POLITECNICO DI MILANO

SOFTWARE ENGINEERING II

CODE KATA BATTLE PLATFORM

REQUIRMENTS ANALYSIS AND SPECIFICATION DOCUMENT

Version 1.0

Simona Cai 10752734
Yang Hao Mao 10705881
Jiaxiang Yi 10765316

Contents

1. Introduction

1.1 Purpose

The purpose of the CodeKataBattle (CKB) platform is to provide a competitive and collaborative environment for students to improve their software development skills. The platform allows educators to create programming exercises, known as code kata battles, in which teams of students compete against each other. These battles are part of larger tournaments, and students' performance is evaluated based on various factors, including the functionality of their code, timeliness of submission, and quality of their sources.

In addition to the competitive aspect, the platform also includes a gamification element in the form of badges, which are awarded to students based on their achievements. These badges, defined by educators, serve as a form of recognition and motivation for the students.

Overall, the CKB platform aims to enhance the learning experience by combining education, competition, and gamification, thereby fostering a dynamic and engaging environment for software development education.

Goal	Description
G1	Allows Students to select a language for training.
G2	Allows Students to subscribe to tournaments and battles.
G3	Allows Educators to create tournaments and battles
G4	Allows Students to practice their coding skills with code kata battles
G5	Allows Students to practice in offline mode

1.2 Scope

CKB is an educational community for computer programming. On the platform, students train on programming challenges know as code kata. These discrete programming exercises train a range of skills in a variety of programming languages and are completed within an online integrated development environment. On CKB the community and challenge progression in gamified, with users earning badges and honour for completing code kata, contributing kata, and quality solutions. CKB will alert all subscribed students new upcoming tournaments promptly.

CKB will support the work of two types of actors:

- Students

- Educators

1.2.1 World Phenomena

World phenomena refer to events that occur in the real world and are not directly detectable by the machine.

Identifier	Description
WP1	Educators creating code kata.
WP2	How Students working on code kata on their devices.
WP3	Student wants to commit their solution.
WP4	Student wants to subscribe their interested tournaments and battles.
WP5	Educator rating each student's work.
WP6	Users want to visit others profile.

1.2.2 Shared Phenomena

Shared phenomena refer to events that occur within the machine.

- Phenomena controlled by the world and observed by the machine:

Identifier	Description
SP1	User registration and authentication.
SP2	Creation of tournaments and battles by Educators.
SP3	Student subscriptions and participation of the tournaments.
SP4	Code submission and evaluation by Students.
SP5	Notifications of new tournaments to all subscribed students.
SP6	Badge creation and assignment to Student.
SP7	Educators close the tournament.
SP8	Students want to form the team to participate code kata battle.

- Phenomena controlled by the machine and observed by the World:

Identifier	Description
SP9	CKB sends notification of upcoming tournaments to all subscribed students of CKB.
SP10	CKB shows updated competition rank at the end of the battle

SP11	CKB shows gamification badges assigned to the student
SP12	CKB shows personal tournament score of each student
SP13	CKB shows verification status of student’s code to indicate whether it is correct
SP14	CKB shows reference answer provided by educator at the end of battle

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Student**: An individual engaged in learning or academic activities, typically enrolled in an educational institution or a course. In a model, a Student might have attributes like name, email, titles, and associated possible battles records.
- **Educator**: A person who provides education or instruction, often a teacher, instructor, or professor. In a system, an Educator might have attributes like name, email, tournament admin, battles creator.
- **Battle**: In an educational or competitive context, a Battle could refer to a contest or a challenge where participants compete, often in teams of one or more Student. Attributes might include battle name, tournament reference, participating teams, start and end times, and status.
- **Tournament**: A series of competitions or battles organized around a particular activity or subject. In a model, a Tournament might have attributes like tournament name, list of battles or contests, registration details, badges, and overall status.
- **Code Kata**: A term often used in programming and software development, referring to a small and focused coding exercise to practice programming skills. In a system, a Code Kata might have attributes like problem statement, language, expected outcomes, and associated test case resources.
- **Badge**: Typically a symbol or indicator of achievement, skill, or qualification. In educational or gamified systems, badges are often awarded to participants for completing tasks, reaching titles, or achieving certain standards. Attributes might include badge name, criteria for earning, and the badge's visual representation.
- **Practice Mode**: This could refer to a non-competitive or learning-oriented mode in a system where participants can practice skills, solve problems, or engage in activities without the pressure of competition or assessment. Attributes might include available exercises, and resources.

1.3.2 Acronyms

- **[CKB]:** Code Kata Battle Platform
- **[UI]:** User Interface
- **[UML]:** Unified Modelling Language
- **[RASD]:** Requirement Analysis and Specification Document

1.4 Revision History

- **Version 1.0** (22 December 2023)

1.5 Reference Documents

- Official link of codewar: https://www.codewars.com/
- The specification of the RASD and DD assignment of the Software Engineering II course, held by Professor Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli at the Politecnico di Milano, A.Y 2023/2024.
- Slides of Software Engineering 2 course on WeBeep.

1.6 Document Structure

Mainly the current document is divided in 4 chapters, which are:

1. **Introduction:** it aims to describe the environment and the demands taken into account for this project. In particular it's focused on the reasons and the goals that are going to be achieved with its development.
2. **Overall Description**: it's a high-level description of the system by focusing on the shared phenomena and the domain model (with its assumption).
3. **Specific Requirements:** it describes in very detail the requirements needed to reach the goals. In addition, it contains more details useful for developers (i.e information about HW and SW interfaces).
4. **Formal Analysis**: this section contains a formal description of the main aspect of the World phenomena by using Alloy.
5. **Effort Spent**: it shows the time spent to realize this document, divided for each section.
6. **References:** it contains the references to any documents and to the Software used in this document.

2. Overall Description

2.1 Product perspective

2.1.1 Scenarios

1. **Educator creates Tournament:**

The technology company CPE (CercoProgrammatoriEccelenti) is currently seeking a group of talented university students to join their company as programmers. Mr. Verdi, the HR manager, know from a friend about a platform called CKB, where enthusiasts can participate in various challenges. Verdi decides to register as an organizer on the platform. After successful registration and account verification, he organizes a programming tournament named "CPE Tournament." Verdi sets the tournament start and end times, battles registration start and end times, the maximum number of battles for join it, and the total achievement to be awarded and internship opportunity.

2. **Educator creates Battle:**

Dr. Rossi, a professor at an engineering school, is usually invited to create various programming exercises. One day, the invitation by CPE, Dr. Rossi creates a competition within the CPE Tournament called "CodingForEver." To meet CPE's recruitment requirements, Dr. Rossi skillfully operates the platform. He selects online compilers such as Java, C, C++, C#, provides the content of the challenge, competition test files, and sets the number of students in a team, the start, and end times of the battle.

3. **Student participates in Battle:**

Alice is a programming enthusiast, usually like to practice in CKB platform and participate in some competitions, receives an invitation to the "CPE Tournament." After reviewing the competition details and rewards, she decides to participate in a battle named "LoneCode." She enters the CKB website through the provided link on her smartphone, logs in with her private account email and password, selects the "LoneCode" battle that she likes and seen the requirements, and clicks "Join." She receives a confirmation email, "Registration Successful." Later, she also notices the "GoCode" competition, which coincides with "LoneCode" but ends later. After careful consideration, she decides to participate in this competition as well.

4. **Student invites other students to participate in the Battle:**

Bob and Calvin, after exploring the "CPE Tournament," decide to participate in a battle named "Trinity." However, while forming a team, they realize that it requires at least three members. Bob thinks of inviting his programming friend David. After getting David's agreement over the phone and obtaining his email, Bob, during team creation, inputs David's email in the invitation field and sends an invitation. David receives the invitation, clicks "Agree to Join," and enters

the CKB website through the link to confirm. Bob then receives a "Team Created Successfully" message, and the team successfully registers for the competition.

5. **Team of students test project:**

"ViNCiaMo" is a team of four students. On the day of the battle, they divide the coding tasks among themselves. Vittorio sees the real-time leaderboard, and they are in the second position. This is their last registered battles, but they have one test case left. After a final submission attempt just before the deadline, they pass the test. At the last moment, all team members, Nicola, Carlo, and Monica, see "Pass" on their interfaces. After clicking the "Submit" button, they celebrate excitedly. When the time is up, each team member receives a confirmation email containing details like the number of completed tests, time taken, submission score, and provisional personal ranking.

6. **Educator evaluates Tournament:**

Dr. Rossi receives a notification at 12 PM that the time for the "CPE Tournament" has expired. He logs into the CKB platform using a computer, clicks on the consolidation stage button, selects the top 10 scores, views their profiles, and finds that the student in the second position, Bob, has completed the most battles. Dr. Rossi decides to give him a bonus (the student who completes the most battles can receive an additional bonus). After a second check, the platform reassigns rankings. Dr. Rossi reviews it and selects "Close Consolidation Stage." After confirming again, the platform sends the final ranking information to all participants via email.

7. **Student receives results:**

Vittorio, during dinner, receives a "CPE Tournament final rank" message. Upon opening it and logging in, he sees that his ranking, which was first at submission, has dropped to the second position. Confused, he notices a button below the email saying, "Contact with Educator." The platform provides him with Dr. Rossi's email. After understanding the situation, Vittorio clicks "Confirm Result" (the platform automatically selects it if not clicked during the publication phase).

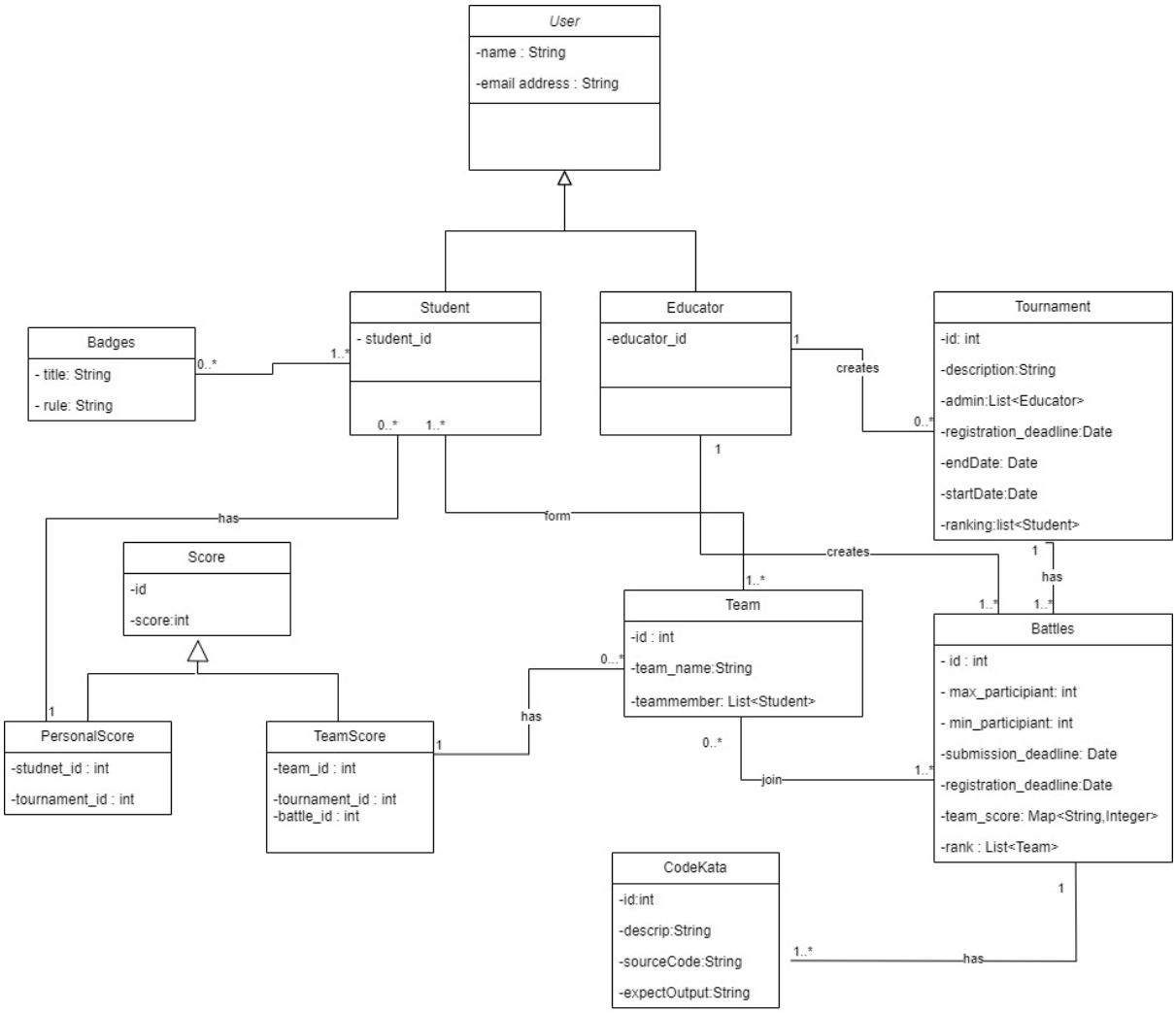
8. **Educator closes Battle and Students receive rewards:**

After the Publication Stage, Dr. Rossi logs into the CKB platform, clicks "Close Tournament," and after confirming, the platform notifies all participating students and educators that the competition is officially over. At this point, the platform automatically distributes rewards to students who meet the ranking requirements and updates their individual competition information, titles, etc.

9. **Student can practice on the platform (additional function):**

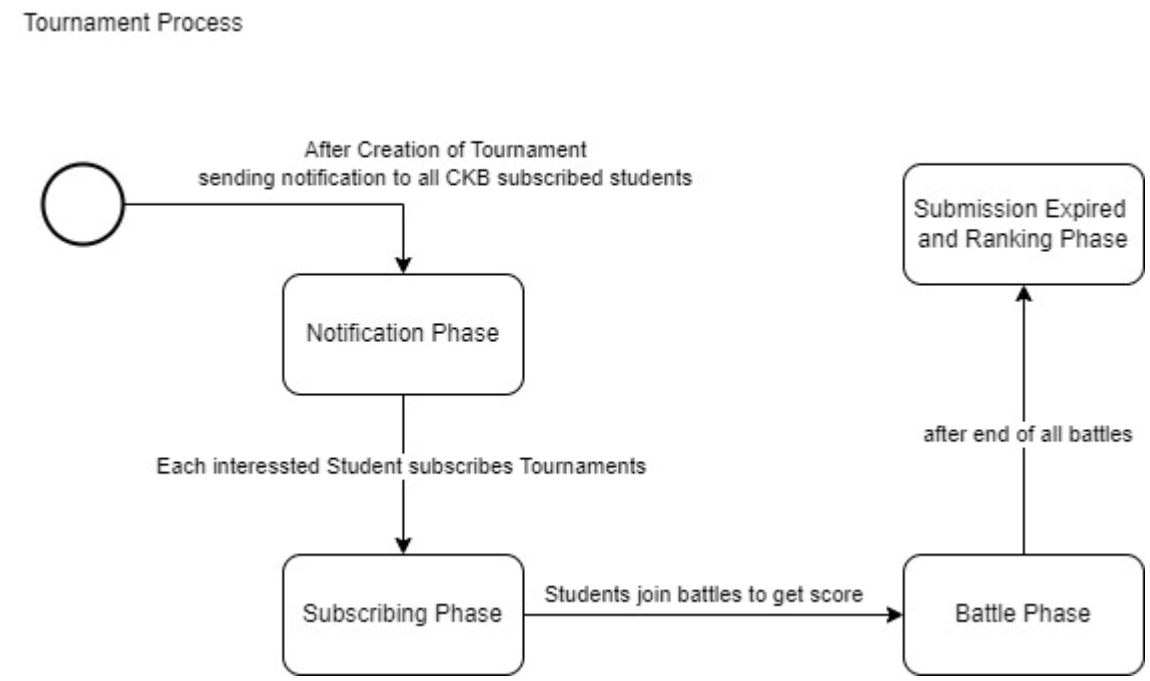
Kevin, preparing for a programming exam, knows from his classmate Alice that the CKB platform has many past competition problems for good practicing programming skills. Kevin decides to register on the platform and, under the "Practice on Yourself" section, sees various competition problems and effective answers published by educators. Kevin clicks on the "CPE Tournament " and "LoneCode" battle, attempting to write and submit his code to the platform. After the platform accepts Kevin's submission in a text file format, it compiles it online, and the interface displays the compilation result.

2.1.2 Domain class diagram



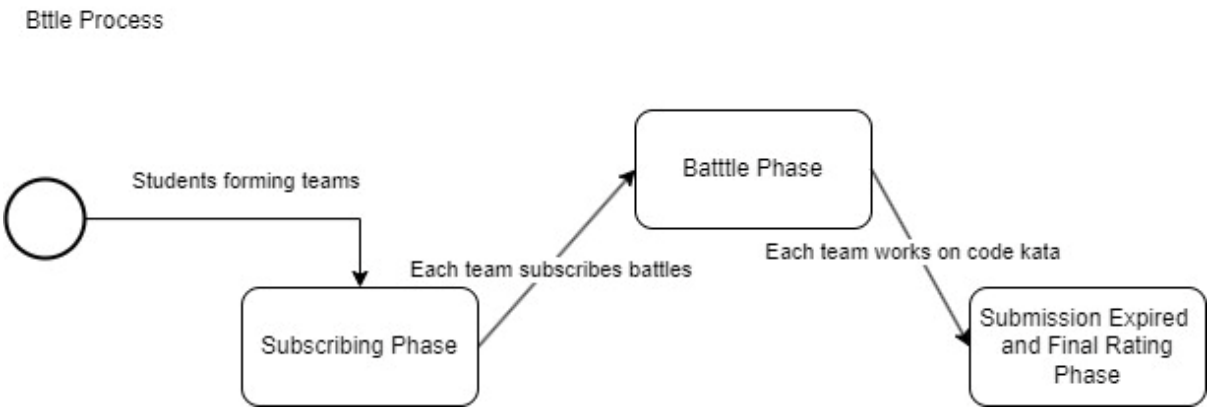
2.1.3 State Diagram

Tournament Process



This state diagram illustrates how a Tournament in CKB can progress through different phases, reflecting the lifecycle of a tournament from subscription to battle execution, rating, and notification. The actual states and transitions may involve additional complexities based on the detailed behaviour of the CKB platform.

Battle Process



This state diagram captures the progression of a battle process in CKB, reflecting the phases of Subscribing, Team Formation, Battle Execution, and Rating & Ranking. The actual states and transitions may involve additional details based on the specific behavior of the CKB platform.

2.2 Product Functions

2.2.1 Students Functionalities

Authentication and Profile Management:

Additionally, providing access to personal profiles enhances user engagement and facilitates a sense of identity within the community. The invitation functionality promotes interaction among students, encouraging them to collaborate and participate in competitions together. The system generates notifications to keep users informed about the success or failure of their actions, ensuring transparency and a smooth user experience.

1. **User Registration and Login:**
    - **Description:** Provide students with the capability to create user accounts through registration and access their accounts through login credentials.
    - **Functionality:**
        i. Allow students to register by providing necessary information.
        ii. Validate registration information for accuracy and completeness.
        iii. Notify students of successful or failed registration.
        iv. Enable students to log in using their registered credentials.
        v. Send notifications for successful or failed logins.
2. **Personal Profile Viewing:**
    - **Description:** Allow students to access and view their personal profiles, providing them with an overview of their account details and activity.
    - **Functionality:**
        i. Enable students to navigate to their profile section.
        ii. Display relevant information such as username, email, and participation history.
        iii. Facilitate a personalProfile interface for easy navigation.
3. **Invitation Functionality:**
    - **Description:** Empower students to invite their peers to participate in battles, fostering a sense of community and collaboration.
    - **Functionality:**
        i. Allow students to send invitations to specific battles.
        ii. Implement constraints to manage the number of invitations according to the number of participants.
        iii. Provide notifications to invitees with details of the invitation.

Battle Participation Features:

The battle participation features aim to enhance the overall engagement of students on the platform. By offering choices in programming languages, facilitating seamless tournament and battle subscriptions, and providing access to GitHub repositories, students can actively participate in coding challenges. Real-time score viewing, notifications, and rankings add a competitive and interactive dimension to their experience. The ability to explore past battles

and engage in post-battle activities, such as practice mode and code testing, contributes to continuous learning and skill development.

1. **Programming Language Selection:**
   - **Description:** Provide students with the flexibility to choose their preferred programming language for training.
   - **Functionality:**
     i. Offer a programming-language interface where students can select their desired programming language.
     ii. Ensure a diverse range of languages to cater to individual preferences and learning goals.

2. **Tournaments Subscription:**
   - **Description:** This feature enables students to participate in tournaments by subscribing to them.
   - **Functionality:**
     i. Allow students to explore and view details of individual tournaments available on the platform.
     ii. Provide a straightforward subscription process for students to sign up for their chosen tournaments.
     iii. Ensure a tournaments interface for easy navigation through the list of available tournaments.

3. **Battles Subscription:**
   - **Description:** This functionality allows students to subscribe to specific battles within the tournaments they have already joined.
   - **Functionality:**
     i. Provide an option for students to explore and select specific battles within the tournaments they are interested in.
     ii. Facilitate a distinct subscription process for battles, ensuring clarity in the battle selection and enrollment process.
     iii. Ensure seamless navigation to find and subscribe to battles within the chosen tournament.

4. **GitHub Repository Access:**
   - **Description:** Facilitate students' access to the GitHub repository relevant to their participating battle.
   - **Functionality:**
     i. Platform sends students a GitHub link associated with their selected battle.
     ii. Students utilize the provided link to upload their battle-related code, ensuring a seamless integration between the platform and GitHub for repository management.

5. **Score Viewing and Notifications:**
   - **Description:** Provide students with real-time access to their competition scores and notifications for relevant events.
   - **Functionality:**

i. Display a dedicated section for students to view their scores obtained in battles.
ii. Implement a notification system to alert students about various events such as new tournaments, battle updates, or score announcements.

6. **Rankings Exploration:**
   - **Description:** Enable students to explore and view rankings within both individual battles and overall tournaments.
   - **Functionality:**
     i. Provide a rankViewing interface for students to access battle rankings.
     ii. Differentiate between rankings within specific battles and broader tournament rankings.

7. **Past Battles Overview:**
   - **Description:** Allow students to revisit and review details of past battles they participated in.
   - **Functionality:**
     i. Provide a comprehensive list of past battles in which the student has taken part.
     ii. Include details such as battle names, dates, and performance summaries.

8. **Practice Mode Selection:**
   - **Description:** Enable students to opt for a practice mode after completing a battle.
   - **Functionality:**
     i. Allow students to choose the practice mode for ongoing learning and skill enhancement.
     ii. Provide a practice interface for uploading and testing code within the practice mode.
     iii. Display practice rankings to highlight individual performance in a non-competitive environment.

2.2.2 Educators/Platform Functionalities

Educator Registration and Management Features:

The Educator Registration and Management features involve the processes related to the registration and login of educators. Educators have dedicated registration and login functionalities tailored to their roles.

Educator Registration and Login:

- **Description:** Provide educators with the ability to register and log in to the CodeKataBattle platform.
- **Functionality:**
  i. Implement a dedicated and secure registration process for educators, allowing them to create accounts with relevant information.
  ii. Develop an educatorLogin interface that enables educators to access their accounts with proper authentication.

iii. Ensure that the registration and login interfaces meet the specific needs of educators, emphasizing security and ease of use.

Tournament and Battle Management Features:

The Tournament and Battle Management features cater to the needs of educators, providing them with tools to organize and oversee coding tournaments and battles. The creation of tournaments is a fundamental aspect, allowing educators to set the stage for engaging coding challenges. The inclusion of permission management ensures collaborative tournament creation, where multiple educators can contribute. The creation of battles within tournaments is enriched with various configuration options, from uploading code kata to defining participant limits and deadlines. Grading parameter configuration, including security percentages, adds a layer of customization to meet the specific requirements of each battle. These features collectively contribute to a robust and flexible system for educators to curate coding challenges tailored to their educational goals.

1. **Tournament Creation and Permissions:**
   - **Description:** Empower educators to create tournaments and manage permissions for battle creation.
   - **Functionality:**
     i. Provide a tournament interface for educators to create new tournaments on the platform.
     ii. Implement a permission system allowing educators to grant specific individuals the authority to create battles within a tournament.
     iii. Include options for setting titles and rules for tournaments, referred to as "tournament badges."
2. **Battle Creation and Configuration:**
   - **Description:** Facilitate educators in creating battles with diverse configuration options.
   - **Functionality:**
     i. Enable educators to upload code kata with corresponding answers (answers revealed after the competition).
     ii. Allow educators to set participant limits for each battle to manage the number of participants.
     iii. Provide options to set registration deadlines, ensuring timely and organized battle sign-ups.
     iv. Implement submission deadlines to define the period during which participants can submit their solutions.
     v. Allow educators to configure grading parameters, including security percentages, to customize the evaluation process.

Notification and Communication Features:

The Notification and Communication features are integral to keeping students informed and engaged during their participation in tournaments and battles on the CodeKataBattle platform. These features contribute to a seamless user experience by providing timely updates on events, GitHub repository links, and battle outcomes. Furthermore, the system allows insight into user

preferences and actions through notifications related to specific choices, such as opting for practice mode.

1. **Tournament Creation Notification:**
    - **Description:** Notify students about the creation of a new tournament on the platform.
    - **Functionality:**
        i. Automatically send notifications to all registered students upon the creation of a new tournament.
        ii. Include essential details such as the tournament name, start date, and relevant information.
        iii. Ensure timely delivery of notifications to maximize student awareness.
2. **Upcoming Battles Notification:**
    - **Description:** Keep students informed about upcoming battles.
    - **Functionality:**
        i. Automatically send notifications to students in advance of upcoming battles.
        ii. Include details such as battle names, dates, and other relevant information.
        iii. Allow students to plan their participation and preparation accordingly.
3. **GitHub Repository Link Notification:**
    - **Description:** Notify students of the GitHub repository link associated with a battle after the registration deadline.
    - **Functionality:**
        i. Send notifications to registered participants with the GitHub repository link.
        ii. Ensure students have access to the repository for code submission and collaboration.
        iii. Include clear instructions on utilizing the GitHub repository for the battle.
4. **Tournament/Competition End Notification:**
    - **Description:** Inform students about the conclusion of a tournament or individual battle.
    - **Functionality:**
        i. Automatically send notifications to all participants when a tournament or battle ends.
        ii. Include details such as final scores, rankings, and any post-battle activities.
        iii. Provide a summary of the battle and acknowledge participant efforts.

Evaluation and Scoring Features:

The Evaluation and Scoring features are designed to ensure a comprehensive and fair assessment of student code on the CodeKataBattle platform. The automatic code evaluation functionality leverages GitHub commits and practice mode submissions to provide timely and objective feedback. This includes the extraction, analysis, testing, and calculation of scores

based on various criteria. Educators are granted access to view student code in GitHub repositories, allowing for a deeper understanding of the submitted work.
For scenarios where manual evaluation is required, educators have the flexibility to assess code after the submission deadline. This manual evaluation involves in-depth analysis, testing, and the application of personalized criteria to determine scores. The combination of automatic and manual evaluation ensures a thorough assessment process that considers both objective and subjective aspects of student performance. Overall, these features contribute to a robust evaluation and scoring system within the CodeKataBattle platform.

- **Automatic Code Evaluation:**

1. **GitHub Commits and Practice Mode Evaluation:**
    - **Description:** Automatically assess student code submitted through GitHub commits or during practice mode.
    - **Functionality:**
        i. Implement a system to extract the latest code from student GitHub repositories.
        ii. Analyze and test the code against predefined criteria, such as code kata.
        iii. Calculate scores based on functional aspects, timeliness, and quality levels.
        iv. Record and upload/update the automatically generated scores to the platform.

2. **Educator Code Viewing:**
    - **Description:** Allow educators to view student code in GitHub repositories for comprehensive evaluation.
    - **Functionality:**
        i. Provide educators with access to the code submitted by students through GitHub commits.
        ii. Facilitate a codeViewing interface for educators to navigate and review the code.
        iii. Ensure secure and controlled access to maintain the integrity of the evaluation process.

- **Manual Code Evaluation by Educators:**

  Manual Code Evaluation After Submission Deadline:
    - **Description:** Enable educators to manually evaluate student code after the submission deadline.
    - **Functionality:**
        i. Allow educators to analyze and test the submitted code manually.
        ii. Calculate scores based on a personalized assessment, considering factors beyond automated evaluations.
        iii. Provide educators with the ability to enter and send manual scores to the platform.

Result and Ranking Management Features:

The Result and Ranking Management features are designed to streamline the post-battle processes on the CodeKataBattle platform. After each battle, the platform undertakes actions

to record scores, update tournament standings, and maintain individual student profiles with relevant statistics. The announcement and updating of scores contribute to transparency and recognition for participants.
Educators have the capability to access and view rankings within both battles and tournaments, allowing them to gauge student performance and identify areas for improvement. The combination of these features creates a dynamic and engaging environment for both students and educators, fostering healthy battle and continuous learning within the CodeKataBattle platform.

1. **Post-Battle Actions:**
    - **Description:** Enable the platform to perform various actions after each battle concludes.
    - **Functionality:**
        i. Record and store battle scores, ensuring an accurate and comprehensive database.
        ii. Add or update tournament scores based on the results of individual battles.
        iii. Record the number of code uploads by students and update the counts in their individual profiles.
        iv. Announce and update battle and tournament scores on the platform.
        v. Send battle ranking notifications to participating students, providing them with feedback and recognition.
        vi. Upload solutions for practice mode to enhance the learning experience.
2. **Educator Access to Rankings:**
    - **Description:** Allow educators to access and view rankings within both individual battles and overall tournaments.
    - **Functionality:**
        i. Provide educators with a rankViewing interface to explore rankings within specific battles.
        ii. Facilitate easy navigation to view rankings within broader tournament categories.
        iii. Display detailed information, including participant names, scores, and any additional relevant data.
        iv. Ensure secure and controlled access to ranking information.

Tournament Closure and Badge Awarding Features:

The Tournament Closure and Badge Awarding features are crucial components of the CodeKataBattle platform, offering educators control over tournament closure and providing students with recognition through badges.
Educators can initiate the closure of a tournament, ensuring that this action is intentional and well-regulated. After each tournament, a series of post-tournament actions are executed. Students receive notifications regarding the closure of the tournament, and the platform checks if they meet the badge requirements. If criteria are met, badges are uploaded or updated on individual student profiles, acknowledging their achievements.

These features contribute to the efficient and organized management of tournaments, providing closure when necessary and recognizing students for their accomplishments through the awarding of badges.

1. **Tournament Closure:**
    - **Description:** Empower educators to officially close a tournament on the CodeKataBattle platform.
    - **Functionality:**
        i. Provide educators with the authority to close a tournament through a dedicated interface.
        ii. Ensure that closure is a controlled and deliberate action, preventing accidental closures.
        iii. Include necessary checks and confirmations to verify the intent of the closure.
2. **Post-Tournament Notifications and Badge Awarding:**
    - **Description:** Implement actions to be performed after each tournament concludes.
    - **Functionality:**
        i. Notify students of the closure of the tournament, providing them with relevant information.
        ii. Conduct checks to determine if students meet the current tournament's badge requirements.
        iii. Upload or update badges to individual student profiles based on the badge criteria.
        iv. Ensure that badge awarding is accurate and aligned with the achievements of participating students.

2.3 User Characteristics

The CKB platform caters to two main user roles: students and educators. Each user's experience is tailored to their respective needs and objectives within the platform.

2.3.1 Students

Students engage with the platform to enhance their coding skills through code kata battles. Their experience revolves around participation in challenges, collaborating with peers, and submitting solutions within specified deadlines. Students also have the opportunity to view and showcase earned badges on their profiles, reflecting their achievements and participation in various tournaments.

2.3.2 Educators

Educators play a crucial role in creating and overseeing the learning environment on the CKB platform. They have the responsibility to design and upload code katas, set parameters for

battles, and establish deadlines. Educators can also grant permissions to colleagues to create battles within a specific tournament. The platform empowers educators to assess student performance, assign scores, and define badges and rules to recognize and reward achievements. Additionally, educators have access to tools that allow them to track student's progress and overall engagement with the platform.

2.4 Assumptions, Dependencies and Constraints

Domain Assumptions

Identifier	Description
D1	The CKB platform assumes that users have a stable internet connection to access and interact with its features.
D2	Once the registration deadline for tournaments or battles has passed, students cannot register for any ongoing or upcoming events.
D3	After the deadline of a tournament, no new battles can be created within the scope of that tournament.
D4	The system must ensure a necessary timestamp between the end of the last battle of a tournament and the conclusion of that tournament.
D5	Users must undergo a secure authentication phase during the login process to access the CKB platform.
D6	CKB relies on access to GitHub functionality for its features to operate effectively.
D7	Educators can only rate student submissions after the submission deadline has passed.
D8	The platform is expected to deliver notifications to users within a time frame of one minute.
D9	The overall tournament score is the sum of scores from all individual battles within that tournament.
D10	Each badge earned by a student is associated with a specific tournament, reflecting the achievements within that context.
D11	Students must respect to the specified maximum and minimum number of group members when forming teams for battles.
D12	Each student is expected to participate in a single team for a specific battle, promoting fairness and avoiding potential conflicts of interest.

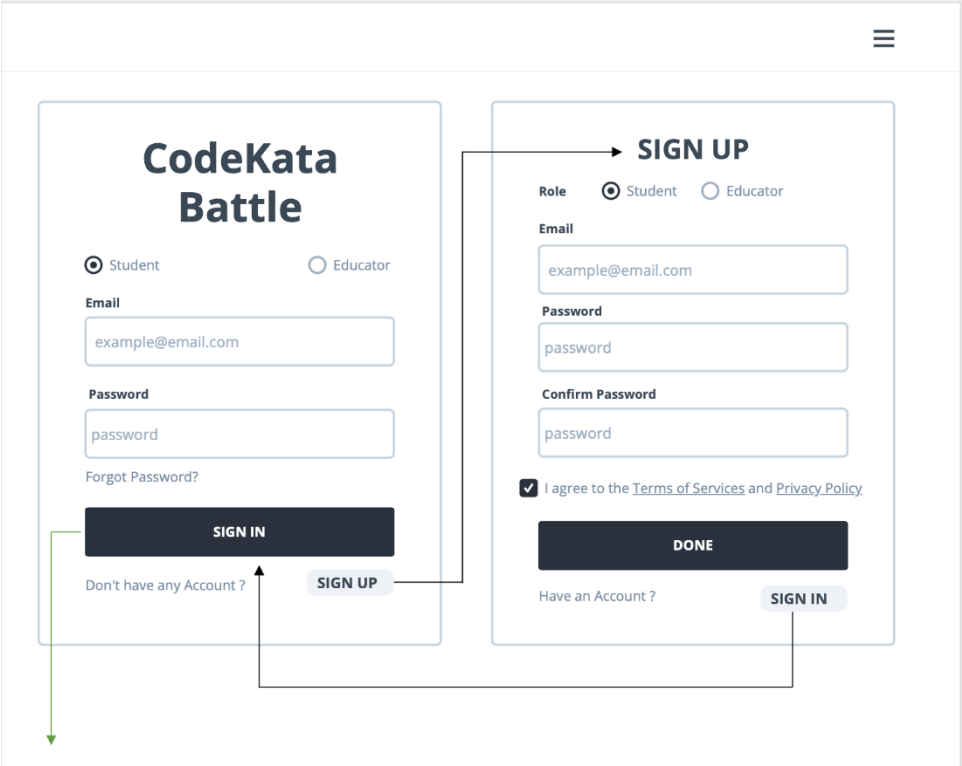
These assumptions, dependencies, and constraints provide a foundation for the functioning of the CKB platform and guide the development and user interaction processes.

3. Specific Requirements

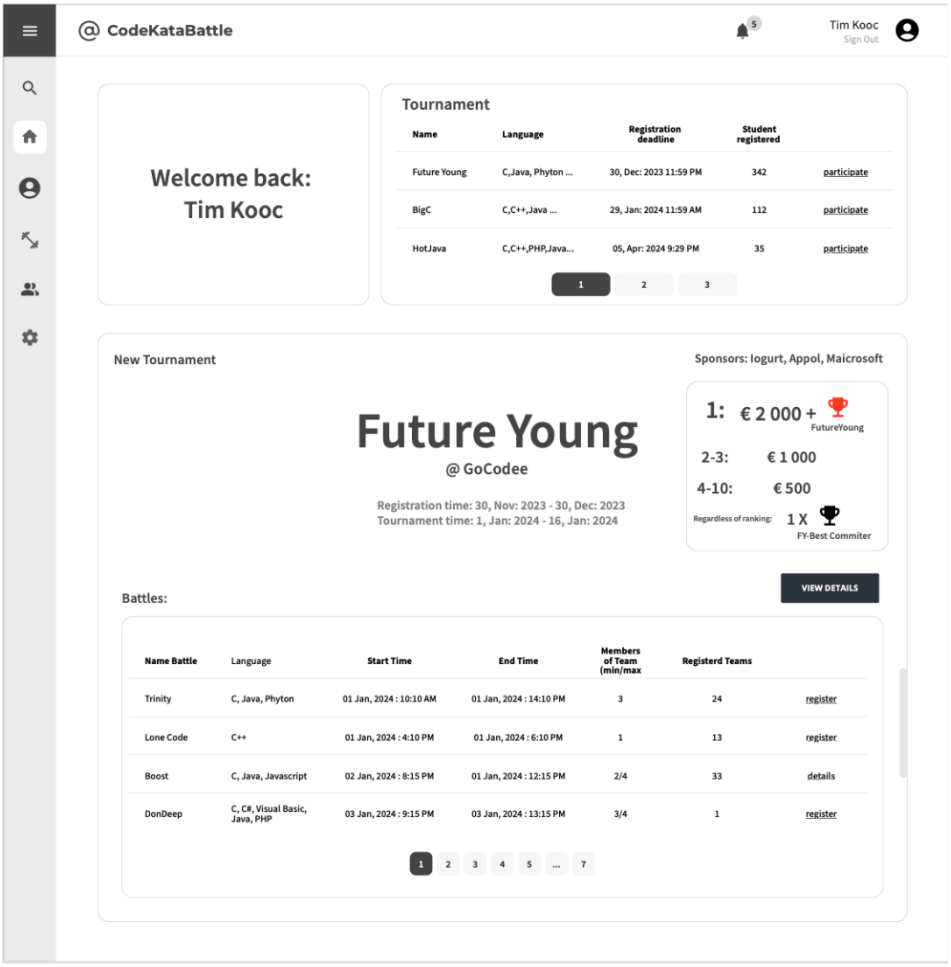
3.1 External Interface Requirements

3.1.1 User Interfaces

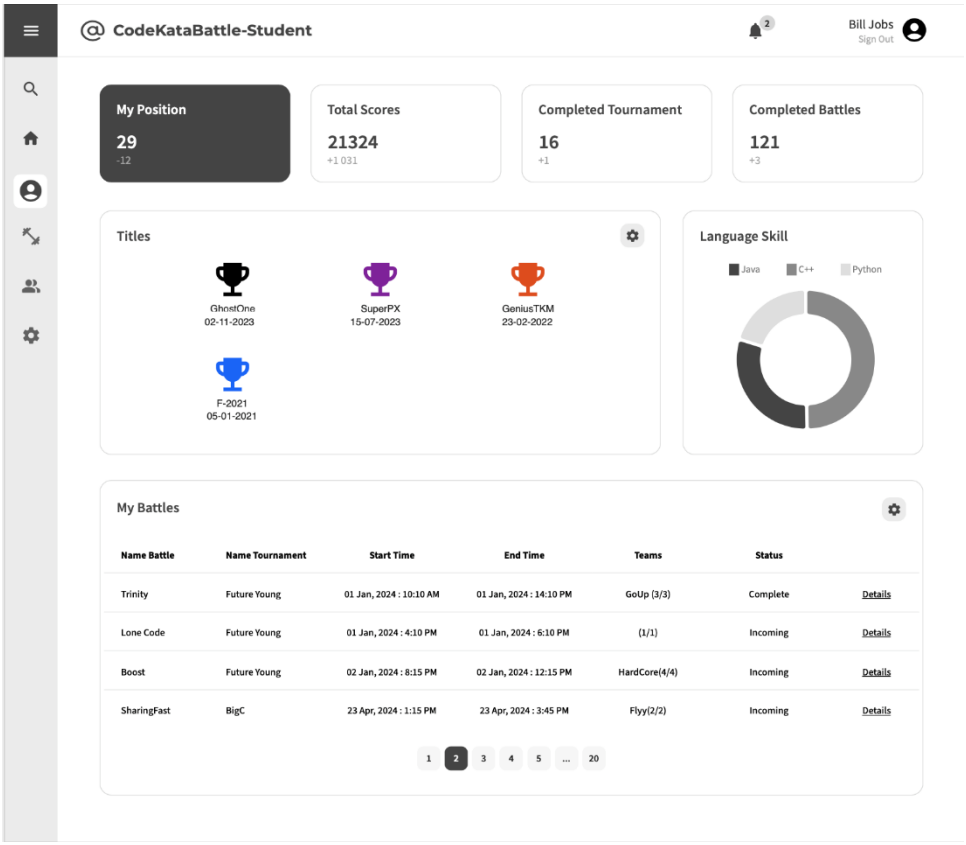
In this section of the document is presented the UI of CodeKataBattle platform: the views both the Student and the Educator are going to see when using the web application.
A Students or an Educator needs to sign in to access the site, if he/she doesn't have registered, the last one can sign up:



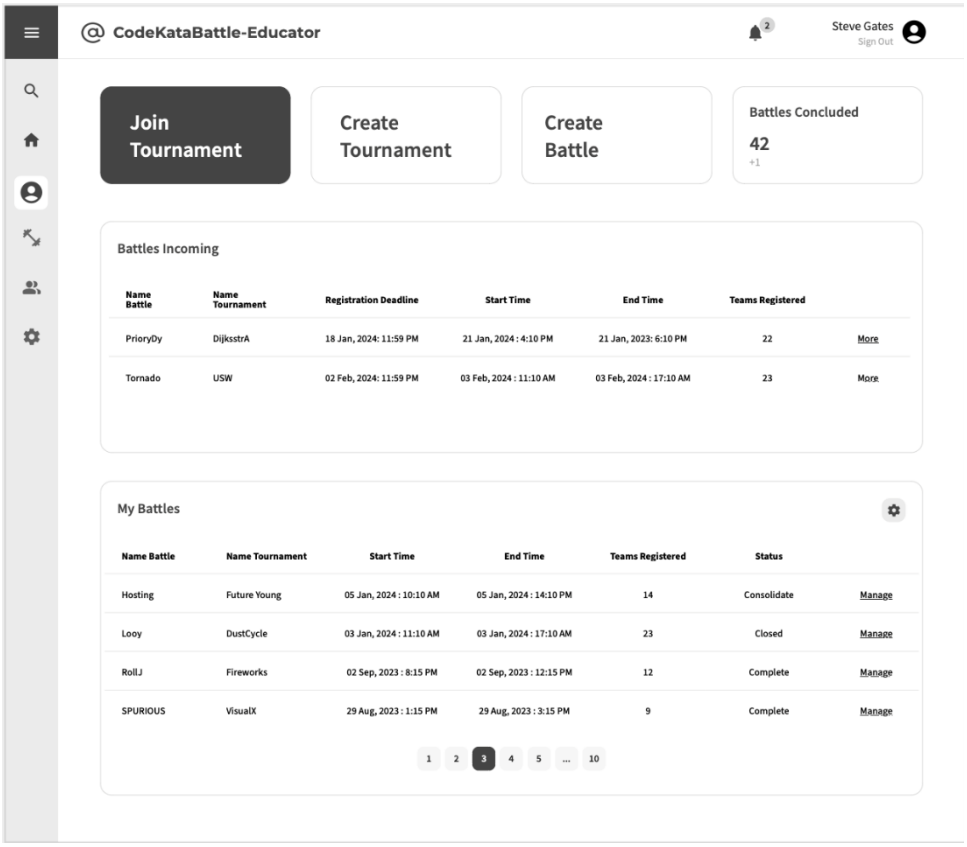
Once is logged in, he/she can see the homepage: there's all sorts of information about tournaments and battles available, as well as times and awards:



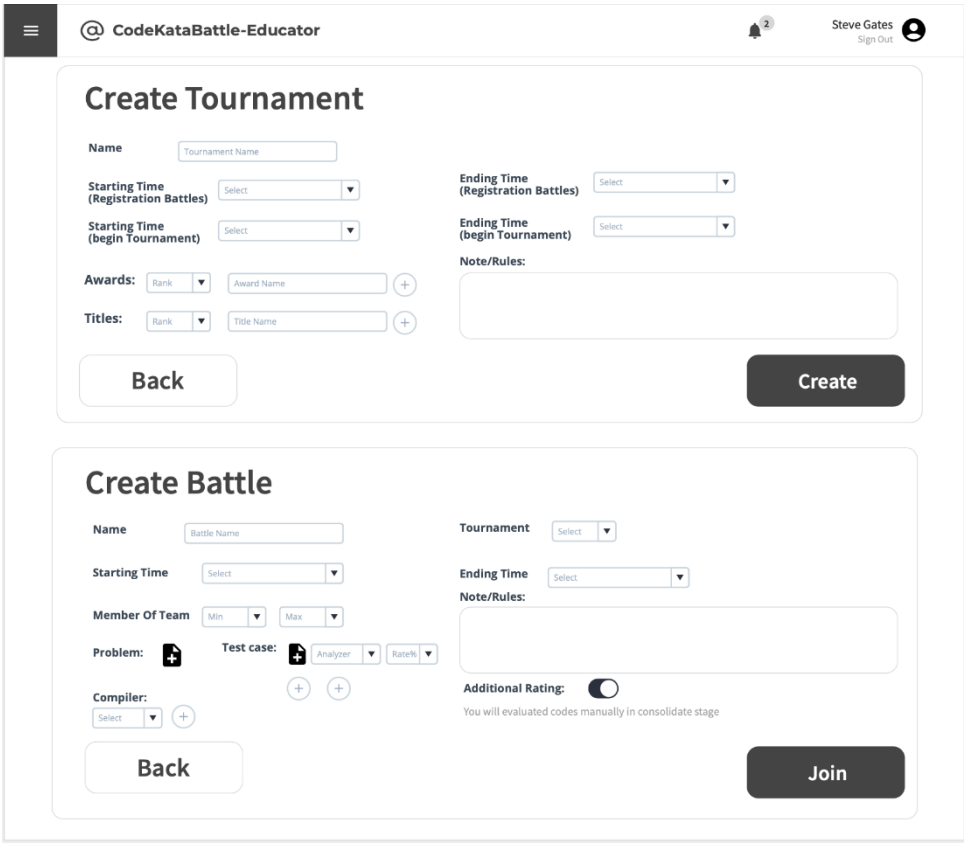
Of course, he/she can also find more information from his personal information, such as the overall ranking of the Student at CKB, scores, tournaments and battles participated in, personal titles, and battles to be participated in:



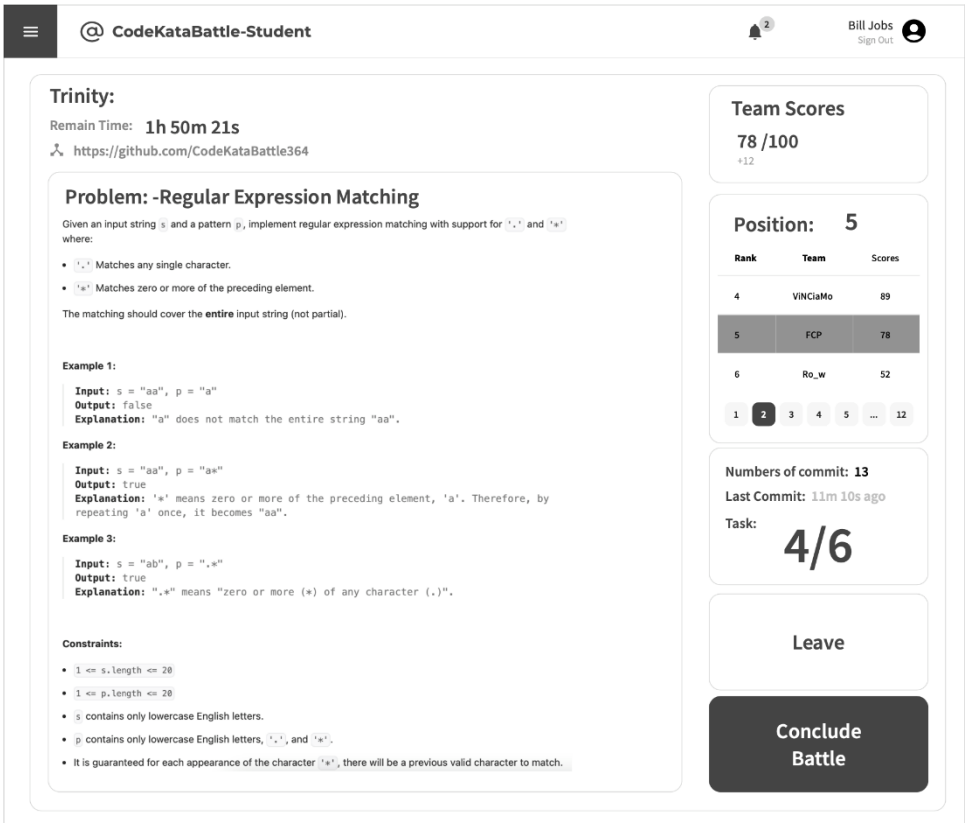
On the Educator's side, it is possible to: view created battles or tournaments, battles awaiting results, and upcoming battles; in addition, the Educator can create a new battles or tournaments：



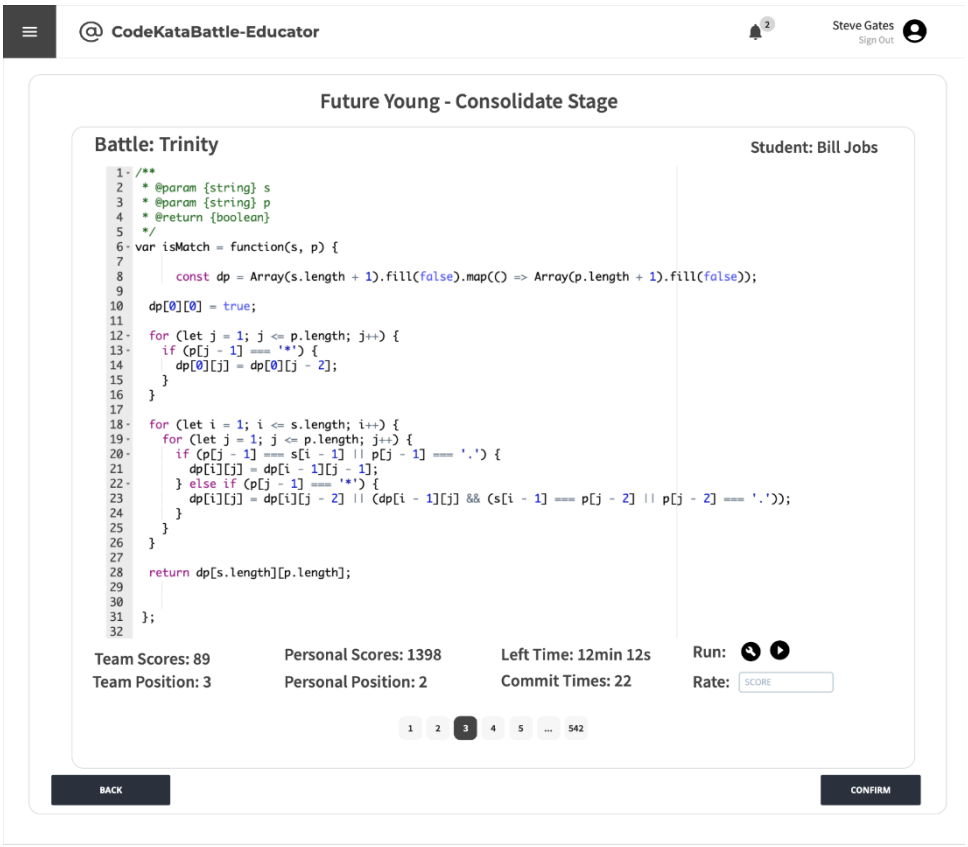
During the creation process, he had many options to choices to complete the creation:



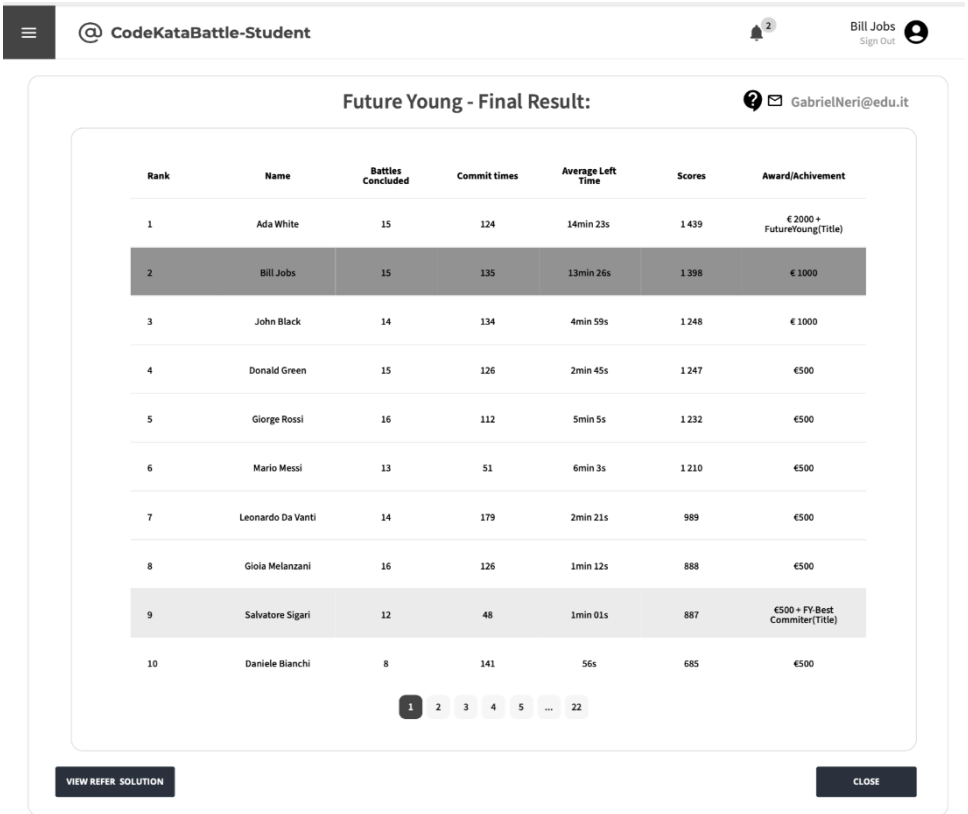
When the battle starts, the student can see assignment, scores, ranking, remain time; he/she can also click to “leave” button for a while if he has other Battle:



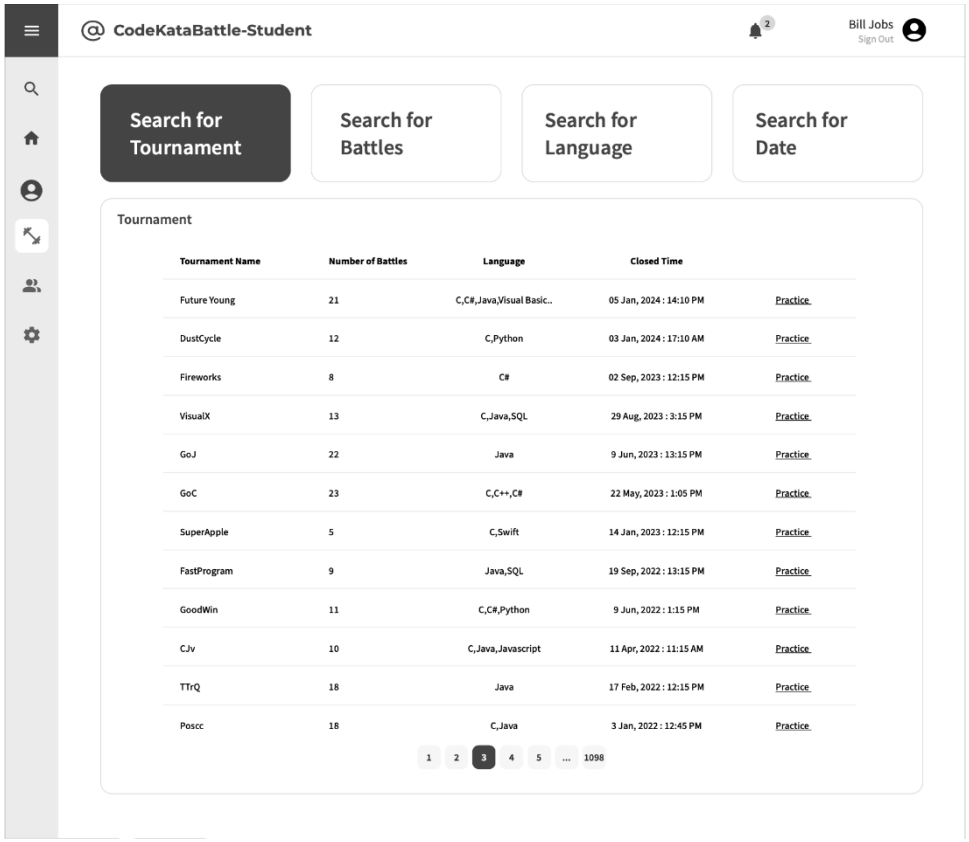
At the end of the tournament (Consolidation Stage), Educator can refer to the students' code and can run and evaluate it:

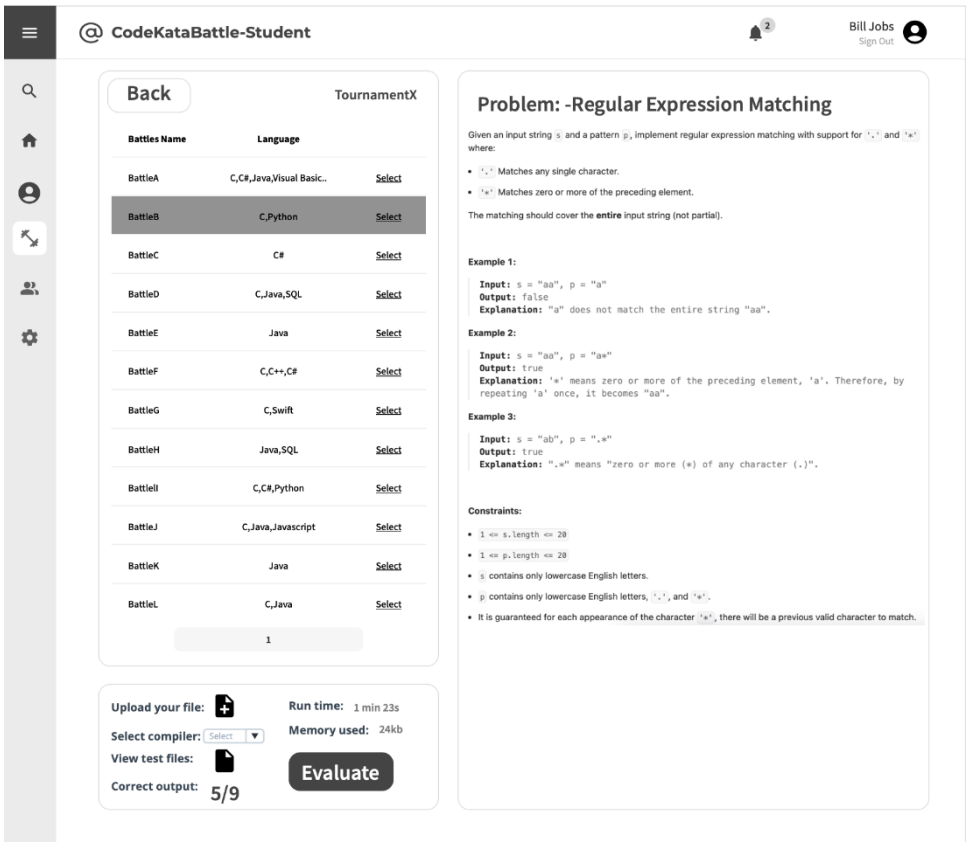


At the end, students can see the final rankings：



In additional, students are free to practice past tournaments (with corresponding battles):





3.1.2 Hardware Interfaces

The service offered by the System via the computer with internet connection and possible mobile device which can also connected with internet.

3.1.3 Software Interfaces

The System required several software interfaces to provide its services:

- Online compilers, provide Users to write, compile and run code directly in their browsers without the need for local development environments
- Statistical analysis tools, the System provides to Educators and/or Students to assess the quality of source code by examining various aspects
- GitHub API, provide services by GitHub to allow User to programmatically interact with GitHub's features and data, such as: accessing user information, collaborate code with other Users etc.

3.1.4 Communication Interfaces

The system exploits the internet connection for the communication to and from all personal computer and/or laptop of the Users with possible mobile devices.

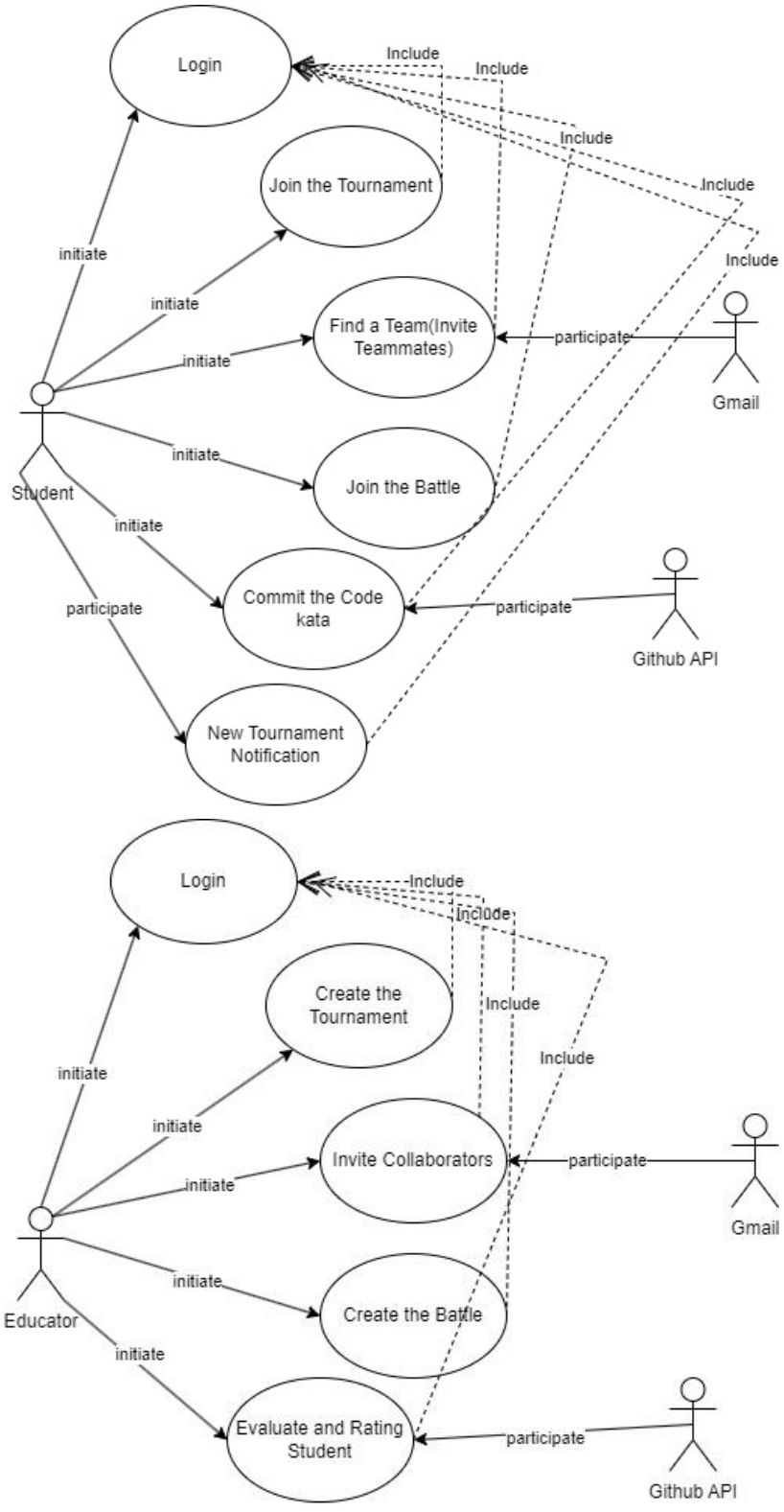
3.2 Functional Requirements

The following specifies all the requirements that the system must fulfil to function properly:

Requirement	Description
R1	The system allows Students to sign up.
R2	The system allows Students to sign in.
R3	The system allows Educators to sign up.
R4	The system allows Educators to sign in.
R5	The system allows Students to access the profile section.
R6	The system allows Students to send invitations for specific battles.
R7	The system allows Students to choose their preferred programming language.
R8	The system allows Students to explore and view details of individual tournaments.
R9	The system allows Students to subscribe to tournaments.
R10	The system allows Students to explore and select specific battles with tournaments.
R11	The system allows Students to subscribe to battles.
R12	The system notifies Students of a GitHub repository link associated with the selected battle.
R13	The system allows Students to view their scores obtained in battles.
R14	The system allows Students to access battle raking.
R15	The system allows Students to access tournament ranking.
R16	The system allows Students to access a comprehensive list of battles in which they have previously participated.
R17	The system allows Students to choose the practice mode for ongoing learning and skill enhancement.
R18	The system allows Educators to create new tournaments.
R19	The system allows Educators to create new battles.
R20	The system allows Educators to grant specific individuals the authority to create battles within a tournament.
R21	The system allows Educators to setting titles and rules for tournaments, referred to as “Tournament Badges”.

R22	The system allows Educators to upload code kata with referenced answer.
R23	The system allows Students to access referenced answer.
R24	The system allows Educators to set participant limits, registration deadlines and submission deadlines for each battle.
R25	The system allows Educators to configure grading parameters (e.g. security percentages...).
R26	The system notifies Students upon the creation of a new tournaments.
R27	The system notifies Students upon the updates of a new battle.
R28	The system notifies Students in advance of upcoming battles.
R29	The system notifies Educators and Students once the tournament has concluded.
R30	The system notifies Educator and Students once the battle has concluded.
R31	The system notifies Students of the battle results and the overall tournament results, including scores and rankings.
R32	The system allows Educators to access to the code submitted by Students through GitHub commits.
R33	The system allows Educators to send manual scores to the platform.
R34	The system allows Students to invite other students to join the team for the battle.

3.2.1 Use Cases Diagrams



3.2.2 Use Cases

[UC1]	Student Login
Actor	Student
Entry Condition	The student accesses the login page.
Event Flow	1. The student clicks on the “Student” radio button. 2. The student enters registered credentials (Email and Password). 3. The student clicks on the “SIGN IN” button. 4. The system validates the login information. 5. The page shows the proper dashboard.
Exit Condition	The user has logged in successfully.
Exception	1. The student enters incorrect credentials. 2. The account has not yet been validated.

[UC2]	Student Registration
Actor	Student
Entry Condition	The student accesses the login page.
Event Flow	1. The student clicks on the “SIGN UP” button. 2. The student clicks on the “Student” radio button. 3. The student enters email address and password. 4. The student clicks on the “DONE” button. 5. The system validates the sign-up information. 6. The system sends a verification email to the user. 7. The student verifies their email address by clicking on the provided link. 8. The system displays the approval message. 9. The system redirects to the login page.
Exit Condition	Registration has been successful. The student’s data will be stored in the system’s database and the student will then be able to sign into the platform.
Exception	1. The provided information is incomplete or inaccurate. 2. The chosen email is already in use, the system notifies the user to choose a different one.

[UC3]	Educator Login
Actor	Educator
Entry Condition	The educator accesses the login page.
Event Flow	1. The educator clicks on the “Educator” radio button. 2. The educator enters registered credentials (Email and Password). 3. The educator clicks on the “SIGN IN” button. 4. The system validates the login information. 5. The page shows the proper dashboard.
Exit Condition	The educator has logged in successfully.
Exception	1. The educator enters incorrect credentials. 2. The account has not yet been validated.

[UC4]	Educator Registration
Actor	Educator
Entry Condition	The educator accesses the login page.
Event Flow	1. The educator clicks on the “SIGN UP” button. 2. The educator clicks on the “Educator” radio button. 3. The educator enters email address and password. 4. The educator uploads the Teacher’s Certificate. 5. The educator clicks on the “DONE” button. 6. The system validates the sign-up information. 7. The system checks for certificate validation (within 24 hours). 8. The system sends a verification email to the user. 9. The educator verifies their email address by clicking on the provided link. 10. The system displays the approval message. 11. The system redirects to the login page.
Exit Condition	Registration has been successful. The educator’s data will be stored in the system’s database and the educator will then be able to sign into the platform.

Exception	1. The provided information is incomplete or inaccurate. 2. The certificate is invalid. 3. The chosen email is already in use, the system notifies the educator to choose a different one.

[UC5]	Creates a Tournament
Actor	Educator
Entry Condition	The educator is logged into the CKB platform.
Event Flow	1. The educator accesses the CKB platform. 2. The educator navigates to the tournament creation section by clicking on the “Create Tournament” button. 3. The system displays a form for creating a tournament. 4. The educator configures tournament settings, such as tournament name and rules, badges definitions and variables associated with the tournament, and sets the deadline for student subscriptions to tournaments. 5. The educator clicks on the “Join” button. 6. The system displays the message of approval of the addition of the battle. 7. The educator clicks on the “Create” button. 8. The system displays a Confirm message. 9. The educator clicks on the “Confirm” button. 10. The system displays the message of approval for the creation of the tournament. 11. The system sends an email to all student users on the CKB platform notifying them that a new tournament has been created. 12. The system redirects to the tournament page.
Exit Condition	The tournament is successfully created, and students are notified of the new tournament.
Exception	

[UC6]	Creates a Battle
Actor	Educator

Entry Condition	1. The educator is logged into the CKB platform. 2. The educator has the necessary permissions to create a battle. 3. The tournament in which this battle is taking place is in progress.
Event Flow	1. The educator accesses the CKB platform. 2. The educator navigates to the Tournaments Section. 3. The educator selects the tournament in which to create a battle 4. The educator clicks on the “Create Battle” button. 5. The system displays a form for creating a battle. 6. The educator enters the name of the tournament name of the battle. 7. The educator enters the details of the battle, including its name and rules, sets the deadline for student submissions and subscriptions to battle, the number of the team member, uploads the problem and the test cases, and adds compiles. 8. The educator clicks on the “Join” button. 9. The system displays the message of approval for the creation of the battle. 10. The system sends an email to all student users on the CKB platform and educator users with permission to create battles at this tournament, notifying them that a new battle has been created. 11. The system redirects to the battle’s tournament page.
Exit Condition	The battle is successfully created, and students and educators subscribed to the current tournament are notified of the new battle.
Exception	1. The educator is not authorized to create battle in the current tournament. 2. The tournament in which this battle is taking place is ended. 3. The tournament does not exist.

[UC7]	Subscribes to a Tournament
Actor	Student

Entry Condition	1. The student is logged into the CKB platform. 2. The tournament is open for subscription.
Event Flow	1. The student accesses the CKB platform. 2. The student navigates to the list of unsubscribed tournaments. 3. The student subscribes to a specific tournament by clicking on the “participate” button. 4. The system displays the approval message. 5. The system redirects to the battle’s tournament page.
Exit Condition	The student is successfully subscribed to the tournament.
Exception	The tournament subscription deadline has passed.

[UC8]	Subscribes to a Battle
Actor	Team
Entry Condition	1. The team members are logged into the CKB platform. 2. The team members have joined the tournament to which the current battle belongs. 3. The battle is open for subscription. 4. The number of registered teams is not full.
Event Flow	1. The team member accesses the CKB platform. 2. The team member navigates to the Tournaments Section. 3. The team member selects the specific tournament which the battle belongs. 4. The system displays the list of ongoing battles. 5. The team member selects and subscribe for a battle by clicking on the “register” button. 6. The team member clicks on the “Join the Battle” button. 7. The system displays the approval message. 8. The system redirects to the battle page.
Exit Condition	The team is successfully subscribed to the battle.
Exception	1. The deadline for subscripting to the tournament has ended. 2. The number of registered team is full.

[UC9]	Join an Existing Team
Actor	Student
Entry Condition	1. The student is logged into the CKB platform. 2. The student is associated with the specific battle.
Event Flow	1. The user accesses the CKB platform. 2. The user navigates to the Battles Section. 3. The user selects the specific battle for which they want to join the team for. 4. The student clicks on the “Join an existing team” button. 5. The system displays the team search input box. 6. The student enters team name. 7. The student clicks on the “join” button. 8. The system displays the approval message to join of the team.
Exit Condition	The student is successfully joined to the battle.
Exception	1. Application to join the team rejected. 2. The team is full. 3. The deadline for subscripting to the battles has ended.

[UC10]	Student Create a New Team
Actor	Student
Entry Condition	1. The student is logged into the CKB platform. 2. The student is associated with the specific battle.
Event Flow	1. The user accesses the CKB platform. 2. The user navigates to the Battles Section. 3. The user selects the specific battle for which they want to create a team. 4. The student clicks on the “Create a new team” button. 5. The system displays the Create Team form. 6. The student completes the team information (the name of the team and the number of members). 7. The student clicks on the “register” button. 8. The system displays the approval message to create the team.
Exit Condition	The student is successfully created the battle.
Exception	The team's name already exists.

[UC11]	**Grant Permission to Create Battles**

Actor	Educator
Entry Condition	1. The educator is logged into the CKB platform. 2. The educator is the administrator of the current tournament.
Event Flow	1. The educator accesses the CKB platform. 2. The educator navigates to the Tournaments Section. 3. The educator selects the tournament in which to grant permission to create battles. 4. The educator clicks on the “Set Parameters” button. 5. The educator grants permission to specific colleagues by providing their email address or selecting them from a list of registered educators. 6. The educator clicks on the “Confirm” button. 7. The system displays the approval message. 8. The system redirects to the tournament page.
Exit Condition	The educator has successfully granted the permissions to colleagues.
Exception	Incorrect email address.

[UC12]	Invitation to Battle
Actor	Student
Entry Condition	1. The student is logged into the CKB platform. 2. The student is associated with the specific battle.
Event Flow	1. The student accesses the CKB platform. 2. The student navigates to the ongoing code kata battle in which they are participating. 3. The student clicks on the “Invite” button. 4. The student invites others by providing their email address. 5. The student may include a personalized message along with the invitation. 6. The system sends invitations to the specified students, notifying them of the invitation to join the code kata battle. 7. The invited students receive notifications and can accept or decline the invitation.
Exit Condition	The student has successfully invited other students to join the team for the battle.

Exception	1. Incorrect email address. 2. Invited students declining the invitation. 3. Maximum team size exceeded. 4. The deadline for subscripting to the battle has ended.

[UC13]	Closes a Tournament
Actor	Educator
Entry Condition	1. The educator is logged into the CKB platform. 2. The educator is the administrator of the current tournament.
Event Flow	1. The educator accesses the CKB platform. 2. The educator navigates to the Tournaments Section. 3. The educator selects the specific tournament they want to close. 4. The educator clicks on the “Close” button. 5. The system displays a Confirm message. 6. The educator clicks on the “Confirm” button. 7. The system displays the message of approval for the end of the tournament. 8. The system performs final calculations for the tournament, including generating the final tournament rank based on individual battle scores. 9. The system evaluates the rules and assigns tournament badges to eligible students. 10. The system notifies all students involved in the tournament of the closure and provides access to the final tournament rank.
Exit Condition	The tournament is successfully closed.
Exception	1. The tournament is incomplete (e.g., ongoing battles). 2. The educator doesn’t have the necessary permission to the close the tournament.

[UC14]	Views User Profile
Actor	User (Student or Educator)

Entry Condition	The user is logged into the CKB platform.
Event Flow	1. The user accesses the CKB platform. 2. The user navigates to the Profile Section. 3. The user can view their own profile details, including personal information, tournament scores, and badges earned. 4. If the user wishes to view another user's profile: a. The user initiates a search for the specific user by entering their email address. b. The system retrieves and displays the profile information of the selected user.
Exit Condition	The user has successfully viewed the desired profile information.
Exception	Profile not found.

[UC15]	Views Battle Rankings
Actor	User (Student or Educator)
Entry Condition	The user is logged into the CKB platform.
Event Flow	1. The user accesses the CKB platform. 2. The user navigates to the Battles Section. 3. The user selects the specific battle for which they want to view rankings. 4. The system displays the current rankings, including scores and positions of all participating teams or individuals. 5. The user can navigate through different sections of the rankings, such as overall rankings, scores for specific evaluation criteria, and individual battle scores.
Exit Condition	The user has successfully viewed the desired battle rankings and related information.
Exception	

[UC16]	Views Tournament Rankings
Actor	User (Student or Educator)
Entry Condition	The user is logged into the CKB platform.
Event Flow	1. The user accesses the CKB platform.

	2. The user navigates to the Tournament Section. 3. The user selects the specific tournament for which they want to view rankings. 4. The system displays the current overall rankings, including aggregated scores and positions of all participating tournament. 5. The user can navigate through different sections of the rankings, such as individual battle scores, badges earned, and other relevant information.
Exit Condition	The user has successfully viewed the desired tournament rankings and related information.
Exception	

[UC17]	Participates in Practice Mode
Actor	Student
Entry Condition	1. The student is logged into the CKB platform. 2. The code kata battle has ended.
Event Flow	1. The student accesses the CKB platform. 2. The student navigates to the Practice Section. 3. The system presents a list of available code katas or programming exercises for practice. 4. The student selects a specific code kata from the list. 5. The system provides a description of the selected code kata, including requirements, constraints, and any provided code snippets or templates. 6. The student starts working on the code kata by writing code to meet the specified requirements. 7. The student completes the code kata and submits their solution to the system for evaluation. 8. The system automatically assesses the solution, score, providing feedback on correctness, efficiency, and best practices. 9. The student reviews the evaluation feedback and may choose to revisit and improve their solution.
Exit Condition	The student successfully completes the selected code kata in practice mode.
Exception	The battle is still in progress and practice mode is not available.

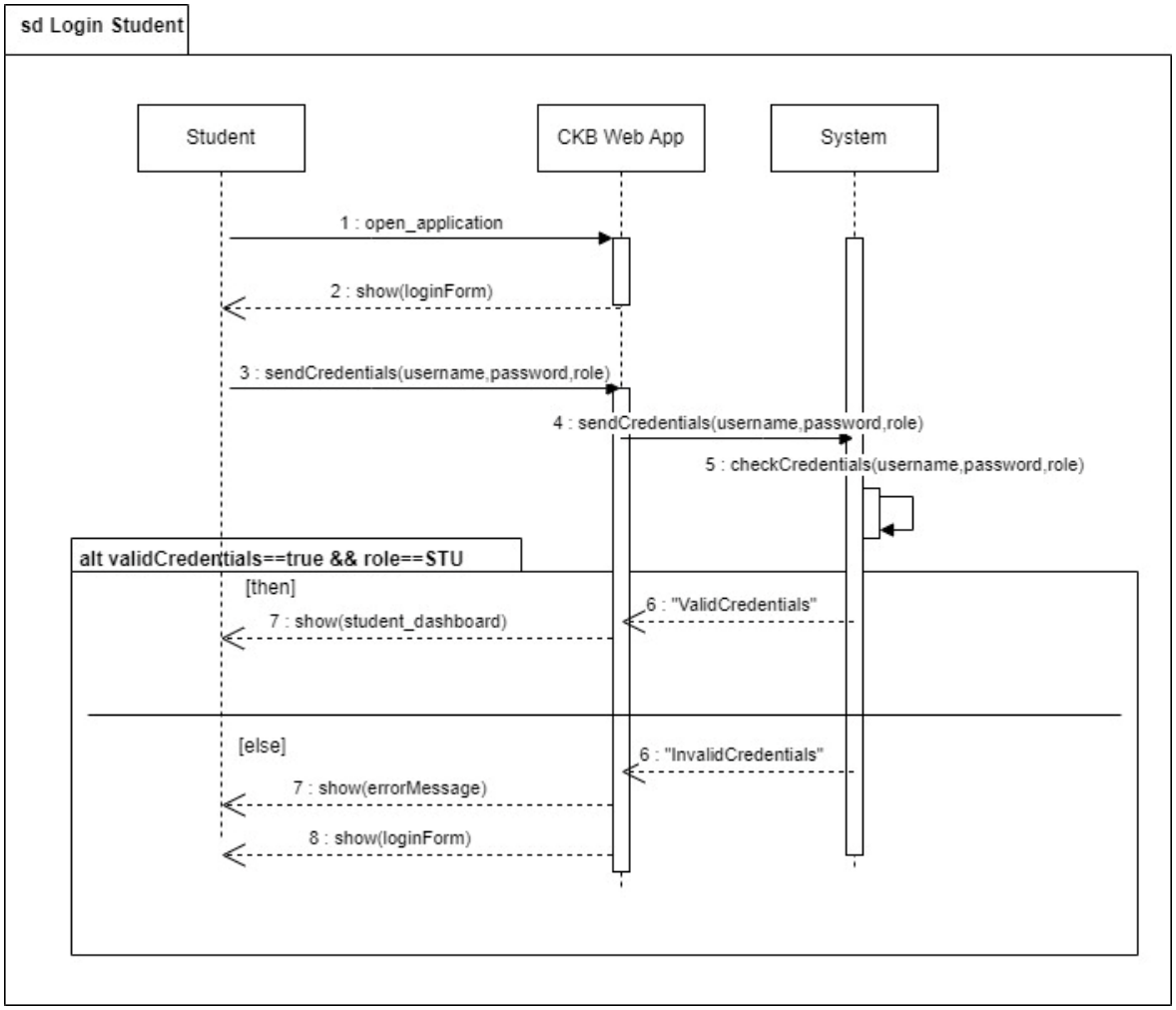
[UC18]	Manual Evaluation
Actor	Educator
Entry Condition	1. The educator is logged into the CKB platform. 2. The educator is the administrator of the current battle. 3. The battle has reached the consolidation stage where manual evaluation is required.
Event Flow	1. The educator accesses the CKB platform. 2. The educator navigates to the Battle Section. 3. The educator selects the specific battle for which manual evaluation is required. 4. The educator clicks on the “Evaluation” button. 5. The system provides a list of teams or individuals participating in the battle, along with their submitted work. 6. The educator reviews the source code, test cases, and other relevant materials submitted by each team or individual. 7. The educator assigns scores of feedback based on predetermined evaluation criteria, such as code quality, adherence to best practices, and overall completeness. 8. The educator submits the final evaluations and scores for each team or individual by clicking on the button “Submit”. 9. The system displays the approval message. 10. The system redirects to the battle page with the team list.
Exit Condition	The educator has successfully completed the manual scoring of the battle and the scores are recorded in the system.
Exception	

[UC19]	Commits Code Kata
Actor	Student
Entry Condition	1. The student is logged into the CKB platform. 2. The student is enrolled in a specific code kata battle or practice mode session.
Event Flow	1. The student accesses the CKB platform. 2. The student navigates to the Battle Section or Practice

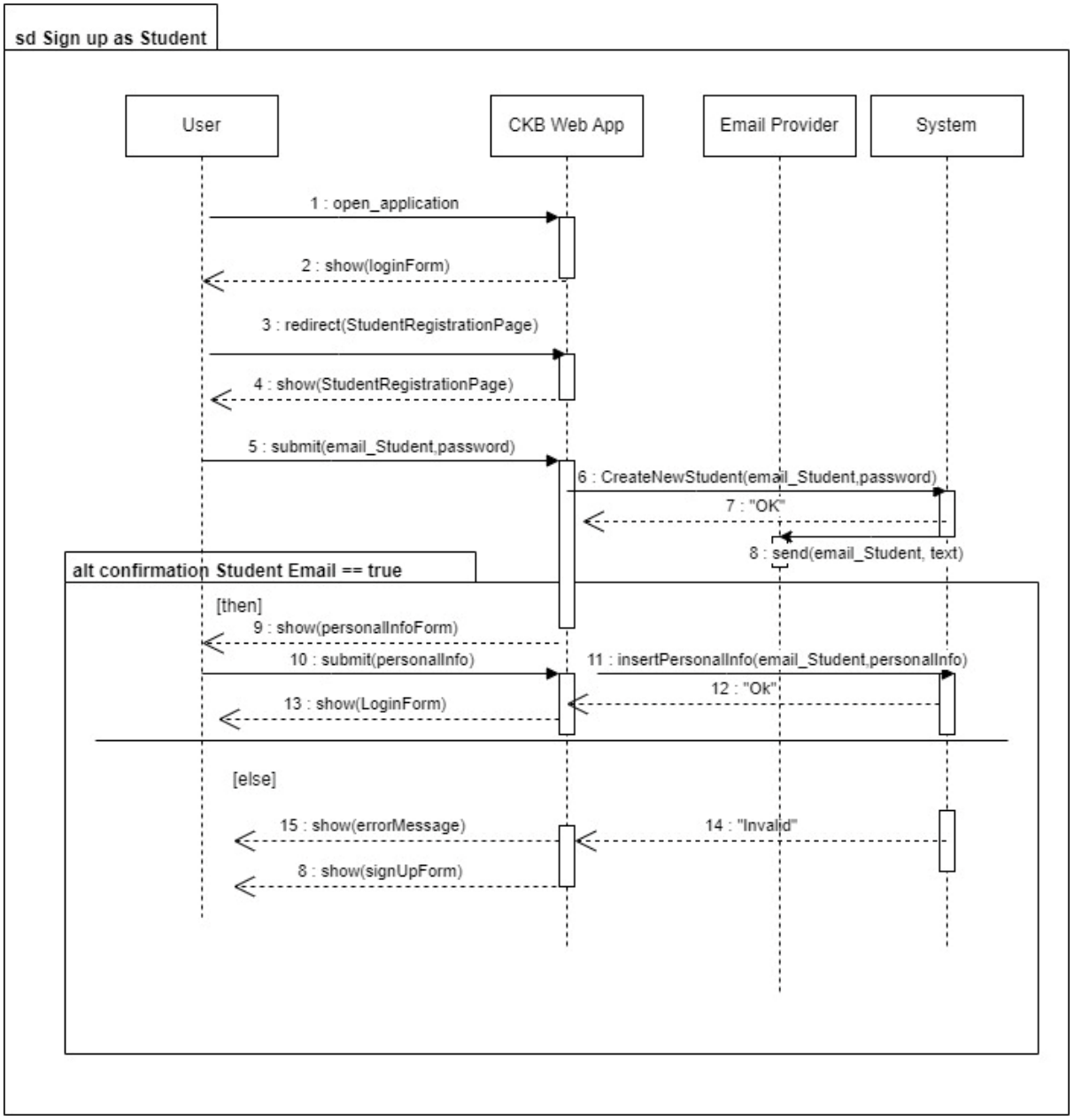
	Mode Section. 3. The student selects the specific battle for which want to commit the solution. 4. The student initiates a commit by pushing their code changes to the main branch of the associated repository. 5. The system detects the new commit. 6. The system retrieves the latest source code from the repository. 7. The system performs automatic analysis of the code by using “SonarQube” API. 8. The system updates the scores for the student based on the results of the automatic evaluation. 9. The student receives feedback on the evaluation, including scores, information on test case outcomes, code quality, and other relevant criteria.
Exit Condition	The student is successfully committed the code kata solution.
Exception	Commits code after the submission deadline.

3.2.3 Sequence Diagrams

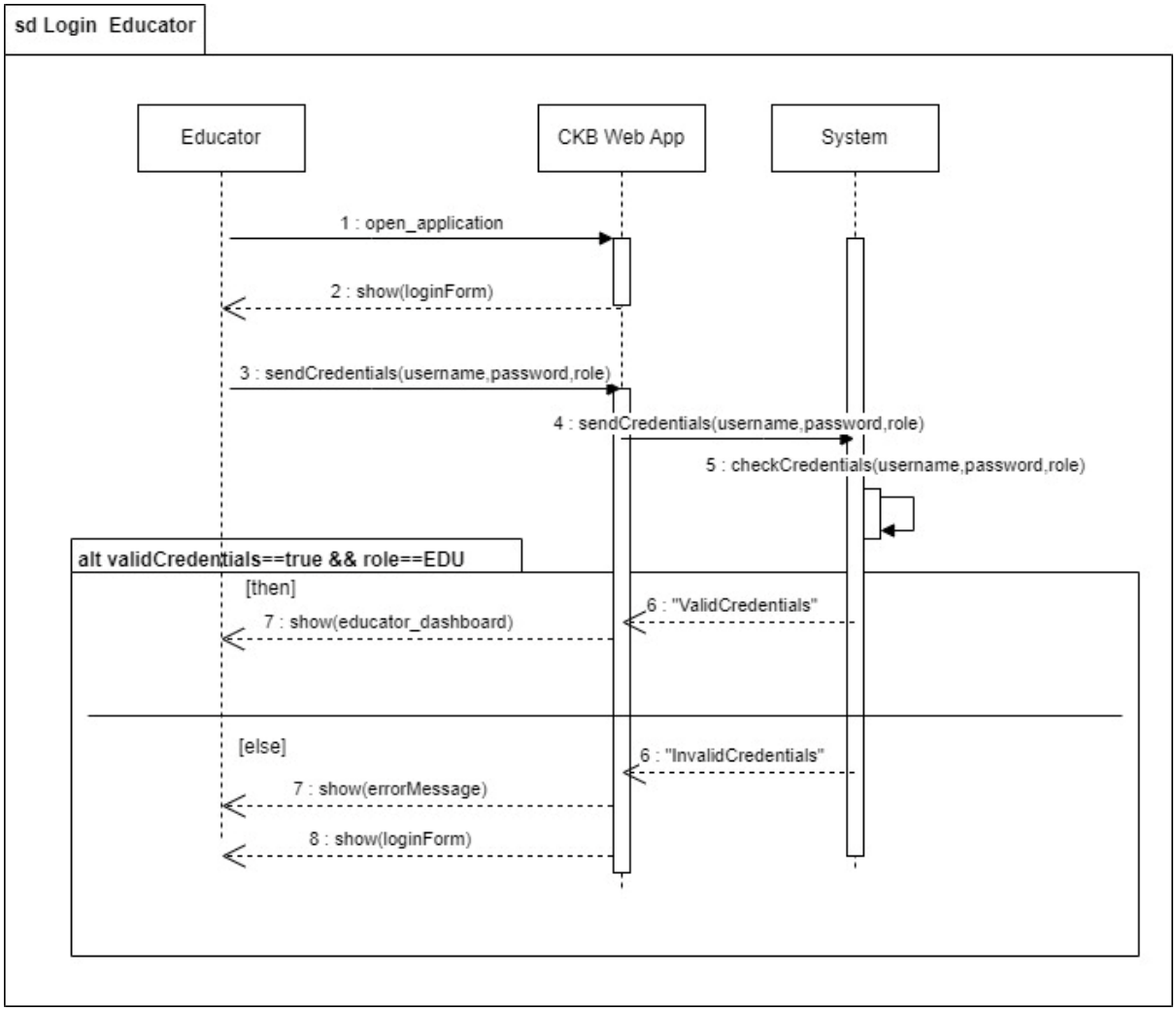
UC1 – Login as Student



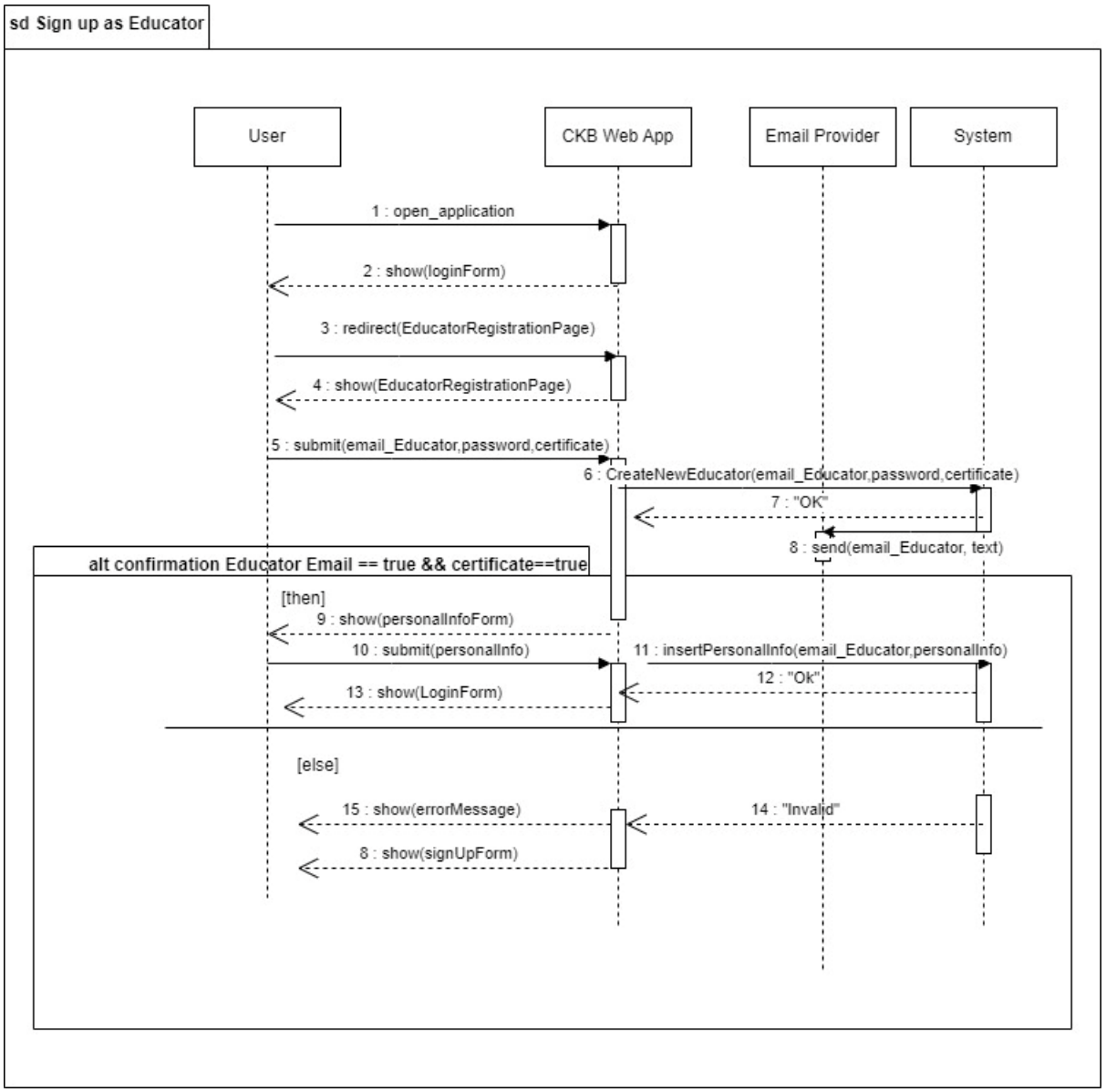
UC-2 Sing up as Student



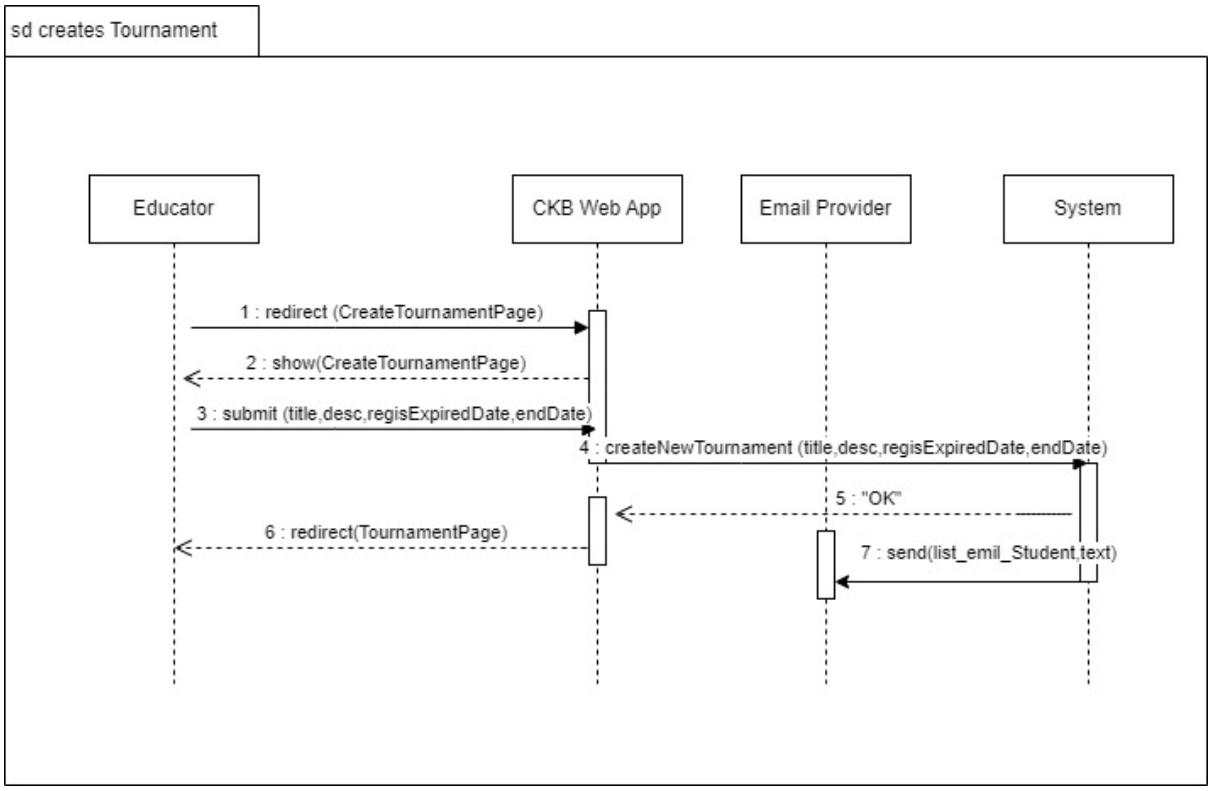
UC-3 Login as Educator



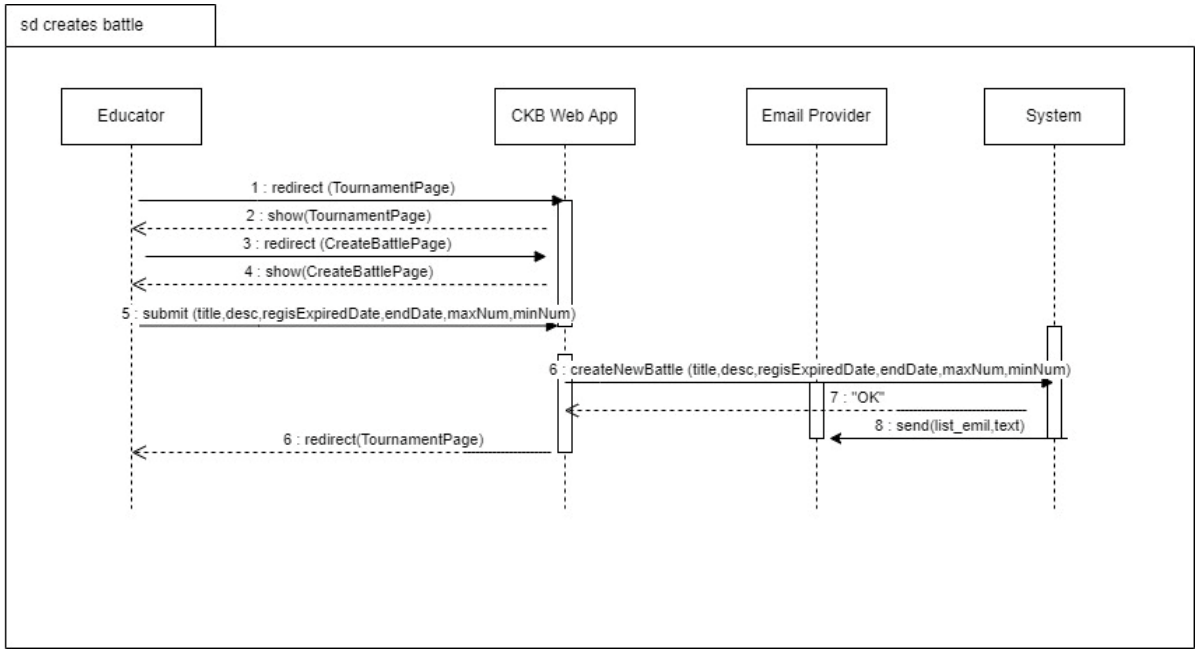
UC-4 Sign up as Educator



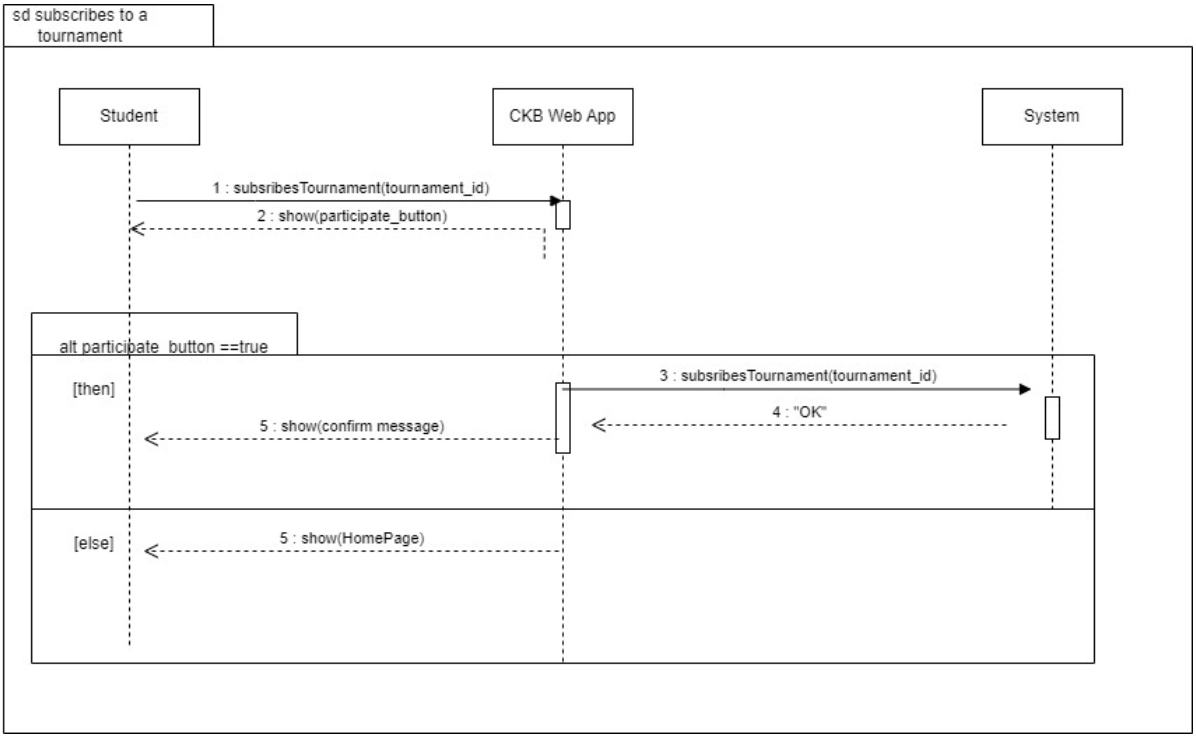
UC- 5 Creates Tournament



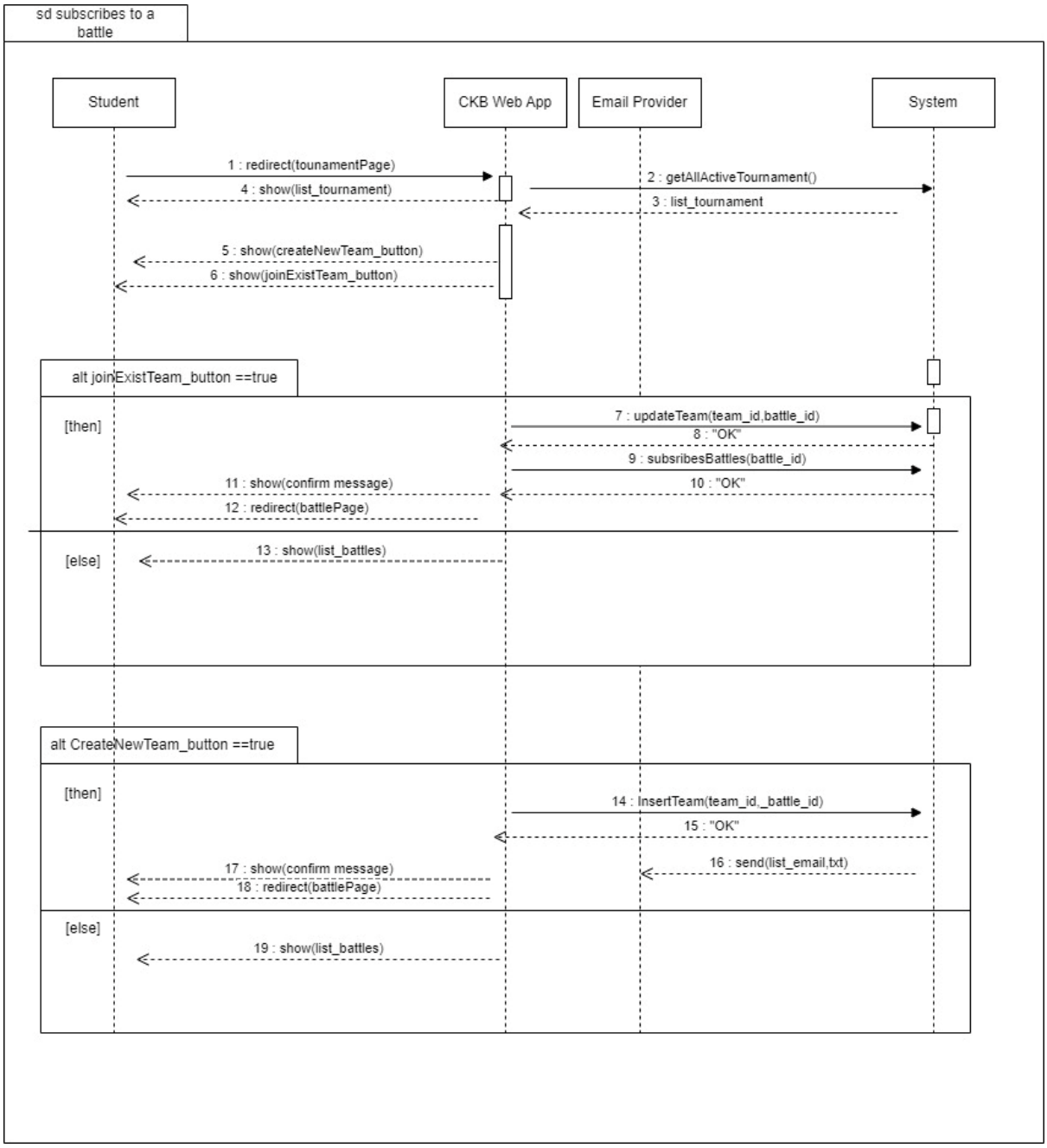
UC-6 Creates Battle



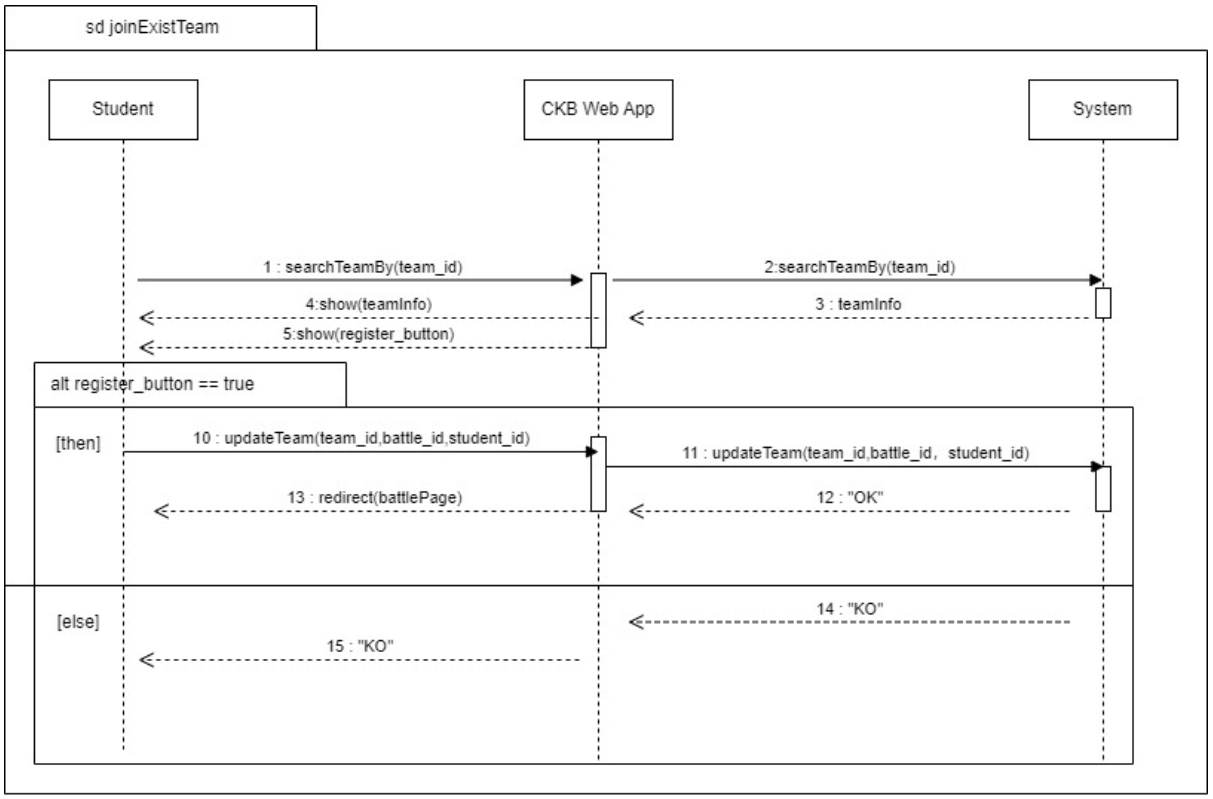
UC-7 Subscribes to a Tournament



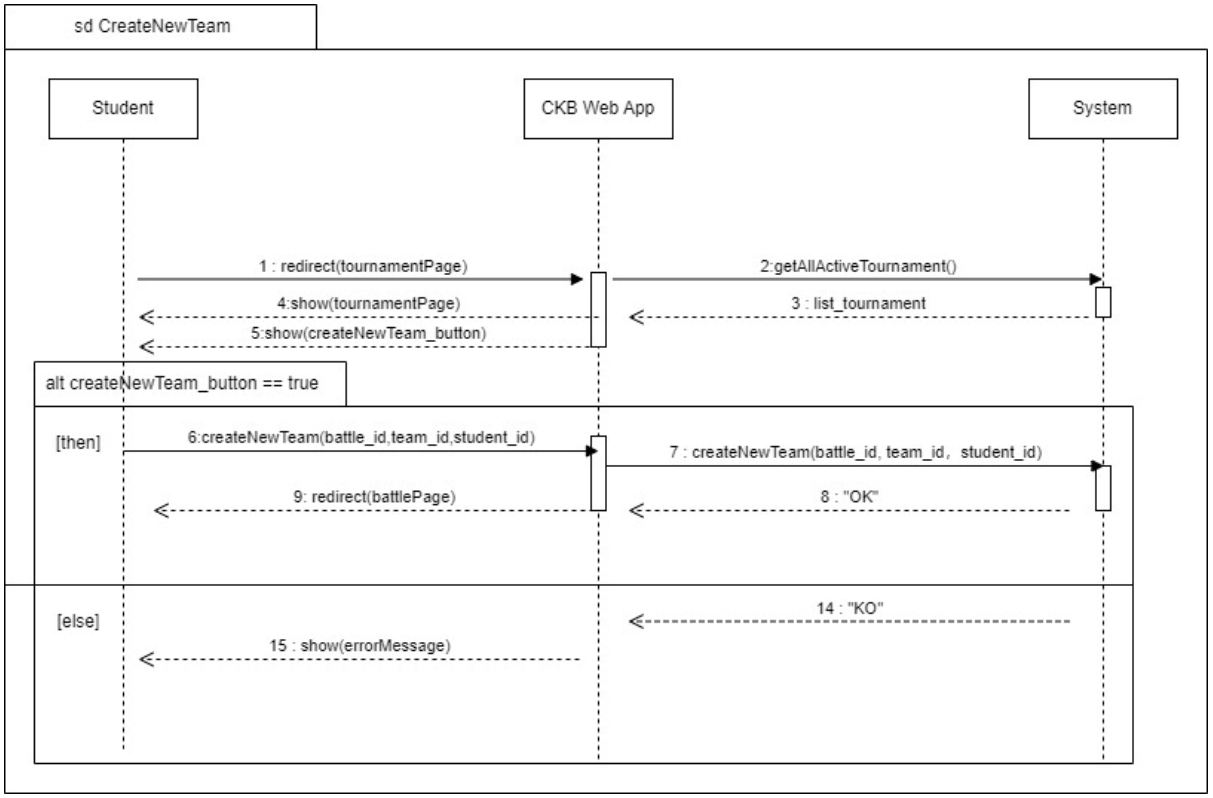
UC-8 Subscribers to a Battle



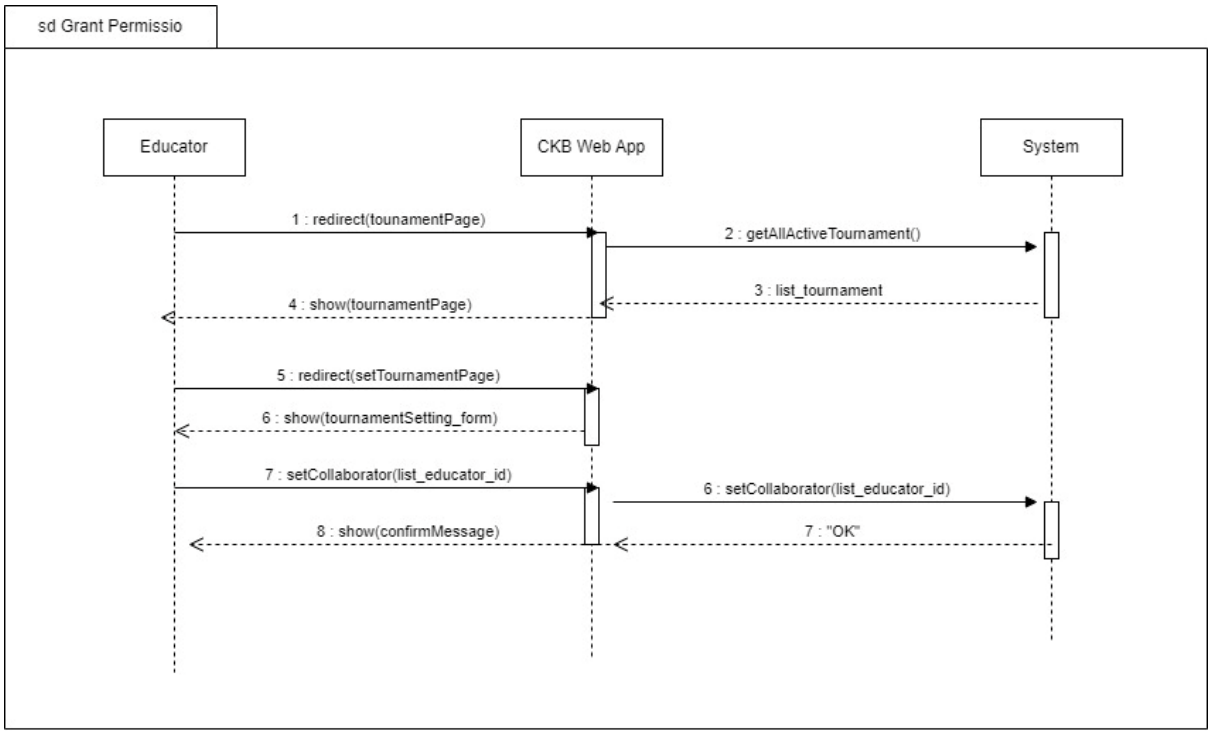
UC-9 Join an Existing Team



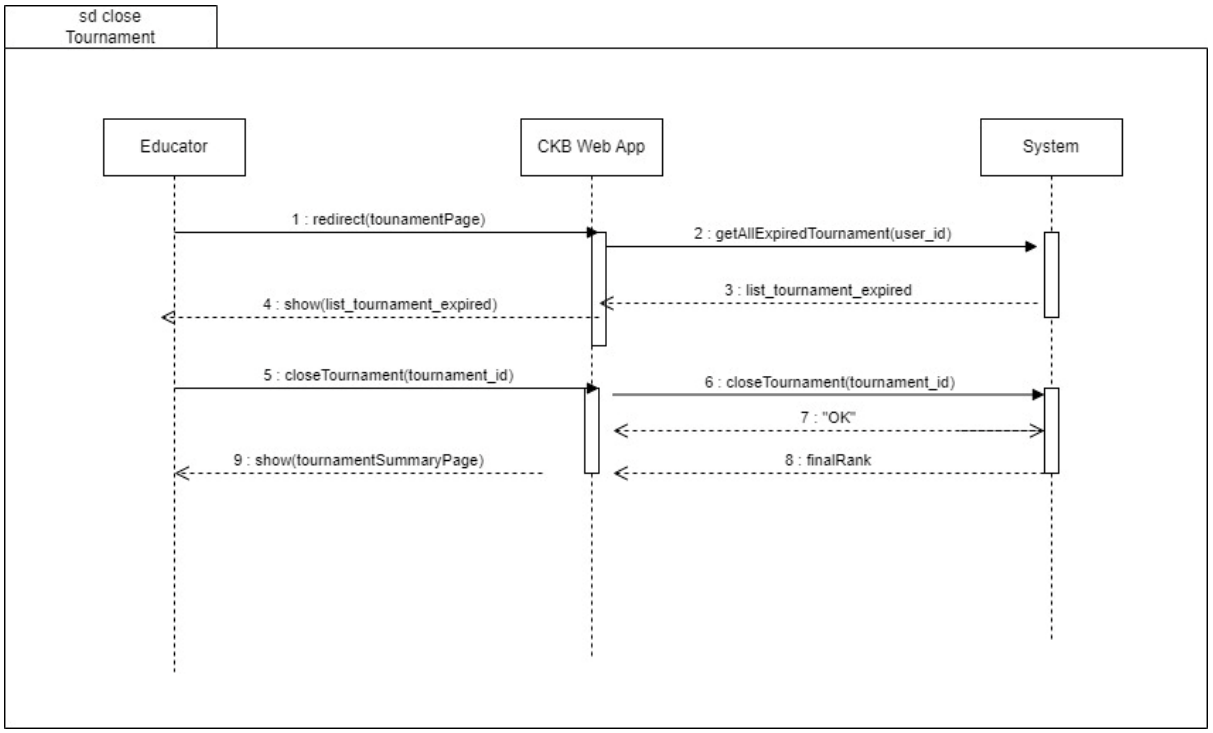
UC-10 Create a New Team



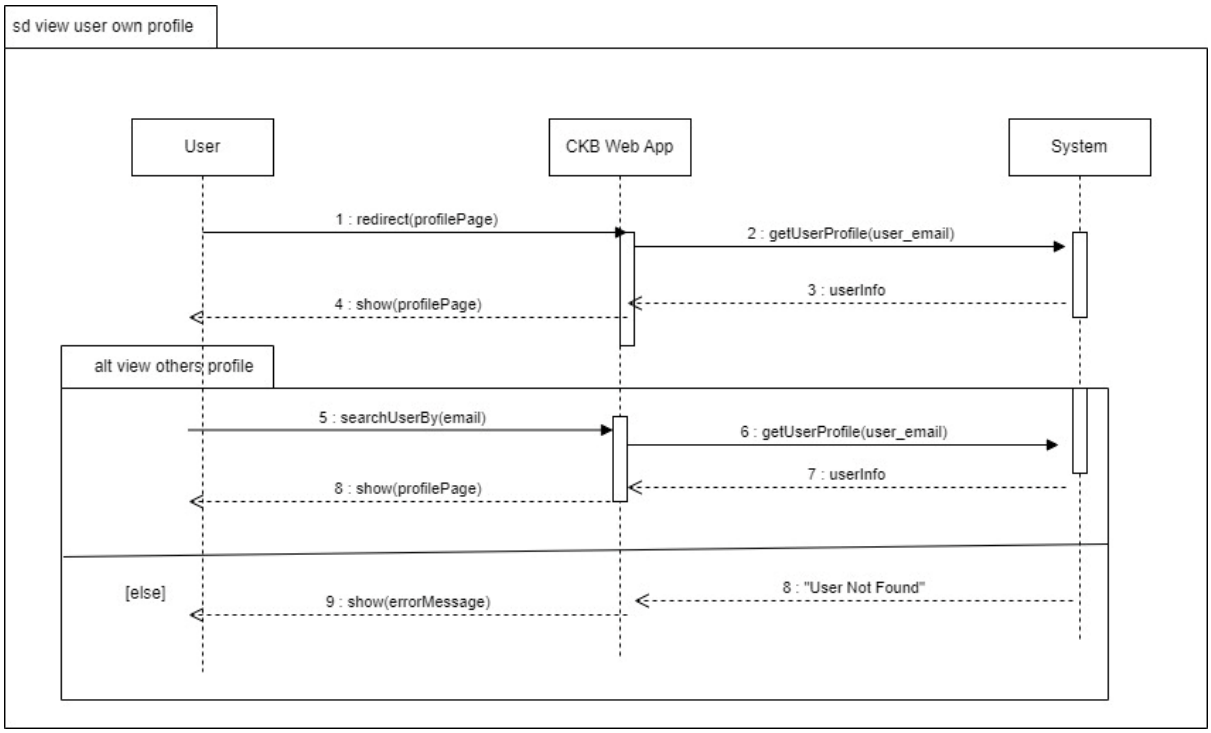
UC-11 Granting Permission



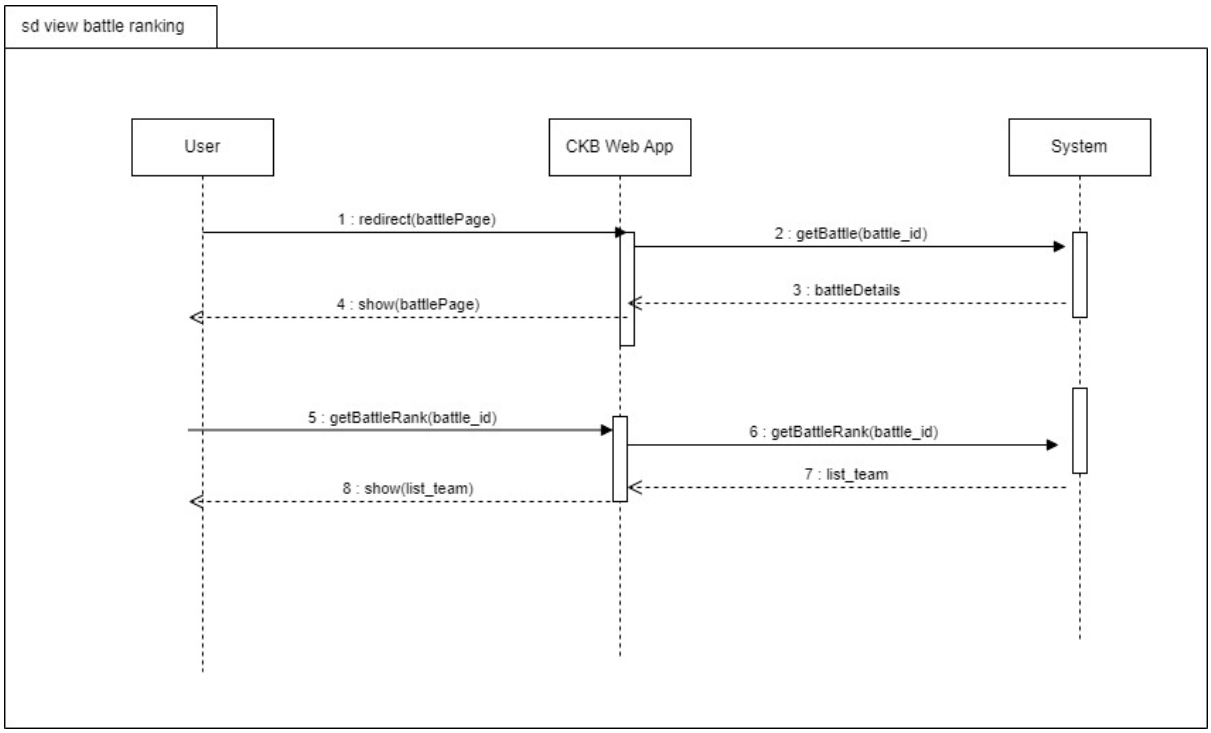
UC-13 Close Tournament



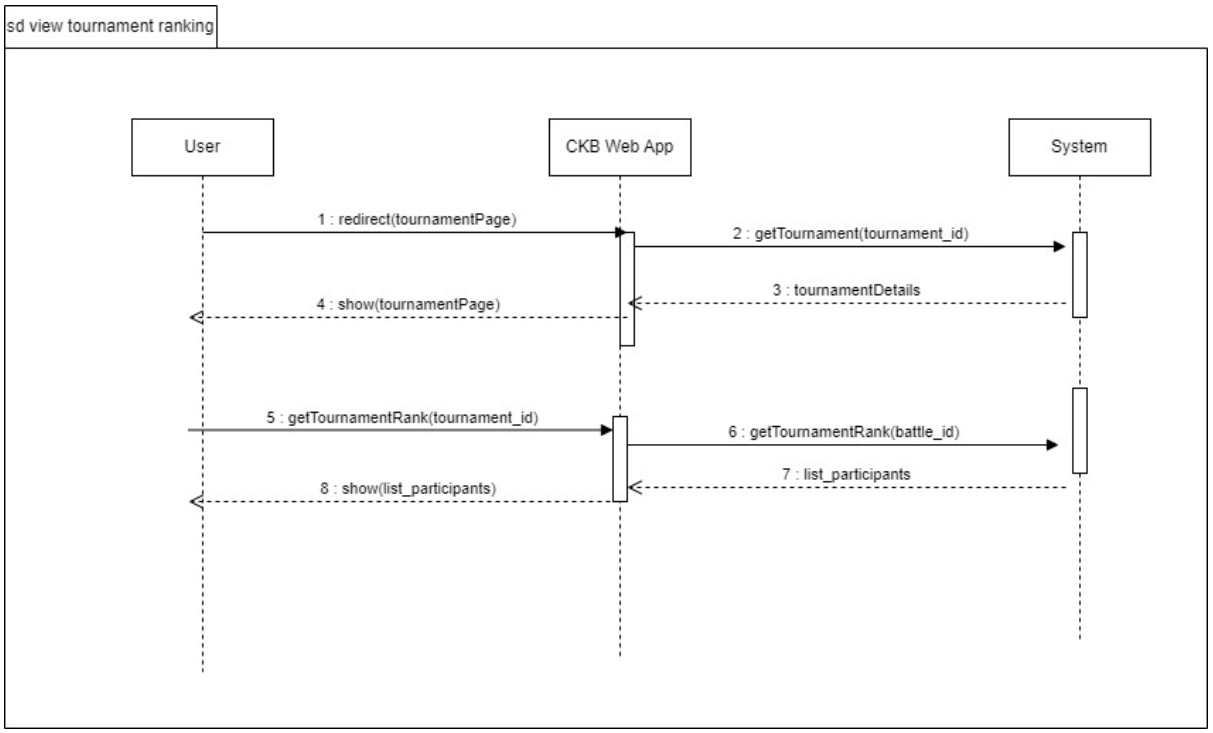
UC-14 View profile



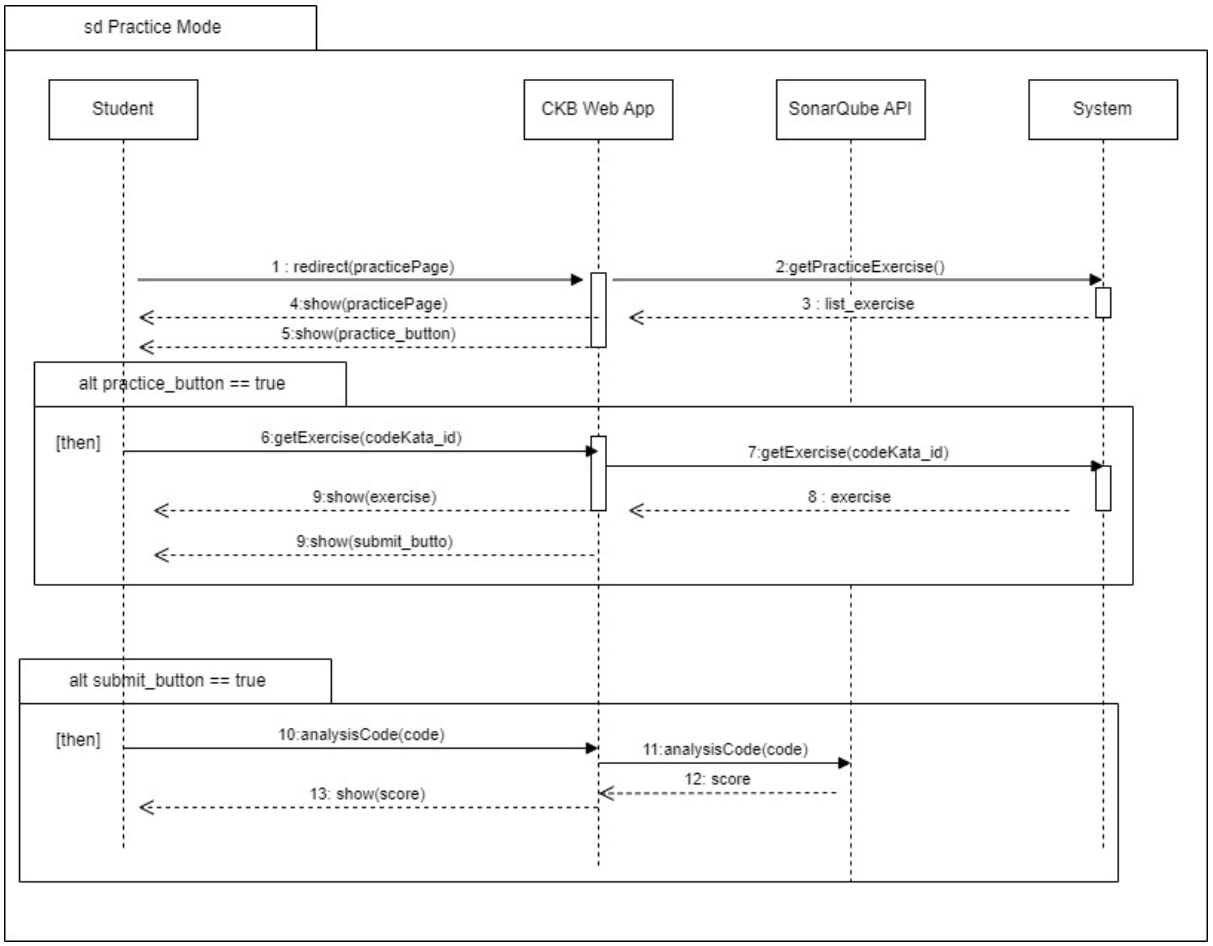
UC-15 View Battle Ranking



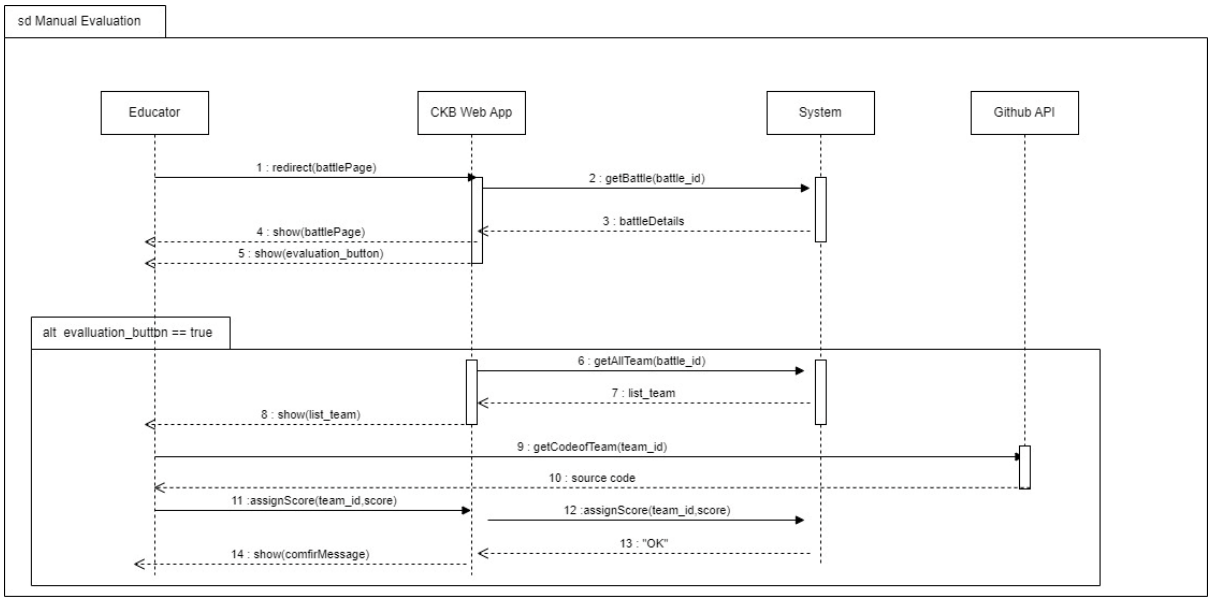
UC-16 View Tournament Ranking



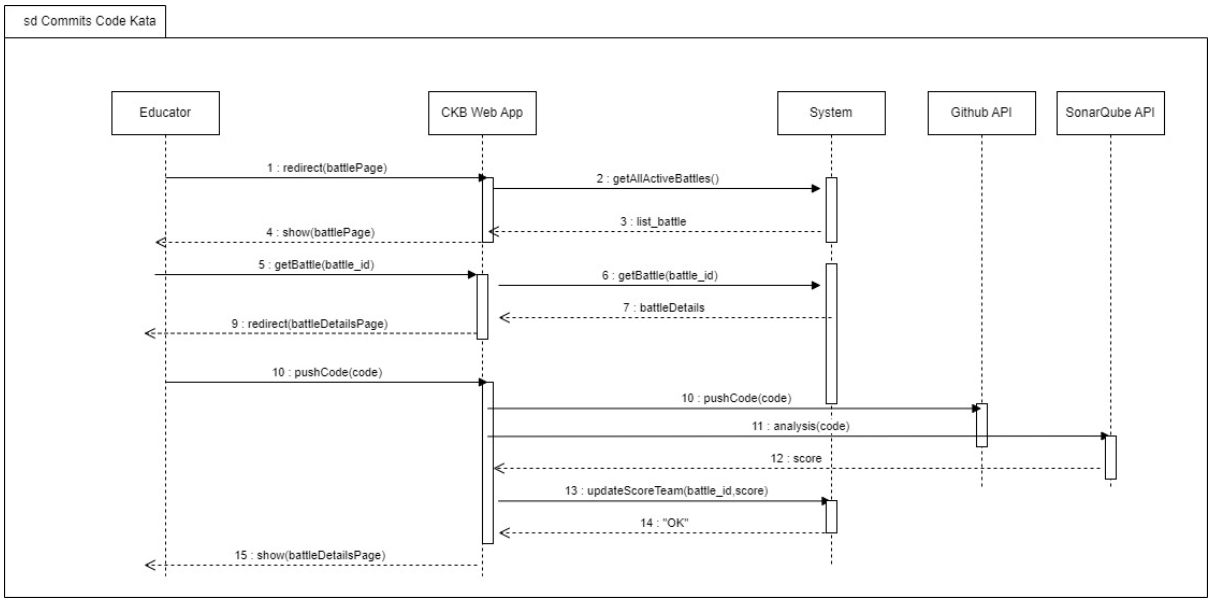
UC-17 Practice Mode



UC-18 Manual Evaluation



UC-19 Commits Code Kata



3.2.4 Requirements Mapping

[G1] Allows Students to select a language for training	
[R1] The system allows Students to sign up. **[R2]** The system allows Students to sign in. **[R5]** The system allows Students to access the profile section. **[R7]** The system allows Students to choose	**[D1]** The CKB platform assumes that users have a stable internet connection to access and interact with its features. **[D5]** Users must undergo a secure authentication phase during the login

their preferred programming language. **[R16]** The system allows Students to access a comprehensive list of battles in which they have previously participated.	process to access the CKB platform.

[G2] Allows Students to subscribe to tournament and battles.	
[R6] The system allows Students to send invitations for specific battles. **[R8]** The system allows Students to explore and view details of individual tournaments. **[R9]** The system allows Students to subscribe to tournaments. **[R10]** The system allows Students to explore and select specific battles with tournaments. **[R11]** The system allows Students to subscribe to battles. **[R13]** The system allows Students to view their scores obtained in battles. **[R14]** The system allows Students to access battle raking. **[R15]** The system allows Students to access tournament ranking. **[R34]** The system allows Students to invite other students to join the team for the battle.	**[D1]** The CKB platform assumes that users have a stable internet connection to access and interact with its features. **[D2]** Once the registration deadline for tournaments or battles has passed, students cannot register for any ongoing or upcoming events. **[D4]** The system must ensure a necessary timestamp between the end of the last battle of a tournament and the conclusion of that tournament. **[D6]** CKB relies on access to GitHub functionality for its features to operate effectively. **[D8]** The platform is expected to deliver notifications to users within a time frame of one minute. **[D11]** Students must respect to the specified maximum and minimum number of group members when forming teams for battles. **[D12]** Each student is expected to participate in a single team for a specific battle, promoting fairness and avoiding potential conflicts of interest.

[G3] Allows Educators to create tournaments and battles	
[R18] The system allows Educators to create new tournaments. **[R19]** The system allows Educators to create new battles. **[R20]** The system allows Educators to grant specific individuals the authority to create	**[D1]** The CKB platform assumes that users have a stable internet connection to access and interact with its features. **[D3]** After the deadline of a tournament, no new battles can be created within the scope

battles within a tournament. **[R21]** The system allows Educators to setting titles and rules for tournaments, referred to as “Tournament Badges”. **[R22]** The system allows Educators to upload code kata with referenced answer. **[R24]** The system allows Educators to set participant limits, registration deadlines and submission deadlines for each battle. **[R25]** The system allows Educators to configure grading parameters (e.g. security percentages...). **[R29]** The system notifies Educators and Students once the tournament has concluded. **[R30]** The system notifies Educator and Students once the battle has concluded. **[R32]** The system allows Educators to access to the code submitted by Students through GitHub commits. **[R33]** The system allows Educators to send manual scores to the platform.	of that tournament. **[D4]** The system must ensure a necessary timestamp between the end of the last battle of a tournament and the conclusion of that tournament. **[D7]** Educators can only rate student submissions after the submission deadline has passed. **[D10]** Each badge earned by a student is associated with a specific tournament, reflecting the achievements within that context. **[D11]** Students must respect to the specified maximum and minimum number of group members when forming teams for battles.

[G4] Allows Students to practice their coding skills with code kata battles	
[R12] The system notifies Students of a GitHub repository link associated with the selected battle. **[R23]** The system allows Students to access referenced answer. **[R26]** The system notifies Students upon the creation of a new tournaments. **[R27]** The system notifies Students upon the updates of a new battle. **[R28]** The system notifies Students in advance of upcoming battles. **[R31]** The system notifies Students of the battle results and the overall tournament results, including scores and rankings.	**[D1]** The CKB platform assumes that users have a stable internet connection to access and interact with its features. **[D2]** Once the registration deadline for tournaments or battles has passed, students cannot register for any ongoing or upcoming events. **[D4]** The system must ensure a necessary timestamp between the end of the last battle of a tournament and the conclusion of that tournament. **[D6]** CKB relies on access to GitHub functionality for its features to operate effectively. **[D8]** The platform is expected to deliver

	notifications to users within a time frame of one minute. **[D9]** The overall tournament score is the sum of scores from all individual battles within that tournament. **[D10]** Each badge earned by a student is associated with a specific tournament, reflecting the achievements within that context.

[G5] Allows Students to practice in offline mode	
[R17] The system allows Students to choose the practice mode for ongoing learning and skill enhancement.	**[D1]** The CKB platform assumes that users have a stable internet connection to access and interact with its features. **[D9]** The overall tournament score is the sum of scores from all individual battles within that tournament.

3.3 Performance Requirements

To ensure the CodeKataBattle (CKB) platform delivers a responsive, reliable, and scalable user experience, the following performance requirements are defined:

3.3.1 Response Time

- User Interaction:
  - The system must respond to user interactions (e.g., code submissions, page loading) within 2 seconds under normal operating conditions.
- Code Compilation:
  - Code compilation processes should complete within 10 seconds for small to medium-sized projects. Larger projects may have varying compilation times, but the system aims to optimize and minimize latency.

3.3.2 Concurrent Users

- Concurrency Handling:
  - The platform must support a minimum of 500 concurrent users without significant degradation in performance. This includes users participating in code kata battles, accessing profiles, and engaging in real-time interactions.

3.3.3 Scalability

- User Growth:
  - The system should be designed to handle a growth in user base by at least 20% per year. Scalability considerations should be in place to accommodate additional users without compromising performance.
- Battle and Tournament Scalability:
  - The platform must efficiently scale to support an increasing number of battles and tournaments. This includes optimizing database queries, resource allocation, and load balancing mechanisms.

3.3.4 Reliability and Availability

- System Uptime:
  - The platform must maintain a minimum of 99.5% uptime over a rolling 12-

month period, excluding scheduled maintenance windows. This ensures users have reliable access to the system.

- Fault Tolerance:
    - The system should be designed with fault-tolerant mechanisms to handle unexpected failures or disruptions, minimizing downtime and ensuring continuity of service.

3.3.5 Data Throughput

- Submission Processing:
    - The system should be capable of processing a minimum of 100 code submissions per minute during peak usage periods.

- Email Notifications:
    - Email notifications to users, such as tournament announcements, should be sent promptly, with an average delivery time not exceeding 1 minute.

These performance requirements are crucial for providing a seamless and efficient experience for users on the CodeKataBattle platform. Regular performance monitoring and optimizations will be conducted to meet or exceed these specifications.

3.4 Design Constraints

3.4.1 Standards Compliance

The CKB platform adheres to various standards to ensure compatibility, security, and best practices in software development.

- Coding Standards:
    - All code generated and submitted on the platform must adhere to widely accepted coding standards in the respective programming languages (e.g., Java, Python). This ensures consistency and readability.

- Security Standards:
    - The platform complies with industry-standard security practices to protect user data, including secure transmission of information and secure storage of credentials. It follows encryption protocols and secure authentication mechanisms.

- Web Standards:
    - The user interfaces and interactions follow web standards to ensure cross-browser compatibility, accessibility, and responsiveness. This includes adherence to HTML, CSS, and JavaScript standards.

- Privacy Regulations:
    - The platform complies with relevant data protection and privacy regulations, ensuring that user data is handled responsibly and ethically. It provides mechanisms for users to control their privacy settings.

3.4.2 Hardware Limitations

The CKB platform operates within specific hardware limitations to ensure optimal performance and resource utilization.

- Minimum System Requirements:
    - Users accessing the platform are required to have devices that meet minimum system requirements, including a compatible web browser and sufficient processing power to handle coding tasks.
- Network Connectivity:
    - The platform relies on network connectivity for user access and data exchange. Users must have a stable internet connection to use CKB effectively.
- Scalability:
    - While efforts are made to optimize platform performance, there are inherent hardware limitations that may impact scalability. The system is designed to handle a reasonable number of simultaneous users, and scalability considerations are continuously monitored.
- Compatibility with Development Tools:
    - Users are encouraged to use up-to-date development tools and environments that are compatible with the platform. Compatibility issues with outdated tools may impact the user experience.

These design constraints are essential to maintaining the integrity, security, and performance of the CodeKataBattle platform. Continuous monitoring and updates are conducted to adapt to evolving standards and ensure a seamless user experience.

3.5 Software System Attributes

3.5.1 Reliability

The system has to be able to run continuously without any interruptions for long periods. To be fault-tolerant the system backend deployment must take advantage of some sort of replication and redundancy. The system has to have offline backups of the data storage to exploit in disaster recovery after a data loss.

3.5.2 Availability

Given the fact that CKB is not an emergency service or anything related to critical situations, the system must provide availability of 99.9%. This means that the average time between the occurrence of a fault and service recovery (MTTR) has to be contained at around 0.365 days per year.

3.5.3 Security

The system stores some sensitive personal information about users, so the security aspect cannot be under- estimated. The central database must be protected with all the available measures to avoid any external or internal attack. The passwords inside the data store have to be encrypted and in case of password recovery, this must never be sent in clear. To communicate over the internet CKB must use some sort of encryption to avoid
traffic sniffing and spoofing, thus avoiding cheating attacks and guaranteeing privacy and consistency.

3.5.4 Maintainability

The system must guarantee a high level of maintainability. Appropriate design patterns should be used, together with good standards. The code must be well documented and hard-coding must be avoided. A testing routine has to be provided and it has to cover at least 75% of the entire codebase, excluding interface code.

3.5.5 Portability

The web application must run on any OS (like Windows, Mac OS, Linux, etc) that supports a web browser.

4. Formal Analysis Using Alloy

4.1 Signatures

```
// Defines a signature Email representing email addresses.
sig Email {}

// Defines a signature DateTime with fields of type Int. The constraint ensures that the date is
// non-negative.
sig DateTime {
    date: one Int,
    hour: one Int,
    minute: one Int,
    second: one Int
} {
    date >= 0 and
    hour >=0 and hour < 24
    minute >= 0 and minute < 60
    second >=0 and second <60
}

// Signature for a programming language.
sig Language {}

 // An abstract signature representing a badge.
abstract sig Badge{}

// An abstract signature representing a test case.
abstract sig TestCase{}

// Signature for a title with a name and a timestamp.
sig Title{
    name: some String,
    time: one DateTime
}

// An abstract signature representing a user.
abstract sig User {}

// Defines a signature Student that extends User. The constraint ensures that all battles in
// closedBattles have status Closed.
sig Student extends User {
```

```
    email: one Email,
    title: set Title,
    closedBattles: set Battle, // student participated battles in the past time
    score: set PersonalScore
}{
    closedBattles.status = Closed
}

// Defines a signature Educator that extends User. The constraint ensures that both closed
// battles and closed tournaments must have the status Closed.
sig Educator extends User {
    email: one Email,
    closedBattles: set Battle,
    closedTournament: set Tournament
}{
    closedBattles.status = Closed and closedTournament.status = Closed
}

// Defines a signature Team. The constraints ensure that the number of members is between
// minMembers and maxMembers, and the sum of autoScores and manualScores is between
// 0 and 100.
sig Team {
    name: set String,
    minMembers: one Int,
    maxMembers: one Int,
    members: some Student,
    battle: one Battle,
    scores: one TeamScore
}{
    #members >= minMembers and #members <= maxMembers and
    scores.autoScores + scores.manualScores <= 100 and
    scores.autoScores >= 0 and scores.manualScores >= 0
}

// Defines a signature Score. The constraints ensure that totScore is the sum of autoScores
// and manualScores, and both autoScores and manualScores are non-negative.
sig Score {
    totScore: one Int,
    autoScores: one Int,
    manualScores: one Int
} {
    totScore = autoScores + manualScores and autoScores >= 0 and manualScores >= 0
}
```

```
// Defines a signature PersonalScore that extends Score and includes references to a student
// and a tourenament.
sig PersonalScore extends Score {
    student: one Student,
    tournament: one Tournament
}

// Defines a signature TeamScore that extends Score and includes references to a team and
// a battle.
sig TeamScore extends Score{
    team: one Team,
    battle: one Battle
}

// Define a signature battle. Constraints ensure a chronological order for ongoing battles.
// Subscription time (regStartTime and regEndTime) must be earlier than the battle start time
// (begStartTime and begEndTime), and the starting time must be earlier than the ending
// time.
sig Battle {
    name: set String,
    regStartTime: one DateTime,
    regEndTime: one DateTime,
    begStartTime: one DateTime,
    begEndTime: one DateTime,
    tournament: one Tournament,
    languages: some Language,
    status: one Status,
    teams: some Team,
    testFile: some TestCase
}

// Define signature tournaments. Constraints ensure chronological ordering, no common
// educators between collaborators and admins, subscription times (regStartTime and
// regEndTime) must be earlier than the battle starts times (beginStartTime and begEndTime),
// and the start time must be earlier than the end time.
sig Tournament {
    name: one String,
    regStartTime: one DateTime,
    regEndTime: one DateTime,
    begStartTime: one DateTime,
    begEndTime: one DateTime,
    awards: some Badge,
    admin: one Educator,
```

```
    status: one Status,
    cooperator: set Educator,
    battles: some Battle
}{
    cooperator & admin = none
}

// Defines an abstract signature Status and two sub-signatures: Ongoing and
// Closed, representing different statuses.
abstract sig Status {}

one sig Ongoing extends Status {}

one sig Closed extends Status {}
```

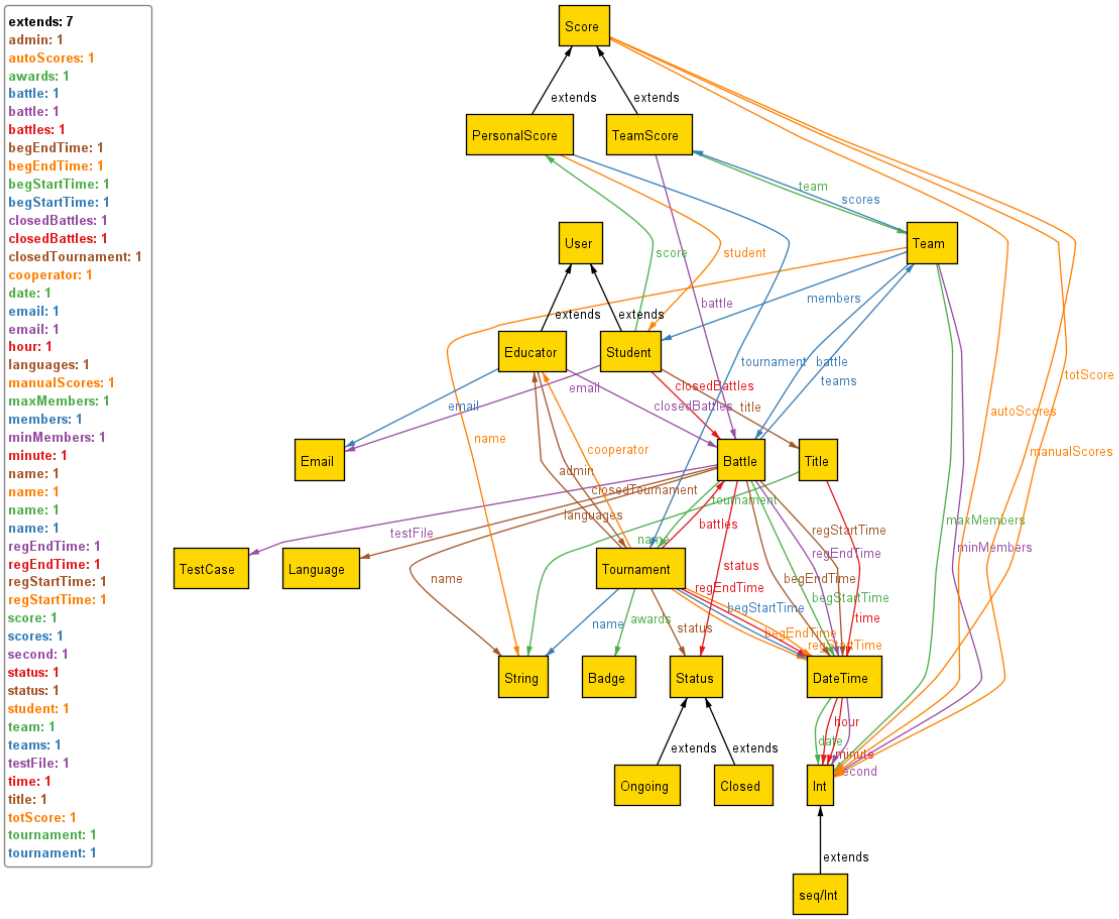
4.2 Facts

```
// It is asserted that there are no instances of both a Student and an Educator with the same
// email address.
fact noSameEmailStudentAndEducator {
    no disj s: Student, e: Educator | s.email = e.email
}

// Ensures that within a battle, there are no two distinct teams with the same name.
fact noDuplicateTeamInOneBattle{
    all b: Battle| no disj t1, t2: b.teams |   t1.name = t2.name
}

// Ensure that no two distinct teams share the same team members within a battle.
fact noDuplicateTeamMember {
    all b: Battle| all disj t1, t2: b.teams | t1.members & t2.members = none
}

 // Asserts that the total score of a team is the sum of the personal scores of its members for a
// specific tournament.
fact personalScoreIsSumofTeamScore {
    all t: TeamScore, p:PersonalScore | p.student in t.team.members
    and p.tournament = t.battle.tournament implies t.totScore = sum( p.totScore)
}

// If a battle is associated with a tournament, it will not be able to continue if the tournament
// is closed.
fact noBattleIfTournamentClosed {
    all b: Battle, t: Tournament | b in t.battles implies not( b.status = Ongoing and t.status =
```

```
        Closed)
}

// Enforce time constraints for battles in a tournament, ensuring proper start and end times.
 fact timeBattleInTournament {
        all b: Battle, t: Tournament | b in t.battles implies
        (isEarlier[t.begStartTime, b.begStartTime] and
        isEarlier[b.begEndTime, t.begEndTime] and
        isEarlier[b.regStartTime, t.begStartTime] and
        isEarlier[b.regEndTime, t.begStartTime]
        )
}

// Establish time constraints for battles, including the order of start and end times.
 fact BattleTimeConstraints{
        all b: Battle |
        isEarlier[b.regStartTime, b.regEndTime] and
        isEarlier[b.regEndTime, b.begStartTime] and
        isEarlier[b.begStartTime, b.regEndTime] and
        (b.status = Ongoing implies (b.begEndTime = none and b.begStartTime != none))

}

// Specifies the start and end times for tournaments within a given time frame.
fact TournamentTimeConstraints{
        all t: Tournament |
        isEarlier[t.regStartTime, t.regEndTime] and
        isEarlier[t.regEndTime, t.begStartTime] and
        isEarlier[t.begStartTime, t.regEndTime] and
        (t.status = Ongoing implies (t.begEndTime = none and t.begStartTime != none))
}

// Defines a predicate isEarlier to compare two DateTime instances, determining if the first
// one is earlier than the second.
pred isEarlier[t1: DateTime, t2: DateTime]{
        t1.date < t2.date or
        (t1.date = t2.date and (t1.hour < t2.hour or
        (t1.hour = t2.hour and (t1.minute < t2.minute or
        (t1.minute = t2.minute and t1.second < t2.second)))))
}

// Defines a predicate show with assertions about the counts of tournaments and battles in
// various statuses.
pred show [] {
```

```
    #Tournament = 2
    #Battle = 2
    #{b1: Battle | b1.status = Ongoing} = 1
    #{b2: Battle | b2.status = Closed} = 1
    #{t1: Tournament | t1.status = Ongoing} = 1
    #{t2: Tournament | t2.status = Closed} = 1
}

run show for 3
```

```
Executing "Run show for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20 Mode=batch
  40926 vars. 765 primary vars. 105160 clauses. 199ms.
  No instance found. Predicate may be inconsistent. 12ms.
```



Metamodel showing the states of the tournaments and battles. It is also possible to see other constraints of the system. Such as one student cannot join two different team in same battle.

5. Time Spent

Chapter	Effort (in hours)
1	5
2	16
3	19
4	13

The timetables written below represent just an approximation of the time spent for the writing and discussions the team had for each specific chapter of this document. These times have not been measured while producing this document and are just based on the personal perception the team members have of the time spent.

Simona Cai

Yang Hao Mao

Chapter	Effort (in hours)
1	5
2	12
3	21
4	16

Jiaxiang Yi

Chapter	Effort (in hours)
1	5
2	13
3	28
4	9

6. References

- Diagrams made with: draw.io
- Alloy models made with: Alloy, Alloy VSCode
- Alloy models runned and checked with: alloytools
- Interface’s graph made with moqups.com