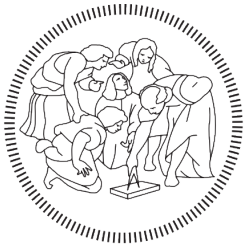
Software Engineering 2 - Prof. Di Nitto Elisabetta
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

CodeKataBattle

RASD
Requirement Analysis and Specification Document

November 1, 2023

Marco Cerino (matricola)
Mattia De Bartolomeis(matricola)



POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

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

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

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

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

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

1.1.1 Goal

The system is characterized by the following goals:

G1 Educators can manage coding tournaments and battles

G2 Students can form teams for coding battles

G3 Students can participate in coding battles

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

G5 Projects are evaluated in an automated way

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

---

1.2 Scope

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

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

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

1.2.1 World phenomena

WP1	Educators evaluate the projects
WP2	Students solve the challenges

1.2.2 Shared phenomena

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

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

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

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

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

1.3.2 Acronyms

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

1.3.3 Abbreviations

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

1.4 Revision history

1.5 Reference Documents

The creation of this document is based on :

- Slides of Software Engineering 2 course

1.6 Document Structure

This document is composed of six sections:

1. **Introduction**: In this section, we present an overview of the document's content and structure. We outline the problem at hand and articulate the goals that the system aims to achieve. The subsection on scope delves into a comprehensive description of various world and shared phenomena relevant to our subject matter. Additionally, essential information is provided to facilitate proper understanding, including definitions and abbreviations.
2. **Overall Description**: This section offers a holistic description of the system. We provide insights into the system's architecture and functionality, accompanied by an outline of the users and their primary functionalities. Domain diagrams are introduced to illustrate key components, and various scenarios are detailed to enhance comprehension. Finally, we present the fundamental assumptions within the system's domain.
3. **Specific Requirements**: In this section, we delve into the specific requirements of the system. Both functional and non-functional requirements are elucidated to provide a comprehensive understanding. Use case diagrams are included to visually represent system interactions, accompanied by detailed descriptions of each use case and related sequence diagrams. The section concludes with a mapping of requirements onto both overarching goals and individual use cases.
4. **Formal Analysis Using Alloy**: Here, we conduct a formal analysis of the system using Alloy. This includes a rigorous examination of the system's specifications and constraints to ensure coherence and reliability.

5. **Effort Spent**: In this section, we provide an estimate of the effort invested by each member of the group. This allows for a transparent understanding of the distribution of work within the team.
6. **References**: The document concludes with a comprehensive list of references used during the research and development phases. This section serves as a valuable resource for readers seeking additional information or verification of the document's content.

2 Overall Description

2.1 Product perspective

2.1.1 Scenarios

A. Educator Registration
Marco, un docente presso l'Università di Napoli, ha recentemente scoperto la piattaforma CKB grazie a un collega. Interessato alle sue potenzialità nel migliorare le competenze di sviluppo software degli studenti, Marco ha deciso di iscriversi come educatore. Visitando la home page di CKB, ha selezionato l'opzione "Registrati" e inserito il suo indirizzo email, scelto una password e fornito nome e cognome. Dopo aver inserito queste informazioni, è stato richiesto a Marco di verificare la sua email cliccando su un link inviatogli per garantire la sicurezza del suo account. Una volta completata questa verifica, ha proseguito con la registrazione selezionando l'opzione "Iscriviti come educatore". Con la registrazione completata con successo, Marco è stato reindirizzato alla sua dashboard personale all'interno di CKB, dove ha avuto accesso a una panoramica delle funzionalità per educatori, inclusa la possibilità di creare nuovi tornei e battaglie.

B. Student Registration
Andrea, uno studente dell'Università di Bologna, ha scoperto l'applicazione CKB grazie al consiglio del suo professore. Affascinato dall'opportunità di affinare le sue competenze di sviluppo software attraverso sfide di code kata, Andrea ha deciso di iscriversi alla piattaforma. Una volta sulla home page di CKB, ha selezionato l'opzione "Registrati" e fornito il suo indirizzo email, creando una password. Ha anche inserito il suo nome e cognome, scegliendo l'opzione "Iscriviti come studente" per completare la registrazione. Tuttavia, prima di finalizzare il processo, il sistema ha richiesto a Andrea di verificare il suo indirizzo email cliccando su un link inviatogli, per assicurare l'autenticità del suo account. Durante questo passaggio, Andrea ha ricevuto un messaggio di errore: l'indirizzo email era già presente nel sistema di CKB. Ha prontamente corretto l'indirizzo email, utilizzando un account valido. Una volta verificata con successo la sua email e completato il processo di registrazione, Andrea è stato reindirizzato alla sua dashboard personale su CKB. Qui, ha potuto esplorare le varie funzionalità disponibili per gli studenti, come i tornei attivi, le competizioni imminenti e le informazioni sui suoi progressi nella piattaforma.

C. Tournament Creation
Daniele, professore di informatica presso il Politecnico di Milano, desidera creare un torneo sulla piattaforma CKB per permettere ai suoi studenti di sviluppare e migliorare le loro competenze di programmazione attraverso la partecipazione a battaglie di code kata.

Daniele accede alla piattaforma utilizzando le sue credenziali di accesso come professore e naviga alla sezione dedicata alla creazione di tornei sulla piattaforma CKB.

Successivamente, avvia il processo di creazione di un nuovo torneo selezionando l'opzione appropriata dalla dashboard principale. Il sistema richiede a Daniele di inserire informazioni sul torneo, come il nome, una data di scadenza per la registrazione al torneo da parte degli studenti e la lista di colleghi che possono creare battaglie all'interno di tale torneo.

Il professore completa il modulo e, dopo aver inserito le informazioni richieste, conferma la creazione del torneo. Il sistema verifica la validità delle informazioni fornite e, nel caso il nome del torneo risulti già esistente, invierà il seguente warning: "Il nome del torneo è gia esistente". Altrimenti, in caso di verifica positiva, il sistema registra il nuovo torneo all'inteno del sistema. Tutti gli studenti iscritti alla piattaforma ricevono notifiche sulla creazione del nuovo torneo.

D. Battle Creation
Giovanni, un professore con privilegi di creazione di battaglie all'interno del torneo "Algorithms and data structures" sulla piattaforma CKB, decide di creare una nuova battaglia. A tale scopo,dopo aver effettuato l'accesso, naviga alla sezione dedicata alla creazione di battaglie e avvia il processo di creazione di una nuova battaglia. Il sistema richiede a Daniele di inserire informazioni essenziali sulla battaglia, tra cui una breve descrizione testuale, il progetto software con gli script di automazione della build, il numero minimo e massimo di studenti per gruppo, la data di scadenza per la registrazione e per la consegna del progetto. Inoltre Daniele imposta informazioni aggiuntive che andranno ad incidere sulla valutazione del punteggio, come sicurezza, mantenimento e affidabilità. Infine, il sistema integra la nuova battaglia nella piattaforma. Gli studenti iscritti al torneo pertinente ricevono notifiche riguardo alla prossima battaglia.

E. Tournament registration
Sarah, una studentessa di ingegneria informatica, si ritrova desiderosa di una nuova sfida per elevare le sue abilità di programmazione. Decide allora di collegarsi alla piattaforma CKB. Dopo aver effettuato l'accesso, esplora nella sezione dedicata ai tornei disponibili e ne individua uno in particolare - "Python Challenge". Sarah clicca sul torneo e legge la sua descrizione e nota che la data di scadenza per la registrazione non è terminata. Senza esitazione, decide di partecipare alla Python Challenge. Cliccando su 'Iscriviti al torneo' Sarah riceve un messaggio di conferma e fa ufficialmente parte del torneo. A questo punto può visualizzare tutte le battaglie in programma all'interno del torneo e verrà anche notificata alla creazione di battaglie future .

F. Battle registration
Leonardo, studente iscritto al torneo "Super Tournament" su CKB, riceve una notifica che cattura la sua attenzione: "Chiamata alla Battaglia - Python Coding Challenge". Volentoroso di mettere alla prova le sue abilità di programmazione, Leonardo, dopo aver effettuato l'accesso, entra nella sezione dedicata alla battaglia, legge con attenzione i dettagli forniti e decide di iscriversi alla battaglia cliccando su "Iscriviti alla battaglia". Successivamente la piattaforma permette a Leonardo di scegliere se partecipare individualmente o invitare altri colleghi a formare un team per la battle, rispettando il numero minimo e massimo di partecipanti consentiti. Dopo la scadenza della fase di registrazione, la piattaforma CKB genera automaticamente una repository dedicata per il progetto della "Python Coding Challenge". Successivamente, il link diretto alla repository viene consegnato ai membri del team e reso disponibile nella sezione "Le tue battaglie" dell'applicazione.

G. Face the battle
Il team "CodeCrafters", composto da appassionati studenti desiderosi di immergersi nella battle, riceve tramite mail il link alla repository su GitHub dedicata alla sfida "Algoritmi Avanzati" su CKB. Con un semplice click, procedono con il fork di tale repository, contenente il codice kata, dando così vita al loro spazio di lavoro virtuale. Per garantire una valutazione continua e precisa del loro lavoro, i CodeCrafters configurano GitHub Actions, che assicura una tempestiva comunicazione con la piattaforma. Immersi nella stimolante sfida degli "Algoritmi Avanzati", i CodeCrafters avviano il processo di sviluppo adottando l'approccio test-first. Con creatività, creano le prime implementazioni, le sottopongono a test e le committano nella repository principale, tracciando così ogni passo del loro iterativo percorso. Ogni push prima della scadenza della battaglia attiva la piattaforma, che valuta automaticamente gli aspetti funzionali e la tempestività del lavoro dei CodeCrafters. Successivamente, la piattaforma calcola e aggiorna il loro punteggio di battaglia, offrendo feedback in tempo reale sulla qualità del loro prezioso contributo.

H. Ranking display
Matteo, durante la sua partecipazione a una battaglia su CKB, accedendo alla sezione del Torneo e della battaglia d'interesse può monitorare in tempo reale la posizione della sua squadra attraverso una classifica dinamica. Al termine della battaglia, si trova di fronte a una fase di consolidamento, durante la quale l'educatore può decidere se assegnare un punteggio personale al progetto o affidarsi esclusivamente al punteggio automatico. Alla conclusione di questa fase, Matteo può consultare la classifica finale recandosi nella sezione dedicata alla battaglia appena conclusa, ottenendo così una panoramica completa delle prestazioni della sua squadra.

Il punteggio ottenuto in questa specifica battaglia si somma a quelli accumulati nelle battaglie precedenti, contribuendo a formare il suo punteggio complessivo nel torneo. Matteo e tutti gli utenti hanno la possibilità di esplorare la classifica nella sezione dedicata al torneo stesso , accessibile a tutti gli utenti interessati.

Infine, una volta che l'educatore chiude definitivamente il torneo, Matteo e gli altri iscritti a CKB possono consultare la classifica finale del torneo sempre nella sezione relativa a quel torneo.

I. Evaluates project
Una volta scaduta la fase di sottomissione di una battaglia del torneo "Algoritmi Avanzati" su CKB, il Professore Bianchi, creatore della sfida, si prepara ad eseguire una valutazione manuale dei progetti presentati dai team partecipanti.Accede alla piattaforma, seleziona il torneo e la battaglia appena conclusa. All'interno dell'interfaccia, trova una lista completa dei progetti degli studenti se è stata aggiunta la valutazione manuale in fase di creazione della battaglia. Inizia esaminando attentamente il codice sorgente prodotto da ciascun team, analizzando gli aspetti che non possono essere completamente valutati in modo automatico. Durante questa fase, il Professore Bianchi attribuisce un punteggio personale a ciascun team in base alla sua esperienza e conoscenza approfondita, in

particolare premiando la creatività, l'ingegnosità e l'approccio strategico dei team partecipanti. Terminata la valutazione lo score finale per ciascun team viene determinato facendo la media tra il punteggio assegnato manualmente dal Professore Bianchi e quello generato automaticamente dalla piattaforma.

J. **Tournament Closure** Luisa, docente di informatica presso l'Università di Torino, ha gestito con successo il torneo "Logic Masters" sulla piattaforma CKB.Dopo l'ultima battaglia, decide di concludere il torneo.Luisa accede a CKB con le sue credenziali e naviga alla sezione del torneo "Logic Masters".Nella dashboard del torneo, trova l'opzione "Chiudi il Torneo". Dopo aver cliccato selezionato tale opzione, appare un messaggio di warning: "Sei sicuro di voler chiudere il torneo?". Luisa deve scegliere tra "Conferma" per procedere con la chiusura o "Annulla" per annullare l'operazione. Con la conferma della chiusura, il sistema, se non vi sono ancora battaglie attive, chiude il torneo.

2.1.2 User interface

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

2.1.3 Software interface

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

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

2.1.4 Interfaccia Hardware

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

- Computer Desktop e Laptop
- Smartphone e Tablet

2.2 Domain class diagram

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

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

conduct a manual evaluation or not, enhancing the evaluation process when educator intervention is desired

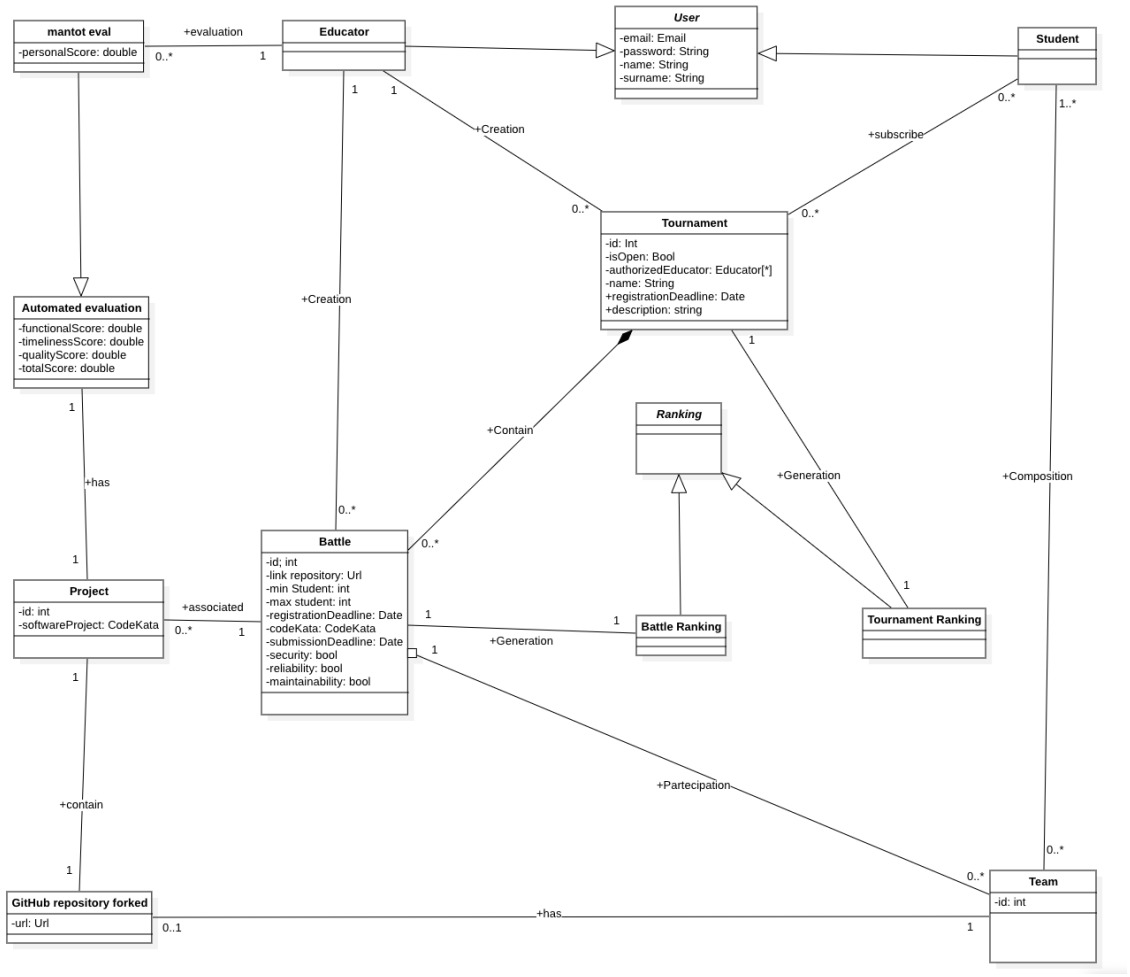


Figure 1: Descrizione dell'immagine

2.3 StateChart Diagrams

Lo StateChart Diagram è un diagramma utilizzato per descrivere il comportamento di classi in termini di stato. Il diagramma mostra gli stati che sono assunti dalla classe in risposta ad eventi esterni.

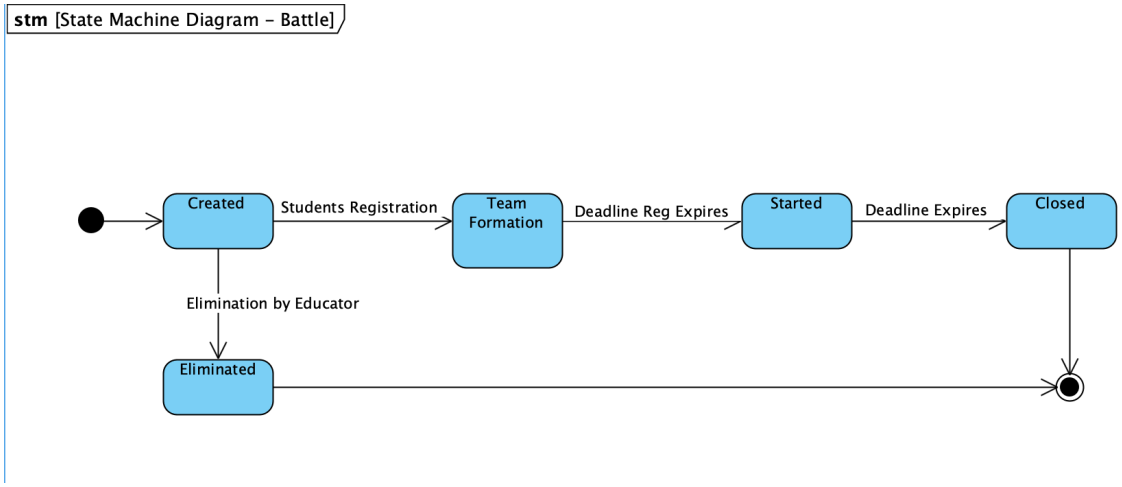


Figure 2: Didascalia dell'immagine

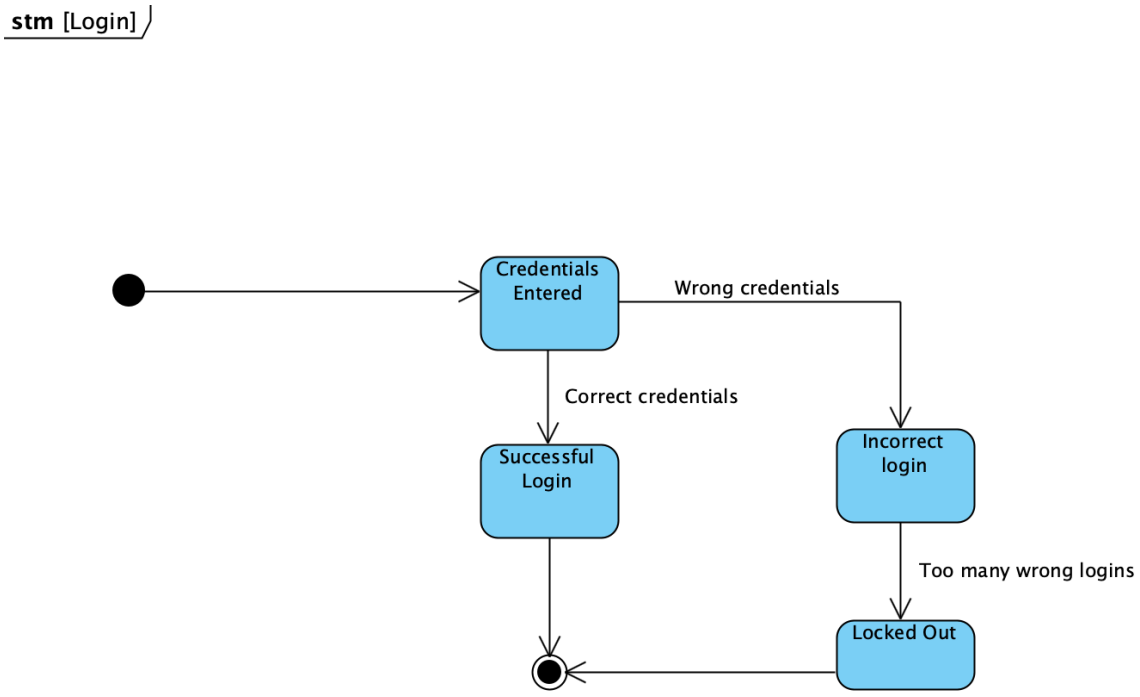


Figure 3: Didascalia dell'immagine

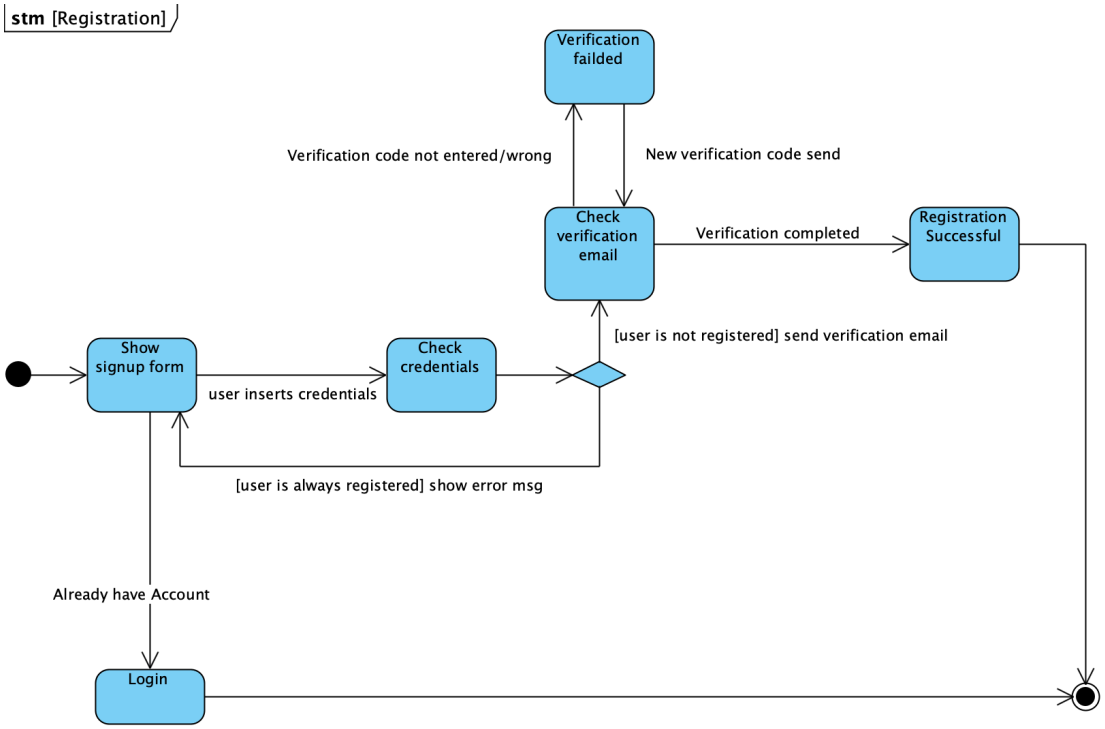


Figure 4: Didascalia dell'immagine

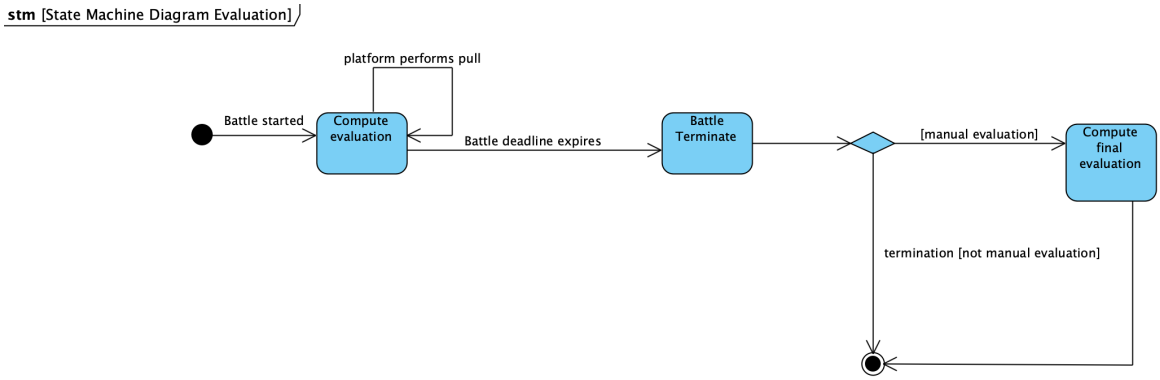


Figure 5: Didascalia dell'immagine

Evaluation process
The diagram outlines a process initiating with the 'Compute evaluation' stage. At this stage, following the commencement of a battle, the platform conducts an automated assessment of the projects. Every time pull operations are performed to obtain the latest code updates submitted by students through GitHub pushes, the system reverts to the 'Compute evaluation' stage for a fresh assessment. This is followed by the 'Battle deadline expires' transition, indicating the end of the project submission deadline and that no new commits for evaluation are accepted.
Next, the process moves to the 'Battle Terminate' state, marking the end of the battle. Here, all automatic scores have been assigned, and no further contributions are accepted.
If a 'manual evaluation' is set, the process can diverge towards a manual evaluation. In this case, if the educator needs to manually assess the students' work, the process proceeds to the 'Compute final evaluation' state. The educator reviews the projects, assigns a manual score which is added to the automatic score to determine the final score of the battle. Otherwise, without a manual evaluation, the process concludes, relying solely on the automatic evaluation for the final score.

Battle Lifecycle
The Statecharts diagram illustrates the flow of events for a code kata battle within the CKB platform. The battle begins with the "Created" state, indicating the initiation of a new challenge by the educator. In this phase, if no students sign up before the registration deadline or if the educator decides to cancel the battle, it transitions to the "Eliminated" state, leading to the removal of the battle from the platform. However, if at least one student signs up, it proceeds to the "Team Formation" state, where students have the opportunity to form teams. After the registration deadline, it enters the "Started" state, marking the actual commencement of the battle, allowing students to start working on their code projects. This state persists until the final submission deadline. Finally, the "Closed" state represents the conclusion of the battle.

2.4 Product Functions

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

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

2.5 User characteristics

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

2.5.1 Student

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

2.5.2 Educator

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

2.6 Assumptions, dependencies and constraints

2.6.1 Assumptions

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

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

Table 1: Domain assumptions.

2.6.2 Hardware Constraints

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

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

2.6.3 Regulatory policy

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

3 Specific Requirement

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

3.1 Requirements

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

R2 The system must allow users to login.

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

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

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

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

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

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

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

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

R11 The system must allow students to invite other students to create a team.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

R31 The system must create and update the tournament rankings.

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

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

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

3.1.1 Mapping between Goal,Domain Assumption and requirements

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

G1 - Educators can manage coding tournaments and battles.

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

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

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

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

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

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

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

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

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

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

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

G2 - Students can form teams for coding battles.

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

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

*R*11 The system must allow students to invite other students to create a team.

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

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

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

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

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

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

*DA*4 Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

G3 - Students can participate in coding battles

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

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

*R*07 The system must notify via mail subscribed students to the platform upon the creation of new tournaments.

*R*08 The system must allow students to subscribe to tournament on the CKB platform before a specific deadline.

*R*09 The system must notify via mail all students subscribed in the tournament whenever a new battle is created within that tournament.

*R*10 The system must allow students to subscribe in the battle.

*R*18 The system must allow user to see the list of available battles.

*R*19 The system must integrate with GitHub for repository management and automate the process of code submission and evaluation.

*R*22 The system must automatically create a GitHub repository containing the code kata upon the expiration of the battle's registration deadline

*R*23 The system must notify via email the link to the repository containing the code kata to all students who are members of a team registered for the battle.

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

*DA*2 Students must have a Github account.

*DA*4 Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

*DA*5 The integration with GitHub and GitHub Actions functions correctly, allowing for seamless repository management, code submission, and automated workflow for the students' projects

*DA*6 Users who register on the CKB platform in the role of 'student' have familiarity with pro- gramming languages, GitHub, and test-driven development methodologies.

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

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

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

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

*R*18 The system must allow user to see the list of available battles.

*R*20 The system must allow educators who have set manually evaluation to see the submitted projects.

*R*21 The system must allow educators who have set manually evaluation to upload a manually score for each project.

*R*34 The system must ensure that the final score of each battle for a team falls within the range of 0 to 100.

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

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

*DA*4 Users have consistent and reliable access to the internet and the necessary technology (com- puters, software development tools) to participate in coding battles and tournaments.

G5 - Projects are evaluated in an automated way

*R*24 The system must automatically pull and analyze the latest sources from student repositories upon each commit

*R*25 The system must support automated test execution and static analysis of submitted code.

*R*26 The system must calculate battle scores based on functional aspects, timeliness, and quality level of source

*R*34 The system must ensure that the final score of each battle for a team falls within the range of 0 to 100.

*DA*5 The integration with GitHub and GitHub Actions functions correctly, allowing for seamless repository management, code submission, and automated workflow for the students' projects.

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

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

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

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

*R*18 The system must allow user to see the list of available battles.

*R*26 The system must calculate battle scores based on functional aspects, timeliness, and quality level of sources.

*R*27 The system must update battle scores in real-time as students push new commits.

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

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

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

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

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

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

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

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

3.2 External interface requirements

3.3 Use case diagrams

3.4 User's use case

3.4.1 Educator registration

ID	1
Name	Educator Registration
Actor	Educator, email API
Entry Conditions	The educator has opened the web page on his computer

Events Flow	1. On the homepage, the educator clicks on the "Register/Login" section 2. The system shows two options: • Register • Login 3. Educator selects "Registration" 4. The system shows a list of fields that the educator needs to enter: • Name • Surname • Email • Password • A checkbox to select only if you are an educator 5. Educator enters the data, selects the checkbox and accepts the "Terms of Service" 6. Educator clicks on the "Confirm" button 7. The system shows the acceptance of the registration and invites the educator to go to his mailbox to confirm the registration 8. The educator opens his mailbox, searches for the email sent by the platform and clicks on the "accept registration" link
Exit Conditions	• Registration successfully completed: now the educator's data has been entered into the database. • The educator clicks on "You already have an account: is redirected to the Login section"
Exceptions	• The educator enters an email that is already present in the database. So, after the user clicks on the "Confirm" button, the platform shows the same page with an error message inviting the user to change the email since it is already stored in the database • Educator enters an incorrect email. So, after the user clicks on the "Confirm" button, the platform shows the same page with an error message inviting the user to modify the email since it is incorrect. • The educator does not receive the registration confirmation message in his mailbox. So, he clicks on "Send a new verification email" and the system sends a new registration confirmation link to the educator's email.

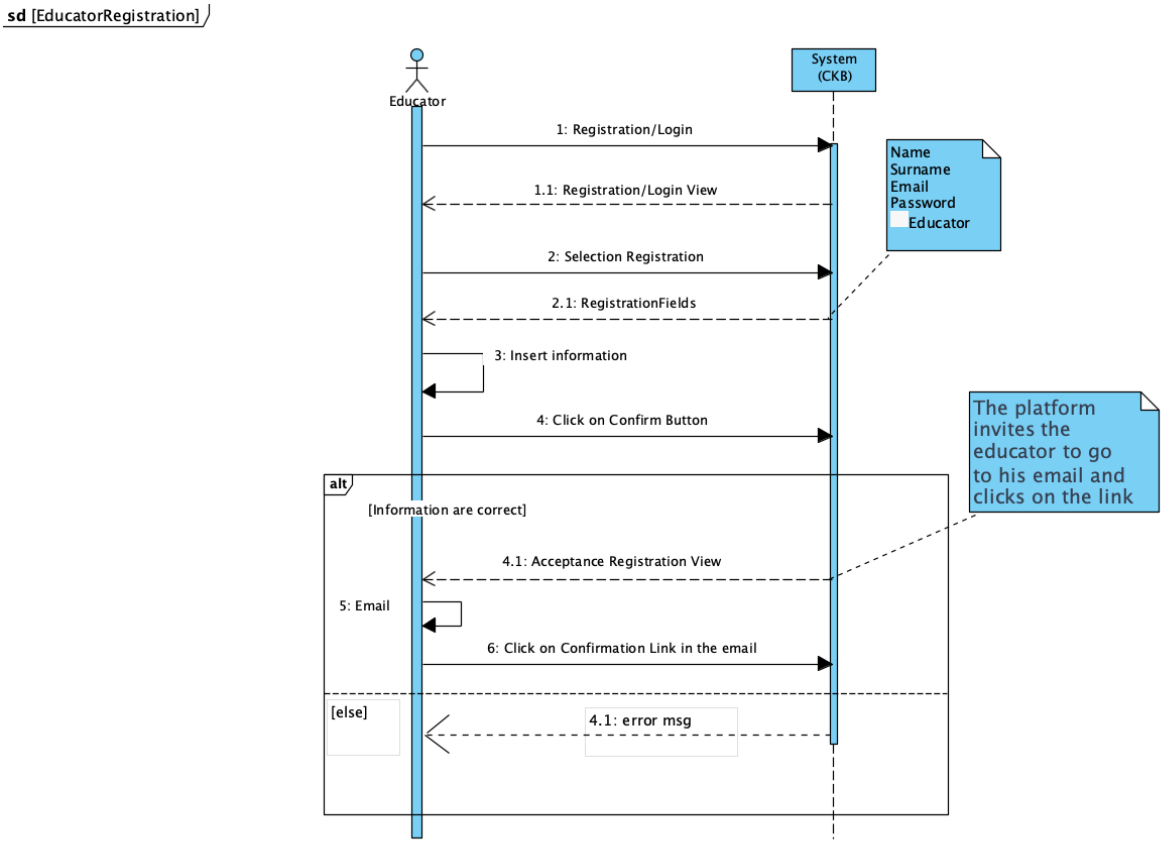


Figure 6: Didascalia dell'immagine

3.4.2 Student registration

ID	2
Name	Student Registration
Actor	Student, email API
Entry Conditions	The student has opened the web page on his computer

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

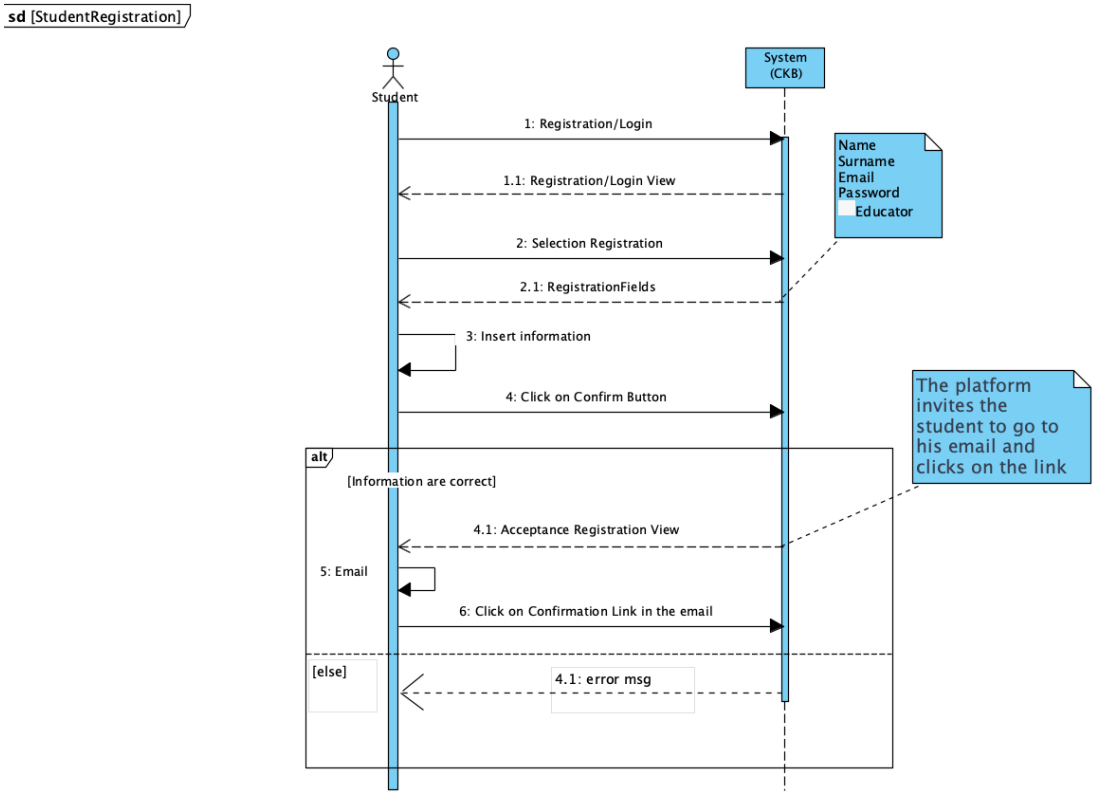


Figure 7: Didascalia dell'immagine

3.4.3 User Login

ID	3
Name	User Login
Actor	Educator or Student
Entry Conditions	The user has opened the web page on his computer

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

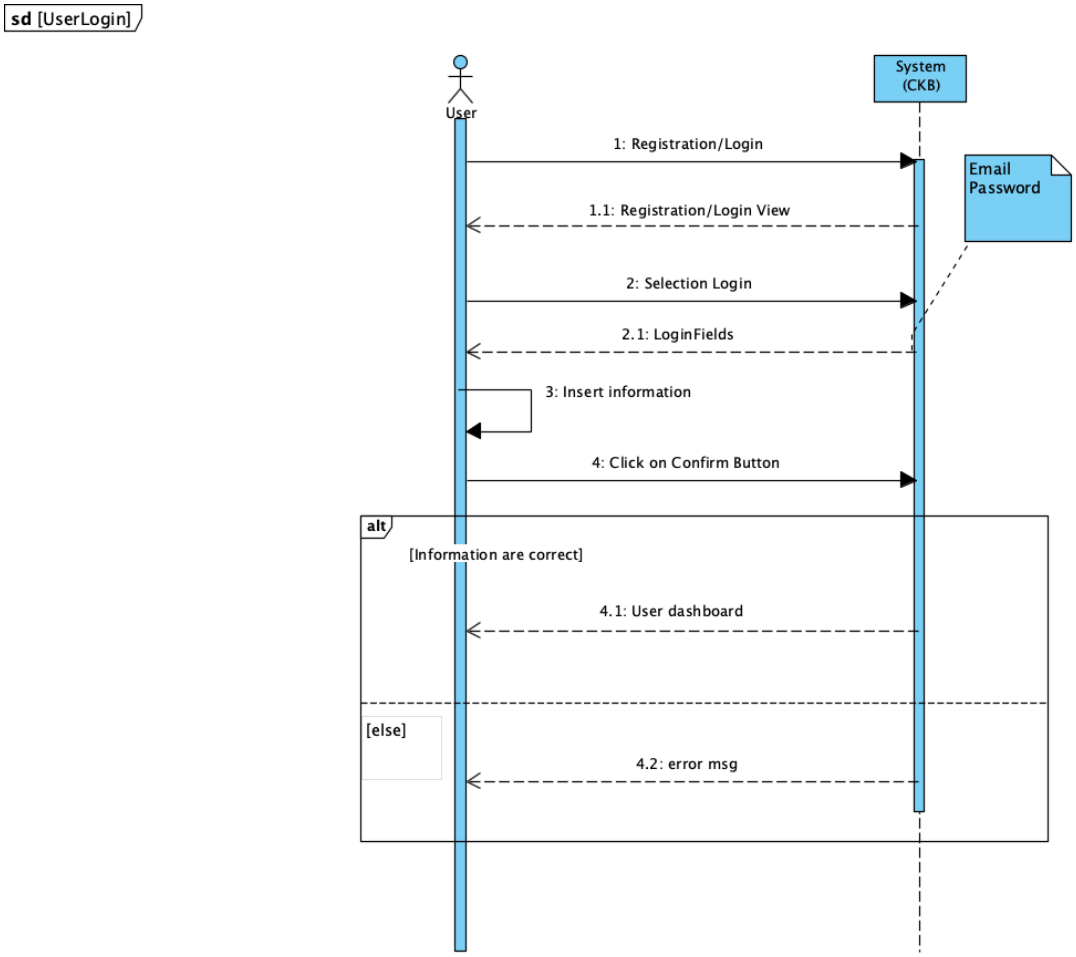


Figure 8: Didascalia dell'immagine

3.4.4 Tournament creation

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

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

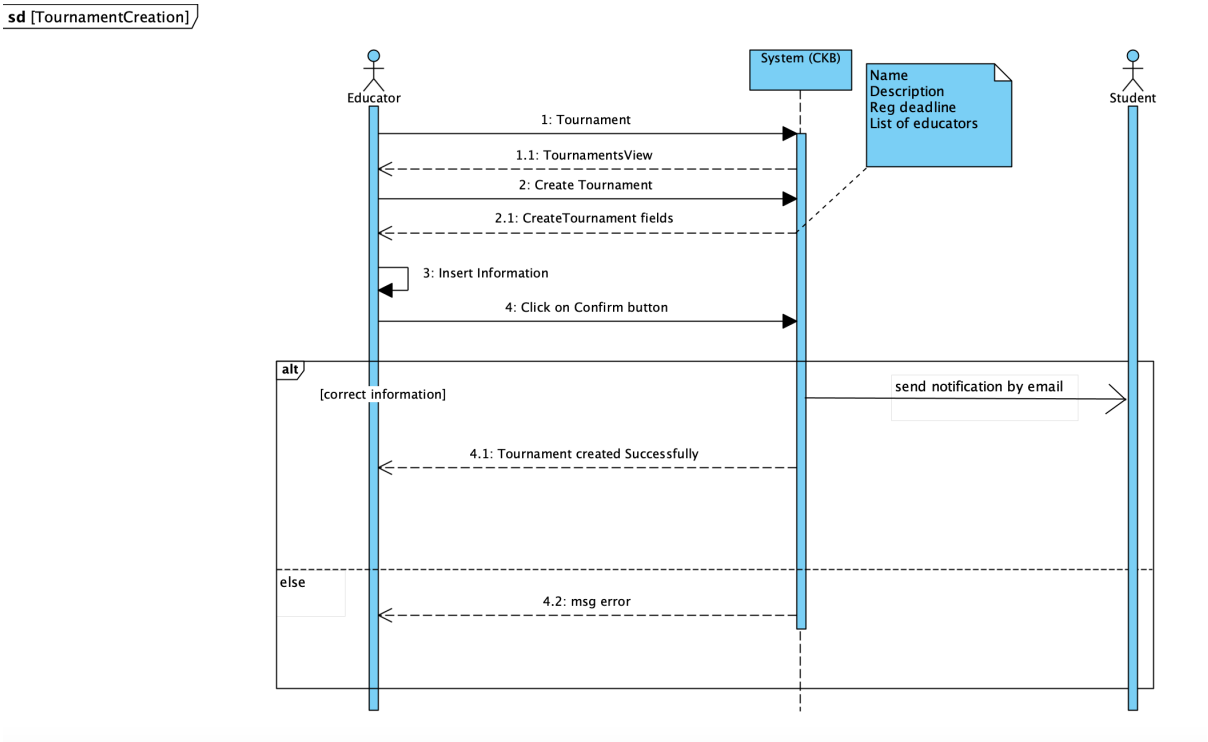


Figure 9: Didascalia dell'immagine

3.4.5 Battle creation

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

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

Exceptions	• The battle name entered by the educator is already assigned to another battle in the same tournament. Therefore, when the educator clicks the "Confirm" button, the platform displays the same page with an error message instructing the educator to change the name as it already exists.

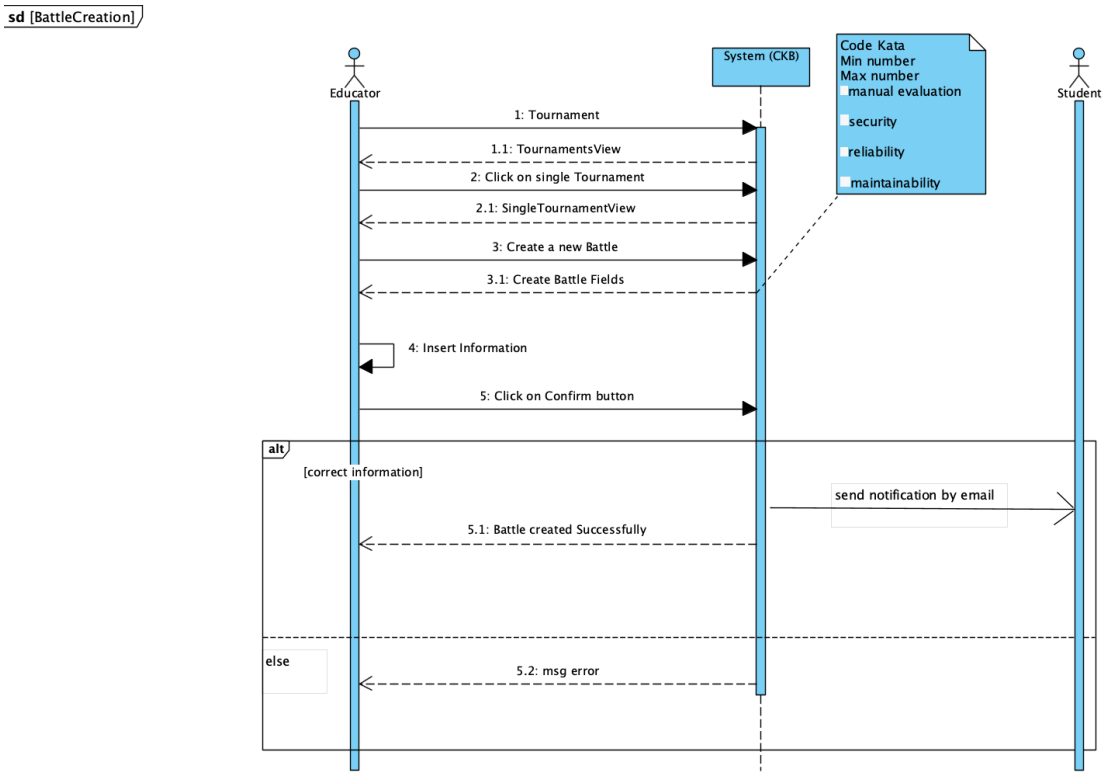


Figure 10: Didascalia dell'immagine

3.4.6 Student registers for the tournament

ID	7
Name	Student registers for the tournament
Actor	Student
Entry Conditions	The student has logged in the system.

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

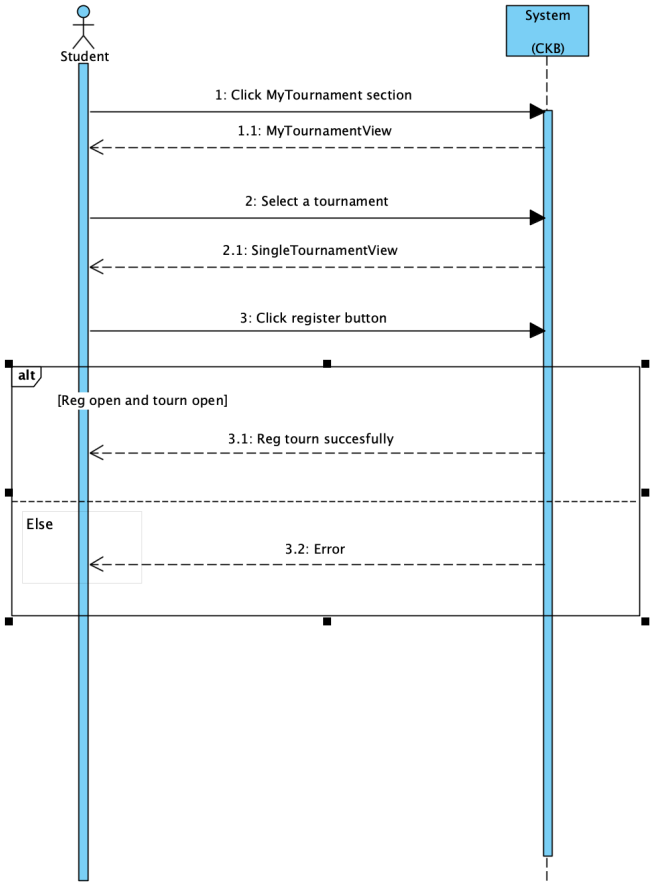


Figure 11: Didascalia dell'immagine

3.4.7 Student registers for the battle

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

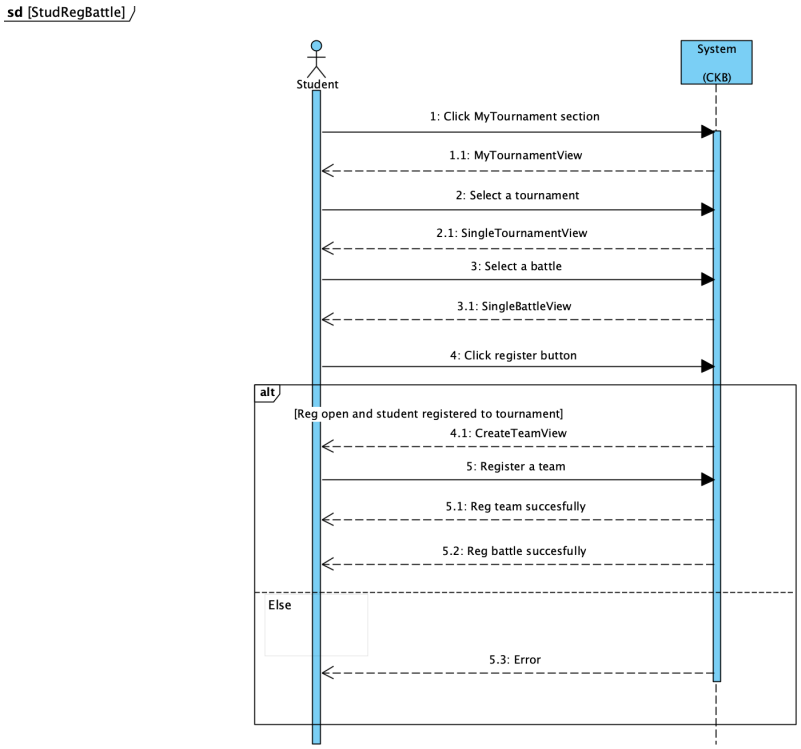


Figure 12: Didascalia dell'immagine

3.4.8 Student invites other students to create a team

ID	9
Name	Student invite other student to create a team
Actor	Student, email API
Entry Conditions	The student has clicked "Register" for a specific battle on the platform and is at the section for forming a team.
Events Flow	1. The student inserts a team name. 2. The student select the students to form the team 3. The student click on "Create team"
Exit Conditions	The system creates team and notify all the students invited via email .

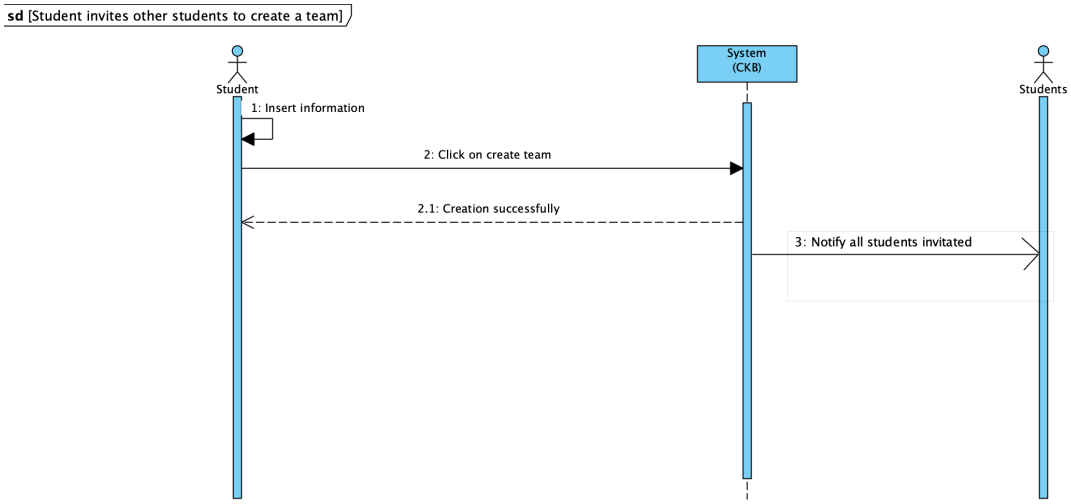


Figure 13: Didascalia dell'immagine

3.4.9 Students accept invitations and become part of the team

ID	10
Name	Students accept invitations and become part of the team
Actor	Student, email API
Entry Conditions	The student has logged in the system.
Events Flow	1. The students receive a link via email from the system. 2. The students click on the link and is reported to CKB home page. 3. The system displays a confirmation message, in which the student can choose "Yes" to accept the invitation. 4. Once the students have accepted the invitation, they are ufficialy part of the team. 5. System send a confirmation notifications via email to all team members.
Exit Conditions	• Students join the team and system updates the database accordingly.

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

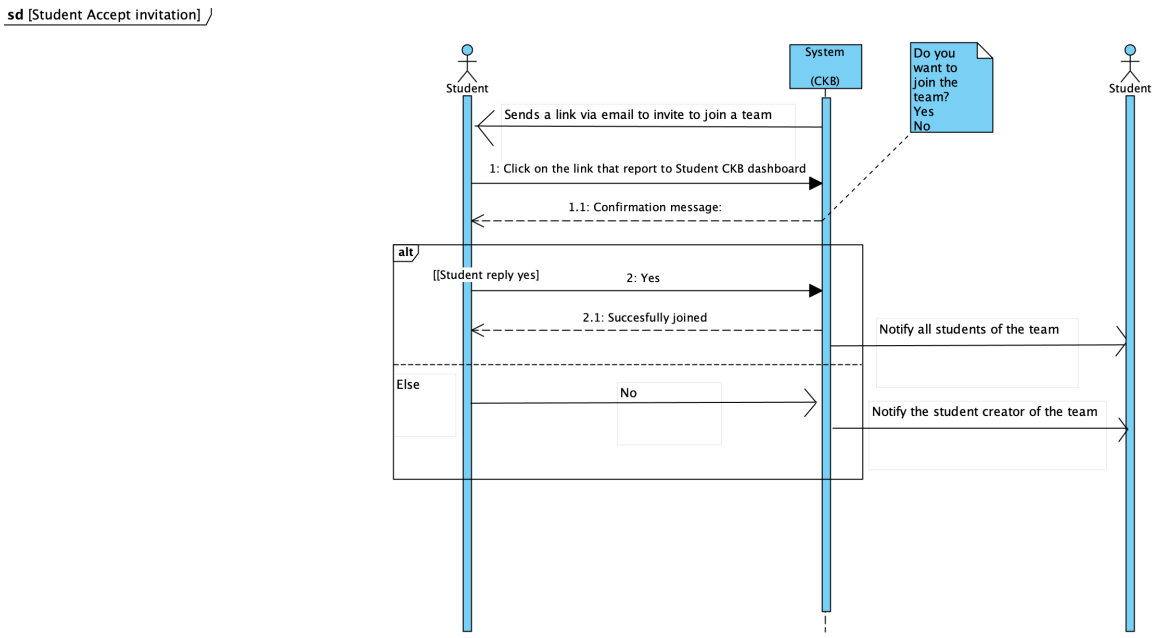


Figure 14: Didascalia dell'immagine

3.4.10 Battle Setup

ID	11
Name	Battle Setup
Actor	Student, GitHub Actions
Entry Conditions	The registration deadline for the battle expires

Events Flow	1. The platform creates a GitHub repository containing the code kata(the project) 2. The platform sends the link via email to all students who registered to the battle. 3. Students fork the repository they've been sent 4. Students set up an automated workflow through GitHub Actions
Exit Conditions	The battle environment is successfully setted up and students start coding.

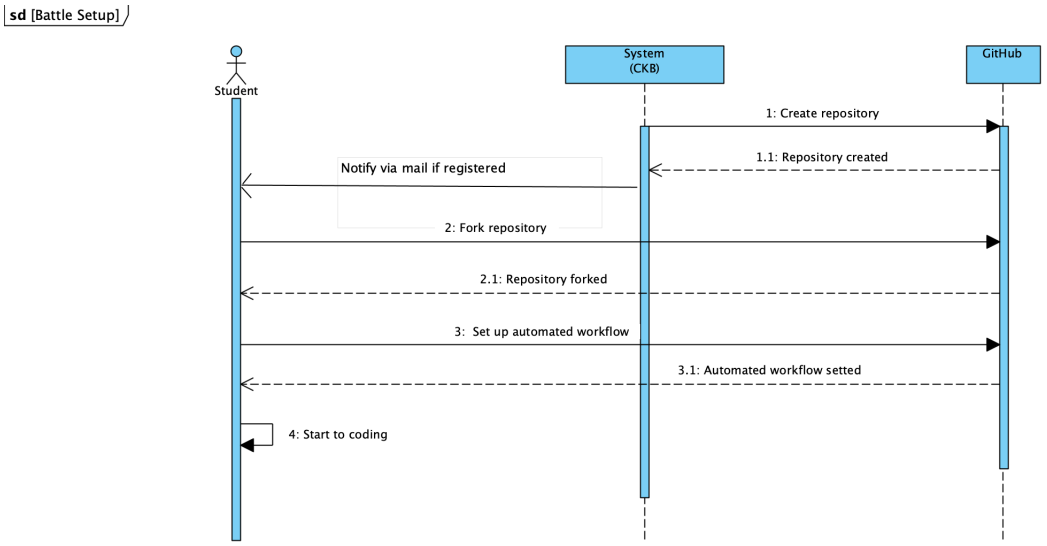


Figure 15: Didascalia dell'immagine

3.4.11 Students start to work

ID	12
Name	Students start to work
Actor	Student, GitHub Actions
Entry Conditions	The battle environment has been successfully set up.

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

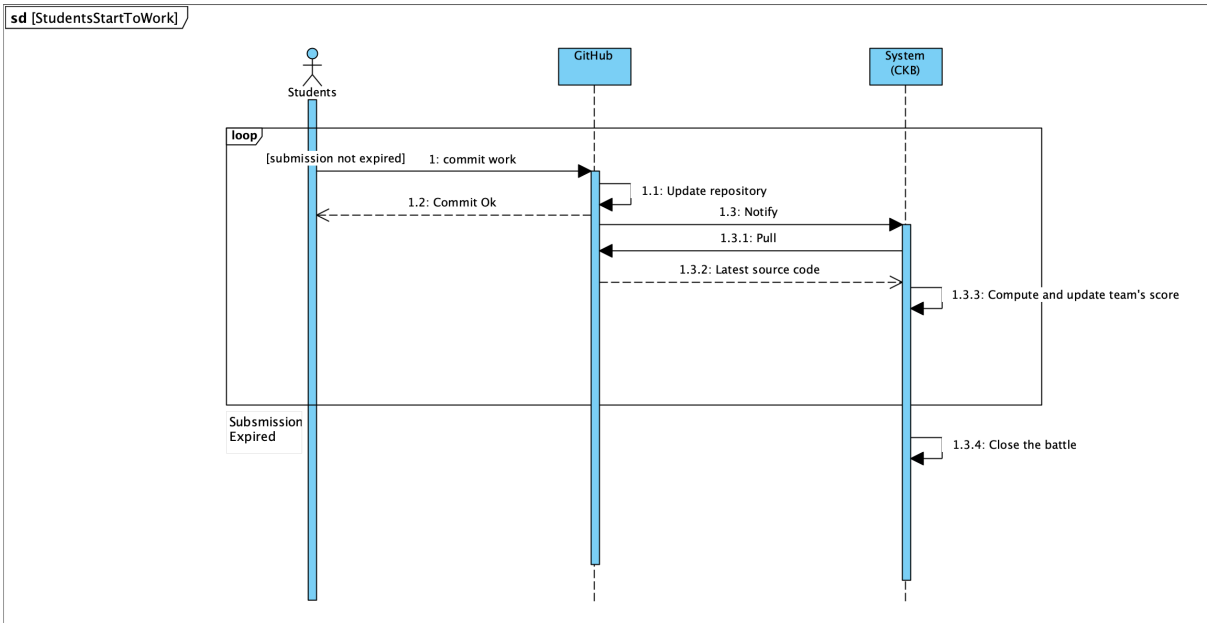


Figure 16: Didascalia dell'immagine

3.4.12 Educator evaluation during the consolidation process

ID	13
Name	Educator evaluation during the consolidation process
Actor	Educator
Entry Conditions	The Code Kata Battle has reached its submission deadline

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

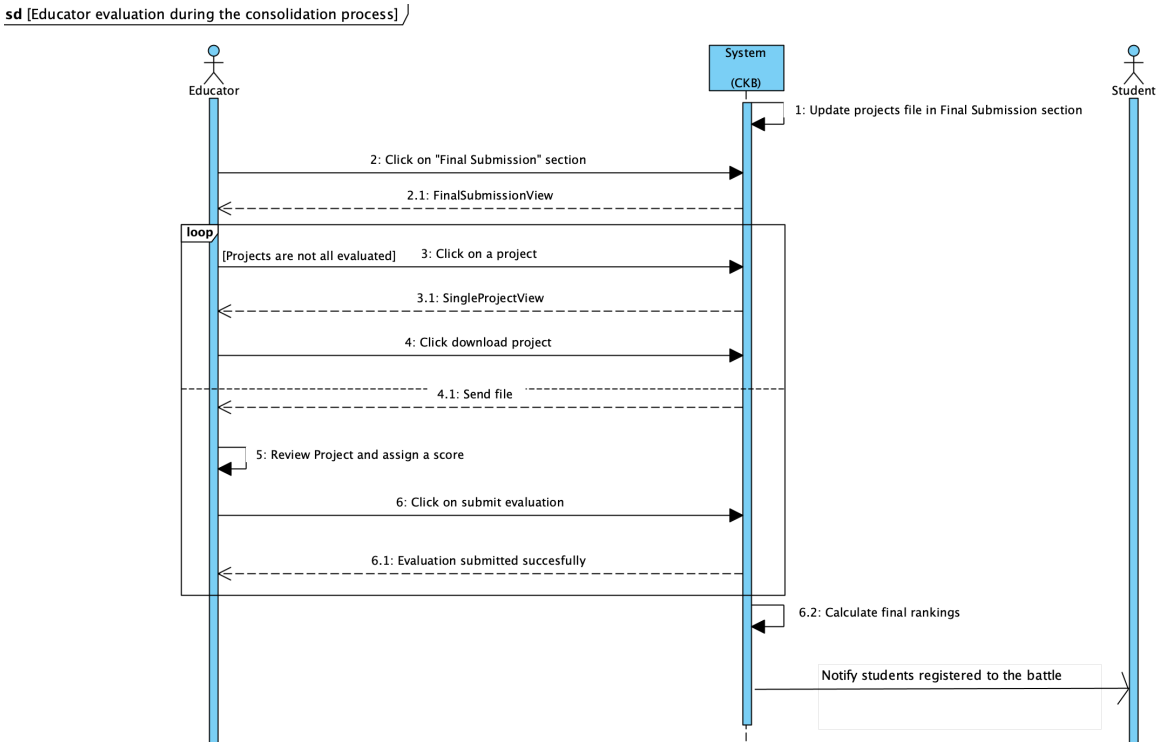


Figure 17: Didascalia dell'immagine

3.4.13 Educator monitors battle ranking

ID	14
Name	Educator monitors battle ranking
Actor	Educator
Entry Conditions	The educator has logged into the system.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on "All Tournaments". 4. The system displays a page that includes all tournaments of the platform, even those for which the educator does not have permission to create battles. 5. The educator clicks on a specific tournament. 6. The system displays the dashboard containing battles for the selected tournament to the educator. 7. The educator selects the battle they want to view the ranking of.
Exit Conditions	The system displays the battle ranking to the educator, containing the following fields: • Team ID • Team name • Team Score calculated in real-time by the system

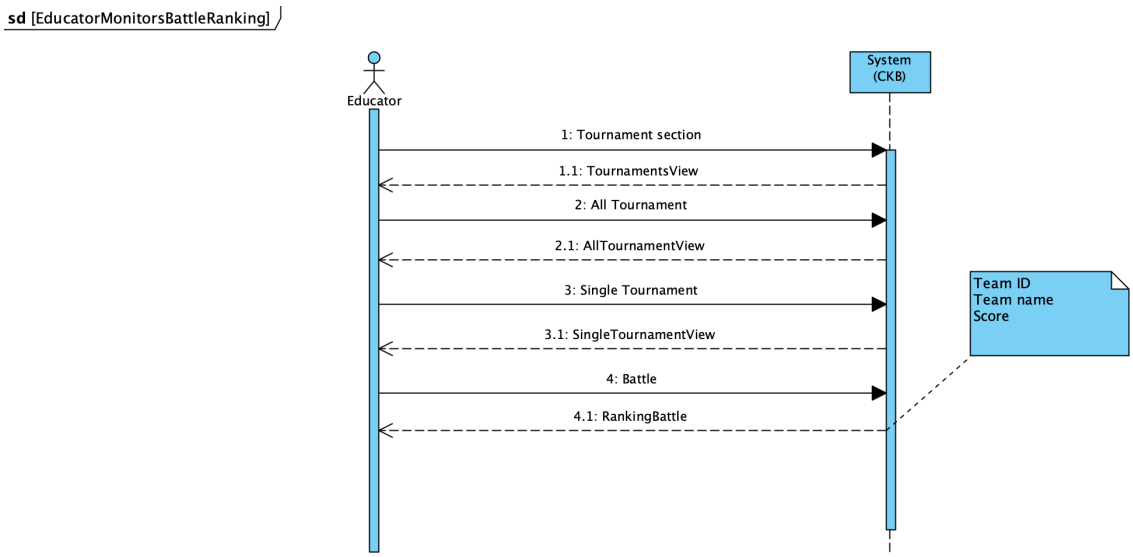


Figure 18: Didascalia dell'immagine

3.4.14 Student monitors battle ranking

ID	15
Name	Student monitors battle ranking
Actor	Student
Entry Conditions	The student has logged into the system.
Events Flow	1. The student, through the homepage, clicks on the "Tournament" section. 2. The system presents a list of all the tournaments available on the platform. 3. The student clicks on a specific tournament. 4. The system displays the dashboard containing battles for the selected tournament to the student. 5. The student selects the battle they want to view the ranking of.
Exit Conditions	The system displays the battle ranking to the student, containing the following fields: • Team ID • Team name • Team Score calculated in real-time by the system

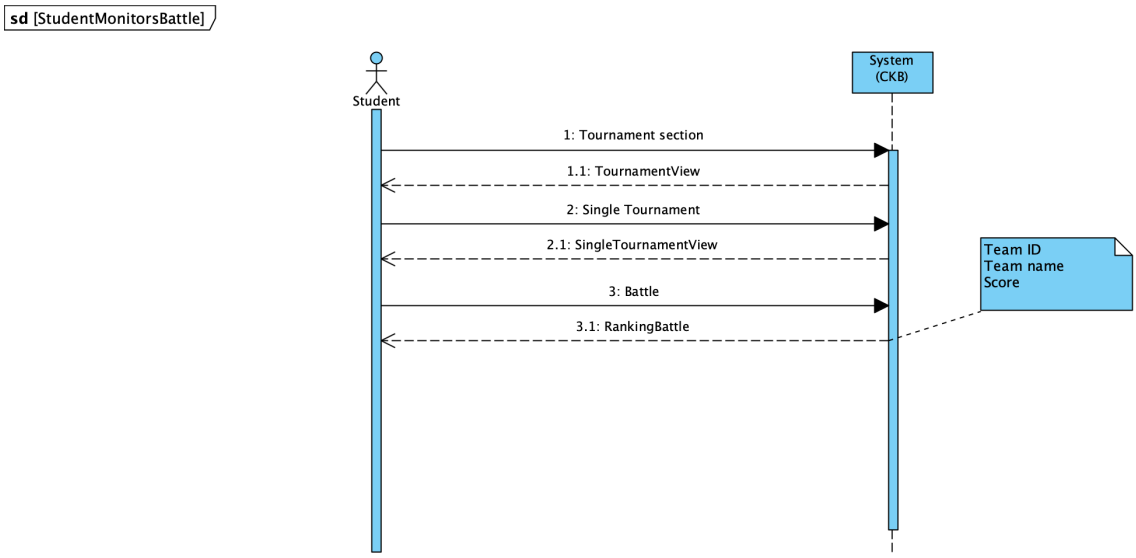


Figure 19: Didascalia dell'immagine

3.4.15 Educator monitors tournament ranking

ID	16
Name	Educator monitors tournament ranking
Actor	Educator
Entry Conditions	• The educator has logged into the system. • The tournament of interest has been created, and the educator has the necessary permissions.
Events Flow	1. The educator, through the homepage, clicks on the "Tournament" section. 2. The system presents a control dashboard that displays the tournaments created by the educator or those for which they have permission to organize battles. 3. The educator clicks on "All Tournaments". 4. The system displays a page that includes all tournaments of the platform, even those for which the educator does not have permission to create battles. 5. The educator clicks on a specific tournament. 6. The system displays the dashboard containing battles for the selected tournament to the educator and the "Tournament ranking" button. 7. The educator clicks on "Tournament ranking".

Exit Conditions	The system displays the tournament ranking containing various fields: • Student ID • Student Name • Student Surname • Student Score

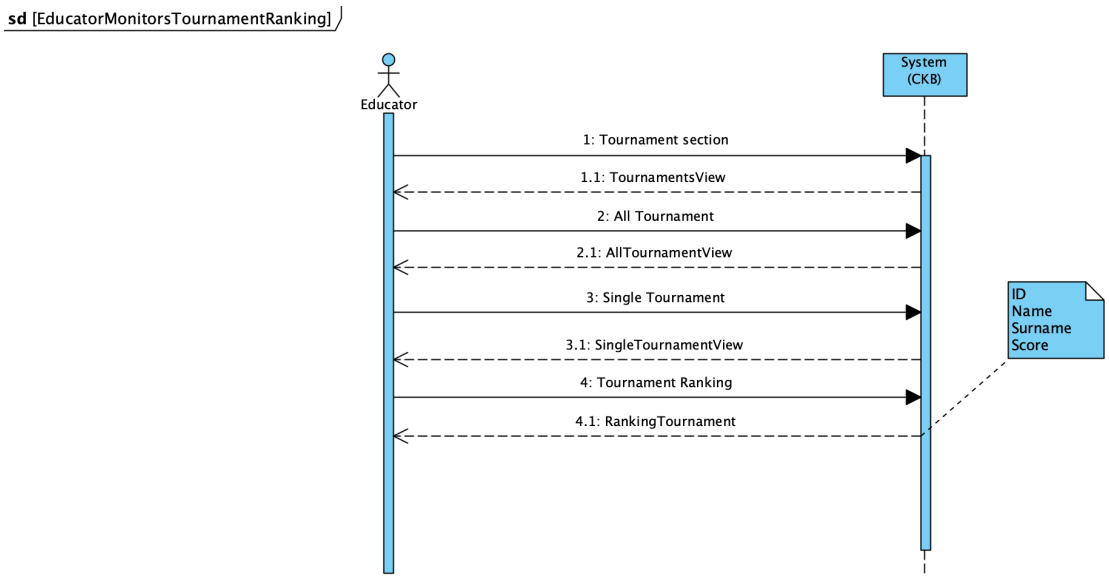


Figure 20: Didascalia dell'immagine

3.4.16 Student monitors tournament ranking

ID	17
Name	Student monitors tournament ranking
Actor	Student
Entry Conditions	The student has logged into the system.

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

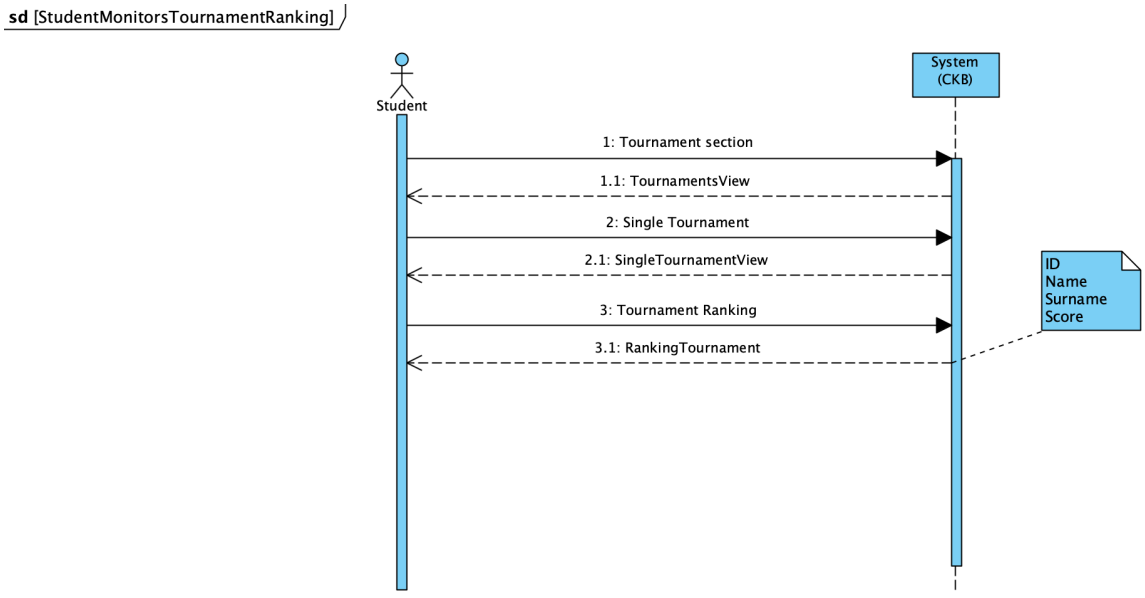


Figure 21: Didascalia dell'immagine

3.4.17 Tournament Closure

ID	18
Name	Tournament Closure
Actor	Educator

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

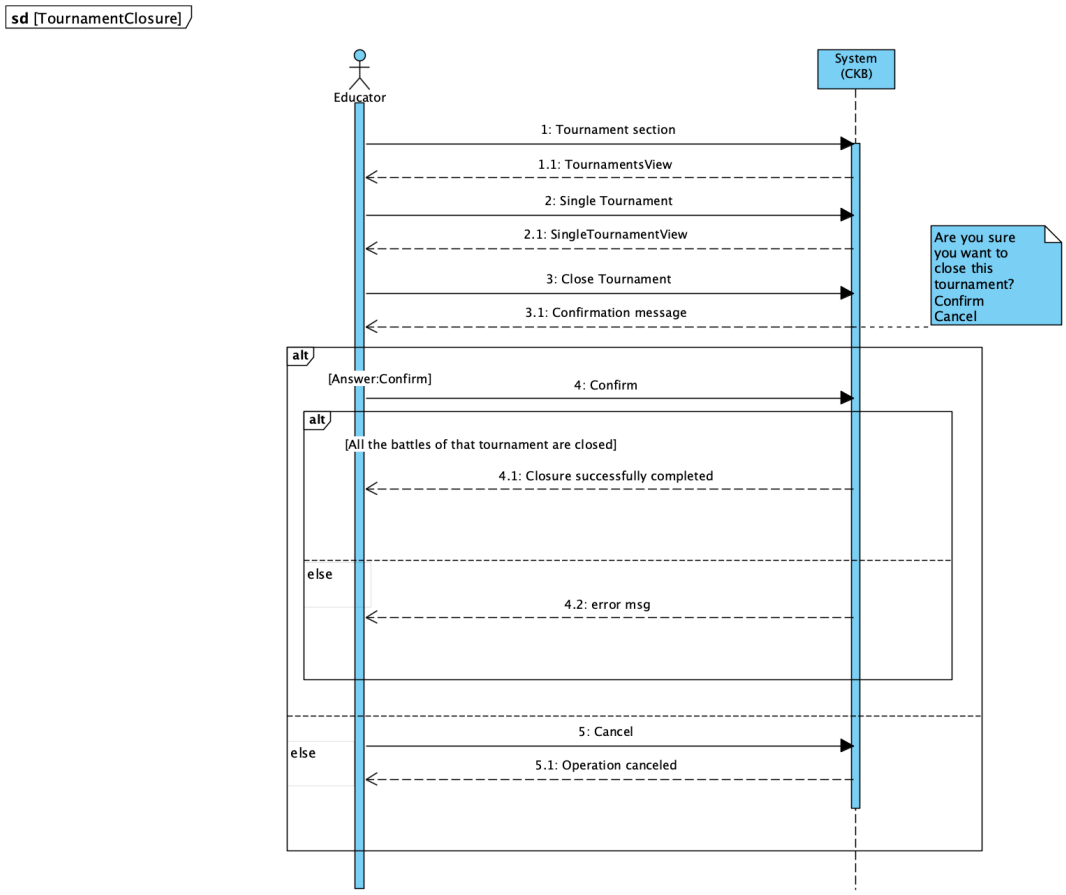


Figure 22: Didascalia dell'immagine