POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering 2 Requirements Analysis and Specification Document

Author(s): **Federico Zanca**
Federico Costantini
Michele Zhenghao Zhuge

Academic Year: 2023-2024
v2.0

Contents

1 | Introduction

CodeKataBattle (CKB) emerges as a pivotal platform in the evolving landscape of software development education. As the paradigm shifts towards enhancing coding proficiency, CKB provides a dynamic arena where students engage in collaborative code kata battles, refining their skills through hands-on challenges.

Educators orchestrate these battles within tournaments, defining parameters such as group sizes, deadlines, and scoring configurations. Facilitating a test-first approach, CKB seamlessly integrates with GitHub, automating workflows and enabling real-time evaluations. As scores evolve, students and educators gain insights into individual and team performances, fostering a competitive yet collaborative learning environment.

Beyond individual battles, CKB aggregates personal tournament scores, offering a comprehensive view of students' progress. Introducing gamification with badges, educators recognize achievements, enhancing the overall learning experience. In essence, CodeKataBattle propels software development education into a new era, blending competition, collaboration, and skill refinement within a cutting-edge platform.

1.1. Purpose

1.1.1. Goals

`CodeKataBattle(CKB)` platform serves as a dynamic and interactive space for students and educators, promoting continuous learning, healthy competition, and recognition of individual and team accomplishments in the realm of software development.

Students can engage in coding challenges by joining battles individually or forming teams within specified limits. By actively contributing to code repositories on GitHub, following a test-first approach, they strive to achieve high scores in battles, contributing to their personal tournament ranking. Eventually work towards earning gamification badges based on their performance and achievements in tournaments.

Educators instead, develop and publish code kata battles, setting criteria for evaluation and deadlines. They will assess student submissions, providing manual scores if required and shaping the final rankings, also defining gamification badges with specific rules and criteria to recognize outstanding achievements. Further more they will oversee the progression of tournaments, including opening, closing, and notifying participants.

Below there's a table that lists all the goals of the `CKB` platform:

ID	Description
G1	The platform should allow educators to set up tournaments.
G2	The platform should allow educators to set up code kata battles with configurable parameters.
G3	The platform should allow third party platforms integration.
G4	The platform should allow students to join battles individually or form teams within specified size limits.
G5	The platform should allow users to see real time updates on current tournaments and battles.
G6	The platform should have automated evaluations of code submissions.
G7	The platform should allow educators to manually evaluate and assign scores for optional factors at the end of each battle.
G8	The platform should allow users to visualize informations about another user.
G9	The platform should allow educators to create new gamification badges.

Table 1.1: The goals.

1.2. Scope

The scope section of the RASD document focuses on tidentifying the CKB platform and its domain outlining the main phenomena that may happen in the context of the application.

1.2.1. World phenomena

ID	Description
WP1	Educators create new tournaments for coding challenges.
WP2	Students seek opportunities to improve their software development skills.
WP3	Educators define code kata battles with project details and evaluation criteria.
WP4	Students aim to participate in code kata battles individually or form teams.
WP5	Students are informed of upcoming battles when subscribed to a tournament.
WP6	Students form teams within specified limits for a code kata battle.
WP7	Students aim to visualize their personal tournament scores and earned badges.
WP8	Educators define gamification badges and rules when creating a tournament.
WP9	At the end of each battle, the CKB platform notifies students of the final battle rank and updates personal tournament scores.
WP10	Educators can close tournaments, and the platform notifies all students when the final tournament rank becomes available.
WP11	Students and educators seek information about ongoing tournaments, ranks, and profiles with badges.

Table 1.2: World Phenomenas.

1.2.2. Shared phenomena

ID	Description	Controller	Observer
SP1	Student subscribes to the CKB platform for notifications.	Student	CKB Platform
SP2	Educator creates a new tournament, specifying details and criteria.	Educator	CKB Platform
SP3	CKB Platform notifies subscribed students about a new tournament.	CKB Platform	Student
SP4	Student forms a team within specified limits for a code kata battle.	Student	CKB Platform
SP5	Educator sets up a code kata battle, including project details and evaluation criteria.	Educator	CKB Platform
SP6	CKB Platform generates a GitHub repository for a code kata battle.	CKB Platform	Student
SP7	Student forks the GitHub repository and sets up automated workflows.	Student	CKB Platform
SP8	Student pushes commits to GitHub, triggering automated evaluation.	Student	CKB Platform
SP9	Automated evaluation includes functional correctness, timeliness, and code quality.	CKB Platform	Student
SP10	Educator manually evaluates optional factors, such as personal scores.	Educator	CKB Platform
SP11	CKB Platform updates battle scores in real-time based on new commits.	CKB Platform	Student
SP12	Students and educators observe the evolving rank during a battle.	Student	CKB Platform
SP13	CKB Platform notifies students when the final battle rank is available.	CKB Platform	Student
SP14	CKB Platform updates personal tournament scores at the end of a battle.	CKB Platform	Student
SP15	Educator closes a tournament, and the platform notifies students of the final rank.	Educator	CKB Platform
SP16	Educator defines gamification badges and rules when creating a tournament.	Educator	CKB Platform

SP17	Students earn gamification badges based on their performance and achievements.	Student	CKB Platform
SP18	Students and educators visualize ongoing tournaments, ranks, and profiles with badges.	Student	CKB Platform

Table 1.3: Shared Phenomenas.

1.3. Definition, Acronyms, Abbreviations

Acronyms	Definition
CKB	Code Kata Battle
RASD	Requirements Analysis and Specification Document

Table 1.4: Acronyms used in the document.

1.4. Revision history

This is the second version of the document. This version includes the following changes:

- Added mailing system to dependencies. This is used to send notifications via email to users.
- Fixed typos and grammar errors.

1.5. Reference Documents

- The specification document Assignment RDD AY 2023-2024.pdf

1.6. Document Structure

The document is structured in six sections, as described below.

First section introduce the goals of the project, purposes, and a brief analysis on world and shared phenomena; abbreviations and definitions useful to understand the problem are listed as well.

The following section, the second one, provides an overall description of the problem: here further details on domain and scenarios are included, aside from more product and user characteristics, assumptions, dependencies and constraints.

Later on, the third section focuses on the specific requirements and provides a more detailed analysis of external interface requirements, functional requirements and performance requirements.

Lastly, the fourth section provides a formal analysis, using Alloy 6. This chapter is crucial to prove the correctness of the model described in the previous sections, and should focus on reporting results of the checks performed and meaningful assertions.

Section five reports the effort spent by each group member in the redaction of this document, meanwhile the last section simply lists bibliography references and other resources used to redact this document.

2 | Overall Description

2.1. Product perspective

2.1.1. Class Diagrams

The diagram below represents and describes the classes involved in the system, their basic functionalities, attributes, and the relationships between them. From the diagram, it is easy to see how the tournament entity function as a bridge between different components. More specifically, in our design we assigned to this entity the task to hold track of battles, leaderboard of the tournament itself and badge assignment. All the other functionalities are esily understandable from within the diagram or will be later discussed.
Here, we want to stress that, even though just an high level view of the system-to-be, some consideration of implementation can be done. More specifically, some design pattern can be taken into consideration:

- Decorator Pattern: to implement the logic behind the scoring system. Educator's choice of different aspect to consider will be easily managable
- Observer patter: to implement the structure behind the notification system.
- Factory pattern: for the creation of different Badges with different characteristics and requirements.

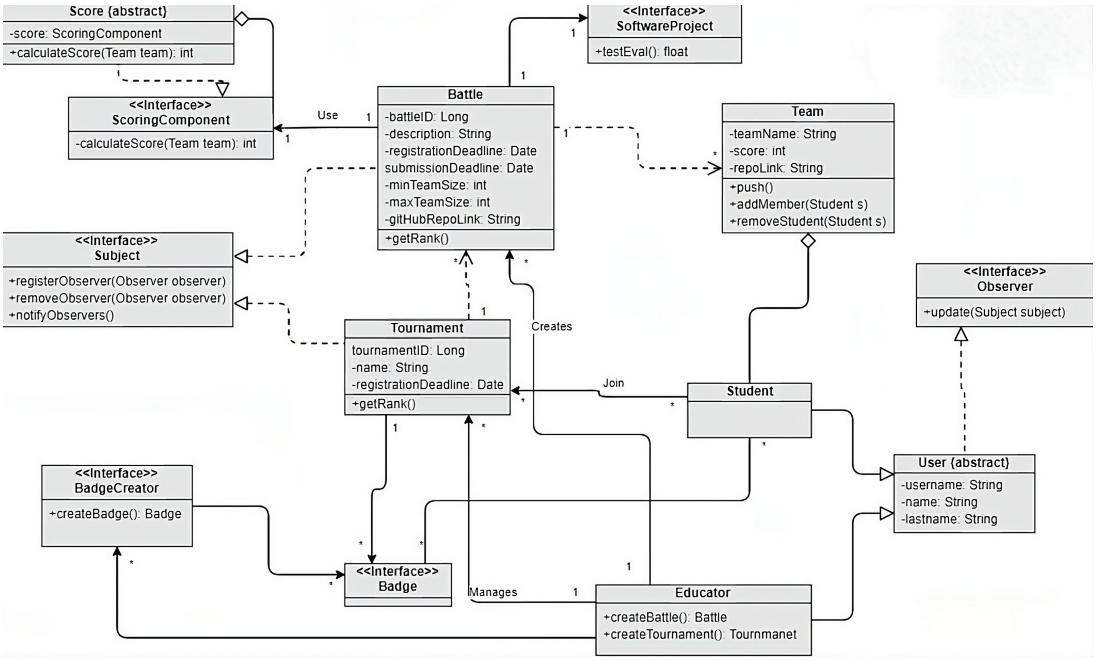

Figure 2.1: A simplified Class Diagram

2.1.2. Scenarios

Unregistered student creates an account. Gianmarco is a computer science student from Politecnico of Milan. After attending several courses about software engineering, he is looking for a way to practice his skills. Fortunately, one of his professor suggests him the CodeKataBattle platform. Gianmarco immediately proceeds to create an account. He navigates to the platform and goes to the "sign up" section. The system asks Gianmarco an email linked to an institutional profile valid for the CKB platform (for example name.surname@mail.polimi.it). He is then redirected to the log in page of his institution. After a successful authentication, Gianmarco will be prompted with a data access request page for sharing data about his university profile. This way the CKB platform can determine wheter Gianmarco is a student or a professor and act accordingly. Once accepted the data access request, he is redirected to the creation profile page, where he is asked to insert an unique username. After accepting the terms & conditions and submitting everything, the system creates his account, if info are correct, and Gianmarco can begin to use the platform.

User Logs in. Emily, the coding enthusiast, is all set for her daily dose of challenges on the CodeKataBattle (CKB) platform. To embark on her coding adventure, she clicks the login button and enters her institutional email. Emily is then redirected to the access page of her institution where she performs the authentication. Once authenticated, she

is reconnected to the CKB platform that welcomes her in the homepage; she's now ready to dive into the coding battles. In case there's a typo in her email, the platform gently guides Emily through the process, ensuring a seamless login experience.

Student creates a team. Francesca, a student on the CodeKataBattle (CKB) platform, is eager to participate in the "Algorithmic Challenge" CKB. Despite the possibility to participate individually to this battle, Francesca is aware that winning it all by herself is a daunting task, so she decides to form a team. She navigates to the tournament details page and clicks on the "Create Team" button and is prompted to input the size of the team compliant to the settings of the battle (min/max number of students per team). Francesca wants to form a team of three students, so she chooses the number three from the drop-down menu. After deciding that the name of her team will be "The Three Musketeers," she clicks on the "Create Team" button. She will now be able to send other students team invitation and accpete or decline joining requests.

Student invites another student to join a team. Imagine Marco, an enthusiastic student navigating the CodeKataBattle (CKB) platform, keen on forming a proficient coding team for an upcoming battle. Having already gathered part of his team, Marco identifies Sofia, another student he believes would enhance their collective skills. On the tournament page, Marco locates Sofia's profile in the lists of students not in a team yet and clicks the "Invite" button, signaling his desire to collaborate. The CKB platform efficiently checks team capacity, ensuring seamless integration. In the digital realm, an invitation is generated and dispatched to Sofia through the platform's communication channels. Marco eagerly awaits the platform's response, hopeful for a positive outcome. Upon success, the invitation is sent, propelling Marco's team towards a cohesive coding alliance.

Student accepts an invitation to join a team. Sofia, a student passionate about coding, is actively using the CKB platform to participate in coding tournaments. Today, as she's logged in, he receives a pop-up notification indicating that she has been invited by Marco to join a team for a CKB. Intrigued, she clicks on the "Accept" button within the invitation pop-up as she knows that her and Marco would be a good team. Behind the scenes, the CKB platform swiftly processes this action. If the team didn't fill up in the meantime, the CKB platform conveys the acceptance information to the student who sent the invitation, ensuring a seamless interaction between both students involved. Sofia and the sender of the invitation are promptly notified of the outcome. The invitation has been successfully accepted, and Sofia is now part of the team.

Student rejects an invitation to join a team. Sofia, a dedicated student engaged in coding challenges on the CKB platform, is logged in and actively participating. Today, she receives a pop-up notification, signaling that she has received an invitation to join a team for a CKB. After thoughtful consideration, Sofia decides to decline the invitation. With a click on the "Reject" button within the invitation pop-up, she communicates her decision to the CKB platform. Behind the scenes, the platform ensures this information is promptly relayed to the student who extended the invitation. Both Sofia and the sender of the invitation receive immediate communication from the CKB platform, confirming the outcome of the declined invitation. The respectful rejection is acknowledged, allowing the students to proceed with their individual coding journeys.

Student joins a battle without a team. Emma, an enthusiastic student with a passion for coding challenges, is actively participating in a coding tournament on the CKB platform. Excited about a specific battle, she navigates to the tournament page to explore the available challenges. Spotting a captivating battle in the list, Emma decides to join without forming a team as she is confident on her own skills. She selects the desired battle and clicks the "Join Battle" button at the bottom of the list. Behind the scenes, the CKB platform promptly verifies the registration deadline for the chosen battle. Upon completion of this process, the platform communicates the outcome of Emma's action. Whether met with success or needing adjustment, Emma receives immediate feedback, allowing her to seamlessly engage in the selected coding challenge.

Professor creates a tournament. Achille is a cyber security professor from university of Milan. He has an active profile on the CKB platform. During his last lesson, he annouced to his students that he will create monthly tournaments with weekly Battle for them to practise. He then proceeds to create the first tournament. He logs in the CKB platform from the "sign-in" page. Then, from the menu option in the home page, he selects "Create a new tournament". He is prompted with the page of the setup for the tournament itself. Achille enters a name for the tournament and the registration deadline. He then toggles the "advanced options" section. From here, he can decide wheter to include or not badges, and, in the first case, which badges are available in the tournament. Achille is prompted with a set of pre-existing badges (created from the platform or in previous tournaments). He can decide to create new badges in addition to the ones already present. When choosing to create a new badge, a pop-up window appears where he can select and combine variables and rules to obtain the new badge. Once he finishes setting up everything, he pushes the "Create tournament" button at the bottom of the page.

Professor closes a tournament. Logged into the CKB platform, Professor Lucia navigates to the "Tournaments" page. She locates the "Programming Prodigy Challenge" in the list and clicks on it to access the tournament details. Satisfied with the students' participation and battle outcomes, she clicks the "Close Tournament" button. A confirmation pop-up appears, and without hesitation, Professor Lucia confirms the action. The CKB platform processes the request, communicates the successful tournament closure, computes the final ranking based on accumulated scores, and notifies student about possible badges acquisition if necessary. The platform promptly makes the final ranking available for participants. Professor Lucia, acknowledging the students' efforts, briefly reviews the rankings.

Professor creates a new CKB. Professor Roberto, logged into his CKB platform account, navigates to the tournament's details page named "Algorithmic Mastery Challenge." Excited to introduce a new coding challenge, he clicks on the "Create New Code Kata Battle" button. Professor Roberto is prompted to upload a code kata for the battle: he selects a well-prepared code kata that includes a comprehensive description and a software project with test cases and build automation scripts. He ensures that all necessary components are included before uploading. Then he decides that groups should consist of a minimum of two students and a maximum of four students for this battle, so Professor Roberto sets these values accordingly in the "Group Size" field. The educator proceeds to set a registration deadline and a final submission deadline, providing students with ample time to prepare and submit their solutions. Curious about the advanced scoring options, Professor Roberto clicks on the "Additional Configurations for Scoring" button. The platform displays the scoring section, allowing him to set additional configurations if needed. Satisfied with the default scoring, he proceeds to click the "Create" button.

Push action on GitHub Let's consider a team named "CodeWarriors" consisting of two students, Alice and Bob. They are participating in a battle in one of the tournaments created by their professor, Achille. Alice and Bob are working on the code kata for the battle. They are in a brainstorming session, discussing the problem and proposing solutions. They write some initial code and test it locally on their machines. After some time, they are satisfied with their progress and decide to push their code to GitHub. Bob commits their code with a meaningful commit message and pushes it to their team's repository on GitHub. As soon as the push action is performed, the CKB platform is triggered. It pulls the latest sources from the team's repository, analyzes them, and runs the tests on the corresponding executables. The platform then calculates the team's score based on the functional aspects, timeliness, and quality level of the sources. Alice and

Bob receive a notification from the CKB platform about their updated score. They review the feedback, discuss the areas they need to improve, and continue working on the code kata. They are now in the process of refining their code and planning their next push action.

Professor grants permissions to another professor to add CKB to a tournament. Professor Marta, logged into her CKB platform account, navigates to the details page of the tournament named "Coding Challenge Extravaganza," which she initiated. Realizing the workload involved in creating battles, she decides to grant permissions to another educator, Professor Elena. She clicks on the "Add Administrator" button, prompting the CKB platform to request the email or username of the educator to whom she wants to grant permissions. Professor Marta confidently inputs Professor Elena's email. Professor Elena receives a notification from the CKB platform, informing her that she has been granted permissions to add battles to the tournament named "Coding Challenge Extravaganza." Now, Professor Elena, can access the tournament details page and contribute by creating new Code Kata Battles.

Professor creates a new Badge. Professor Sofia, logged into her CKB platform account and currently on the details page of the "Coding Excellence Showcase" tournament, is inspired to introduce a special badge for outstanding performances. Excited about the idea, she clicks on the "Create New Badge" button located in the "Advanced Options" section. The CKB platform prompts her to input the title of the badge. Professor Sofia, with a clear vision in mind, names the badge "Elite Coder." Next, the platform allows Professor Sofia to define the rules for the badge. She navigates through a user-friendly wizard, creating rules that capture exceptional coding skills. Each rule represents a milestone that, when achieved, contributes to earning the "Elite Coder" badge. Professor Sofia completes the rule-setting process, ensuring a fair and challenging criteria for students to meet. Satisfied with her creation, she clicks on the "Create" button.

Professor manually evaluates teams in a CKB. Professor Marco, logged into his CKB platform account and currently on the details page of the "Algorithmic Challenge" CKB, a challenge that is ended and is in post evaluation phase. The team already has a score, automatically computed by the platform, but Professor Marco wants to provide a more detailed evaluation. Eager to ensure a fair evaluation, he clicks on the "Manually Evaluate Teams" button. The platform presents him with a list of teams that participated in the "Algorithmic Challenge." Professor Marco carefully selects a team from the list, keen on assessing their coding skills and approach to the given challenge. Upon selecting

a team, the CKB platform displays the team's submitted sources. Professor Marco thoroughly reviews the materials and, based on his expert judgment, assigns a comprehensive score to the team's performance. Satisfied with the evaluation, Professor Marco clicks the "Save" button to record his assessment. This meticulous manual evaluation process repeats as Professor Marco proceeds to assess other teams involved in the "Algorithmic Challenge." Each score contributes to the overall ranking of the teams.

User consults the leaderboard of a tournament. Sofia, an active user on the CodeKataBattle (CKB) platform, decides to check the current ranking of the "Data Structures Madness" tournament. She navigates to the "Tournaments" page and selects "Data Structure Madness" tournament. Sofia proceeds opening the details page and effortlessly views the real-time leaderboard with a single click. The platform promptly presents the current standings, allowing Sofia to gauge the performance of each student subscribed to the tournament.

User views the profile of another user to check their badges. Francesca, a student on the CodeKataBattle (CKB) platform, is intrigued to explore the profile of a student, Alex, to gain insights into his achievements. With a simple click on Alex's profile from the "Users" page, Francesca is presented with a comprehensive view. The platform seamlessly showcases the badges earned by Alex. To delve deeper, Francesca clicks on a specific badge and is presented with a detailed description of the badge and the criteria for earning it.

User views the ranking of an ongoing CKB. Emma, an enthusiastic student on the CKB platform, is eager to witness the live dynamics of an ongoing coding tournament. Logging into CKB, Emma navigates to a specific tournament, selecting a Code Kata Battle (CKB) of interest. Within the chosen CKB, Emma clicks on "Ranking". In an instant, the CKB platform reveals the real-time rankings, showcasing the fluid movements of teams based on their pushes to the repository, automatically evaluated by the system.

2.1.3. State Diagrams

Below are presented state diagrams relative to some of the scenarios described before

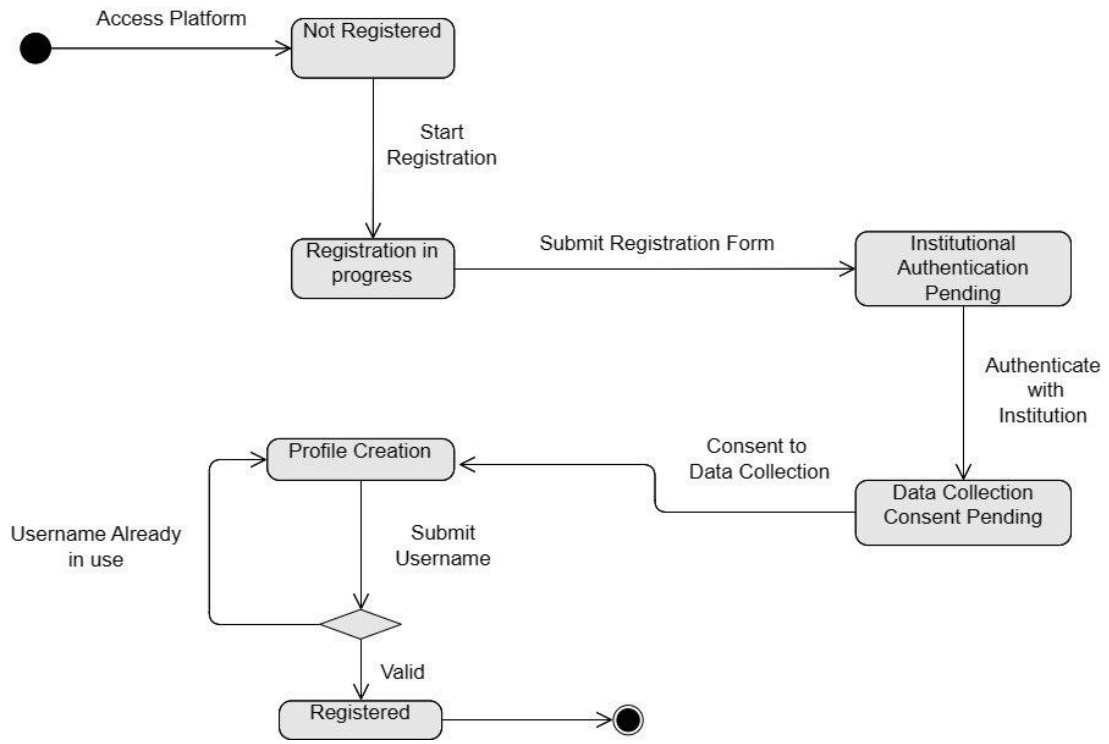

Figure 2.2: User registration state diagram

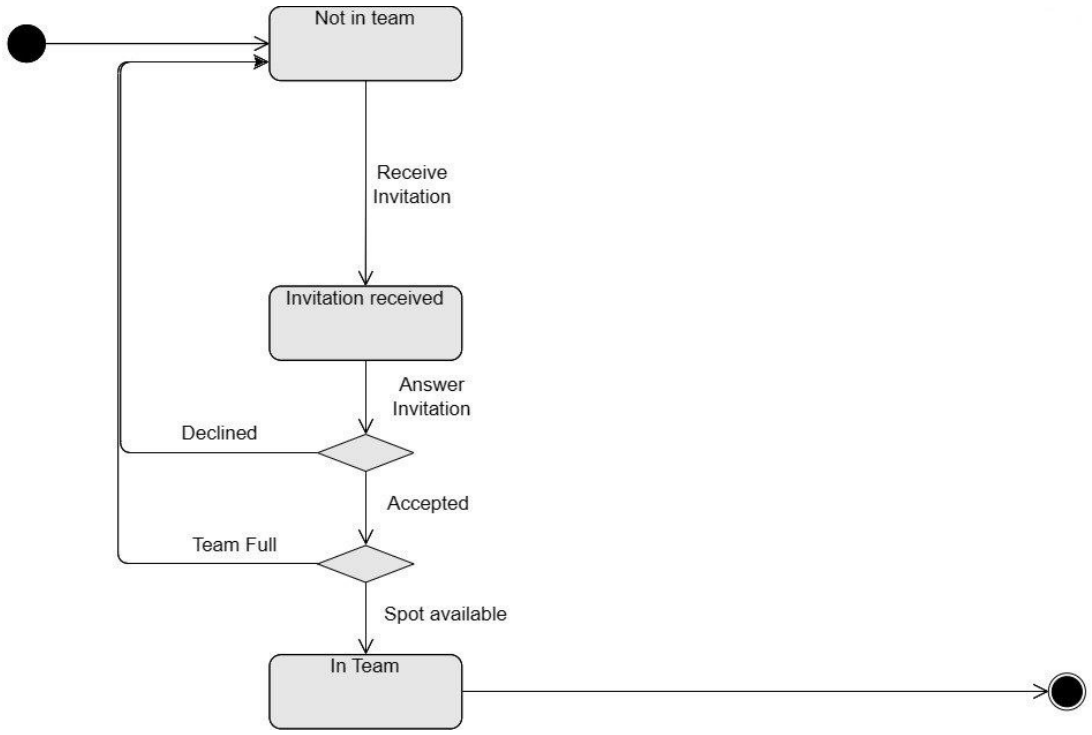

Figure 2.3: Join team state diagram

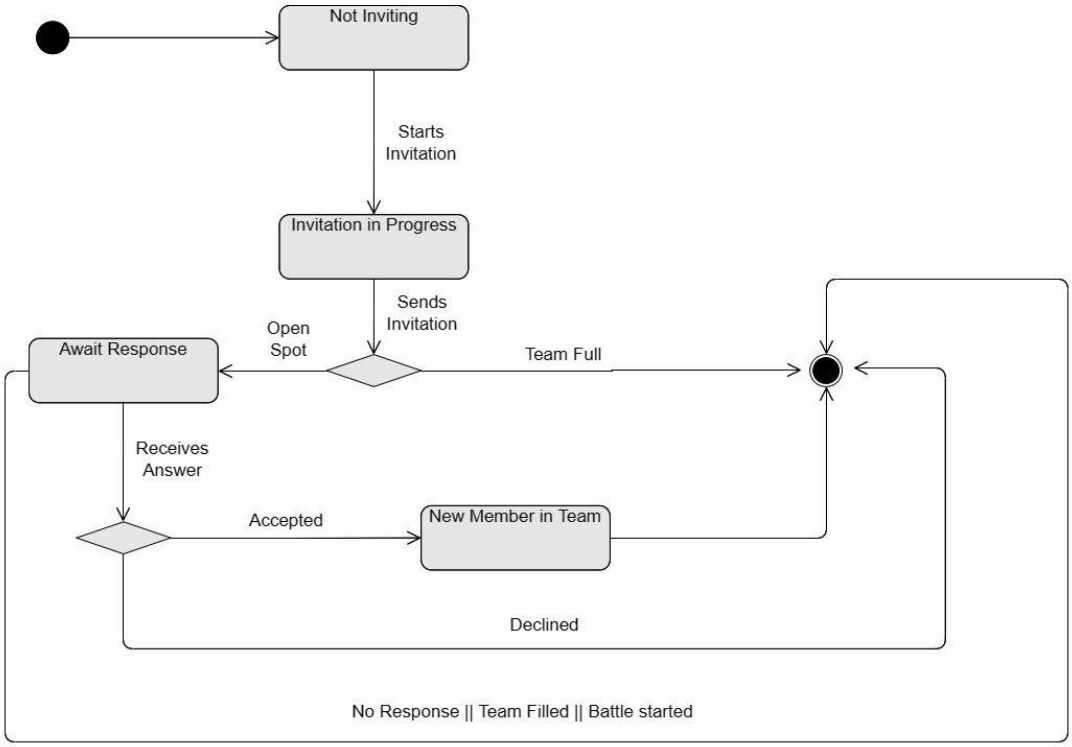

Figure 2.4: Student invitation state diagram

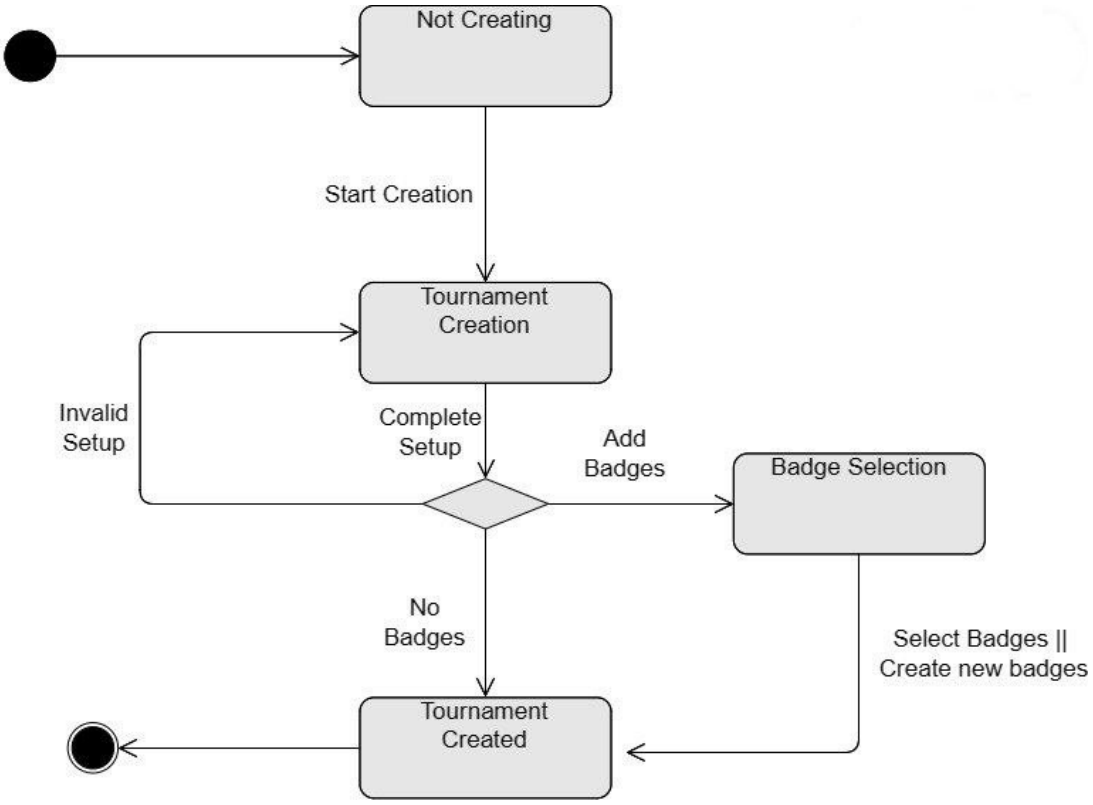

Figure 2.5: Tournament creation state diagram

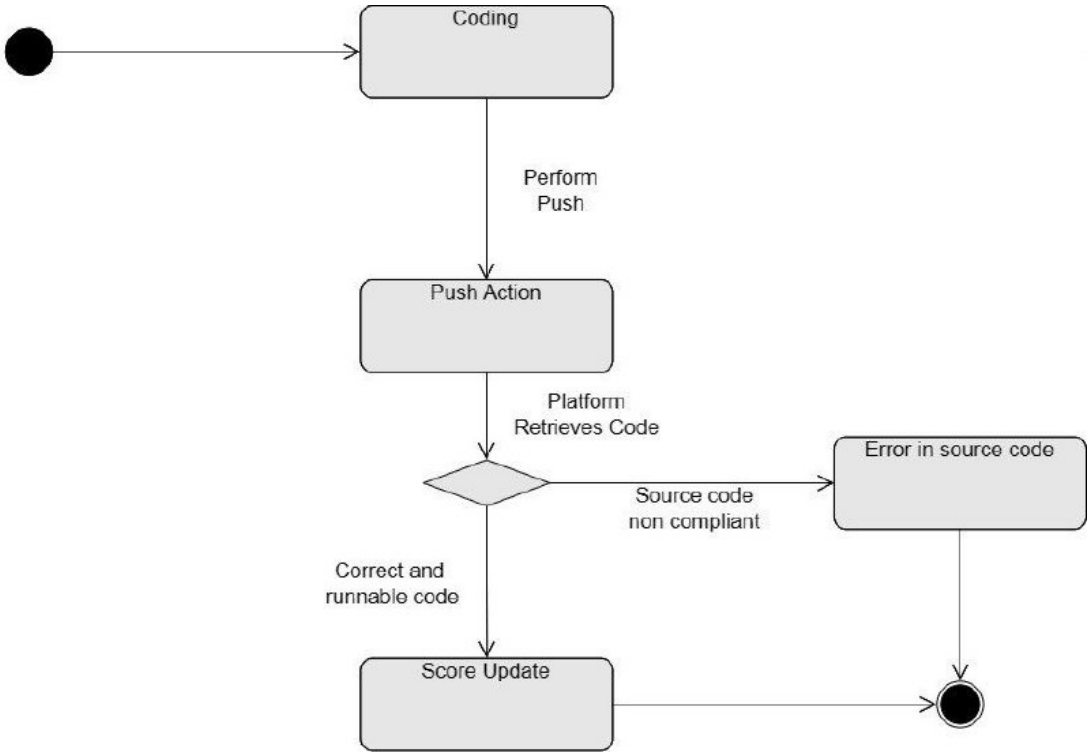

Figure 2.6: Push action state diagram

2.2. Product Functions

CodeKataBattle

CodeKataBattle was born with the intention of helping students improve their software development skills by training with peers on code kata. Educators use the platform to challenge students by creating code kata battles in which teams of students can compete against each other, thus proving (and improving) their skills.

Tournaments and Battles

CodeKataBattle (CKB) platform offers a dynamic and interactive service for creating tournaments and battles that plays a pivotal role in fostering collaborative coding endeavors. Educators leverage this functionality to design engaging coding competitions between students. CKB platform handles tournaments as a collection of battles, each with a specific set of parameters and rules. Each tournament has its ranking, which consist of a leaderboard of single students. Battles hold their own ranking, which consist of a leaderboard of teams. The platform provides a user-friendly interface that allows educators to create tournaments and battles, set up scoring systems, and manage the

overall tournament dynamics. Students, in response, actively participate in competing in battles, laying the groundwork for collaborative problem-solving experiences.

Team Formation

The team formation feature within the CodeKataBattle (CKB) platform stands as a pivotal element, streamlining the collaborative dynamics of problem-solving among students. This integral aspect of the platform caters to an array of user scenarios, presenting students with versatile options to either autonomously join battles or take a proactive role by extending and receiving invitations. This flexibility empowers students to shape their teams based on preferences, expertise, or collaborative strategies. The platform ensures a user-friendly team creation process, allowing students to seamlessly navigate the formation landscape. The invitation system is a precision tool, enabling users to send tailored invitations to specific individuals. It adheres to the battle-specific constraints, considering the predefined minimum and maximum number of students per group. This precision minimizes ambiguity in team size, ensuring a cohesive and efficient team-building process. Moreover, the acceptance mechanism is seamlessly integrated, facilitating swift and hassle-free team assembly. The platform's intuitive design considers user experience at its core, ensuring that the team formation process is not only functional but also enhances the overall collaborative learning experience.

Automated Scoring for Code Evaluation

Automated scoring is a core feature, evaluating various facets of student submissions. The platform is capable of performing automated analysis on the student' submissions. This analysis covers functional aspects, including test case pass rates, timeliness of submissions, and the quality of source code. The quality of the source code is determined by multiple factors, including security, reliability, and maintainability. Educators can leverage automated tools for static code analysis to ensure a comprehensive evaluation but are also able to manually evaluate submissions if needed.

Notifications

Automated notifications are dispatched to students when educators close or create a tournament. These notifications include final tournament ranks, providing closure to the competition and acknowledging the achievements of participating students. Moreover, the platform sends notifications to students when they are invited to join a team or when their invitation is accepted or rejected.

Gamification Badges

To inject a gamified element, educators define badges to recognize and reward student achievements. Badges, with titles like "Top Performer" or "Code Guru," serve as visual representations of accomplishments, enhancing student profiles and fostering a sense of achievement. Educators can include badges in tournaments by selecting from a list of pre-existing badges or creating new ones. When creating new badges, educators must define rules and variables that determine when a badge is awarded to a student. Rules and variables can either be selected from a list of pre-existing ones or created from scratch. In order to create a new variable, educators can make use of metrics exposed by the platform that offer information that could be relevant for scoring. These metrics can be combined to arithmetic operations to create new variables. New rules can be defined by performing operation on variables (avreage, max, min, sum, etc.), resulting in complex requirements the students must meet to earn the badge. Mathematical operations to define rules and variables are supported by the platform, which leverages Google Spreadsheets to compute the results.

2.3. User characteristics

The CKB platform caters to various user roles, each with specific responsibilities and access levels:

- **Guest:** Users who are not registered on the CKB platform but have the potential to become either students or educators. Guests have limited access and functionality until they complete the registration process. Once registered, they can assume the roles of either Student or Educator based on their role in the institution they belong to.
- **Educator:** Educators who have successfully registered within the CKB system. Identified by a unique identifier, registered educators, depending on permissions granted, may assume the role of Tournament Administrator. They possess the ability to create CKBs, evaluate student submissions, and engage in other system functionalities.
- **Student:** Users, both registered and unregistered, participating in code kata battles facilitated by the CKB platform. Students can form teams, submit solutions, and engage in the competition. Their scores, rankings, and achievements contribute to their overall performance, visible in the context of tournaments.
- **Tournament Administrator:** Educators with privileges to perform actions within

a specific Tournament. Tournament Administrators, typically the creators of a Tournament, can delegate administrative powers to other educators. These administrators have the authority to initiate new CKBs, evaluate submissions, and oversee tournament-related activities.

2.4. Assumptions, Dependencies, and Constraints

2.4.1. Assumptions

ID	Assumptions
D1	Educators using CKB platform have the necessary technical knowledge and skills to create and manage code kata battles. This includes the ability to create programming exercises, write test cases, and set up build automation scripts.
D2	All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary for accessing the platform, downloading code kata, submitting code, and receiving notification.
D3	Educators are expected to have the ability to evaluate the work done by students and assign scores if manual evaluation is required.
D4	Educators managing a torunament will not lose access to their institutional email during the whole duration of the tournament itself.
D5	Educators and students have a good understanding of GitHub and GitHub actions.
D6	Educators are capable of creating and closing tournaments, without leaving them open undefinetly when no more battles are scheduled or in session.
D7	The scoRing system is transparent, consistent, and coherent with the criteria of the rules (quality of sources, timeliness, ...)
D8	Badge rules and varibales are well-defined and understood by the platform. They are meaningfull with respect to the tournament associated with.
D9	Students have basic knowledge of at least one programming language supported by the CKB platform.
D10	Educators and students are linked to one and only one institutional email.
D11	Github repo collaborators are coherent with respect to groups created in the CKB platform for a given battle.

Table 2.1: Assumptions.

2.4.2. Dependencies

- Affiliated universities have an automated system to provide data about an authenticated student/educator to the CKB platform.
- GitHub needs to provide APIs to allow the CKB platform to receive notification when Actions are performed and be able to automatically pull the repositories of each group for evaluation.
- Google needs to provide APIs to allow the CKB platform to make use of Google Spreadsheets functionalities to validate and compute Spreadsheet formulas defined by educators to create new badges.
- Mail services need to be available to allow the CKB platform to send notifications to students and educators.

3 | Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

The `CodeKataBattle (CKB)` platform is accessed via an intuitive and responsive web interface compatible with major browsers. Educators enjoy a dedicated dashboard for creation and management of tournaments, battles, and badges. This dashboard provides a comprehensive view of ongoing battles, tournament scores, and badge achievements. Students utilize a user-friendly dashboard for team formation, battle participation, and progress tracking. It streamlines team formation, displays upcoming battles, current ranks, and summarizes earned badges. GitHub seamlessly integrates into the platform for code versioning and automated testing. Students can easily fork repositories, set up GitHub Actions, and monitor build and test results within the `CKB` platform. To keep all stakeholders informed, the platform employs a robust notification system. This system supports both email notifications and in-platform alerts, ensuring timely updates for educators and students on critical events like upcoming deadlines or changes in battle status.

3.1.2. Hardware Interfaces

The `CKB` platform prioritizes accessibility by ensuring compatibility across a diverse range of devices. Users can seamlessly access the platform from desktop computers, laptops, tablets, and smartphones. The platform's responsive design ensures that the user interface adapts fluidly to different screen sizes, providing an optimal experience regardless of the device used. This commitment to device compatibility aims to enhance user convenience and flexibility, promoting a versatile and user-centric engagement with the `CKB` platform.

3.1.3. Software Interfaces

The CKB platform seamlessly communicates with GitHub via APIs, enabling functionalities such as repository creation, commit tracking, and automated test processes. To ensure a secure integration, the platform should smoothly connect with GitHub APIs, facilitating automated workflows triggered by student commits, and Google Spreadsheet APIs, to help educators defining new rules and variables for gamification badges. Additionally, the platform harnesses static analysis tools to assess code quality comprehensively. Incorporating these tools seamlessly, educators can tailor automated evaluations by configuring specific aspects like security, reliability, and maintainability.

3.1.4. Communication Interfaces

The platform actively engages in communication with students, delivering notifications, battle updates, and final results in a secure manner through HTTPS. A reliable messaging system ensures timely information dissemination to students. Similarly, educators stay well-informed through the platform, receiving notifications and updates on battle progress and final results. Secure communication channels, similar to those used for students, guarantee the confidentiality and reliability of information relayed to educators. For the configuration of badges and rules, educators seamlessly use the platform to define new badges, rules, and associated variables. Ensuring a user-friendly interface, educators can make real-time adjustments, with changes promptly reflecting across the platform. Notifications are sent to students and educators via email and in-platform alerts. This ensures that all stakeholders are well-informed of critical events like upcoming deadlines or changes in battle status.

3.2. Functional Requirements

3.2.1. Requirements

The CKB platform offers several functionalities to both educators and students. In the following table they are listed all the detected requirements that the platform should respect in order to guarantee the satisfiability of the goals:

Educators

ID	Description
R1	The CKB platform shall allow educators to create an account.
R2	The CKB platform shall allow educators to log in.
R3	The CKB platform shall allow educators to create a new tournament.
R4	The CKB platform shall allow educators to set the minimum and maximum number of students per group for a tournament.
R5	The CKB platform shall allow educators to upload a code kata battle.
R6	The CKB platform shall allow educators to grant permissions to other educators to create battles within a specific tournament.
R7	The CKB platform shall enable educators to include a battle to a specific tournament.
R8	The CKB platform shall allow educators to include a description for a battle.
R9	The CKB platform shall allow educators to include a software project with test cases.
R10	The CKB platform shall allow educators to set a registration deadline for a battle within a tournament.
R11	The CKB platform shall allow educators to set a final submission deadline for a battle within a tournament.
R12	The CKB platform shall allow educators to set additional configurations for scoring, including functional aspects and quality level criteria.
R13	The CKB platform shall allow educators to close a tournament.
R14	The CKB platform shall allow an educator to define optional manual evaluation criteria for score assignment in battles.
R15	The CKB platform shall allow educators to manually evaluate and assign scores to teams.
R16	The CKB platform shall allow educators and students to visualize the gamification badges.
R17	The CKB platform shall allow educators to define new badges for gamification.
R18	The CKB platform shall allow educators to define new rules associated with the badges.
R19	The CKB platform shall allow educators to define new variables associated with the badges.
R20	The CKB platform shall allow all students and educators to see the ranking of each ongoing tournament with the score of each student subscribed.

Table 3.1: Educator Requirements.

Students

ID	Description
R21	The CKB platform shall not allow students to participate a tournament after the registration deadline.
R22	The CKB platform shall not allow students to participate a battle after the registration deadline.
R23	The CKB platform shall allow students to subscribe to a tournament within a specified deadline.
R24	The CKB platform shall allow students to create teams for a specific battle within a tournament.
R25	The CKB platform shall allow students to join teams for a specific battle within a tournament.
R26	The CKB platform shall allow students to invite other participants to the same group.
R27	The CKB platform shall allow students to accept an invitation.
R28	The CKB platform shall allow students to reject an invitation.
R29	The CKB platform shall allow students to join a battle without a team.

Table 3.2: Student Requirements.

Platform

ID	Description
R30	The CKB platform shall notify all subscribed students of a new battle and its details within a specific tournament.
R31	The CKB platform shall notify all subscribed students of a new tournament and its details.
R32	The CKB platform shall create a GitHub repository for each battle.
R33	send a link to the Github repository associated to a battle to all members of subscribed teams upon expiration of the registration deadline.

R34	The CKB platform shall be able to be informed of new students' commits by Github Actions workflows.
R35	The CKB platform shall be able to pull the latest sources from the forks of the Github repository provided.
R36	The CKB platform shall be able to run the testcases on the code uploaded by students and determine if the code is a valid solution for the exercise.
R37	The CKB platform shall inform students of the mandatory automated evaluation criteria, including functional aspects, timeliness, and source code quality.
R38	The CKB platform shall automatically update the battle score of a team based on GitHub commits and test results.
R39	The CKB platform shall automatically close a finished battle.
R40	The CKB platform shall assign or update battle scores to each team of the battle.
R41	The CKB platform shall calculate and update the personal tournament score of each student based on their performance in battles.
R42	The CKB platform shall be able to create a ranking of teams for every tournament.
R43	The CKB platform shall keep track of time elapsed from the start of a CKB and the final submissions of each team.
R44	The CKB platform shall be able to use static analysis tools to evaluate the quality of the code submitted by teams in terms of security, reliability, maintainability and other aspects defined by the educator who created the battle.
R45	The CKB platform shall notify all students involved in a tournament when it is closed and the final ranking is available.
R46	The CKB platform shall visualize ongoing tournaments and their ranks for all users.
R47	The CKB platform shall display collected badges on the profile of both students and educators.

Table 3.3: Platform Requirements.

3.2.2. Mapping on goals

In the following section it is shown how the relation $R \wedge D \models G$ holds. In particular, at first it is shown a traceability matrix that associates domain assumptions and requirements to each goal. After that, to facilitate reading, the section reports the text of all the

assumptions and all the requirements related to each goal.

Goal	Domain assumptions	Requirements
G1	D2,D4,D6,D10	R1,R2,R3,R4,R10,R13
G2	D1,D2	R1,R2,R5,R6,R7,R8,R11
G3	D2,D5	R32,R33,R34,R35,R38
G4	D2	R24,R25,R26,R27,R28,R29
G5	D2	R30,R31,R45,R46
G6	D1,D2,D7	R9,R34,R35,R36,R37,R38,R41,R44
G7	D1,D2,D3	R14,R15
G8	D2	R16,R20,R46,R47
G9	D2,D8	R17,R18,R19

Table 3.4: Mapping on goals.

In this section, it will be shown the functional requirements and the domain assumption related to each goal.

- **[G.1] The platform should allow educators to set up tournaments.**
  - **[R.1]** The `CKB` platform shall allow educators to create an account.
  - **[R.2]** The `CKB` platform shall allow educators to log in.
  - **[R.3]** The `CKB` platform shall allow educators to create a new tournament.
  - **[R.4]** The `CKB` platform shall allow educators to set the minimum and maximum number of students per group for a tournament.
  - **[R.10]** The `CKB` platform shall allow educators to set a registration deadline for a battle within a tournament.
  - **[R.13]** The `CKB` platform shall allow educators to close a tournament.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.
  - **[D.4]** Educators managing a torunament will not lose access to their institutional email during the whole duration of the tournament itself.

  - **[D.6]** Educators are capable of creating and closing tournaments, without leaving them open undefinetly when no more battles are scheduled or in session.
  - **[D.10]** Educators and students are linked to one and only one institutional email.

- **[G.2] The platform should allow educators to set up code kata battles with configurable parameters.**
  - **[R.1]** The CKB platform shall allow educators to create an account.
  - **[R.2]** The CKB platform shall allow educators to log in.
  - **[R.5]** The CKB platform shall allow educators to upload a code kata battle.
  - **[R.6]** The CKB platform shall allow educators to grant permissions to other educators to create battles within a specific tournament.
  - **[R.7]** The CKB platform shall enable educators to include a battle to a specific tournament.
  - **[R.8]** The CKB platform shall allow educators to include a description for a battle.
  - **[R.11]** The CKB platform shall allow educators to set a final submission deadline for a battle within a tournament.
  - **[D.1]** Educators using CKB platform have the necessary technical knowledge and skills to create and manage code kata battles. This includes the ability to create programming exercises, write test cases, and set up build automation scripts.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.

- **[G.3] The platform should allow third party platforms integration.**
  - **[R.32]** The CKB platform shall create a GitHub repository for each battle.
  - **[R.33]** send a link to the Github repository associated to a battle to all members of subscribed teams upon expiration of the registration deadline.
  - **[R.34]** The CKB platform shall be able to be informed of new students' commits by Github Actions workflows.

  - **[R.35]** The CKB platform shall be able to pull the latest sources from the forks of the Github repository provided.
  - **[R.38]** The CKB platform shall automatically update the battle score of a team based on GitHub commits and test results.
  - **[D.1]** Educators using CKB platform have the necessary technical knowledge and skills to create and manage code kata battles. This includes the ability to create programming exercises, write test cases, and set up build automation scripts.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.
  - **[D.5]** Educators and students have a good understanding of GitHub and GitHub actions.

- **[G.4] The platform should allow students to join battles individually or form teams within specified size limits.**
  - **[R.24]** The CKB platform shall allow students to create teams for a specific battle within a tournament.
  - **[R.25]** The CKB platform shall allow students to join teams for a specific battle within a tournament.
  - **[R.26]** The CKB platform shall allow students to invite other participants to the same group.
  - **[R.27]** The CKB platform shall allow students to accept an invitation.
  - **[R.28]** The CKB platform shall allow students to reject an invitation.
  - **[R.29]** The CKB platform shall allow students to join a battle without a team.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.

- **[G.5] The platform should allow users to see real time updates on current tournaments and battles.**
  - **[R.30]** The CKB platform shall notify all subscribed students of a new battle and its details within a specific tournament.

  - **[R.31]** The CKB platform shall notify all subscribed students of a new tournament and its details.
  - **[R.45]** The CKB platform shall notify all students involved in a tournament when it is closed and the final ranking is available.
  - **[R.46]** The CKB platform shall visualize ongoing tournaments and their ranks for all users.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.

- **[G.6] The platform should have automated evaluations of code submissions.**
  - **[R.9]** The CKB platform shall allow educators to include a software project with test cases.
  - **[R.34]** The CKB platform shall be able to be informed of new students' commits by Github Actions workflows.
  - **[R.35]** The CKB platform shall be able to pull the latest sources from the forks of the Github repository provided.
  - **[R.36]** The CKB platform shall be able to run the testcases on the code uploaded by students and determine if the code is a valid solution for the exercise.
  - **[R.37]** The CKB platform shall inform students of the mandatory automated evaluation criteria, including functional aspects, timeliness, and source code quality.
  - **[R.38]** The CKB platform shall automatically update the battle score of a team based on GitHub commits and test results.
  - **[R.41]** The CKB platform shall calculate and update the personal tournament score of each student based on their performance in battles.
  - **[R.44]** The CKB platform shall be able to use static analysis tools to evaluate the quality of the code submitted by teams in terms of security, reliability, maintainability and other aspects defined by the educator who created the battle.
  - **[D.1]** Educators using CKB platform have the necessary technical knowledge and skills to create and manage code kata battles. This includes the ability to

create programming exercises, write test cases, and set up build automation scripts.

  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.
  - **[D.7]** The scoring system is transparent, consistent, and coherent with the criteria of the rules (quality of sources, timeliness, ...)

- **[G.7] The platform should allow educators to manually evaluate and assign scores for optional factors at the end of each battle.**
  - **[R.14]** The `CKB` platform shall allow an educator to define optional manual evaluation criteria for score assignment in battles.
  - **[R.15]** The `CKB` platform shall allow educators to manually evaluate and assign scores to teams.
  - **[D.1]** Educators using `CKB` platform have the necessary technical knowledge and skills to create and manage code kata battles. This includes the ability to create programming exercises, write test cases, and set up build automation scripts.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.
  - **[D.3]** Educators are expected to have the ability to evaluate the work done by students and assign scores if manual evaluation is required.

- **[G.8] The platform should allow users to visualize informations about another user.**
  - **[R.16]** The `CKB` platform shall allow educators and students to visualize the gamification badges.
  - **[R.20]** The `CKB` platform shall allow all students and educators to see the ranking of each ongoing tournament with the score of each student subscribed.
  - **[R.46]** The `CKB` platform shall visualize ongoing tournaments and their ranks for all users.
  - **[R.47]** The `CKB` platform shall display collected badges on the profile of both students and educators.

  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.

- **[G.9] The platform should allow educators to create new gamification badges.**
  - **[R.17]** The CKB platform shall allow educators to define new badges for gamification.
  - **[R.18]** The CKB platform shall allow educators to define new rules associated with the badges.
  - **[R.19]** The CKB platform shall allow educators to define new variables associated with the badges.
  - **[D.2]** All users of the CKB platform, both educators and students, are assumed to have access to a stable internet connection. This is necessary fir accessing the platform, downloading code kata, submitting code, and receiving notification.
  - **[D.8]** Badge rules and varibales are well-defined and understood by the platform. They are meaningfull with respect to the tournament associated with.

3.3. Use Case Diagrams

3.3.1. Guest

Figure 3.1: Guest Diagram

3.3.2. Student

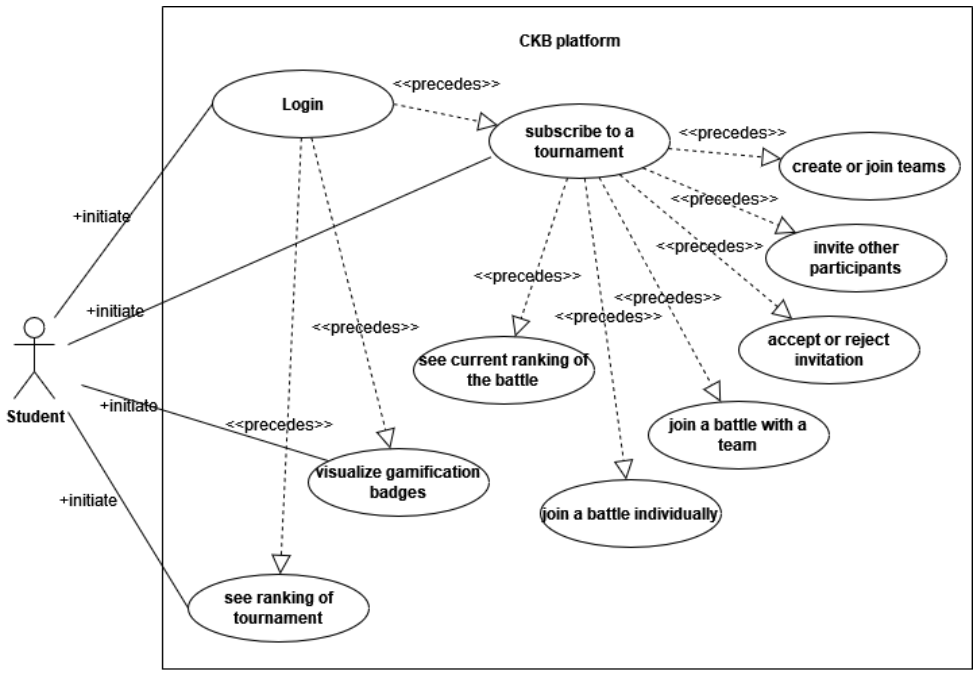

Figure 3.2: Student Diagram

3.3.3. Educator

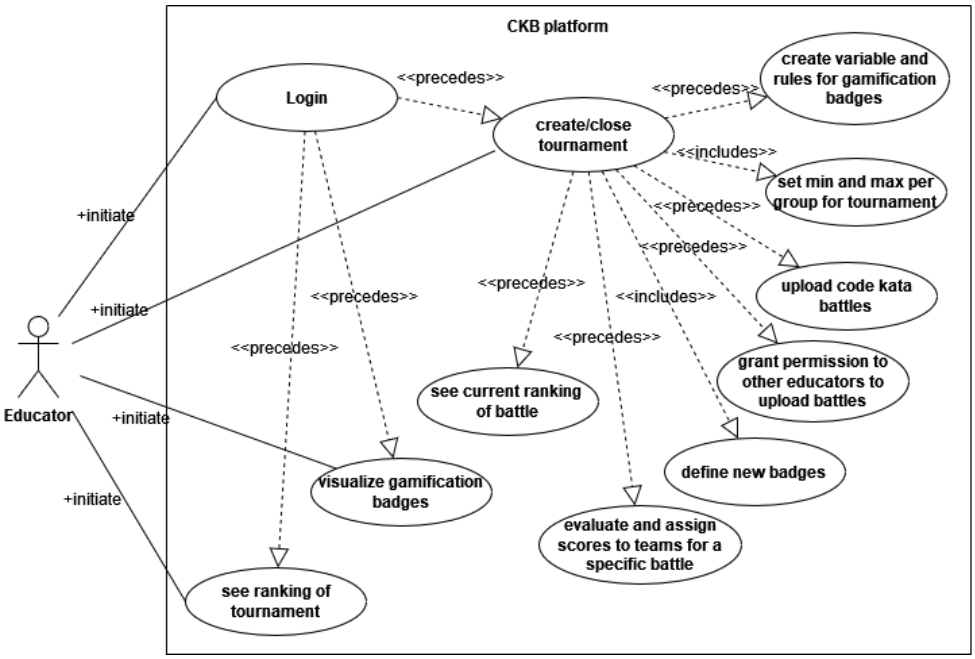

Figure 3.3: Educator Diagram

3.4. Use Cases

In this section, the primary identified use cases are elucidated. Each use case is accompanied by a table delineating entry conditions, event flow, exit conditions, and exceptions. Additionally, a sequence diagram is provided to illustrate the interactions between entities and the functions invoked. This comprehensive representation aims to capture the essential aspects of each use case, ensuring a clear understanding of the system dynamics within the context of the Code Kata Battle (CKB) platform.

UC1. Student Registration

Actor	Unregistered Student
Entry conditions	The student isn't registered on the CKB platform and clicks the sign-up button
Event Flow	1. The CKB platform prompts the unregistered student to input personal information (name, surname, email linked to an institutional profile). 2. The unregistered student fills the form with personal information 3. The student agrees to the platform's "Terms & Conditions," and "Privacy Policy." 4. The CKB platform validates the provided student information. 5. The CKB platform requests the student to input a username for their profile. 6. The student inputs a username. 7. The CKB platform validates the username. 8. The CKB platform sends a verification email containing a 6-digit code to the student's provided institutional email address. 9. The CKB platform prompts the student to input the verification code. 10. The student inputs the verification code. 11. The CKB platform communicates the outcome of the student's registration.
Exit condition	An account is created.
Exceptions	3.1. The student does not agree to the platform's "Terms & Conditions," and "Privacy Policy." The CKB platform shows a message asking the user to agree to such "Terms & Conditions" stopping the sign up operation.

4.1. The CKB platform is unable to validate the student's personal information. In these cases, the unregistered student receives a notification with an error message. 7.1. The student's provided username is already in use. The CKB platform shows a message asking the user to choose another username. 7.2. The student's provided username is in a non acceptable format The CKB platform shows a message asking the user to choos another username 10.1. The student inputs an incorrect verification code. The CKB platform shows a message asking the user to input the correct verification code.

Table 3.5: Student Registration Use Case.

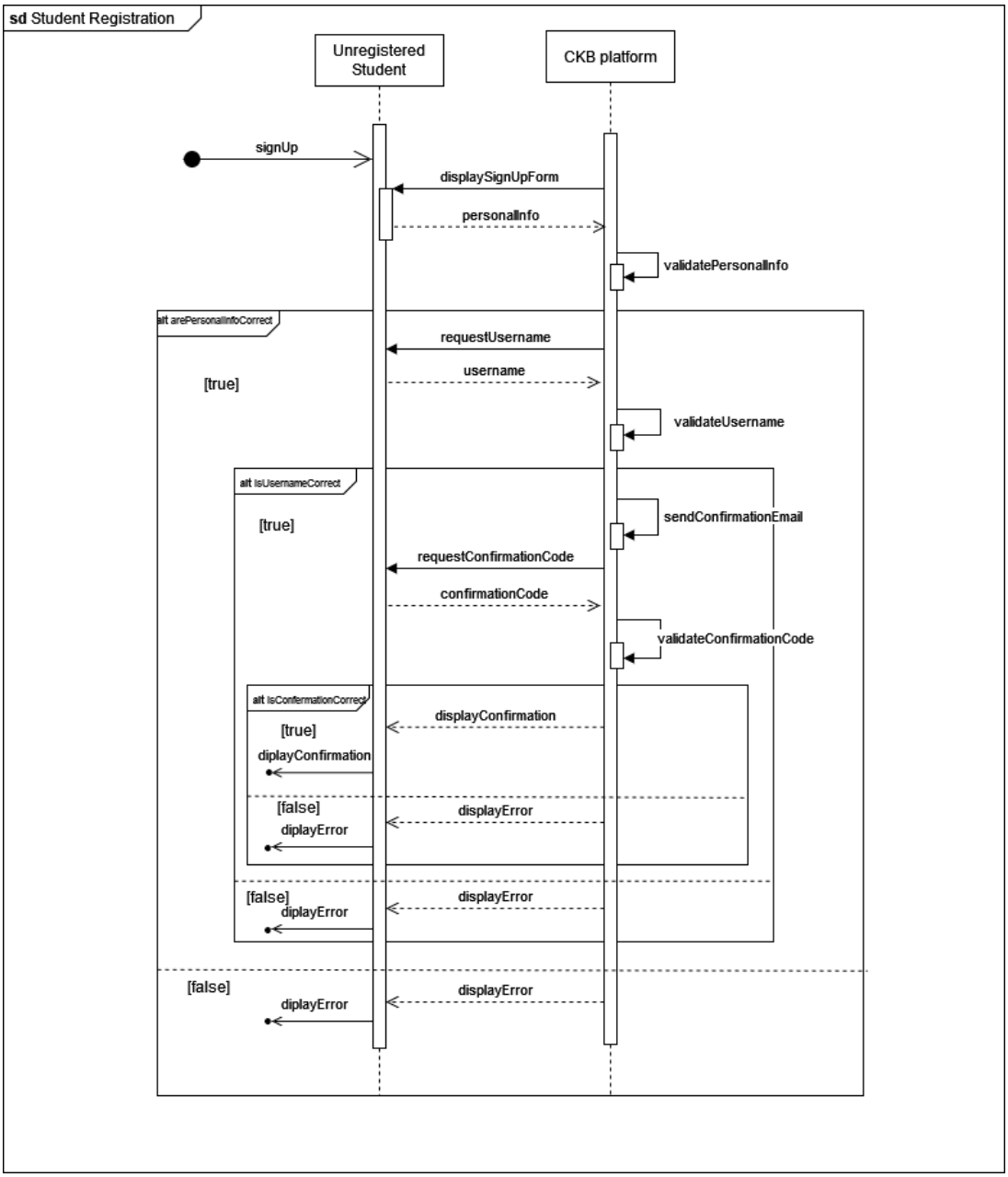

Figure 3.4: Student Registration Sequence Diagram

UC2. User Login

Actor	Registered User
Entry conditions	The User is registered on the `CKB` platform and clicks the login button.
Event Flow	1. The `CKB` platform prompts the user to input institutional email.

	2. The CKB platform validates the email redirecting the user to the institution's page. 3. The CKB platform communicates the outcome of the user's login.
Exit condition	Login completed.
Exceptions	2.1. The user's email isn't valid. 2.2. The user's verification on the institution's page goes wrong. In these cases, the user receives a notification with an error message.

Table 3.6: User Login Use Case.

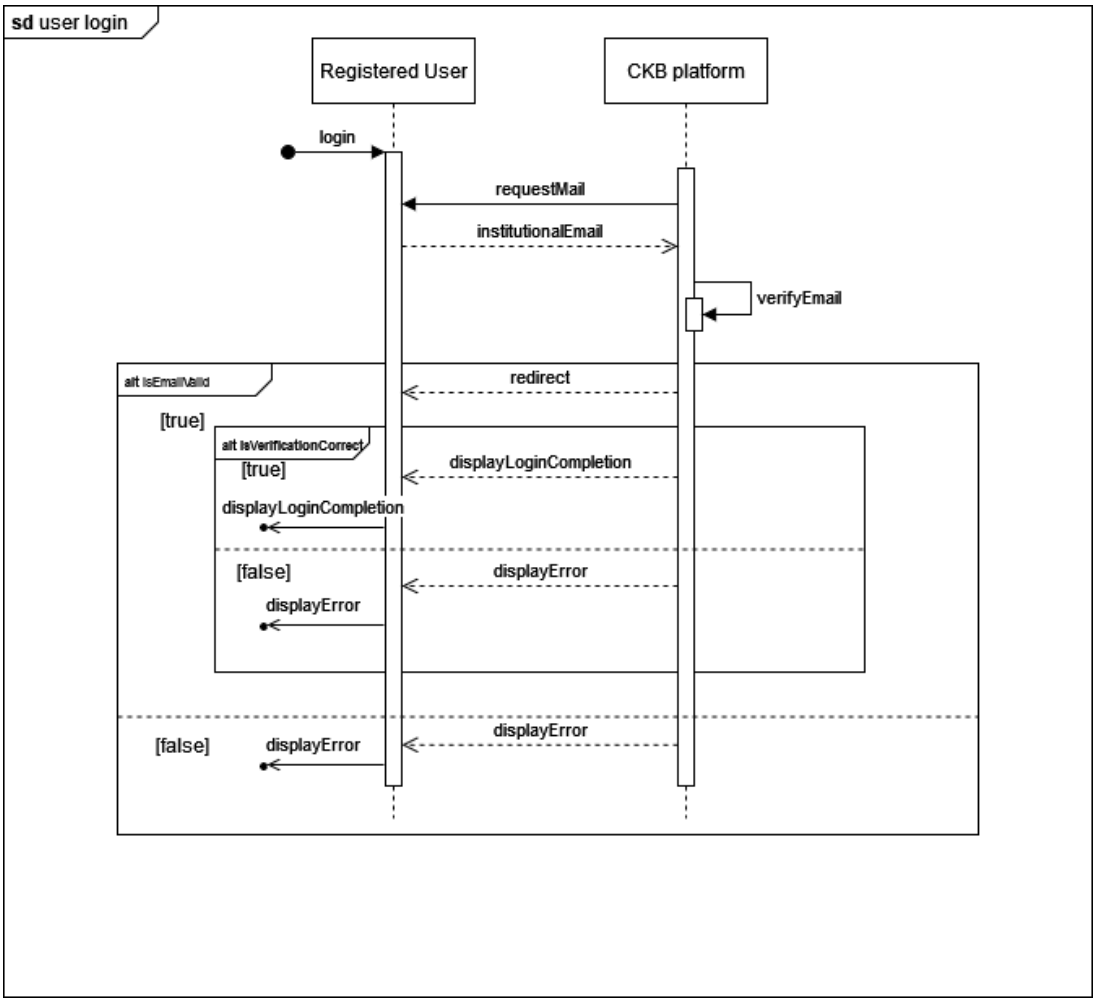

Figure 3.5: User Login Sequence Diagram

UC3. Student Subscription to a Tournament

Actor	Registered Student
Entry conditions	The Student is logged in on the CKB platform and clicks to the subscription button to a tournament.
Event Flow	1. The CKB platform receives the request. 2. The CKB platform checks if the tournament's subscription deadline is over. 3. The CKB platform communicates the outcome of the student's subscription.
Exit condition	Subscription completed.
Exceptions	2.1. The tournament's subscription deadline is over. In these cases, the student receives a notification with an error message.

Table 3.7: Student Subscription Use Case.

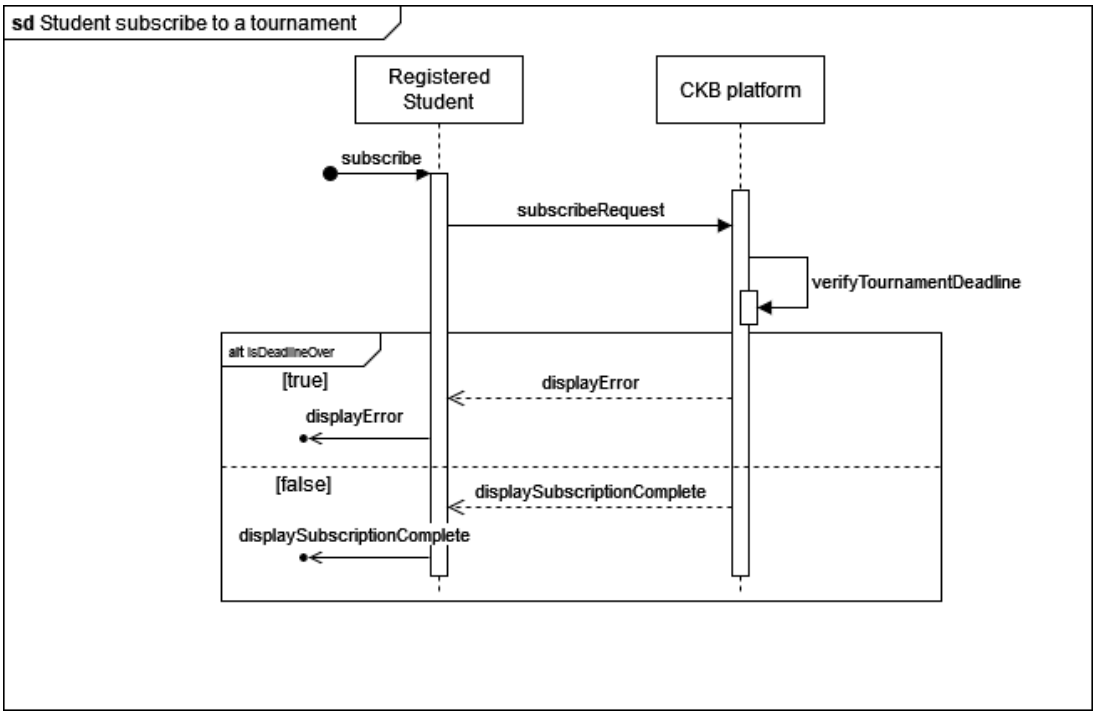

Figure 3.6: Student Subscription Sequence Diagram

UC4. Student creates a team

Actor	Student
Entry conditions	The student is logged in on the CKB platform.
Event Flow	1. The student clicks on the "Create team" button from the tournament page. 2. The CKB platform prompts the student to input the size of the team. 3. The student inputs the size of the team. 4. The CKB platform prompts the student to input the name of the team. 5. The student inputs the name of the team. 6. The CKB platform checks the name of the team. 7. The CKB platform communicates the outcome of the team creation.
Exit condition	A team is created.
Exceptions	2.1. The team size is invalid. 4.1. The team name already exists or it is invalid. In these cases, the student receives a notification with an error message. 2-6.1. The student cancels team creation. In this case, the CKB platform will roll back any action performed and the student receives a notification with an update message.

Table 3.8: Team Creation Use Case.

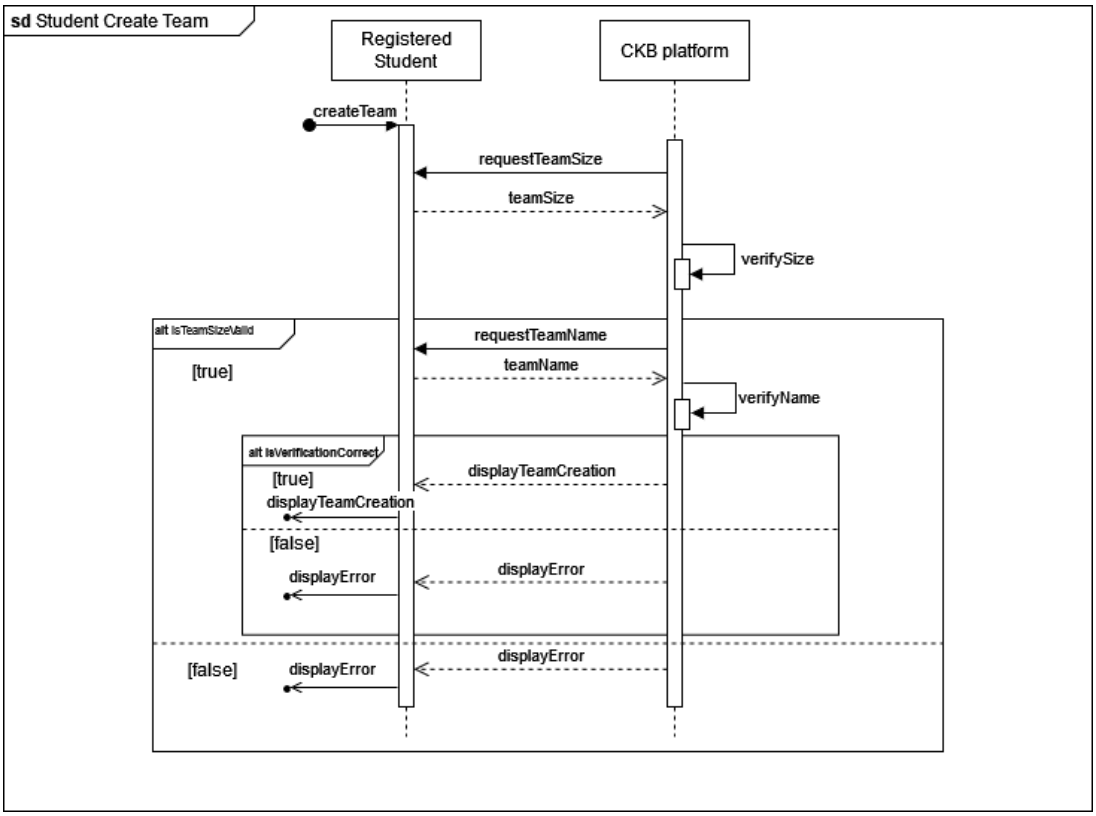

Figure 3.7: Team Creation Sequence Diagram

UC5. Student invites another student to the team

Actor	Student
Entry conditions	The student is logged in on the CKB platform and has already created a team.
Event Flow	1. The student selects a student from the list of students on the tournament page. 2. The student clicks on the invite button on the bottom of the list of students from the tournament page. 3. The CKB platform checks for remaining space in the team. 4. The CKB platform sends the invitation to the selected student. 5. The CKB platform communicates the outcome of the invitation to the student.
Exit condition	An invitation is sent.
Exceptions	3.1. The team size is full. 3.2. The student selected is already in another group.

	In these cases, the CKB platform will notify the student with an error message.

Table 3.9: Invitation Use Case.

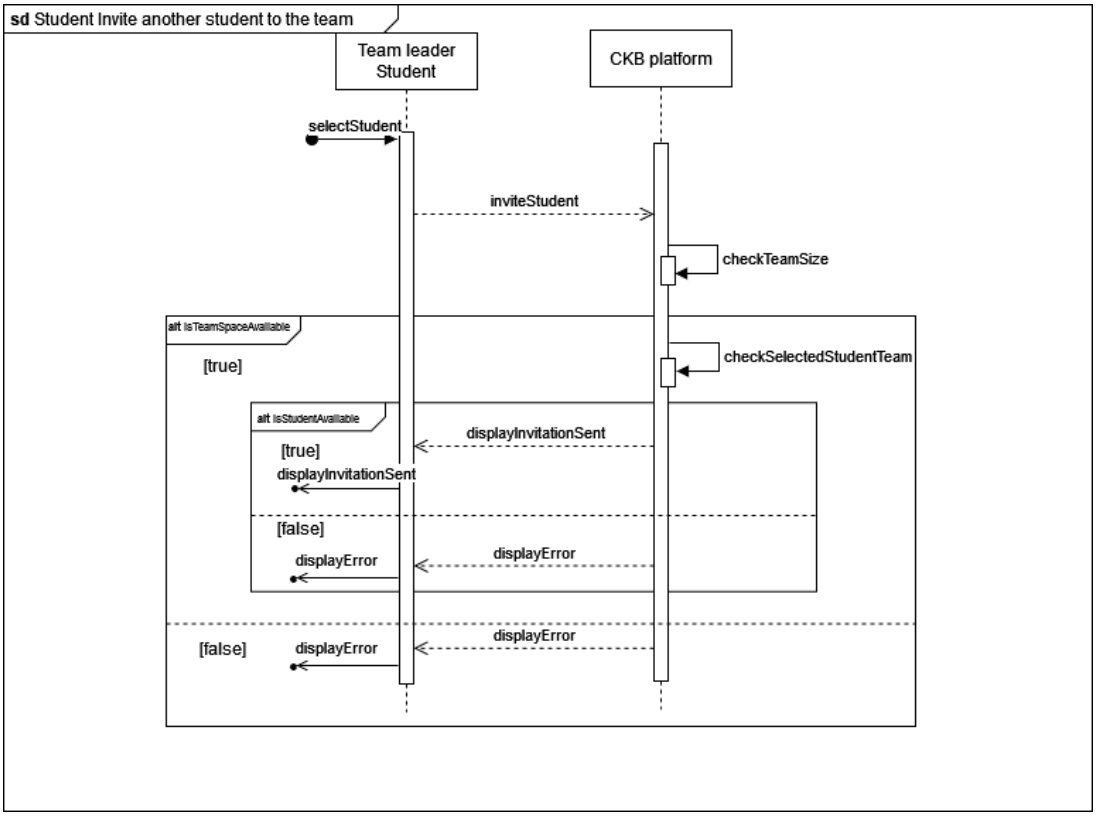

Figure 3.8: Invitation Sequence Diagram

UC6. Student accept an invitation to join a CKB team

Actor	Student
Entry conditions	The student is logged in on the CKB platform and receives and invitation to join a CKB team.
Event Flow	1. The student receives a pop-up norification informing of the invitation. 2. The student clicks on the "Accept" button on the invitation pop-up. 3. The CKB platform sends the information to the invitation sender.

	4. The CKB platform communicates the outcome of the invitation to both students.
Exit condition	The two students are now part of the same team.
Exceptions	3.1. The team is already dissembled. In this case, the CKB platform will roll back any action performed and the student receives a notification with an update message.

Table 3.10: Invitation Accepted Use Case.

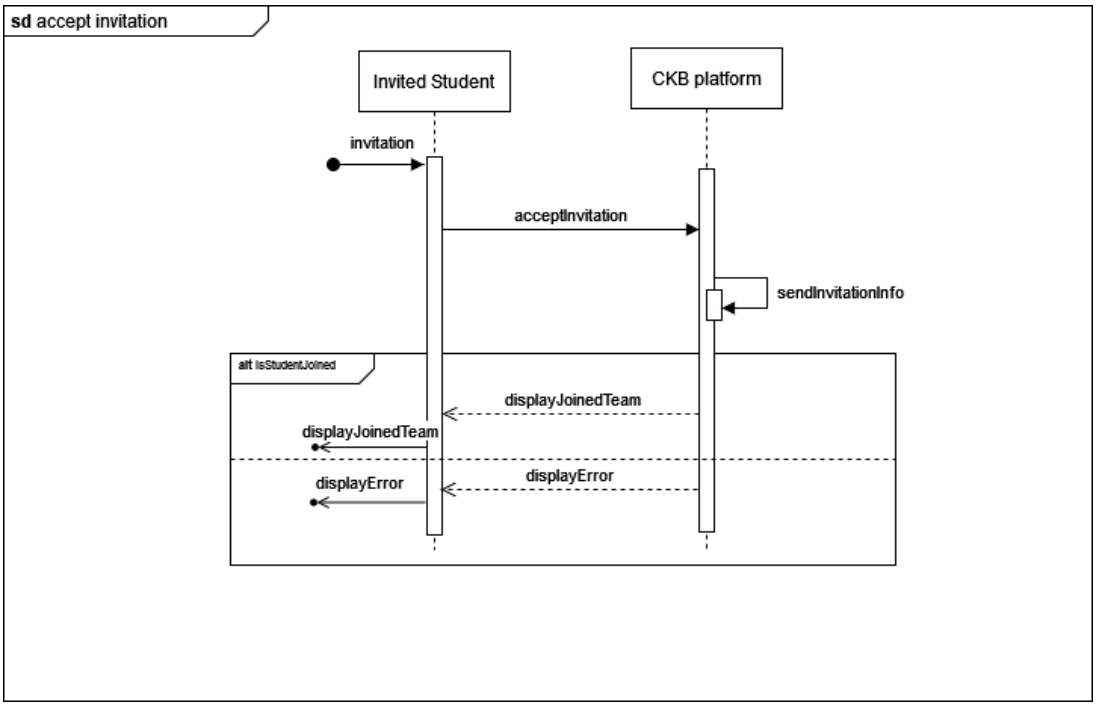

Figure 3.9: Invitation Accepted Sequence Diagram

UC7. Student rejects an invitation

Actor	Student
Entry conditions	The student is logged in on the CKB platform and receives and invitation to join a CKB team.
Event Flow	1. The student receives a pop-up norification of the invitation. 2. The student clicks on the "Reject" button of the invitation pop-up. 3. The CKB platform sends the information to the invitation sender.

	4. The CKB platform communicates the outcome of the invitation to both students.
Exit condition	The students are not part of the same team.

Table 3.11: Invitation Rejected Use Case.

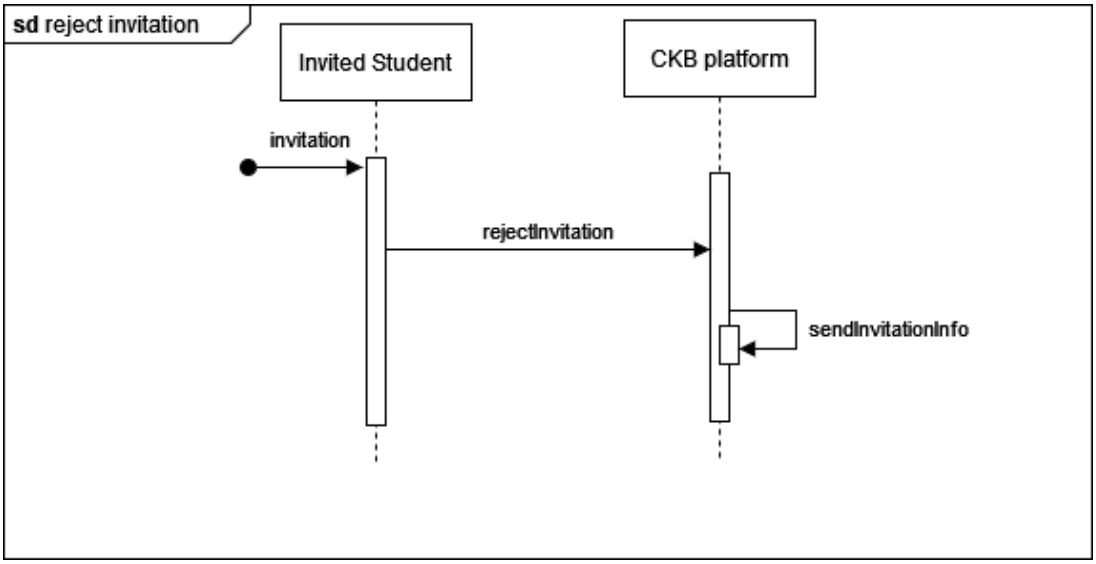

Figure 3.10: Invitation Rejected Sequence Diagram

UC8. Student joins a battle without a team

Actor	Student
Entry conditions	The Student is logged in on the CKB platform and is registered to a tournament.
Event Flow	1. The student selects a CKB from the list of battles in the Tournament page. 2. The student clicks the "Join battle" button on the bottom of the list of battles in the tournament page. 3. The CKB platform verifies the registration deadline for the battle. 4. The CKB platform communicates the outcome of the student's action.
Exit condition	The student joined the battle.
Exceptions	3.1. The battle's registration deadline is over. In this case, the student receives a notification with an error message.

Table 3.12: Student Join battle Use Case.

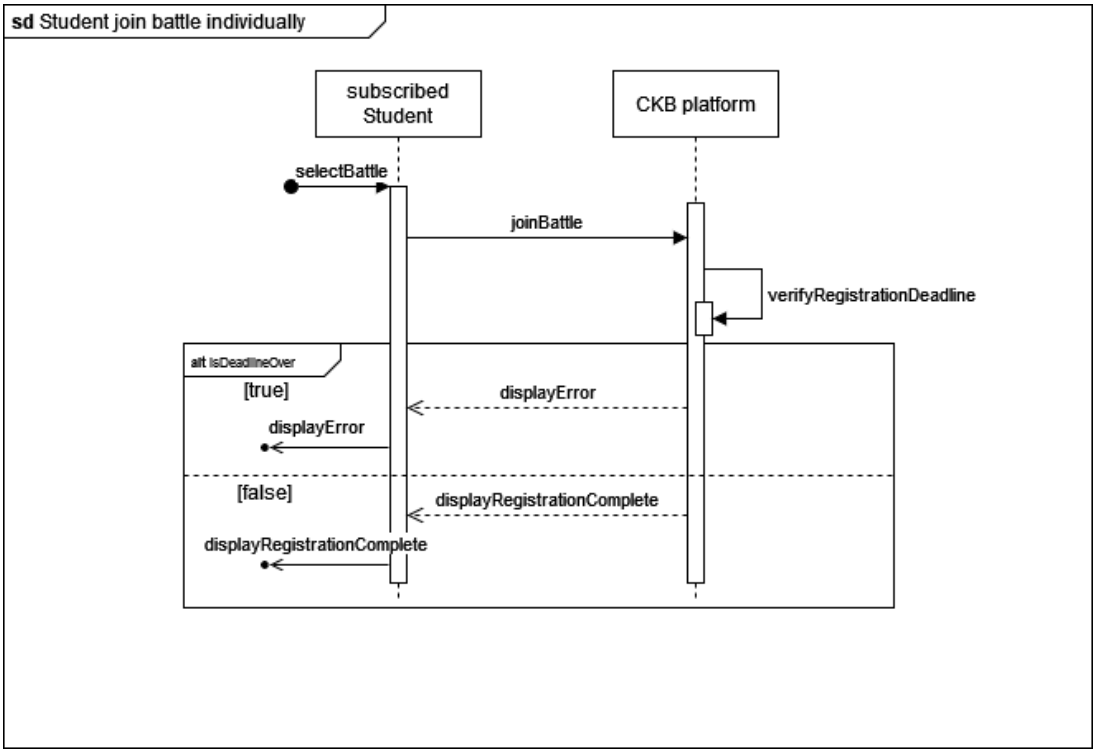

Figure 3.11: Student Join battle Sequence Diagram

UC9. Student joins a battle with a team

Actor	Team of students
Entry conditions	The team is registered on the `CKB` platform in a tournament's battle.
Event Flow	1. The team leader selects a CKB from the list of battles in the Tournament page. 2. The student clicks the "Join battle" button on the bottom of the list of battles in the tournament page. 3. The `CKB` platform verifies the registration deadline for the battle. 4. The `CKB` platform communicates the outcome of the student's action.
Exit condition	The team joined the battle.
Exceptions	2.1. The number of students in the team is not in the range of the battle's minimum and maximum number of students.

	In this case, the student receives a notification with an error message and the team does not join the battle. 3.1. The battle's registration deadline is over. In this case, the student receives a notification with an error message.

Table 3.13: Team Join Battle Use Case.

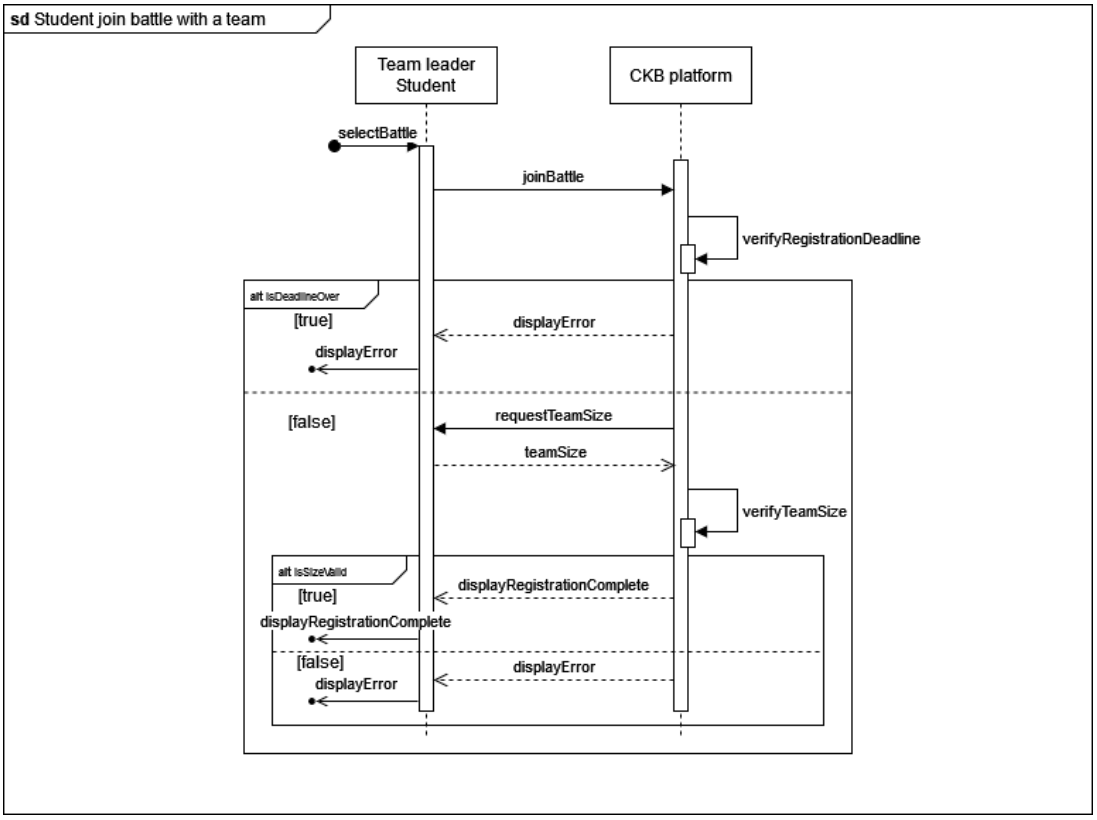

Figure 3.12: Team Join Battle Sequence Diagram

UC10. Educator creates a tournament

Actor	Educator
Entry conditions	The educator is logged in on the `CKB` platform.
Event Flow	1. The educator clicks on the "Create Tournament" button from the Menu in the homepage. 2. The `CKB` platform prompts the educator to input the tournament name.

	3. The educator inputs the tournament name. 4. The CKB platform prompts the educator to input the registration deadline. 5. The educator inputs the registration deadline. 6. The educator toggles the "Advanced Options" section. 7. The CKB platform prompts the educator to choose whether to include badges or not. 8. The educator chooses whether to include badges or not. 9. The CKB platform prompts the educator to choose which badges to include. 10. The educator chooses which badges to include. 11. The CKB platform prompts the educator to choose whether to create new badges or not. 12. The educator chooses to create new badges. 13. The CKB platform opens a pop-up window allowing the educator to create new badges by selecting and combining variables and rules. 14. The educator creates new badges. 15. The educator pushes the "Create Tournament" button. 16. The CKB platform communicates the outcome of the tournament creation. 17. The CKB platform notifies all students that a new tournament is available.
Exit condition	A tournament is created.
Exceptions	3.1. The educator does not input the tournament name. The CKB platform shows a message asking the user to input the tournament name. 5.1. The educator does not input the registration deadline or inputs an invalid date. The CKB platform shows a message asking the user to input the registration deadline. 8.1. The educator does not choose whether to include badges or not. The CKB platform assumes that badges are not included. 10.1. The educator does not choose which badges to include. The CKB platform assumes that no pre-existing badges are included. 12.1. The educator does not choose to create new badges. The CKB platform assumes that no new badges are created.

	14.1. The educator tries to create a new badge without selecting any variable or rule. The CKB platform shows a message asking the user to select at least one variable or rule.

Table 3.14: Tournament Creation Use Case.

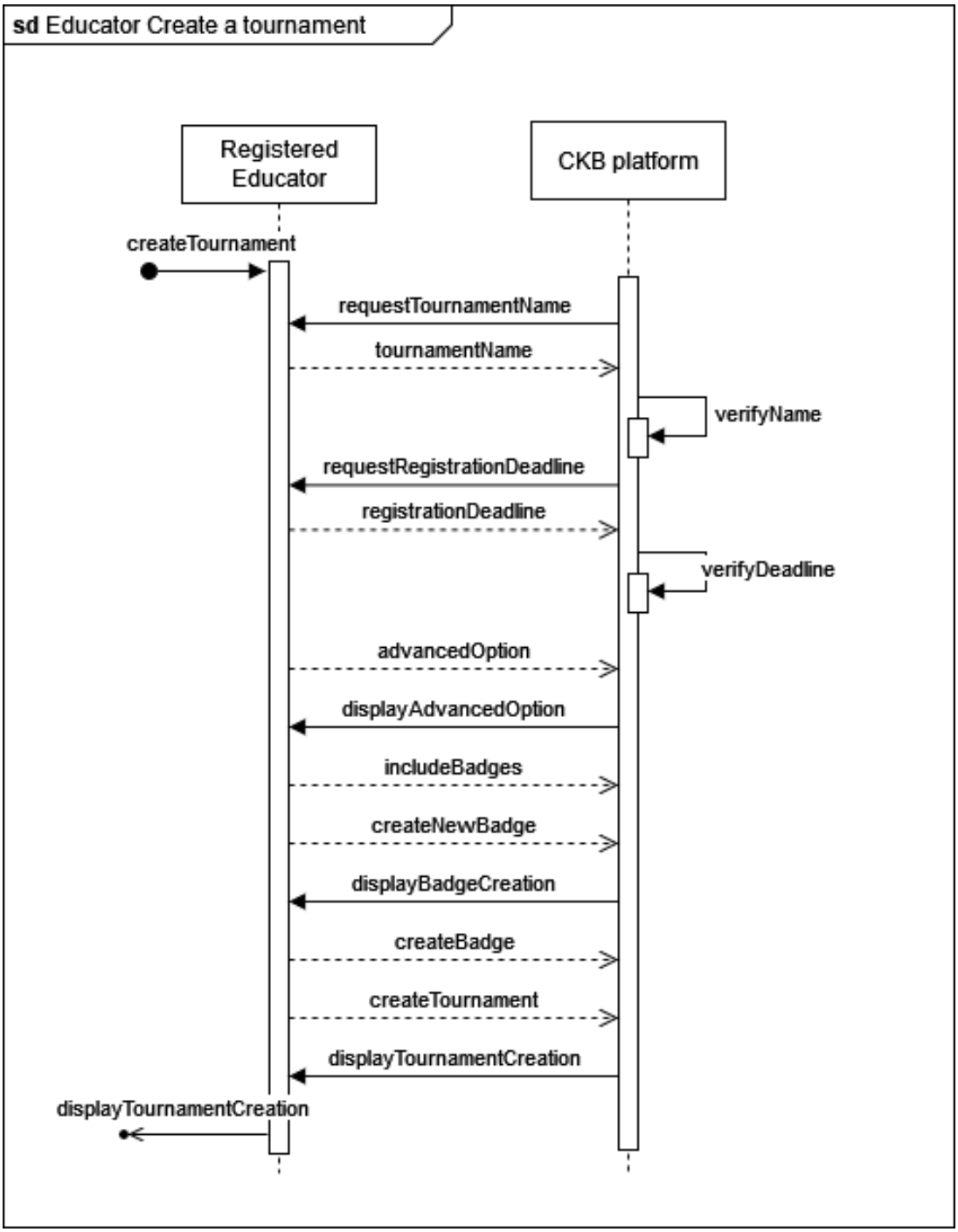

Figure 3.13: Tournament Creation Sequence Diagram

UC11. Educator closes a tournament

Actor	Educator
Entry conditions	The educator is logged in on the CKB platform and has permissions to perform actions on a specific tournament.
Event Flow	1. The educator clicks on a tournament for which he has editing permissions from the list in the "Tournaments" page 2. The CKB platform shows the tournament details page 3. The educator clicks on the "Close Tournament" button 4. The CKB platform shows a pop-up window asking for confirmation 5. The educator confirms the action 6. The CKB platform communicates the outcome of the tournament closing 7. The CKB computes the final ranking of the tournament and makes it available 8. The CKB platform notifies all students that the tournament is closed
Exit condition	The tournament is closed.
Exceptions	5.1. The educator does not confirm the action The CKB platform does not close the tournament.

Table 3.15: Tournament Closing Use Case.

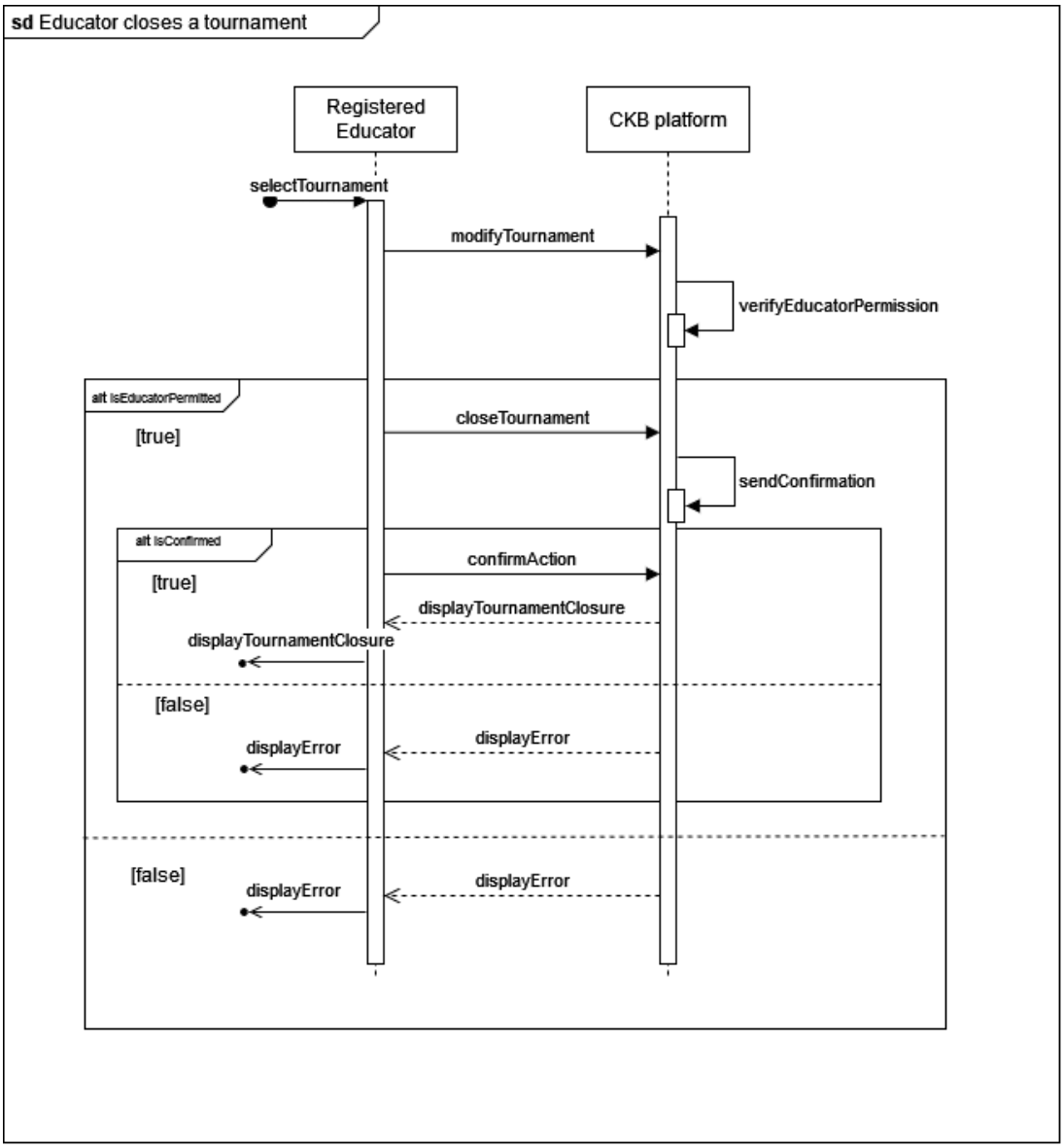

Figure 3.14: Tournament Closing Sequence Diagram

UC12. Educator creates a new Code Kata Battle

Actor	Educator
Entry conditions	The educator is logged in on the `CKB` platform and is on a tournament's details page. The educator has permissions to add new CKB to the tournament.
Event Flow	1. The educator clicks on the "Create New Code Kata Battle" button

	2. The CKB platform asks the educator to upload the code kata for the battle 3. The educator uploads the code kata 4. The CKB platform verifies that the code kata includes a description and a software project with test cases and build automation scripts 5. The CKB platform asks the educator to set the minimum and maximum number of students per group 6. The educator sets the minimum and maximum number of students per group 7. The CKB platform asks the educator to set a registration deadline 8. The educator sets a registration deadline 9. The CKB platform asks the educator to set a final submission deadline 10. The educator sets a final submission deadline 11. The educator clicks on the "Additional Configurations for Scoring" button 12. The CKB platform shows the "Additional Configurations for Scoring" section 13. The educator sets additional configurations for scoring if needed 14. The educator clicks on the "Create" button 15. The CKB platform communicates the outcome of the CKB creation 16. The CKB platform notifies all students subscribed to the tournament that a new CKB is available
Exit condition	The new CKB is added to the tournament.
Exceptions	4.1. The code kata does not include a description The CKB platform shows a message asking the educator to upload a code kata including a description. 4.2. The code kata does not include a software project The CKB platform shows a message asking the educator to upload a code kata including a software project. 4.3. The code kata does not include test cases The CKB platform shows a message asking the educator to upload a code kata including test cases. 4.4. The code kata does not include build automation scripts

	The CKB platform shows a message asking the educator to upload a code kata including build automation scripts. 5.1. The educator does not set the minimum and maximum number of students per group The CKB platform shows a message asking the educator to set the minimum and maximum number of students per group. 7.1. The educator does not set a registration deadline or sets an invalid date The CKB platform shows a message asking the educator to set a registration deadline. 9.1. The educator does not set a final submission deadline or sets an invalid date The CKB platform shows a message asking the educator to set a final submission deadline.

Table 3.16: New CKB creation Use Case.

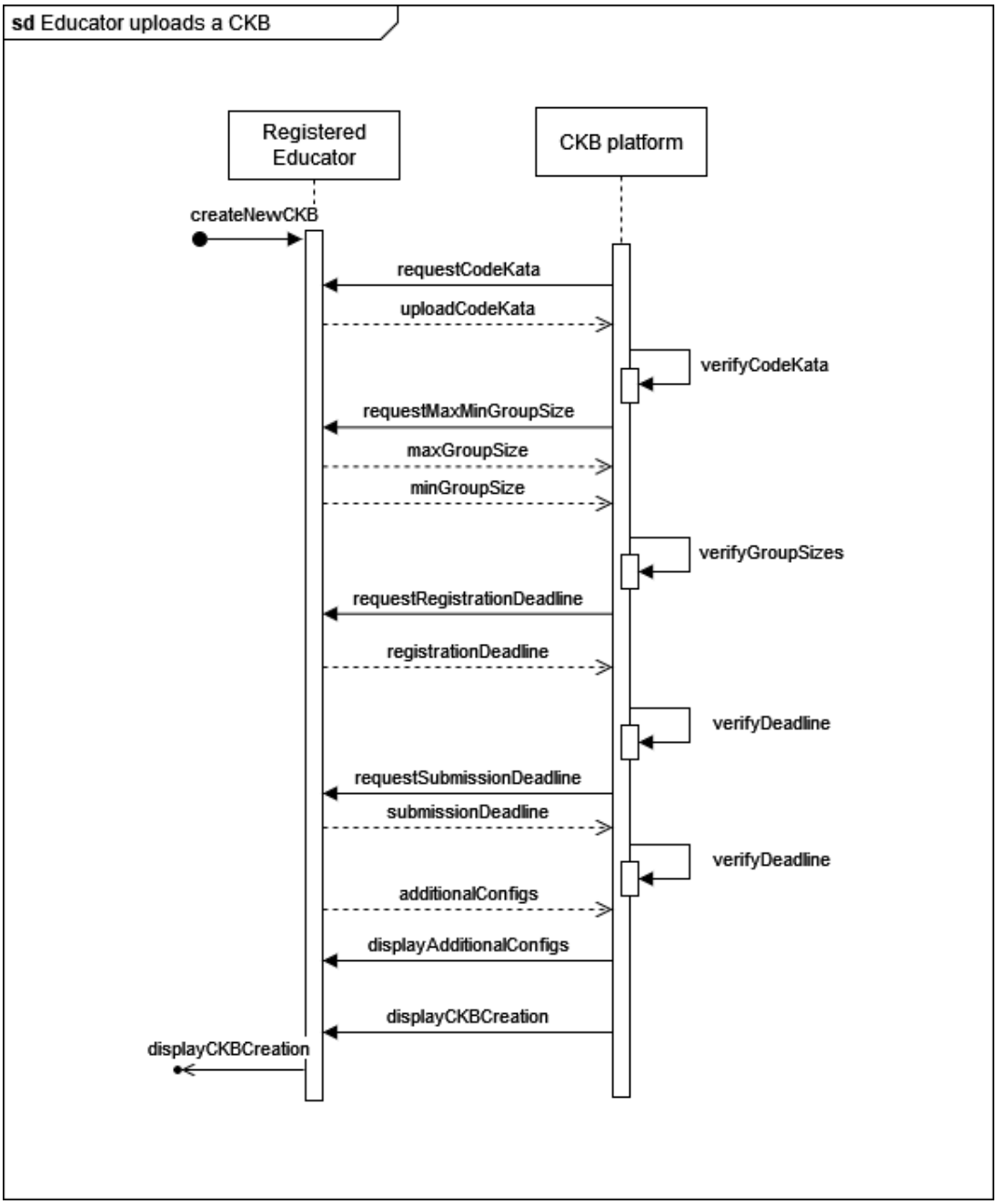

Figure 3.15: New CKB creation Sequence Diagram

UC13. Educator grants another educator permissions to create battles in a tournament

Actor	Educator
Entry conditions	The educator is logged in on the CKB platform and is on a tournament's details page. The educator is the creator of the tournament.
Event Flow	1. The educator clicks on the "Add Administrator" button 2. The CKB platform asks the educator to input the email or username of the educator to whom grant permissions 3. The educator inputs the email or username of the educator to whom grant permissions 4. The CKB platform communicates the outcome of the operation 5. The CKB platform notifies the educator to whom permissions have been granted
Exit condition	The educator to whom permissions have been granted can create battles in the tournament.
Exceptions	3.1. The educator inputs an invalid email or username or one belonging to a user who is not an educator The CKB platform shows a message asking the educator to input a valid email or username.

Table 3.17: Grant permission Use Case.

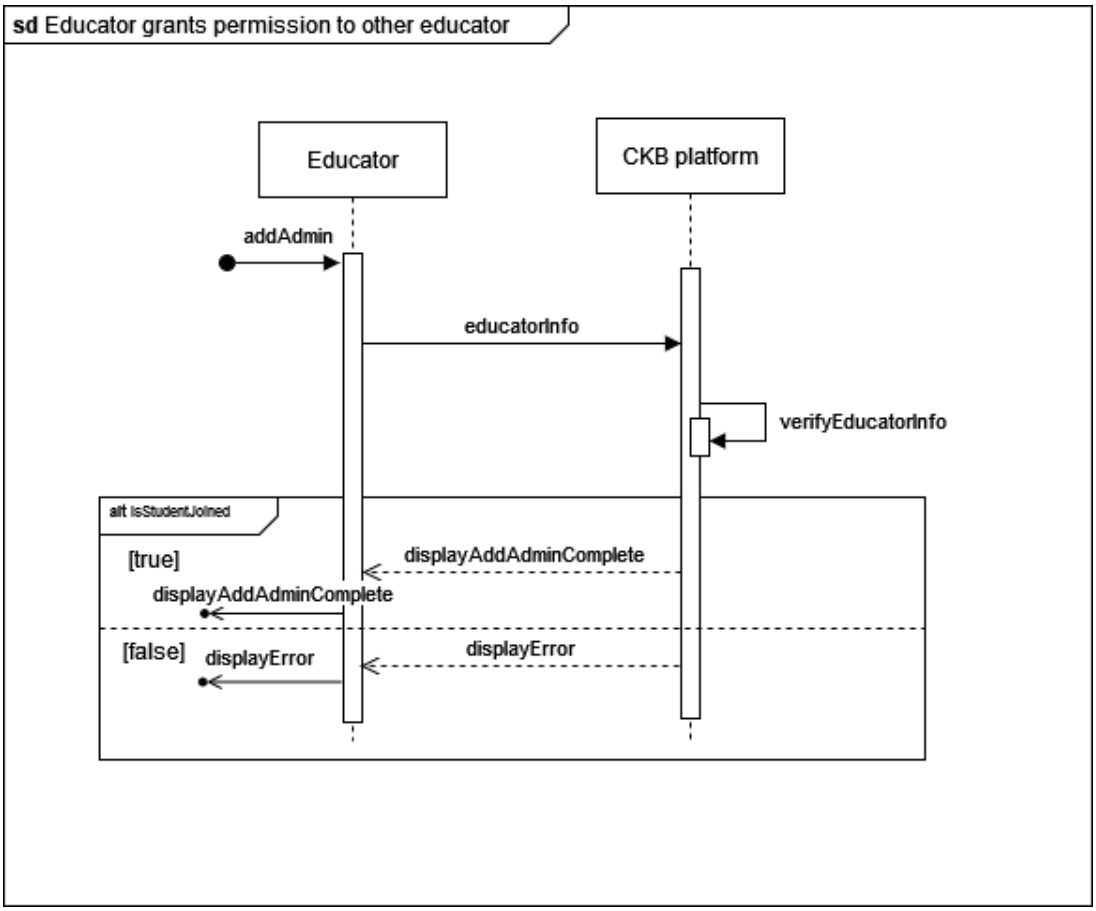

Figure 3.16: Grant permission Sequence Diagram

UC14. Educator creates a new badge

Actor	Educator
Entry conditions	The educator is logged in on the CKB platform and is on a tournament's details page. The educator is creating a tournament.
Event Flow	1. The educator clicks on the "Create New Badge" button in "Advanced Options" 2. The CKB platform prompts the educator to input the title of the badge 3. The educator inputs the title of the badge 4. The CKB platform allows the educator to create new variables and rules for the badge 5. The educator defines the rules of the badge through a wizard 6. The educator clicks on the "Create" button

	7. The CKB platform communicates the outcome of the badge creation
Exit condition	The new badge is added to the tournament.
Exceptions	3.1. The educator does not input the title of the badge The CKB platform shows a message asking the educator to input the title of the badge. 5.1. The educator does not define any rule for the badge The CKB platform shows a message asking the educator to define at least one rule for the badge.

Table 3.18: New Badge creation Use Case.

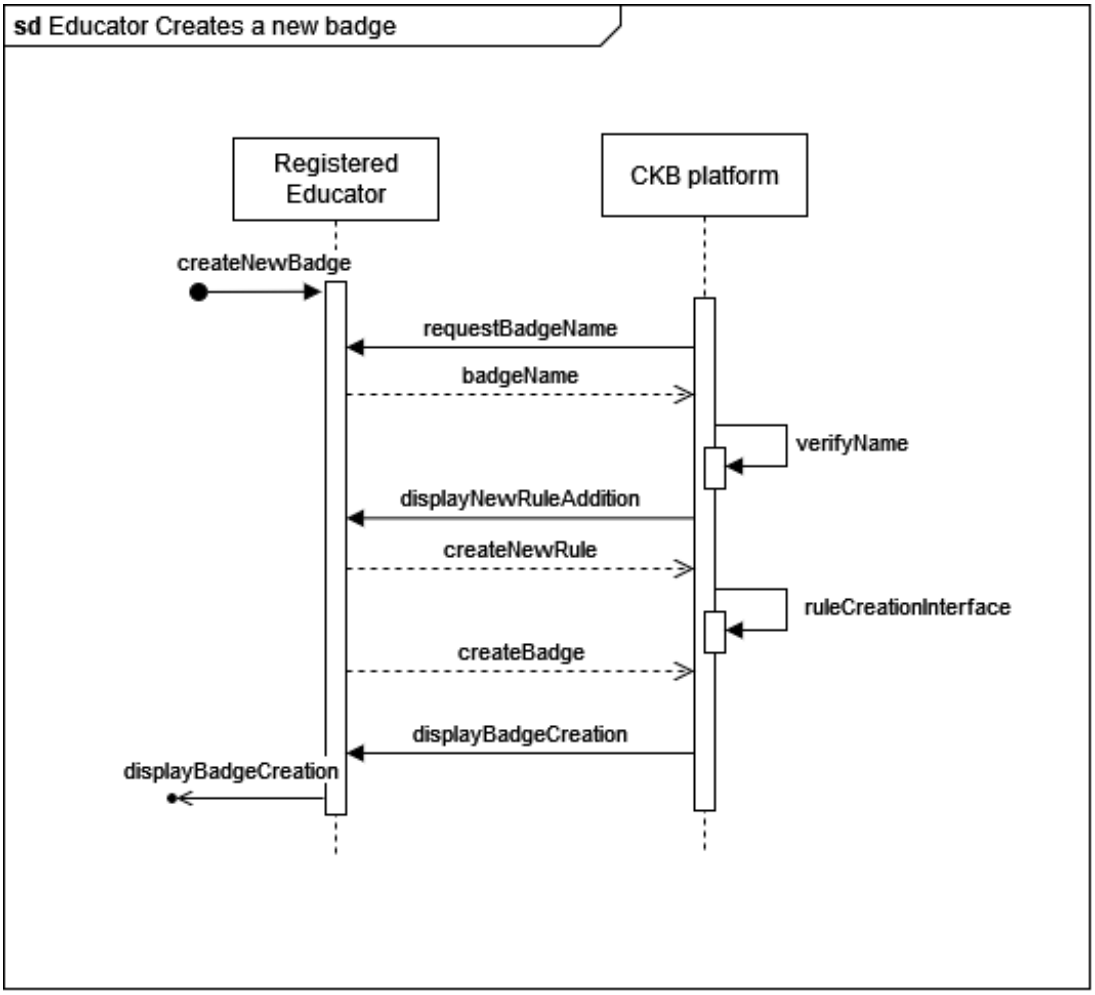

Figure 3.17: New Badge creation Sequence Diagram

UC15. Educator creates a new variable and a new rule for a badge

Actor	Educator
Entry conditions	The educator is logged in on the CKB platform and is creating a new badge.
Event Flow	1. The educator clicks on the "Create New Variable" button at the end of the list of pre-existing variables 2. The CKB platform prompts the educator to input the name of the variable 3. The educator inputs the name of the variable 4. The platform shows a list of metrics exposed by CKB and available for the creation of the variable 5. The educator selects one or more metrics from the list 6. The educator writes an arithmetic expression (Spreadsheet formula) combining the selected metrics in the input box on the right of the list 7. The educator clicks on the "Create" button 8. The CKB platform communicates the outcome of the variable creation 9. The educator clicks on the "Create New Rule" button at the end of the list of pre-existing rules 10. The CKB platform prompts the educator to input the description of the rule 11. The educator inputs the description of the rule 12. The platform shows a list of variables available for the creation of the rule 13. The educator writes a mathematical expression (Spreadsheet formula) combining the selected variables in the input box on the right of the list 14. The educator clicks on the "Create" button 15. The CKB platform communicates the outcome of the rule creation
Exit condition	The new variable and the new rule are added to the badge.
Exceptions	6.1. The educator does not write a valid arithmetic expression 13.1. The educator does not write a valid arithmetic expression

In both cases the CKB platform shows a message asking the educator to write a valid arithmetic expression.

Table 3.19: New Variable and Rule creation Use Case.

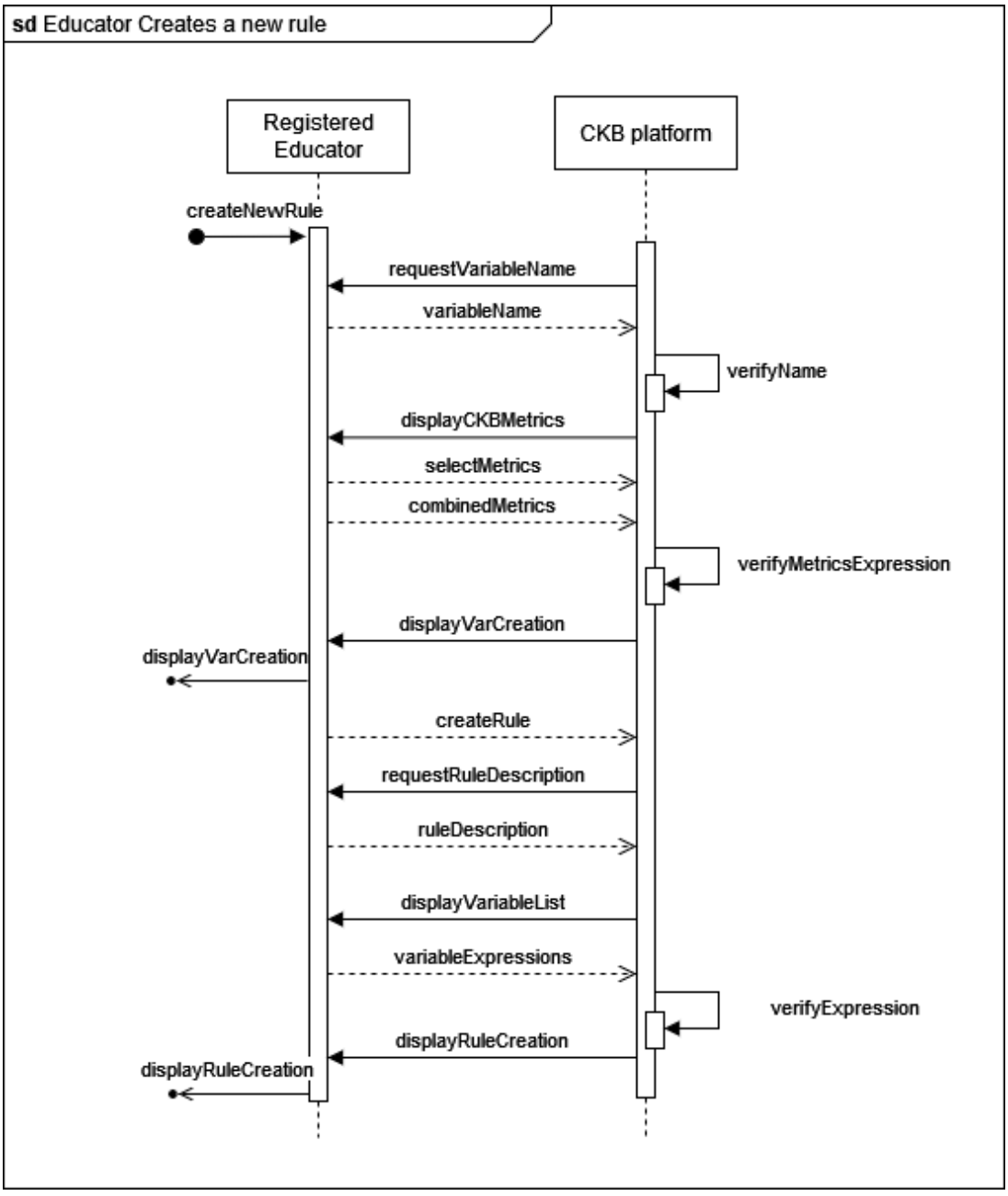

Figure 3.18: New Variable and Rule creation Sequence Diagram

UC16. Educator manually evaluates teams in a CKB

Actor	Educator
Entry conditions	The educator is logged in on the `CKB` platform and is on a CKB's details page. The educator is the creator of the CKB and the CKB has ended.
Event Flow	1. The educator clicks on the "Manually Evaluate Teams" button 2. The `CKB` platform shows the list of teams involved in the CKB 3. The educator clicks on a team for which he wishes to perform a manual evaluation 4. The `CKB` platform shows the team's sources 5. The educator assigns a score to the team 6. The educator clicks on the "Save" button 7. The `CKB` platform communicates the outcome of the operation
Exit condition	The score of the team is updated.

Table 3.20: Manual evaluation Use Case.

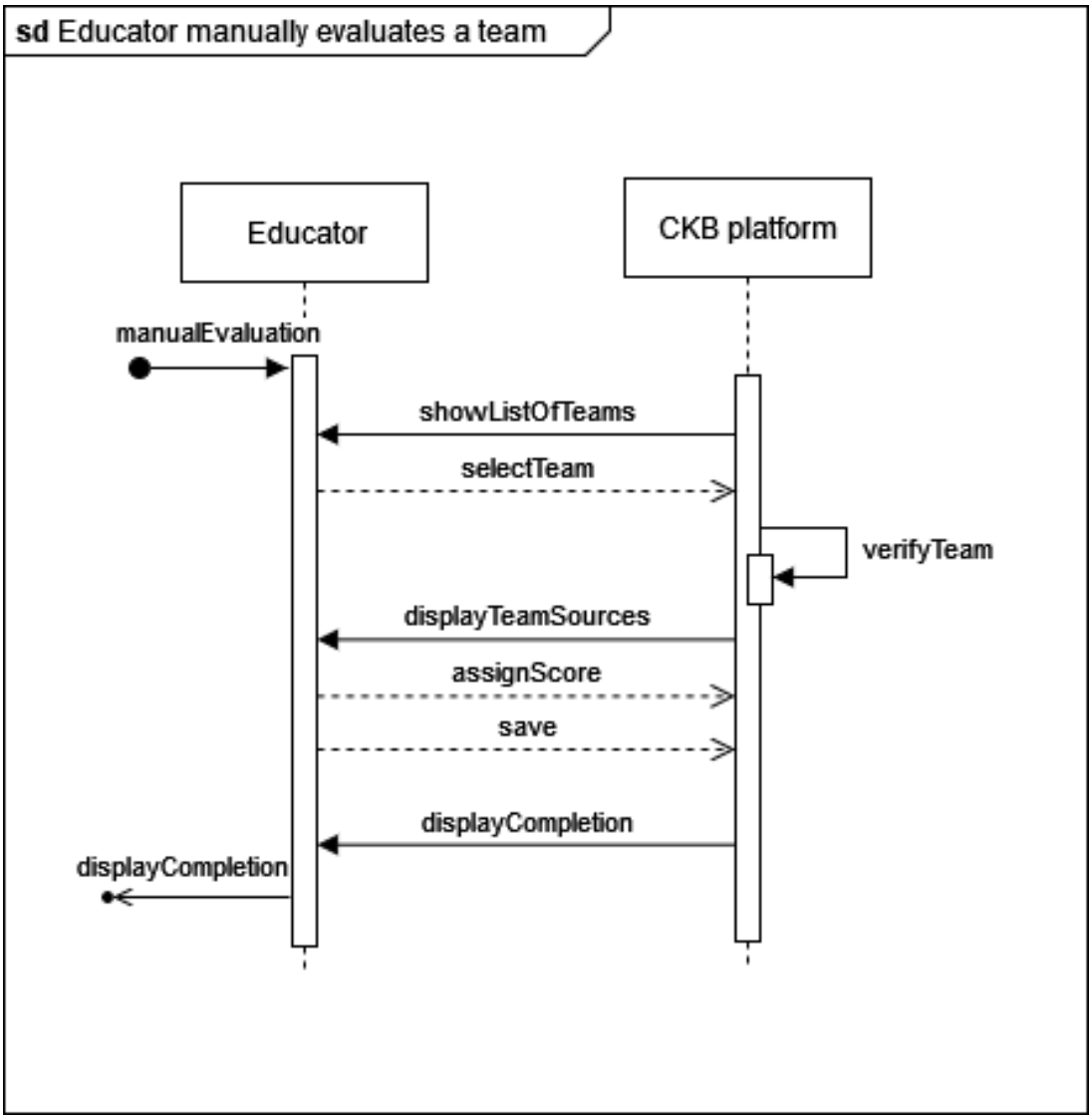

Figure 3.19: Manual evaluation Sequence Diagram

UC17. User visualizes a tournament's current ranking

Actor	Educator/Student
Entry conditions	The user is logged in on the CKB platform and is on the "Tournaments" page
Event Flow	1. The user clicks on a tournament from the list of tournaments in the "Tournaments" page 2. The CKB platform shows the tournament's details page 3. The user clicks on the "Ranking" button 4. The CKB platform shows the tournament's current ranking

Exit condition	The user visualizes the tournament's current ranking.

Table 3.21: Visualization of tournament's leaderboard.

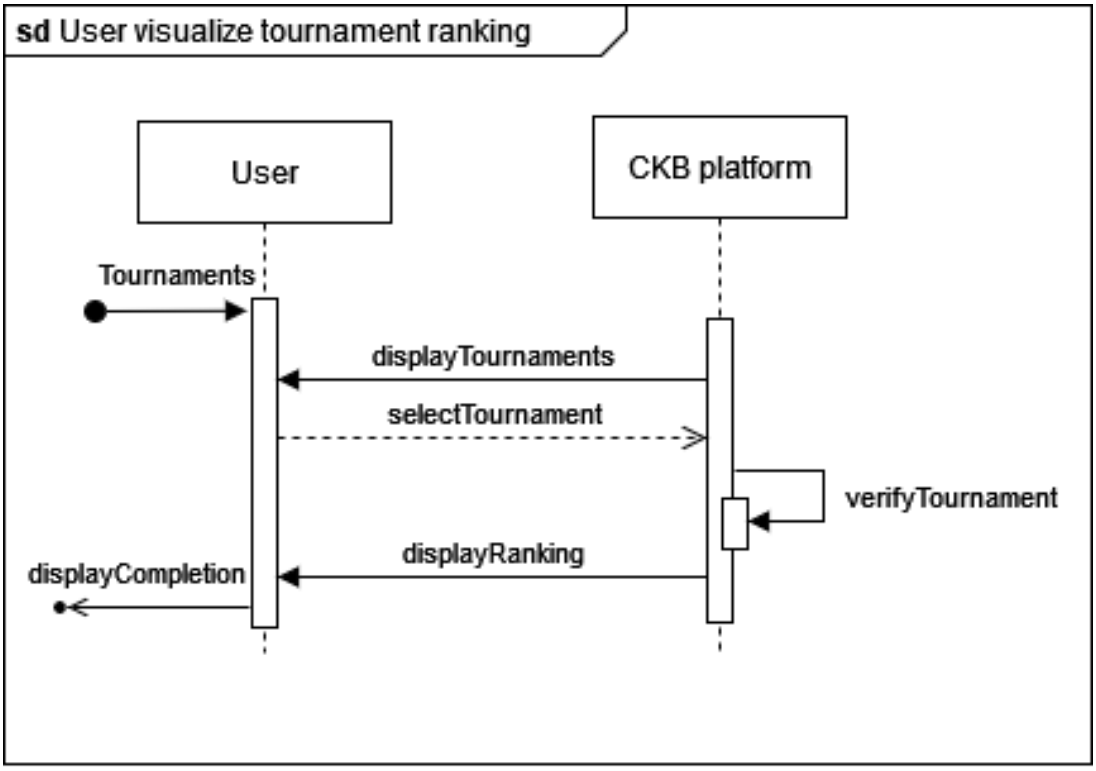

Figure 3.20: Visualization of tournament's Diagram

UC18. User visualizes a student's profile

Actor	Educator/Student
Entry conditions	The user is logged in on the `CKB` platform and is browsing the "Users" page
Event Flow	1. The user clicks on a student from the list of students in the "Users" page 2. The `CKB` platform shows the student's profile page which includes the badges achieved by the student 3. The user clicks on a badge 4. The `CKB` platform shows the badge's details
Exit condition	The user visualizes the student's profile and the badges he earned.

Table 3.22: View Profile Use Case

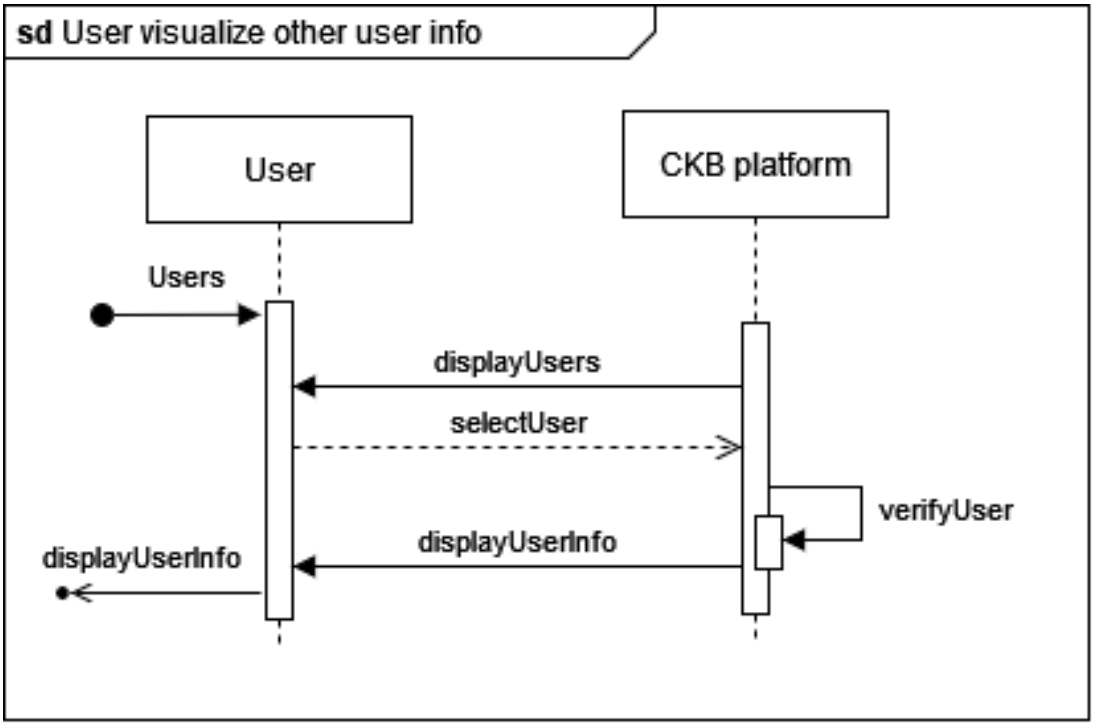

Figure 3.21: View Profile Sequence Diagram

UC19. User visualizes a CKB's current ranking

Actor	Educator/Student
Entry conditions	The user is logged in on the CKB platform. The user is either an educator who administrates the CKB or a student who is participating in the CKB.
Event Flow	1. The user clicks on a tournament in which they are involved from the list of tournaments in the "Tournaments" page 2. The CKB platform shows the tournament's details page 3. The user clicks on a CKB in which they are involved from the list of CKBs in the tournament's CKB list page 4. The CKB platform shows the CKB's details page 5. The user clicks on the "Ranking" button 6. The CKB platform shows the CKB's current ranking
Exit condition	The user visualizes the CKB's current ranking.

Table 3.23: CKB ranking Use Case

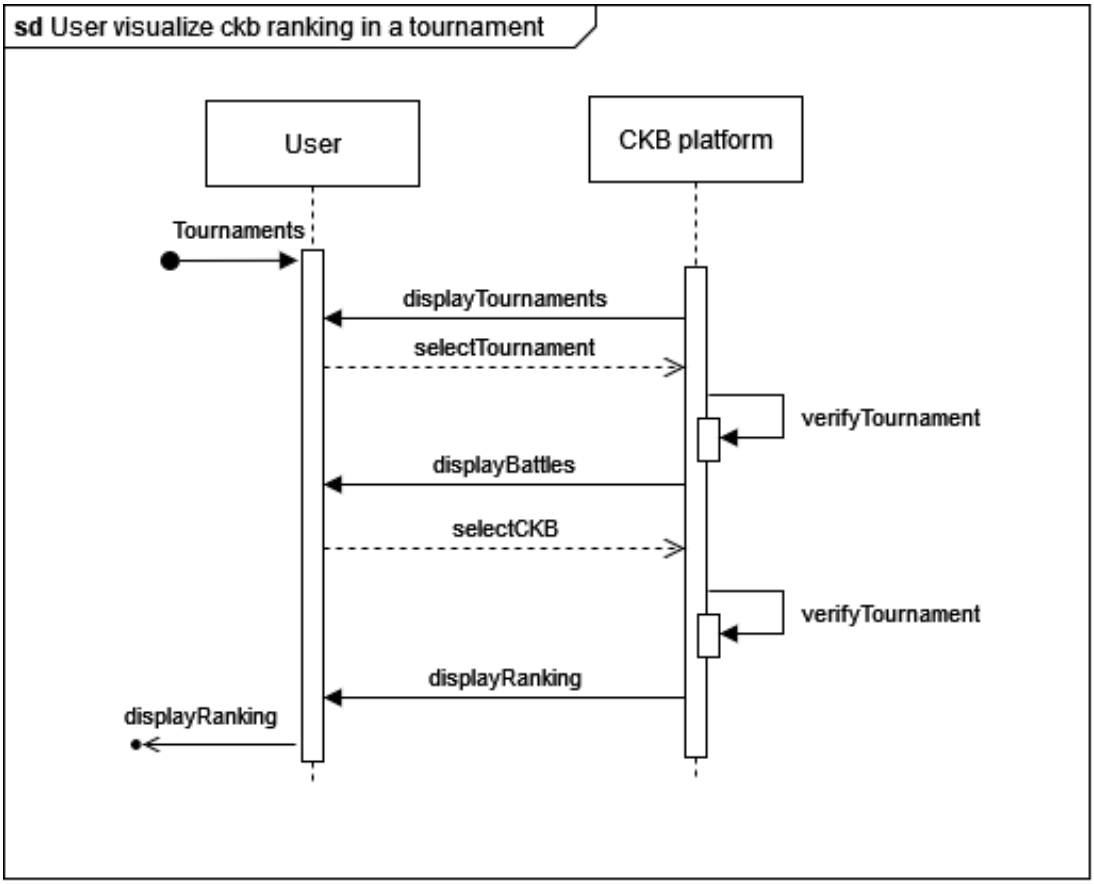

Figure 3.22: CKB ranking Sequence Diagram

3.5. Performance Requirements

1. Response Time:

- The CKB platform shall respond to user interactions within an average time of 2 seconds, measured from the user action initiation to the completion of the corresponding operation.

2. Concurrent Users:

- The platform should support at least 1000 concurrent users without a significant degradation in response time.

3. GitHub Integration:

- GitHub repository creation and updates triggered by student commits shall be processed within 5 minutes of the GitHub action.

4. Automated Evaluation:

- Automated evaluation of submitted code shall be completed within 1 minute of each GitHub push action.

5. Data Retrieval Time:

- Information retrieval for ongoing tournaments, tournament ranks, and personal scores should be performed within an average time of 3 seconds.

6. Consolidation Stage:

- The consolidation stage, where manual evaluation is required, should be completed within 3 days after the submission deadline of a battle.

7. Badge Assignment:

- Badge assignment based on rules and variables should be executed within 1 day after the final tournament rank becomes available.

8. Platform Uptime:

- The platform should maintain a minimum of 99.9% uptime over any given month.

90. Notification Latency:

- Notifications should be sent to all relevant users within 1 minute of the triggering event.

10. Gamification Features:

- Display of badges on user profiles and visualization of tournament-related achievements should be accessible within a response time of 3 seconds.

3.6. Design Constraints

3.6.1. Standards Compliance

The Code Kata Battle (CKB) platform must conform to established standards and legal requirements to ensure ethical and lawful operation. Key considerations include:

1. **Privacy and Data Laws:**
    - The platform must adhere to all laws governing privacy and data treatment, especially regarding user information and data exchange with third parties, such as CodeKata educators.
    - Compliance with the European Union General Data Protection Regulation (EU GDPR) is mandatory to ensure the privacy and rights of users are protected.
    - The system design should align with the principles outlined in Article 5 of the GDPR document, emphasizing the lawful, fair, and transparent processing of personal data.

3.6.2. Hardware Limitations

The design and functionality of the CKB platform need to consider specific limitations for optimal user experience:

1. **Cross-Platform Accessibility:**
    - The platform should be accessible from both web browsers and mobile applications. Mobile applications will be implemented for Android and iOS operating systems.

Note: Cross-platform accessibility, although listed under hardware limitations, primarily involves software compatibility rather than hardware constraints. It highlights the need for the platform to function seamlessly across different operating systems.

3.6.3. Other Constraints

While considering standard compliance and hardware limitations, additional but less restrictive constraints specific to the CKB project should be acknowledged:

1. **Educator Configuration:**
    - The design should facilitate easy configuration by educators, allowing them to

customize scoring criteria, badge rules, and associated variables according to their preferences and requirements.

2. **Automated Testing Tools:**
    - Seamless integration with automated testing tools is essential. The platform should be designed to accommodate any constraints or limitations imposed by these tools during code evaluation.
3. **Educator Workload:**
    - The design should optimize the educator's workload, particularly during the manual evaluation stage. Streamlining the evaluation process will enhance efficiency and user satisfaction.
4. **Gamification Logic:**
    - Constraints associated with the gamification logic, including badge creation, rule definition, and variable association, should be carefully considered to maintain system flexibility without compromising integrity.
5. **Integration with External Systems:**
    - The platform needs to seamlessly integrate with external systems, such as GitHub, while considering any constraints or limitations imposed by these external systems on data exchange and interactions.

These design constraints are vital to guide the development of the Code Kata Battle platform, ensuring legal compliance, optimal user experience, and consideration of specific project-related constraints.

3.7. Software System Attributes

3.7.1. Reliability

The reliability of the Code Kata Battle (CKB) platform is crucial for its users, especially during critical operations such as code evaluations and scoring. While the platform doesn't require absolute perfection in every operation, it must maintain a high level of reliability to instill confidence in educators and students. Considering the nature of the platform, a failure rate between 0.1% and 1% is deemed acceptable. This range positions the CKB system as a robust and dependable tool for code kata battles.

3.7.2. Availability

Availability is a key attribute for the CKB platform, particularly for educators who rely on its functionalities for managing tournaments, battles, and student assessments. Downtime, especially during crucial periods like ongoing battles or tournament deadlines, is unacceptable. Therefore, the platform aims to achieve a remarkable 99.9% uptime, ensuring educators have uninterrupted access to critical features. This commitment to high availability enhances the overall user experience and reinforces the platform's reliability.

3.7.3. Security

Security is paramount for the CKB platform, given its involvement in handling code submissions, communication with external systems like GitHub, and the storage of sensitive user information. The platform ensures data privacy by employing HTTPS for secure communication. Additionally, all stored information is encrypted, providing an extra layer of protection against potential security threats. By prioritizing these security measures, the CKB platform safeguards the integrity and confidentiality of user data, fostering trust among educators and students.

3.7.4. Maintainability

Maintainability is a critical aspect of the CKB platform's design philosophy. The platform is structured into modular components to facilitate efficient maintenance, updates, and future extensions. Each implemented functionality is documented, ensuring clarity for developers involved in maintenance tasks. The design emphasizes a modular approach, allowing updates to specific components without adversely affecting the entire system. This commitment to maintainability ensures that the CKB platform can adapt to evolving

requirements and technologies without compromising its stability.

3.7.5. Portability

The CKB platform prioritizes portability to accommodate a diverse user base. Users can access the platform seamlessly from web browsers and mobile applications on both Android and iOS devices having access to the same features. While cross-platform development tools offer efficiency, separate implementations for Android and iOS are considered for a more tailored user experience. This flexibility in portability ensures that the CKB platform remains accessible and user-friendly, regardless of the chosen device or operating system.

4 | Formal Analysis Using Alloy

4.1. Alloy Code

```
// Alloy model for CodeKataBattle platform – Entities and Facts
abstract sig BattleState{}
one sig OpenBattle extends BattleState{}
one sig StartedBattle extends BattleState{}
one sig InProgressBattle extends BattleState{}

abstract sig TournamentState{}
one sig OpenTournament extends TournamentState{}
one sig ClosedTournament extends TournamentState{}
one sig InProgressTournament extends TournamentState{}


enum BadgeEnum{ParticipationBadge, GroupPowerBadge, DoubleDigitScoreBadge}

sig Badge{
    badgeType: one BadgeEnum
}


// Signatures
abstract sig User{
    username: one Int
}


sig Student extends User{
    sBadges: set Badge
}
```

```
sig Educator extends User{

}

sig Score{
    student: one Student,
    value: Int,
    rank: Int
}


sig Team{
    members: set Student,
    size: Int,
    score: Int // Punteggio del team nella  battle
}


sig Battle{
    teams: disj set Team,
    min_size: Int,
    max_size: Int,
    state: one BattleState,
}


sig Tournament{
    participants: set Student,
    leaderboard: set Score,
    administrators: set Educator,
    battles: disj set Battle,
    tState: one TournamentState,
    tBadges: disj set Badge
}

// ------------------------- USER RELATED FACTS
    -------------------------
```

```
// Req: every user as an unique usernanme

fact uniquesername{
    all u1, u2: User | u1!=u2 implies u1.username != u2.username
}

// ---------------------------- TOURNAMENT RELATED FACTS

// A score can't be in two different tournaments

fact noDoubleScore{
    all t1, t2: Tournament, s1: t1.leaderboard, s2: t2.leaderboard|t1!=t2
       implies s1 != s2

}


// Links the score in the tournament with the participants in the same tournament

fact studentInTournamentMatchesScore {
    all t: Tournament | t.participants = (t.leaderboard.student)
}

// In a tournament, a student is linked to only one score

fact singleStudentScore{
    all tournament: Tournament, s1, s2: tournament.leaderboard | s1 != s2
       implies s1.student != s2.student
}


// Tournament needs to have at least one administrator

fact numAdministrators{
    all t: Tournament | #t.administrators > 0
}

// ------------------------ TEAM RELATED FACTS
```

```
// Just some general variable check

fact generalTeamReq{
    all team: Team | #team.members <= team.size and team.size > 0 and
        team.score >= 0 and #team.members > 0

}

// The size defined for the team must be in battle constraint and Num of members is at
    least minimum in a team

fact correctSize{
    all battle: Battle, team: battle.teams | team.size >= battle.min_size and
        team.size <= battle.max_size and #team.members >= battle.min_size
}

// The students in a team must be all relative to the same tournament

fact possibleTeam{
    all t: Tournament, b: t.battles, team : b.teams, m: team.members | m in
        t.participants
}

// Teams must be linked to a battle

fact teamBattleRel{
    all team: Team| one battle: Battle | team in battle.teams
}

// Students in a battle must be in only one team

fact oneTeamPerStudent{
    all battle: Battle, team1, team2: battle.teams | team1 != team2 implies
        (all s1: team1.members, s2: team2.members | s1 != s2)
}

// ------------------------ BATTLE RELATED FACTS

fact generalBattleReq{
```

```
    all battle: Battle | battle.min_size <= battle.max_size and
        battle.min_size > 0
}

// Battle are related to tournament

fact battleTournamentRel{
    all battle: Battle | one tournament: Tournament | battle in
        tournament.battles
}

// ------------------------- SCORE RELATED FACTS

// general req

fact generalScoreReq{
    all score: Score | score.value >= 0 and score.rank > 0
}

// All ranks are in order given their score

fact ranksAreInSuccession {
    all t: Tournament, s: t.leaderboard | s.rank > 0 and s.rank <=
        #t.leaderboard
    all t: Tournament, s1, s2: t.leaderboard | s1 != s2 implies s1.rank !=
        s2.rank
    all t: Tournament | lone s: t.leaderboard | s.rank = 1
    all t: Tournament | lone s: t.leaderboard | s.rank = #t.leaderboard
}

//For every two students in the participant sets, a student with a higher score in a
    tournament has a lower rank

fact studentsWithHigherScoreHaveLowerRank{
    all t: Tournament, s1, s2: t.leaderboard | s1 != s2 implies ((s1.value >=
        s2.value iff s1.rank <= s2.rank)) //and (s1.value = s2.value implies s1.rank =
        s2.rank))

}
```

```
// A score is linked to a torunament

fact scoreLinkedTournament{
    all score: Score | some tournament: Tournament | score in
        tournament.leaderboard
}

// the score is the some of the scores acquired in the challenges

fact consistentScore{
    all t: Tournament, s: t.leaderboard | s.value = (sum team: {team: Team |
        s.student in team.members && (some battle: t.battles | team in
        battle.teams)} | team.score)
}


// ------------------------ BADGE RELATED FACTS

// Badges exists withnin tournament scope

fact badgeTournamentRel{
    all b: Badge | some t: Tournament | b in t.tBadges
}

// Badges in a tournament are all different

fact allDiffBadgesT{
    all t: Tournament, b1, b2: t.tBadges | b1 != b2 implies b1.badgeType !=
        b2.badgeType
}

// Badges of a student are all different

fact allDiffbadgesS{
    all s: Student, b1, b2: s.sBadges | b1 != b2 implies b1.badgeType !=
        b2.badgeType
}
```

```
// A student can have a badge if he was in a tournament where that badge was acquirable

fact coherenceAcq{
    all s: Student, b: Badge | some t: Tournament | b in s.sBadges implies (b
        in t.tBadges && s in t.participants)
}

// Student that are in a tournament where participationBadge is available , must have a
    partecipationBadge

fact participationsBadgeAcq{
    all t: Tournament, s: t.participants, b: t.tBadges | b.badgeType =
        ParticipationBadge implies (one b2: Badge | b2.badgeType =
        ParticipationBadge && b2 in s.sBadges)
}

// Student in a tournament with GroupPower Badge can acquire it if conditions are
    satisfied

fact groupBadgeAcq{
    all t: Tournament, badge: t.tBadges, battle: t.battles, team: battle.teams
        |
        (badge.badgeType = GroupPowerBadge && #team.members >= 2) implies (all
            s: team.members | one b2: Badge | b2.badgeType = GroupPowerBadge &&
            b2 in s.sBadges)
}

fact groupBadgeAcqR{
    all s: Student, badge: s.sBadges | badge.badgeType = GroupPowerBadge
        implies (some team: Team | #team.members > 2 && s in team.members)
}

// Student in a tournament with DoubleDigitScoreBadge can acquire it if conditions are
    satisfied

fact DoubleDigitScoreBadgeAcq{
    all t: Tournament, badge: t.tBadges, score: t.leaderboard |
        (badge.badgeType = DoubleDigitScoreBadge && score.value > 9) implies
```

```
    (one b2: Badge | b2.badgeType = DoubleDigitScoreBadge && b2 in
        score.student.sBadges)
}

fact DoubleDigitScoreBadgeAcqR{
    all s: Student, badge: s.sBadges | badge.badgeType = DoubleDigitScoreBadge
        implies (some score: Score | s = score.student && score.value > 9)
}
```

4.2. Simulations

Below is presented a simulation of the built model.
The world has been simulated forcing Alloy to generate more entities (Tournaments, Battles, Teams, Badges, etc.) in order to give a deeper insight of the model.

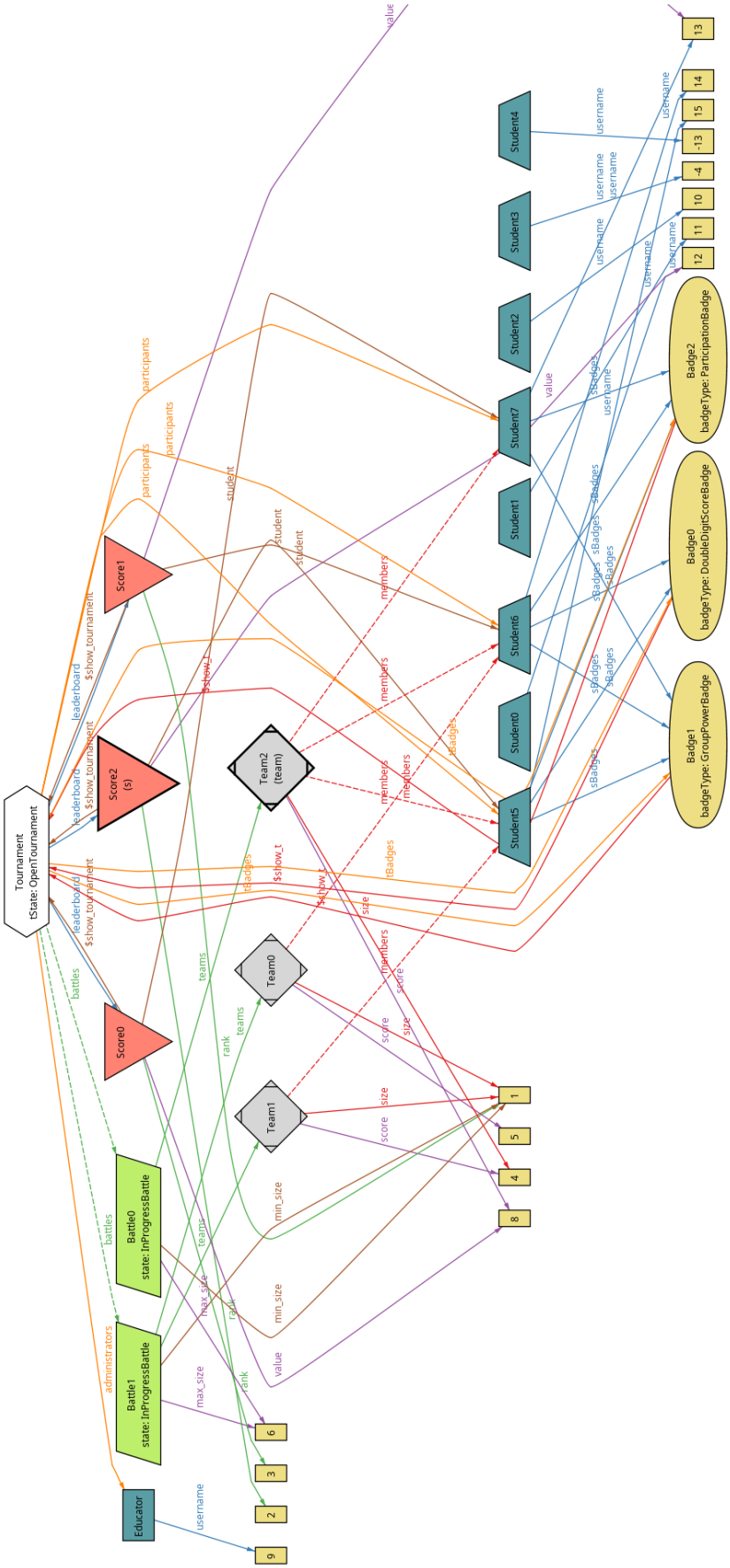

Figure 4.1: Alloy World .

5 | Effort Spent

Member of group	Effort spent	
Federico Zanca	Introduction	$0h$
	Overall description	$11h$
	Specific requirements	$23h$
	Formal analysis	$12h$
Federico Costantini	Introduction	$0h$
	Overall description	$19h$
	Specific requirements	$7h$
	Formal analysis	$18h$
Michele Zhenghao Zhuge	Introduction	$4h$
	Overall description	$8h$
	Specific requirements	$24h$
	Formal analysis	$8h$

Table 5.1: Effort spent by each member of the group.

6 References

6.1. Paper and online references

- The specification document Assignment RDD AY 2023-2024.pdf
- An example of a website with similar features to the CKB platform

6.2. Used tools

- GitHub for project versioning
- Draw.io for UML diagrams
- Notion for reasoning, notes and task assignment between members of the group
- Visual Studio Code as LaTeX and Alloy editor
- Alloy for formal analysis

List of Figures

List of Tables