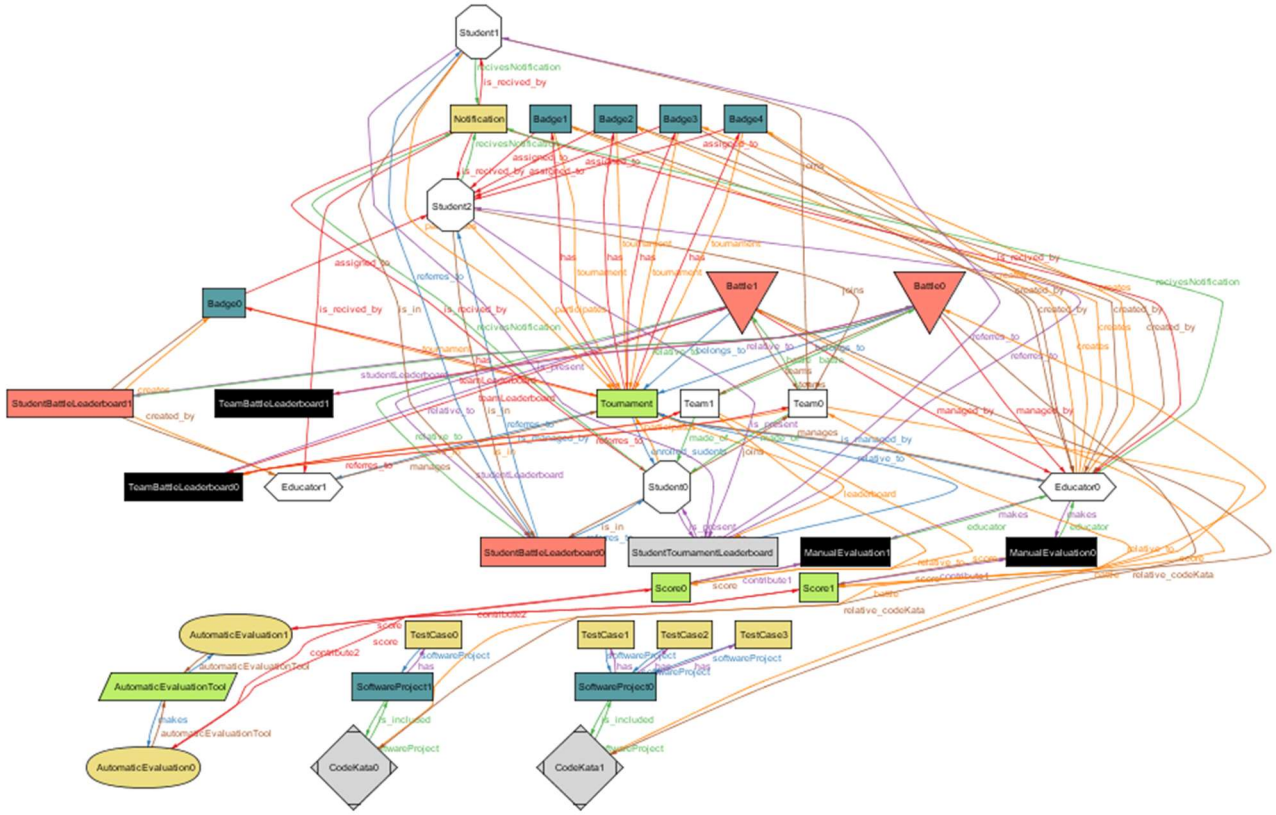


FIGURE 29: THIRD INSTANCE OF THE WORLD OBTAINED BY RUNNING THE ALLOY MODEL.

In conclusion, after several other executions of the code, it was always obtained as a result an instance of the world consistent with the model and with the features of the system that are described in this specification document.

5 EFFORT SPENT

Student 1

Topic	Hours
General reasoning	6:00h
Purpose & Scope	4:00h
Phenomena	3:00h
Scenarios	1:30h
Class diagram	2:00h
Statechart	2:00h
Product functions	1:00h
Domain assumptions	1:00h
User interfaces	2:00h
Hardware, software and communication interfaces	1:00h
Functional requirements	1:30h
Use case & Use case Diagram	5:00h
Sequence Diagrams	3:00h
Design constraints and software attributes	2:00h
Alloy and alloy documentation	8:30h
Document organization	4:30h
Total effort spent	48:00h

Student 2

Topic	Hours
General reasoning	6:00h
Purpose & Scope	4:00h
Phenomena	3:00h

Scenarios	1:30h
Class diagram	2:00h
Statechart	2:00h
Product functions	1:00h
Domain assumptions	1:00h
User interfaces	2:00h
Hardware, software and communication interfaces	1:00h
Functional requirements	1:30h
Use case & Use case Diagram	5:00h
Sequence Diagrams	3:00h
Design constraints and software attributes	2:00h
Alloy and alloy documentation	8:30h
Document organization	4:30h
Total effort spent	48:00h

Student 3

Topic	**Hours**
General reasoning	6:00h
Purpose & Scope	4:00h
Phenomena	3:00h
Scenarios	1:30h
Class diagram	2:00h
Statechart	2:00h
Product functions	1:00h
Domain assumptions	1:00h
User interfaces	2:00h
Hardware, software and communication interfaces	1:00h
Functional requirements	1:30h
Use case & Use case Diagram	5:00h
Sequence Diagrams	3:00h

Design constraints and software attributes	2:00h
Alloy and alloy documentation	8:30h
Document organization	4:30h
Total effort spent	48:00h

6 REFERENCES

- All the diagrams have been made with Draw.io
- Alloy code was made on the specific Analyzer tool
- All the user interfaces have been made with VisualStudioCode using HTML.