

1. INTRODUCTION

1.1 Purpose

The CodeKata project is conceptualised to address the pressing need for continuous skill enhancement among software developers. Recognizing that expertise in software development is not static but evolves through persistent and reflective practice, CodeKata provides a dedicated environment for developers to sharpen their abilities beyond the confines of their routine professional tasks.

As a Platform for Deliberate Practice, CodeKata is an assemblage of meticulously structured coding challenges, or 'katas', each mirroring complex as well as simple, real-world programming scenarios, faced in the software industry. These katas are crafted to target diverse aspects of coding, and allow the developers to engage with and solve problems that parallel those encountered in their professional lives, mixing theoretical concepts and tangible applications.

The philosophy of CodeKata is rooted in the belief that learning is an iterative process. Developers are encouraged not only to solve problems but also to review and enhance their solutions. This iterative refinement promotes a culture of excellence and continuous self-improvement.

CodeKata emphasises the power of collective learning. It is a place where developers can share their solutions, gain feedback, and engage in discussions about alternative approaches. This collaborative environment amplifies the learning experience, providing a platform for communal growth and knowledge sharing.

CodeKata serves as a repository of collective wisdom, where insights and best practices are shared and accessible to all. It is a resource for newcomers to learn and for experts to contribute back to the community, promoting an ecosystem of mutual learning and mentorship.

– Maybe take out this part ?–

The CodeKata project aspires to be more than just a platform; it aims to be a movement that galvanizes software developers towards a journey of perpetual learning and professional growth. It is designed to ensure that developers, regardless of their experience level, can maintain and enhance their technical acumen, staying at the cutting edge of software development.

1.1.2 Goals

[G1] Create a user-friendly as well as efficient web-based platform called CKB whose main goal is to improve students' coding skills.

[G2] Provide Educators with the necessary tools to create, manage, delete tournaments, battles and manually score the results of the latter

[G3] Implement a gamification system based on winning badges by fulfilling certain requests which will encourage students to code. Besides, teachers will be able to create new badges with a set of rules of their choice.

[G4] Allow students to create and join existent groups to participate in tournaments, battles, modify their profile, win badges, submit solutions via GitHub repositories, access code kata descriptions and contact their educators.

[G5] Implement an intuitive interface, easy to use and straightforward.

[G6] Create a platform with a notification system that will notify students when a tournament is created, if they have joined a battle or the score they got in such a battle. Besides, it will also notify educators if a battle is ready to score (if manually scoring has been set up) or if they have received a message.

[G7] Create user profiles attached to each user with relevant information (names, surnames, e-mails, etc) and different characteristics if they are students or educators profiles.

[G8] Develop automatic scoring that will automatically assign each group a score everytime a battle ends according to different parameters like code clarity and quality.

[G9] Develop manual scoring for teachers to evaluate the results of the groups and allow educators to attach comments in case they want to explain the results giving some feedback.

[G10] Enable measures to prevent hacking, protect data by for example adding authentication methods, ensure fault tolerance and data-recovery protocols, creating in this way a robust as well as secure platform.

[G11] Allow communication between students and educators through several ways like e-mail, forums or direct messages.

[G12] Include a section explaining how the several features of the platform work (gamification, sending messages, joining a group, creating a tournament, etc) with an index so users can directly check whatever they need.

1.2 Scope of the CodeKata Project

The scope delineates CodeKata's functional parameters, target audience, and content coverage. It defines what the platform will deliver, setting clear expectations for users and stakeholders.

1.2.1 Target Audience

The primary audience of the CodeKata project is directed towards all the spectrum of people involved in software engineering.

Therefore the main target audience are the all level software developers:

- New Developers: Offering foundational exercises to build their coding skills from the ground up.

- Intermediate Programmers: Providing intermediate challenges to help bridge the gap to advanced coding.
- Expert Coders: Supplying complex katas for honing advanced skills and staying sharp in current technologies.

However secondary targets must be included, such as

- Educators and instructors in the realm of software development seeking structured exercises for their students.
- Tech recruiters and organisations aiming to assess coding abilities of potential hires.

By catering to this broad user base, CodeKata aims to become an essential resource for personal development, professional assessment, and educational purposes.

1.2.2 Geographical Scope

The CodeKata project leverages its digital framework to offer global accessibility from the outset.

Our initial content curation will cater primarily to english-speaking markets, with a special focus on regions such as North America, the United Kingdom, and Australia, where the demand for software development skills is robust and English is the primary language of instruction and business.

However, recognizing the value of diversity and inclusion, we are committed to multilingual expansion. Structured plans are in place to extend our reach to non-English speaking regions in the next phases, starting with the most widely spoken languages in the tech industry.

This phased approach allows us to ensure quality and relevance in our offerings, as we scale to support a truly global community of software developers.

1.2.3 Functional Scope

The CodeKata platform is engineered to offer a comprehensive suite of features designed to empower developers in their learning journey:

- **Competitive Exercises Access & Development of Coding Skills:** An extensive collection of coding challenges is readily available. And developers can decide to tackle any of them, sharpening their skills at their own pace.

This allows the users to challenge themselves against others, by taking part in the competitions.

- **Competitive Exercises Submission:** Instructors and educators can organise coding tournaments, creating custom problems for their students or community members to solve. This feature facilitates an engaging and competitive learning environment, promoting motivation and practical application of skills.
- **Community Interaction:** Robust discussion forums and feedback mechanisms foster a collaborative spirit, allowing users to engage in problem-solving, knowledge sharing, and networking.
- **Performance Tracking:** Tracking system and challenge result analysis provides detailed insights into user progress, allowing him to identify both strengths and areas in need of development, facilitating a tailored growth path.

These features are designed to evolve with our user base, incorporating user feedback to continuously improve and scale the learning experience. Unique to CodeKata is the integration of these elements into a seamless, user-friendly interface that emphasises engagement and personal growth.

1.2.4 Content Scope

The CodeKata platform presents a diverse repository of coding challenges, known as 'Katas', designed to cater to a spectrum of skill levels — from beginners to seasoned experts. These exercises will be oriented to all kinds of software engineering principles and design patterns. The challenges are rooted in practical, real-world problem scenarios, preparing users for the kinds of tasks they may encounter in their professional lives.

To complement hands-on practice, the platform provides a wealth of knowledge resources and tutorials. This curated collection includes comprehensive reading materials, step-by-step walkthroughs, and interactive tutorials, all aimed at reinforcing learning and offering in-depth understanding of complex topics.

Recognizing the dynamic nature of software development, CodeKata ensures the content remains up-to-date and in line with emerging technologies and methodologies. The platform welcomes contributions from the community, enabling experienced developers and educators to share their expertise and insights, thus continuously enriching the educational content.

For those seeking a more structured approach to learning, CodeKata introduces learning paths that guide users through a series of progressively challenging topics, tailored to their developing skills. We also cater to various learning styles by offering

content in multiple formats, ensuring that whether you prefer visual, auditory, or kinesthetic learning, there's something for everyone.

1.2.5 Integration Scope

While the primary platform is web-based:

- Mobile application support for Android and iOS might be considered in future phases.
- Potential integration with other educational platforms or Learning Management Systems (LMS) for seamless content delivery.

1.2.6 World Phenomena – Less bad than before –

– TODO add the world phenomenon –

[WP1] Student improves his/her coding skills during class

[WP2] Educator publishes the dates of future tournaments

[WP3] Student wants to use CKB

[WP4] Educator recommends CKB to his/her students

[WP5] User has internet connection

[WP6] Educator wants to create a tournament

1.2.7 Shared Phenomena – Less bad than before –

– TODO add the shared Phenomena –

1.2.7.1 Shared Phenomena Controlled by the World and Observed by the Machine

[SP1] User inserts his/her credentials

[SP2] User updates his/her profile

[SP3] User logs in

[SP4] User logs out

[SP5] Educator manually scores a group

[SP6] Educator creates a tournament

[SP7] Educator creates a badge

[SP8] Educator creates a battle

[SP9] Student joins a group

1.2.7.2 Shared Phenomena Controlled by the Machine and Observed by the World

[SP10] User earns a badge

[SP11] A groups is assigned with a score

[SP12] The system sends a notification to a student/educator

[SP13] The system denies access to an user with the wrong credentials

[SP14] The platform asks the user to write a review

[SP15] The platform closes a tournament

[SP16] The platform finishes a battle

1.2.8 Limitations

While the CodeKata project offers an immersive and interactive platform for developing software engineering skills, it is important to recognize its scope and limitations:

- The platform is designed to augment and enrich the learning experience rather than serve as a standalone educational institution. It provides practical coding exercises and challenges that ideally complement comprehensive software development courses and formal education.
- While fostering a community of practice and feedback is at the heart of CodeKata, it is not intended to replace professional coding collaboration tools or version control systems. Instead, it serves as a preparatory and intermediary space where developers can refine their skills before applying them in a collaborative professional environment.

The essence of the CodeKata project is to create a supportive ecosystem for continuous learning, offering global access to diverse content that works alongside other educational resources. It is about providing a supplemental toolkit that enhances the real-world applicability of software development skills.

1.3 Definitions, Acronyms, Abbreviations – Add more as we go on –

1.3.1 Definitions

Kata: A term borrowed from martial arts, representing a coding challenge or exercise designed to practise software development skills.

A term borrowed from martial arts that means “shape” or “form”. It is practised in Japanese martial arts as a way to memorise and perfect the movements being executed. In the platform we are developing, a Kata will represent a coding challenge or exercise designed to practise software development skills

Tournament: A competition hosted on the CodeKata platform where multiple users or teams can solve challenges to rank against each other.

CodeKataBattle: The name of the platform to be realised where these coding exercises and tournaments will be hosted.

Challenge: A problem or exercise on the CodeKata platform that requires a solution or code submission from the user.

Leaderboard: A feature within the platform that ranks users or teams based on their performance in challenges or tournaments.

Collaborative Problem Construction (CPC): The process by which educators and mentors can create and publish their own challenges on the platform.

Learning Path: A curated sequence of challenges and resources designed to progressively advance a user's skills in a particular domain of software development.

Personalization Engine: The underlying technology on the CodeKata platform that suggests relevant challenges and resources to users based on their past activity and preferences.

Community Forum: An integrated discussion board on the CodeKata platform where users can interact, share knowledge, and seek help.

Progress Tracking: The system that monitors and reports a user's progress through completed challenges, providing analytics on performance.

LMS Integration: The potential capability for CodeKata to integrate with Learning Management Systems for educational institutions or enterprises.

These terms are essential for understanding the full scope and functionality of the CodeKata platform and will be used consistently throughout this document.

1.3.2 Acronyms

CKB: Code Kata Battle

1.3.3 Abbreviations

- **[Gn]** - It indicates a Goal and its n number.
- **[Rn]** - It indicates a Functional Requirement and its n number.
- **[WPn]** - It indicates the World Phenomenon and its n number.
- **[SPn]** - It indicates the Shared Phenomenon and its n number.
- **[MPn]** - It indicates the Machine Phenomenon and its n number.

1.4 Revision History

Version	Date	Description of Changes	Author
1.0	18/10/2023	Initial Introduction Draft	Adrian Valica
1.1	09/11/2023	Finishing Introduction 2nd Draft	Pablo García Alvarado
1.2	14/12/2023	Finishing points 2 and 3	Pablo García Alvarado

1.5 Reference Documents

Here are references towards di

The document is based on:

- GITHUB: <https://github.com>
- BITBUCKET (we do not need to use this): <https://bitbucket.org/product/>
- KITTENS KATA SITE: <http://codekata.com/>
- WHAT IS TDD?: https://en.wikipedia.org/wiki/Test-driven_development
- EXAMPLE OF SIMILAR SITE: <https://www.codewars.com/>

1.6 Document Structure

1. INTRODUCTION: In this section we will cover the goals and reasons that we will want to accomplish with this project as well as the document structure and clarify concepts, abbreviations and acronyms so anyone can understand the concepts listed along the document.
2. OVERALL DESCRIPTION: With this part we will describe the shared-phenomena and the respective domain model.
3. SPECIFIC REQUIREMENTS: As the name indicates, here we will describe the requirements needed to achieve the aforementioned goals. Besides, we will describe the different interfaces, namely Hardware, Software, User and Communication interfaces. Finally design constraints and the software system attributes.
4. FORMAL ANALYSIS USING ALLOY: It will formally describe the world phenomena by using Alloy.

5. EFFORT SPENT: We will indicate the time spent on every section by all the group members during the development of the project.
6. REFERENCES: We will thoroughly list all the references of the books, websites, and contents that we have used to make the project.

2. Overall Description

2.1 Product Perspective

2.1.1 Scenarios

1. User takes part into a Code Kata Battle:

Alfredo wants to sharpen his skills in Software engineering. In order to do this, he goes on CodeKataBattle to find a challenge. He searches the challenges related to his coding language, and can set some more filters. After timeFrame, the system shows up all the possible challenges and he can browse through the list.

Alfredo clicks on the "Take part" button of the challenge he chose. The systems create a Github repository, and shortly after (timeframe), it shows the link to the repository on the screen. Alfredo can click it and access it.

On the page of the challenge, some runnable tests snippets appear. When clicking on any of the "Run" buttons, or on the "Run all" button, the system proceeds to launch all the tests on the code of the repository. As soon as it is done (timeFrame), the system shows some results (errors, or output of the program), and if missed tests are highlighted in red.

Alfredo and his team can therefore Alfredo, aiming to sharpen his skills in software engineering, navigates to CodeKataBattle in search of a challenge. He specifies his preferred programming language and fine-tunes the search results by applying filters for difficulty level, topic, and expected duration. Within moments, the system displays a curated list of challenges that perfectly align with Alfredo's criteria.

From the list, Alfredo selects a challenge that catches his interest and clicks the "Take part" button. The system immediately generates a private GitHub repository for the selected challenge and promptly provides Alfredo with a direct link to access the newly created repository.

On the challenge page, Alfredo finds runnable test snippets. To evaluate his initial code, he can click on "Run" for individual tests or "Run all" to execute the full suite of tests against his repository code. The system quickly provides detailed feedback,

displaying the output or errors for each test case. Any failing tests are clearly marked in red, allowing Alfredo to easily identify areas that need attention.

Throughout the duration of the challenge, Alfredo and his team engage in collaborative coding, iterating on their solution. The system supports this process by offering real-time feedback and performance metrics, and they can see their score on the global leaderboard at every submission. It is updated at each push of code on the repository.

At any time, Alfredo can submit his final solution. The code is directly taken from the Github repository and is saved as the final result of his participation.

2. User checks the results of the competition

Pierluca, eager to see how he fared, receives an email and in-app notification from CodeKataBattle indicating that the results of the challenge he recently participated in are now available. The message comes after a meticulous consolidating phase where organizers grade and verify all submissions.

He promptly logs into CodeKataBattle and navigates to 'My Challenges' on his dashboard, which efficiently organizes his participations in reverse chronological order. Selecting the challenge in question, Pierluca clicks to uncover a detailed report of the event.

The platform provides him with an interactive scorecard, clearly displaying his overall score out of 100. It delves into his performance, offering a nuanced breakdown across essential factors:

- Functional Correctness: Displayed as the ratio of passed test cases to the total, showcasing his solution's efficacy.
- Timeliness: Measured by the interval between the registration deadline and his last commit, indicating his efficiency.
- Code Quality: Evaluated through static analysis tools that assess various code aspects pre-selected by the challenge creator—such as security, reliability, and maintainability—reflecting the sophistication of his work.

Pierluca is then presented with the final leaderboard, a dynamic display of top-scoring solutions and a comprehensive list of rankings. His own submission is distinctively highlighted, allowing him to instantly locate his standing.

To encourage a culture of learning and improvement, the platform encourages exploration. Pierluca can view profiles and submissions of top-ranking participants, glean insights from their approaches and potentially enhancing his own software engineering acumen for future challenges.

3. User subscribes to an upcoming Challenge

Pablo wants to have some programming exercise and decides to join a new CodeKataBattle challenge. He signs in to the CodeKataBattle platform and employs

filters to select challenges that align with his expertise and preferences. His attention is captured by a challenge scheduled for the upcoming weekend which perfectly fits into his schedule. The interface conveniently displays the time remaining until the challenge begins and prompts him with an "Enroll" option.

Upon clicking "Enroll," Pablo's participation is confirmed, and his account is seamlessly added to the participant list for the challenge. The system acknowledges his enrollment and ensures he's informed about the essential details.

In anticipation of the challenge start date, the platform proactively generates a private GitHub repository for each contestant. This strategic move is designed to distribute system load and prevent delays. Pablo, along with the other participants, is notified as the challenge kicks off, marking the beginning of an exciting coding journey.

At the start of the challenge, Pablo receives an alert, along with a link to his exclusive GitHub repository, which had been kept secret until now. With a click, he accesses his repository and is all set to dive into the coding problems presented by the CodeKataBattle challenge, ready to code his way to solutions.

4. User cancels his participation before the challenge starts

Mario, a student and an aspiring programmer, had enthusiastically registered for an upcoming CodeKataBattle weekend challenge. However, as the week progresses, he realizes that his video game project is likely to require more of his time than he initially thought. He makes the practical decision to cancel his participation in the challenge.

He logs into his account on the CodeKataBattle platform and heads straight to the 'My Challenges' section. There, amongst his list of activities, he easily spots the upcoming challenge. Right next to the challenge details, the 'Withdraw' button catches his eye, indicating a clear option for opting out.

With a purposeful click on 'Withdraw,' a confirmation prompt appears, ensuring that he doesn't accidentally cancel his participation. Mario reaffirms his decision, and clicks 'Confirm.' His action is immediately processed by the system. In response, his account is removed from the registered account for the challenge is cancelled, his dedicated private GitHub repository for the challenge is erased, and the challenge itself vanishes from his 'My Challenges' list.

Now certain that he will not be interrupted by any notifications related to the challenge, Mario focuses his attention back on his video game project. Despite withdrawing, he can effortlessly re-enroll in the challenge any time before it officially ends.

5. User cancels his participation during the challenge

Just as the CodeKataBattle challenge begins, Luigi receives a call from his brother, who urgently needs his help over the weekend. Family comes first for Luigi, and he quickly realizes that he won't be able to participate in the challenge. He decides to withdraw to support his brother.

Luigi logs into CodeKataBattle, heads to the challenge page, and immediately spots the 'Withdraw' button. He clicks it, and a confirmation pop-up appears. Without hesitation, Luigi confirms his decision, understanding the implications.

Upon confirmation, the system efficiently processes Luigi's withdrawal. It removes his account from the list of active participants, ensuring his prompt exit from the competition. The system takes further steps to maintain data integrity and privacy: it erases Luigi's submission history, clears any of his entries from the leaderboard, and deletes the GitHub repository that was allocated for his participation in the challenge. As a final step, the challenge itself is removed from Luigi's list of current participations.

While Luigi steps back from the challenge to focus on family matters, he remains aware that he can rejoin the competition at a later time if his schedule allows. However, Luigi knows that if he chooses to re-enter, he will have to start from scratch, as all his previous progress has been completely reset following his withdrawal.

6. Educator creates a tournament

Dr. Luca, an educator of the computer science class, decides it's time to challenge his students with a practical coding tournament. He plans to use CodeKataBattle to create this exercise session.

He logs into his CKB educator account. His credentials grant him access to an array of tools tailored for educators. Dr. Smith navigates to the 'Create Tournament' section. He's greeted by a user-friendly form designed to capture all the essentials of a compelling coding tournament.

With careful thought, Dr. Smith begins filling in the details of the tournament. He chooses the title of the tournament, and a detailed description. He considers the skill level of his students and sets the tournament's difficulty level accordingly. He drafts a series of problem statements, each designed to test different aspects of coding, and sets two deadlines: one for the students to register in groups to the tournament, and one for the end of the submissions in the tournament challenges.

Next, Dr. Luca uploads several supporting documents. These include resources to help his students understand the problems better and criteria for automatic scoring that ensure a fair and objective evaluation of the submissions.

Before making the tournament live, Dr. Luca takes a moment to review all the information. Confident with the setup, he clicks the 'Publish Tournament' button.

Within moments, the tournament goes live on CKB. Simultaneously, an automated notification is dispatched to all his students, inviting them to participate in this exciting coding adventure.

7. Educator scores a challenge manually

The deadline has passed, and it's time for Dr. Luca to start checking all the submissions that await his review.

He logs into the CodeKataBattle platform, where he's greeted with a dashboard containing the latest submissions. One particular challenge in the tournament, known for its complexity, requires a more nuanced approach—manual scoring.

Dr. Luca carefully opens each submission, his screen transforming into a canvas of code. He meticulously evaluates each key factor such as efficiency, readability, and correctness for every submission.

Dr. Luca assigns a score to each submission. He also writes detailed feedback for each student, comments designed to guide, encourage, and enlighten.

Once the scoring is complete, Dr. Luca publishes the scores and feedback on the CodeKataBattle platform. All the students that took part into the tournament are notified that the grades are available. The manual scoring of Dr. Luca and the automatic scoring from CodeKataBattle are combined to provide a final score, with proportions fixed in advance by the educator.

8. Educator adds a new collaborator

Dr. Calu wants to use the tournament that Dr. Luca created in order to evaluate his students. But he also wants to add some more exercises for his class.

Dr. Luca agrees to this and wants to add him as a collaborator on the tournament management. He logs into the CodeKataBattle account and on the page of his tournament, he clicks on the button “Add Collaborators”. A pop up appears, and he now enter a username to add. He enters Dr. Calu’s username and confirms his choice.

Dr. Calu gets notified on his account that he is invited to become a collaborator on the tournament. As he accepts the invitation, he is brought to the page of the tournament. The interface is the same as the one of Dr. Luca’s, he can edit anything and manage the tournament as its owner, with one difference: he cannot delete it.

Dr. Luca can see the list of the members on the top of the screen and decide to restrain one’s access anytime.

9. User earning a badge

Alice, a diligent and skillful coder, has been a whirlwind of activity on the CodeKataBattle platform, partaking in numerous tournaments. Her dedication to the

craft is evident in her consistent top-tier performances, each submission a testament to her growing prowess.

Behind the scenes, the gamification system of CKB acts as a silent auditor of success, meticulously tracking Alice's achievements. It's not just a counter of points; it's a recognizer of effort and skill. As Alice continues to excel, the system notes her eligibility for one of the most coveted acknowledgments on the platform – the 'Coding Master' badge.

The moment Alice's latest successful submission ticks the final checkbox on the list of criteria, the system springs into action. With digital fanfare, the 'Coding Master' badge is automatically awarded to her, its icon taking pride of place on her CKB profile.

Almost immediately, a notification pops up, a digital drumroll announcing her new achievement. Alice clicks on the alert, and her profile blooms with the new badge, its presence a badge of honor and a milestone in her coding journey.

As Alice views her badge, a symbol of her dedication and expertise, it serves as both a reward and a beacon, inspiring her and her peers to continue pushing the boundaries of their coding capabilities on CodeKataBattle.

10. Users forming a group for a challenge

John and Emma, both keen to leverage their collective problem-solving skills, decide to join forces for a group challenge hosted on CodeKataBattle (CKB). Recognizing the power of collaboration, they are excited to see what they can achieve together.

Taking the first step in their collaborative journey, they log into CKB and set up a group on the platform, seamlessly inviting one another to join. This virtual space is set to become their hub of innovation and teamwork.

With the group established, John and Emma engage in a strategy session. They use the platform's integrated chat feature to communicate in real-time, discussing the challenge's intricacies and plotting their course of action. They divide the tasks according to their strengths, with Emma taking on the algorithmic challenges and John focusing on data structure optimization. For more complex aspects of the project, they turn to external tools that offer real-time collaborative coding capabilities, ensuring that no stone is left unturned.

The challenge is in full swing as they begin to weave their code together. They each contribute parts of the project, merging their individual work into a cohesive whole. After rigorous testing and revision, they are ready to push their combined code to their shared GitHub repository. With a sense of accomplishment, they submit their final solution through CKB, marking the culmination of their joint effort.

As they await the verdict, the CKB system evaluates their submission. Before long, John and Emma receive detailed feedback on their work, along with points that

reflect their hard work and creativity. These points are a boon to their individual profiles, contributing to their standings on the platform and opening up new opportunities for learning and growth.

Their successful collaboration on CodeKataBattle not only strengthens their programming abilities but also cements a partnership that blends two coding minds into a formidable force.

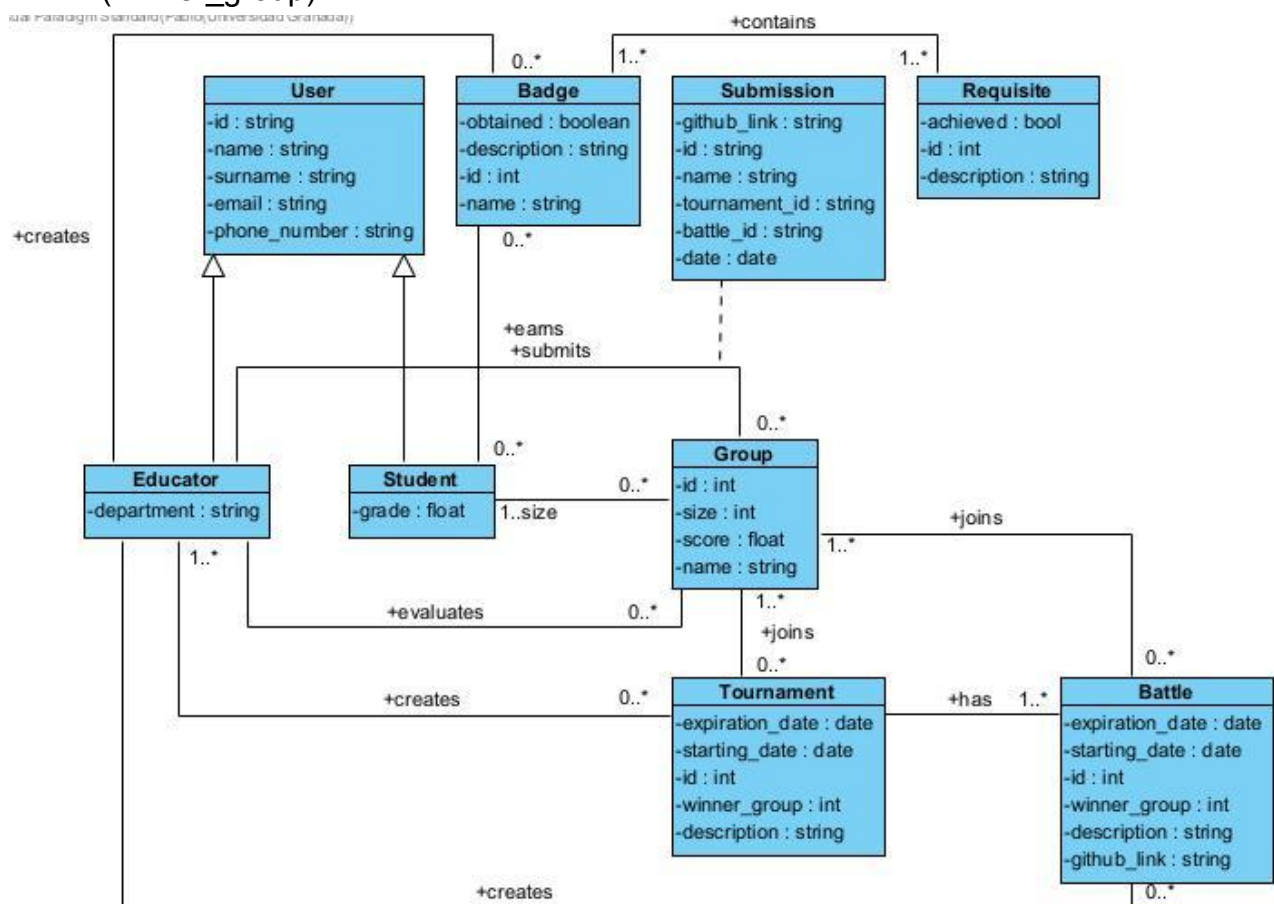
2.1.2 Domain Class diagrams

Figure 2.1 represents the Domain Class Diagram for the CKB platform. It contains all the elements involved in the system's operations and how they interact among them. The elements will be explained in natural language to clarify if needed what does each class represent:

- There are two types of Users (User): Educators (Educator) and Students (Student) each of them having different kinds of interactions with the platform. The former will have a department assigned (department) to identify their field of expertise within the Educational Institution. It will also be able to create as many badges as she/he wishes by assigning their respective rules or not to create badges at all. The same with tournaments and battles by specifying the starting and expiration date as well as the id. An educator user will also be able to score different groups (Group). The latter will have a grade (grade) field that teachers will update accordingly but a student is not allowed to change his/her grade. However, a student will be able to both create groups or join them without a minimum number of groups to join. A student user cannot also be an educator and vice versa. Nevertheless, they will have a few features in common, like personal information, that is an id (id), a name (name), a surname (surname), e-mail (email) and last but not least a phone number (phone_number).
- Many students (Student) or just one can conform to a group (Group) which is identified by an id (id), the size (size) of the group, that is the number of integrants, a name (name) as well as the score (score) which is the overall score they got in all the battles. A group can join as many tournaments as it wants or not join any tournaments at all that also applies to the battles taking place within the tournament.
- A badge (Badge) has at least a requisite (Requisite) to obtain it, a description (description) in clear and concise natural language, an id (id), a name (name) as well as a flag (obtained) that will be true if an user x has earned the badge and false otherwise.
- A requisite (Requisite) will contain a flag (achieved) that will equal true if it has been achieved and false otherwise. Another attribute will be the id (id) and the description (description) that will explain how to achieve that requisite in natural language.
- We have an association class called Submission that will contain the submissions the groups (Group) will send to the teacher (Educator) and it will

contain the following attributes: a GitHub link (github_link), an id (id), a name (name), the tournament id (tournament_id), a battle id (battle_id) and a date (date) that will contain the date when the submission was submitted. A group can submit as many submissions as they wish but only the last one (latest date) will be the one evaluated by the professor.

- Educators can create several tournaments (Tournament) where groups (Group) can join and code kata battles (Battle) will take place. As attributes, a tournament will contain both expiring and starting dates (expiration_date and starting_date respectively), a description (description), an id (id) and a winner group id (winner_group) where it will be stored the id of the group that won the tournament.
- Educators can create several code kata battles (Battle) within a tournament (Tournament), a Battle can only take place in a tournament and several groups can take place in a battle or none of them. It contains both expiration and start dates (expiration_date and starting_date respectively), an id (id), a description (description), a GitHub link (github_link) that redirects users to the repository containing the needed files and finally the id of the winner group (winner_group).



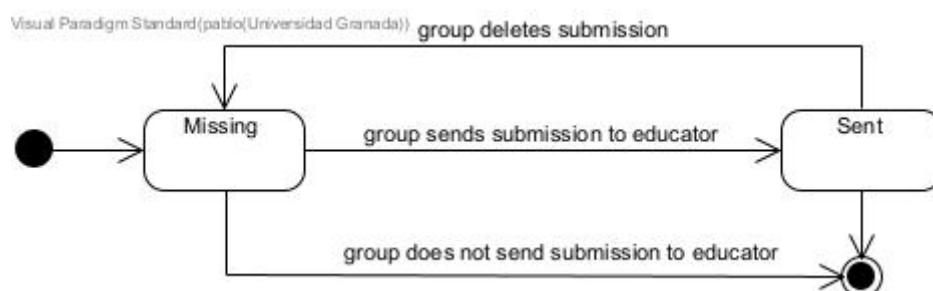
(Figure 2.1) Domain Class Diagram

2.1.3 State Diagrams

In this section we will explain some of the states in which the instances of the diverse classes of our system could be in using both natural language and State Diagrams.

Submission Process

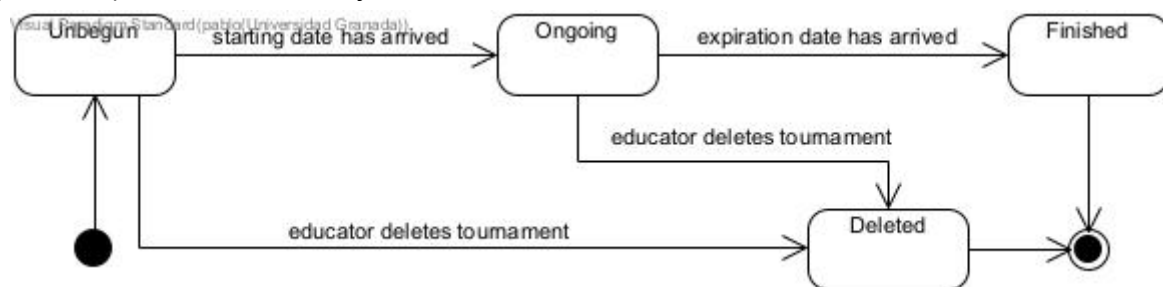
Dr. Agosta creates a task (submission) to be submitted within a week. In the beginning nothing has been sent so the task is missing (Missing), but not for very much longer because the group conformed by Paolo, Federico and Beatrice submits the submission (Sent). However, since they are not happy with the result, they decide to delete the submission (Missing) and after that they forget to submit again the new version (Missing) and the week ends so they cannot upload anything in the end.



(Figure 2.2) Submission State Diagram

Tournament Process

Dr. Baldini creates a tournament with its starting and expiration date. In the beginning the tournament has not yet started (Unbegun), but once the starting date arrives (Ongoing) battles can start taking place and once the expiration date arrives (Finished) the tournament is finished for good. An educator can delete a tournament (Deleted) if it is not already in the Finished state.

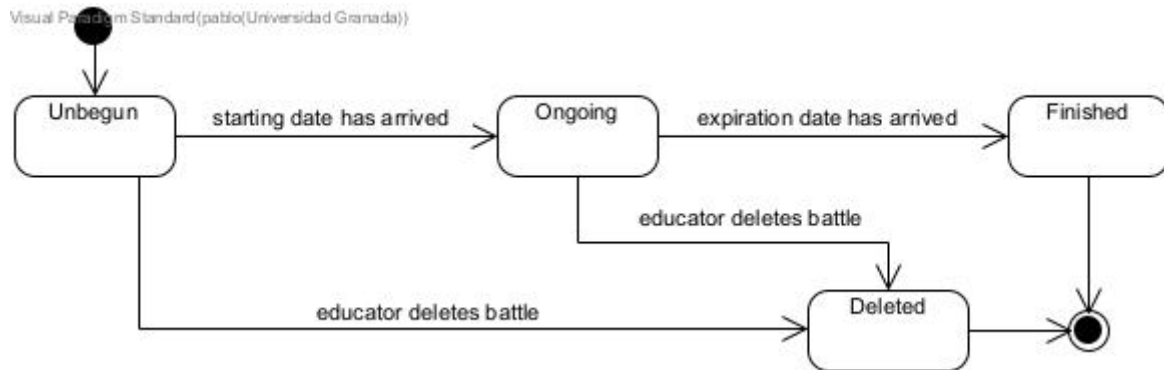


(Figure 2.3) Tournament State Diagram

Battle Process

A battle is created by Dr. Camino with its respective starting and expiration date. The state diagram works in a similar way to the tournament one. First the battle has not started (Unbegun) because the starting date has not arrived, but once it does (Ongoing), the battle is available for the student groups that joined it and they can now participate in it. Once the expiration date arrives (Finished), the battle is finished

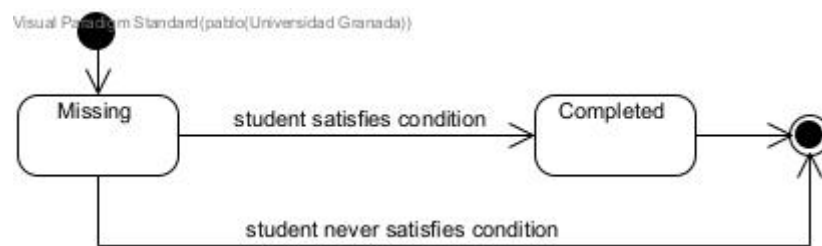
and groups cannot interact with it anymore. An educator can delete a battle (Delete) if it is not already in the Finished state.



(Figure 2.4) Battle State Diagram

Requisite Process

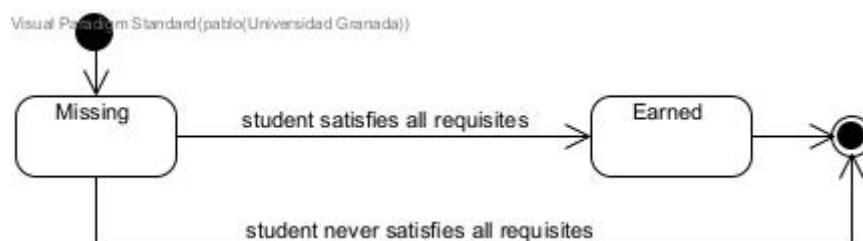
Student Artemisia is coding and submits another task, turning her into the student that more work has done within the month, which will implicitly turn the requisite “Coder of the month” from being missing (Missing) to being fulfilled (Completed). Another student called Francesco never gets to be coder of the month so in that case his requisite will remain in the missing state.



(Figure 2.5) Requisite State Diagram

Badge Process

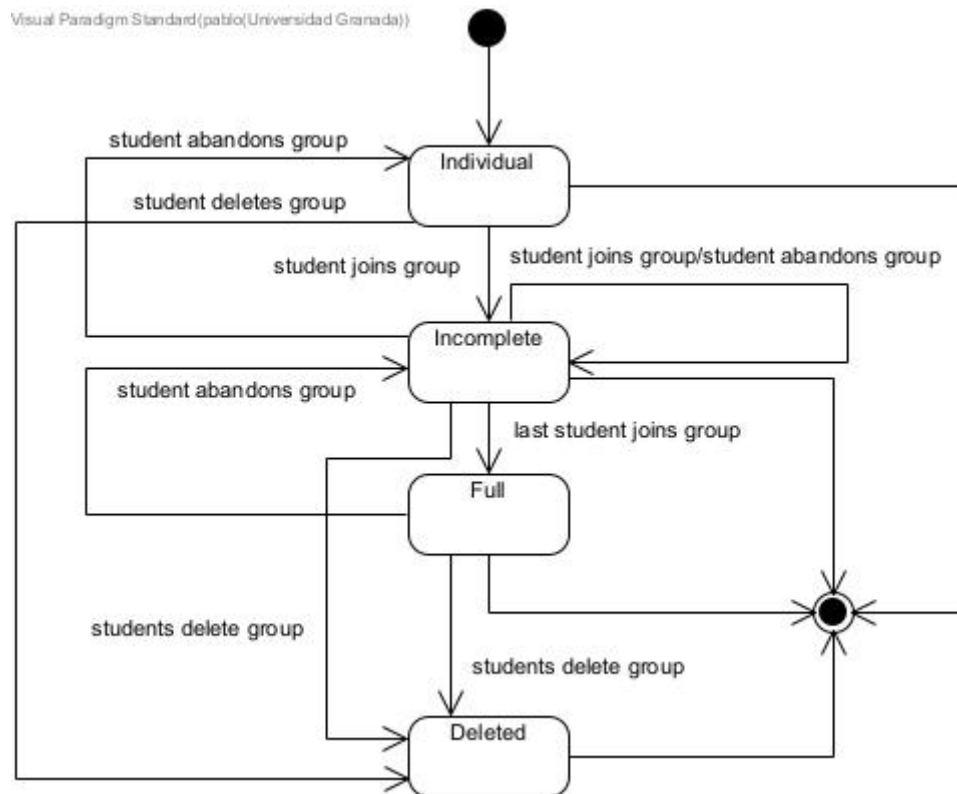
Works in a similar way to the requisite state diagram, but in this case only after all those requisites are in the completed state. To illustrate this, the student Irene earns without noticing the badge “The speed of light” for both having been “Coder of the month” and having achieved the number of submissions in the shortest time compared to previous coders of the month. So his badge goes from being missing (Missing) to being earned (Earned) as both requisites were obtained. But, what happens if one of the requisites is not achieved? Then the badge will never obtain the state Earned and will remain in Missing forever. That could be the case of Artemisia.



(Figure 2.6) Badge State Diagram

Group Process

Let's imagine that student Carla wants to create a group for joining the next exciting tournament, she will insert the name ("Hedy Lamarr & Co.") and the size she wishes the group to be. Once created she will be alone (Individual) but more people can join the group (up to the size field specified), then her friend Matteo joins and the size is 4 then the state will go to Incomplete. Besides, if her friend Alessandra joins the group will remain in the state Incomplete, only after the last student joins it will switch to the state Full. The same applies to the other way around, if someone from the full group abandons the state will switch to incomplete and only after there is only one person left will the group return to Individual. However, both incomplete and individual groups can still join a tournament/party. In addition, a group can be deleted if students wish to (Deleted), obviously a deleted group cannot join a tournament/battle because it does not exist anymore.



(Figure 2.7) Group State Diagram

2.2 Product Functions

2.2.1 Sign Up & Log In

These functions are available to all the types of users. The Sign Up function will allow users to create an account in case they do not have one. They will need to provide the mail of I.I.S.S Bertrand Russell (others will simply not work) a name, a

surname/s and a password as well as an id which will be the user's nickname (the id must not include vulgar or strong words). The platform will detect whether the mail provided contains the domain of an educator or a student to properly create the user and if the nickname is not offensive. On the other hand, the Log In function will allow users that already have an account to access it and use the platform accordingly. All they need to provide is either their id or mail and their password.

2.2.2 Create Tournament

An educator will be able to create a tournament by specifying both the starting and expiration date and adding a description. The system will check if the expiration date is later than the starting date and that both of them have not happened yet, if these conditions are not fulfilled then the system will send the respective message asking the user to try again and explaining what went wrong. Besides, the system will also assign the tournament an unique id if it is properly created.

2.2.3 Create Badge

An educator will also be able to create badges that the students can earn. He/She will need to specify the name of the badge (the more creative the name the better), a clear and concise description in natural language of how to earn that badge and assign the requisite/s needed to earn the badge. Once a badge is created it will automatically check which students have already fulfilled the requisites and automatically grant to them the respective badges.

2.2.4 Grant Badge

The system will automatically grant a badge to a student once he/she has accomplished all requisites similar to how it happens with PlayStation™ trophies.

2.2.5 Join Group

A student will be able to ask to join a group and if he/she is granted access by the group integrators, the student will join the group, otherwise nothing will happen. A student can send as many requests as he/she wishes but the first group to grant access will be the only one to be taken into account.

2.2.6 Join Tournament

A group can join a tournament but only if the tournament has not already started. Once joined they will be allowed to join battles that will take place in the tournament and submit submissions for each battle. The group will be also permitted to see the current state of the tournament.

2.2.7 Rate Group

An educator will be able to choose whether to rate a group's performance taking into account the submission handed in or let the system automatically rate that group taking into account a series of parameters and indications also based on the submission. A group can submit many submissions but only the latest one will be taken into account.

2.2.8 Update Profile

Any user can decide to update their data by adding new information (for example adding a telephone number if the user did not add it before) or modifying the one that is already available, that is for example changing the mail (if the new one is also valid, that is, from the same domain) or changing the nickname (if the new one is not already taken). If the new information is correct, the profile will be updated, otherwise the system will not update anything.

2.2.9 Log Out

Any user will be able to log out from their account so nobody else can access it in case for example they are working in a library or university computer or many people use his/her computer. Next time the user wants to use the platform, he/she will need to log in again.

2.2.10 Delete Account

Any user will be able to delete his/her account in case they have finished their studies at school or simply because they want to. However, to prevent problems

with group mates, deleting an account will only be possible once he/she is not enrolled in any tournament anymore. Deleting an account means that it will disappear from the system, that is it cannot be visited nor updated anymore and the badges the student won will be lost.

2.2.11 Delete Battle

Any educator that once created a battle will be able to delete it only if the battle has not yet finished. Deleting a battle means that it will no longer exist in the system, which means that groups will not be able to find it or to join it anymore.

2.2.12 Delete Tournament

The educator that once created a tournament can later delete it only if that tournament has not yet finished. Deleting a tournament means that it will no longer exist in the system, which means that groups will not be able to find it or to join it anymore.

2.2.13 Delete Group

Members of a group will be able to delete their group only if they previously reached a consensus. The educator that is in charge of rating it will also be able to delete the group, in this case without the need of a consensus. Deleting a group means that it will no longer exist in the system.

2.3 User Characteristics

- **Educators:** They are the primary administrators and content creators of the CKB platform. They have the ability to design challenging and educational code kata battles. Besides, educators are responsible for creating tournaments, battles and defining scoring criteria. In addition, they can evaluate and provide manual scores for student submissions. Finally they are capable of creating and defining gamification badges and associated rules.
- **Students:** The participants in code kata battles that are science students from the I.I.S.S Bertrand Russell, working individually or in teams. They have diverse levels of programming skill and experience. Students can join battles individually or form teams according to battle requirements. They will also be

able to fork GitHub repositories, set up automated workflows and submit code. Finally they will receive both automated and manual scores for their submissions.

- **Administrators:** They oversee the overall functionality and health of the CKB platform. They have technical expertise to manage and maintain the platform. They are also responsible for ensuring platform reliability, security and scalability. They can handle user accounts, permissions and general system administration. Finally, they may be involved in updating or/and upgrading the platform.
- **Platform Visitors:** Users who use the platform without logging in and because of that without participating in battles or tournaments. They have access to view ongoing battles and battle ranks. They can also explore the list of available code kata battles and their descriptions.

2.4 Assumptions, Dependencies, and Constraints

2.4.1 Regulatory policies

[RP1]: The data provided by the users will never ever be given to third parties with commercial purposes, respecting in this way the GDPR policy.

[RP2]: The platform adheres to web accessibility standards to be usable by a diverse user base, including those with disabilities.

2.4.2 Assumptions

[A1]: User Technical Proficiency: Users have intermediate to advanced programming skills, enabling them to understand and solve complex coding challenges.

[A2]: Stable Internet Access: Users have access to a stable and fast internet connection, essential for accessing real-time updates, resources, and submission features on CKB.

[A3]: Device Accessibility: Users own or have access to modern computing devices with adequate processing power and screen size for an optimal coding and learning experience.

[A4]: GitHub Integration Familiarity: Users are familiar with basic GitHub functionalities, such as cloning repositories, committing changes, and managing branches, facilitating smooth code submission processes.

[A5]: Proficiency in English: Users have a good command of English, enabling them to understand and interact with the platform's content, instructions, and community discussions effectively.

2.4.3 Dependencies

[DP1]: Web Framework Reliability: CKB's user experience depends on the reliability and efficiency of chosen web frameworks for UI/UX design.

[DP2]: Database Management Systems: The platform relies on robust database systems to manage large volumes of user data, educational content, and challenge submissions securely.

[DP3]: API Integration: Smooth functionality of CKB's features, such as code submissions and repository management, depends on reliable integration with third-party APIs like GitHub.

[DP4]: Browser Compatibility: Optimal performance and accessibility of CKB across various modern web browsers are crucial for reaching a wider user base.

2.4.4 Constraints

[C1]: Scalability Limitations: The architecture must support scalability to accommodate an increasing number of users without performance issues.

[C2]: Security and Data Privacy: Compliance with data protection laws and implementation of robust security measures to safeguard user data is critical.

[C3]: Content Currency: Regular updates to coding challenges and educational material are necessary to keep pace with evolving software development trends.

[C4]: Resource Allocation: Constraints related to server capacity and bandwidth might affect the platform's ability to handle large user volumes and data-intensive operations.

[C5]: Language Limitation: Initial language support is limited to English, which might restrict access for non-English speaking users.

2.4.5 Domain Assumptions

[D1]: Educators possess comprehensive knowledge in software development and are skilled in crafting coding challenges that mirror real-world problems, encouraging practical learning.

[D2]: Students have foundational knowledge in programming and show a proactive approach towards enhancing their skills through self-paced learning and participation in challenges.

[D3]: The platform's automated scoring system is precise and reliable, capable of evaluating code submissions against a variety of parameters like logic, efficiency, and coding standards.

[D4]: All users, including students and educators, are committed to maintaining a constructive and respectful environment, adhering to a community code of conduct that encourages positive interactions.

[D5]: The user interface of CKB is designed to be intuitive and user-friendly, accommodating users with different levels of technical expertise and providing an efficient learning and navigation experience.

[D6]: Integration with external platforms, particularly GitHub, functions seamlessly, allowing for efficient code management, submission, and retrieval without technical glitches.

[D7]: Timely delivery of notifications and alerts is ensured in less than 3 minutes, keeping users informed about challenge updates, submission deadlines, feedback, and other relevant information.

[D8]: Users must have a stable internet connection.

[D9]: An Educator User has to actually be an Educator.

[D10]: The GitHub repository shall be created within a minute.

[D11]: The User data has to be correct and up-to-date.

[D12]: Users need to know their credentials in order to log in.

[D13]: Students are capable of forking GitHub repositories and setting up automated workflows using GitHub actions.

[D14]: Students adhere to registration and submission deadlines set by educators.

[D15]: A Student user has to actually be a student.

2.4.3 Constraints

The system can ask the user for access to the camera in case he/she wants to take a picture to update/upload the profile picture.

The data provided by the user won't be given to third parties with commercial purposes respecting the GDPR.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

The system has two hardware interfaces for all the users: a web page and a webapp.

- The web page will have visualisation and browsing functions. It can be accessed from any device, such as a phone or a computer, and requires the user to have an internet connection. It will be used by the educators to check the information of their tournaments, but starting any modification procedure

will bring the user on the web app. For the challengers, it will allow them to browse the challenges, subscribe or unsubscribe some of them, and check any information about the challenges that the user is in.

- The web app on the other side will be the main functionality of the platform. For the educators, this will be the place where they can access the creation / modification interface of tournaments and challenges, with all the personalization options + upload of the required files. The grading of the submissions will also be done through the web app. On the other side, for the challengers, their submissions will be evaluated and graded through the web app, they will be able to run the included test through it, and all their progress will be done there.

Having two different hardware interfaces is useful because it allows the user interface to be more responsive, while at the same time not being too demanding for quick browsing. The web page allows users to quickly and resourcelessly check out the content, while the web app enables more performance and reactivity for actual implication in a task (challenge and grading). The web page will have endpoints leading to the web app in those places.

3.1.3 Software Interfaces

The software requires some software interfaces for its functions.

- A Calendar API in order to avoid registering multiple challenges at the same time. It would have an internal calendar associated with each user and would update it for every inscription / removal of challenges.
- A Git API in order to be able to create the repos, detect the commits of the changes and being able to retrieve the code and execute the tests
- A User authentication system, to register information for each user.
- A Code testing API, to be able to execute the challenge tests on the code from the Git.

3.1.4 Communication Interfaces

The software communicates through the internet connection of the user. It accesses the Git repositories thanks to the Git API. The software should also have a communication system between the web page and the web app in order to launch the web app correctly. Depending on the actions selected on the web page, the web app should be launched in the right state, with the user authenticated. It should seem to the user that it is only system, without needing to realize some procedures twice.

3.2 Functional Requirements

3.2.1 Requirements

[R1]: CKB allows an unregistered user that is a I.I.S.S. Bertrand Russel educator to create an educator user account.

[R2]: CKB allows an unregistered user that is a I.I.S.S. Bertrand Russel student to create a student user account.

[R3]: CKB allows all users that already have an account to log in.

[R4]: CKB allows all users to log out.

[R5]: CKB allows educators to create new badges.

[R6]: The platform allows educators to create new requisites.

[R7]: The platform allows users to search for other profiles as well as their own.

[R8]: The system allows users to visit other profiles and their own too.

[R9]: CKB will allow users to update their own profiles which means adding, deleting and/or changing information that was already available.

[R10]: CKB will allow users to delete their account under certain conditions specified in the Class Diagram if the user is a student and without restrictions if it is an educator.

[R11]: The platform automatically scores a group's submission in a battle if a manual scoring is not specified.

[R12]: The system lets educators manually score a group's performance in a submission if they wish to.

[R13]: CKB allows educators to create tournaments in which several code kata battles can take place.

[R14]: An educator can create one or more battles within a tournament.

[R15]: An educator can delete a tournament that has not finished yet.

[R16]: An educator can delete a battle that has not finished yet

[R17]: An educator can allow other educators to create battles in a specified tournament.

[R18]: CKB allows any student to create a group that can join tournaments and consequentially, battles in the tournament.

[R19]: CKB allows any student to join a vacant group.

[R20]: CKB allows a student that belongs to a group to abandon such a group.

[R21]: The platform lets an educator expel a student from a group.

[R22]: The platform lets a group vote to decide to expel a student if the reasons are justified, that is, the student is not doing his/her part of the submission.

[R23]: The system allows student groups to hand in several submissions for each battle, although only the last one to be submitted will be scored.

[R24]: The system allows student groups to delete submissions if they wish to, although it is strongly discouraged.

[R25]: CKB allows educators to update the information (description, winner_group...) of any tournament.

[R26]: CKB allows educators to update the information (description, github_link...) of any battle.

[R27]: The system will automatically grant a badge to a student that has fulfilled the needed requisites.

[R28]: The system will automatically notify students when a new tournament has started and also when it is about to start.

[R29]: The system will automatically notify students when a new battle has started and also when it is about to start.

[R30]: CKB will notify students when an educator/the system has rated his/her group submission to a specific battle.

[R31]: The system will store the users' personal information.

[R32]: Users can check FAQs to understand how the platform works in case they are not sure how they can do something.

[R33]: The platform will allow users to send direct messages amongst them.

[R34]: The platform will allow users to publish messages in a forum.

[R35]: The platform will allow users to send emails to other users.

[R36]: The system will notify educators when a group's submission is ready to rate.

[R37]: The system will notify any user if they have received an email.

[R38]: The system will notify any user if they have received a direct message.

[R39]: The system allows the educator that created a badge to erase it.

[R40]: The system allows the educator that created a requisite to erase it.

3.2.2 Mapping on goals

[G1]: Create a user-friendly as well as efficient web-based platform called CKB whose main goal is to improve students' coding skills

<p>[D2]: Students have foundational knowledge in programming and show a proactive approach towards enhancing their skills through self-paced learning and participation in challenges</p> <p>[D4]: All users, including students and educators, are committed to maintaining a constructive and respectful environment, adhering to a community code of conduct that encourages positive interactions</p> <p>[D5]: The user interface of CKB is designed to be intuitive and user-friendly, accommodating users with different levels of technical expertise and providing an efficient learning and navigation experience</p> <p>[D7]: Timely delivery of notifications and alerts is ensured in less than 3 minutes, keeping users informed about challenge updates, submission deadlines, feedback, and other relevant information</p> <p>[D10]: The GitHub repository shall be created within a minute</p> <p>[D11]: The User data has to be correct and up-to-date</p>	
---	--

<p>[G2]: Provide Educators with the necessary tools to create, manage, delete tournaments, battles, badges and manually score the results of the latter</p>	
<p>[D1]: Educators possess comprehensive knowledge in software development and are skilled in crafting coding challenges that mirror real-world problems, encouraging practical learning</p> <p>[D8]: Users must have a stable internet connection</p> <p>[D9]: An Educator User has to actually be an Educator</p> <p>[D10]: The GitHub repository shall be created within a minute</p>	<p>[R5]: CKB allows educators to create new badges</p> <p>[R6]: The platform allows educators to create new requisites</p> <p>[R12]: The system lets educators manually score a group's performance in a submission if they wish to</p> <p>[R13]: CKB allows educators to create tournaments in which several code kata battles can take place</p> <p>[R14]: An educator can create one or more battles within a tournament</p> <p>[R15]: An educator can delete a tournament that has not finished yet</p> <p>[R16]: An educator can delete a battle that has not finished yet</p> <p>[R17]: An educator can allow other educators to create battles in a specified tournament</p> <p>[R21]: The platform lets an educator expel a student from a group</p> <p>[R25]: CKB allows educators to update the information (description, winner_group...) of any tournament</p> <p>[R26]: CKB allows educators to update the information (description, github_link...) of any battle</p> <p>[R39]: The system allows the educator that created a badge to erase it</p> <p>[R40]: The system allows the educator that created a requisite to erase it</p>

<p>[G3]: Implement a gamification system based on winning badges by fulfilling certain requests which will encourage students to code. Besides, teachers will be able to create new badges with a set of rules of their choice</p>	
<p>[D1]: Educators possess comprehensive knowledge in software development and are skilled in crafting coding challenges that mirror real-world problems, encouraging practical learning</p> <p>[D2]: Students have foundational knowledge in programming and show a proactive approach towards enhancing their skills through self-paced learning and participation in challenges</p> <p>[D7]: Timely delivery of notifications and alerts is ensured in less than 3 minutes, keeping users informed about challenge updates, submission deadlines, feedback, and other relevant information.</p>	<p>[R5]: CKB allows educators to create new badges</p> <p>[R6]: The platform allows educators to create new requisites</p> <p>[R27]: The system will automatically grant a badge to a student that has fulfilled the needed requisites</p> <p>[R39]: The system allows the educator that created a badge to erase it</p> <p>[R40]: The system allows the educator that created a requisite to erase it</p>

<p>[D9]: An Educator User has to actually be an Educator</p> <p>[D11]: The User data has to be correct and up-to-date</p> <p>[D15]: A Student user has to actually be a student</p>	
--	--

[G4]: Allow students to create and join existent groups to participate in tournaments, battles, modify their profile, win badges, submit solutions via GitHub repositories, access code kata descriptions and contact their educators

<p>[D2]: Students have foundational knowledge in programming and show a proactive approach towards enhancing their skills through self-paced learning and participation in challenges</p> <p>[D4]: All users, including students and educators, are committed to maintaining a constructive and respectful environment, adhering to a community code of conduct that encourages positive interactions</p> <p>[D5]: The user interface of CKB is designed to be intuitive and user-friendly, accommodating users with different levels of technical expertise and providing an efficient learning and navigation experience</p> <p>[D6]: Integration with external platforms, particularly GitHub, functions seamlessly, allowing for efficient code management, submission, and retrieval without technical glitches</p> <p>[D7]: Timely delivery of notifications and alerts is ensured in less than 3 minutes, keeping users informed about challenge updates, submission deadlines, feedback, and other relevant information</p> <p>[D8]: Users must have a stable internet connection</p> <p>[D10]: The GitHub repository shall be created within a minute</p> <p>[D11]: The User data has to be correct and up-to-date</p> <p>[D13]: Students are capable of forking GitHub repositories and setting up automated workflows using GitHub actions</p> <p>[D14]: Students adhere to registration and submission deadlines set by educators</p> <p>[D15]: A Student user has to actually be a student</p>	<p>[R18]: CKB allows any student to create a group that can join tournaments and consequentially, battles in the tournament</p> <p>[R19]: CKB allows any student to join a vacant group</p> <p>[R20]: CKB allows a student that belongs to a group to abandon such a group</p> <p>[R23]: The system allows student groups to hand in several submissions for each battle, although only the last one to be submitted will be scored</p> <p>[R24]: The system allows student groups to delete submissions if they wish to, although it is strongly discouraged</p> <p>[R27]: The system will automatically grant a badge to a student that has fulfilled the needed requisites</p>
--	---

[G5]: Implement an intuitive interface, easy to use and straightforward

[D5]: The user interface of CKB is designed to be intuitive and user-friendly, accommodating users with different levels of technical expertise and providing an efficient learning and navigation experience

[G6]: Create a platform with a notification system that will notify students when a tournament is created, if they have joined a battle, the score they got in such a battle or if they earned a badge. Besides, it will also notify educators if a battle is ready to score (if manually scoring has been set up) or if they have received a message

<p>[D3]: The platform's automated scoring system is precise and reliable, capable of evaluating code submissions against a variety of parameters like logic, efficiency, and coding standards</p>	<p>[R28]: The system will automatically notify students when a new tournament has started and also when it is about to start</p> <p>[R29]: The system will automatically notify students when a new battle has started and also when it is about to start</p>
--	---

<p>[D7]: Timely delivery of notifications and alerts is ensured in less than 3 minutes, keeping users informed about challenge updates, submission deadlines, feedback, and other relevant information</p> <p>[D8]: Users must have a stable internet connection</p> <p>[D11]: The User data has to be correct and up-to-date</p>	<p>[R30]: CKB will notify students when an educator has rated his/her group submission to a specific battle</p> <p>[R36]: The system will notify educators when a group's submission is ready to rate</p> <p>[R37]: The system will notify any user if they have received an email</p> <p>[R38]: The system will notify any user if they have received a direct message</p>
--	---

<p>[G7]: Create user profiles attached to each user with relevant information (names, surnames, e-mails, etc) and different characteristics if they are students or educators profiles</p>	
<p>[D9]: An Educator User has to actually be an Educator</p> <p>[D11]: The User data has to be correct and up-to-date</p> <p>[D12]: Users need to know their credentials in order to log in</p> <p>[D15]: A Student user has to actually be a student</p>	<p>[R1]: CKB allows an unregistered user that is a I.I.S.S. Bertrand Russel educator to create an educator user account</p> <p>[R2]: CKB allows an unregistered user that is a I.I.S.S. Bertrand Russel student to create a student user account</p> <p>[R3]: CKB allows all users that already have an account to log in.</p> <p>[R5]: CKB allows educators to create new badges</p> <p>[R6]: The platform allows educators to create new requisites</p> <p>[R7]: The platform allows users to search for other profiles as well as their own</p> <p>[R8]: The system allows users to visit other profiles and their own too</p> <p>[R9]: CKB will allow users to update their own profiles which means adding, deleting and/or changing information that was already available</p> <p>[R10]: CKB will allow users to delete their account under certain conditions specified in the Class Diagram if the user is a student and without restrictions if it is an educator</p> <p>[R12]: The system lets educators manually score a group's performance in a submission if they wish to</p> <p>[R13]: CKB allows educators to create tournaments in which several code kata battles can take place</p> <p>[R14]: An educator can create one or more battles within a tournament</p> <p>[R15]: An educator can delete a tournament that has not finished yet</p> <p>[R16]: An educator can delete a battle that has not finished yet</p> <p>[R17]: An educator can allow other educators to create battles in a specified tournament</p> <p>[R18]: CKB allows any student to create a group that can join tournaments and consequentially, battles in the tournament</p> <p>[R19]: CKB allows any student to join a vacant group</p> <p>[R20]: CKB allows a student that belongs to a group to abandon such a group</p> <p>[R21]: The platform lets an educator expel a student from a group</p> <p>[R25]: CKB allows educators to update the information (description, winner_group...) of any tournament</p> <p>[R26]: CKB allows educators to update the information (description, github_link...) of any battle</p> <p>[R31]: The system will store the users' personal information</p> <p>[R39]: The system allows the educator that created a badge to erase it</p> <p>[R40]: The system allows the educator that created a requisite to erase it</p>

<p>[G8]: Develop automatic scoring that will automatically assign each group a score everytime a battle ends according to different parameters like code clarity and quality</p>

<p>[D3]: The platform's automated scoring system is precise and reliable, capable of evaluating code submissions against a variety of parameters like logic, efficiency, and coding standards</p> <p>[D6]: Integration with external platforms, particularly GitHub, functions seamlessly, allowing for efficient code management, submission, and retrieval without technical glitches</p> <p>[D11]: The User data has to be correct and up-to-date</p> <p>[D14]: Students adhere to registration and submission deadlines set by educators</p>	<p>[R11]: The platform automatically scores a group's submission in a battle if a manual scoring is not specified</p> <p>[R30]: CKB will notify students when an educator/ the system has rated his/her group submission to a specific battle</p> <p>[R31]: The system will store the users' personal information</p>
--	--

<p>[G9]: Develop manual scoring for teachers to evaluate the results of the groups and allow educators to attach comments in case they want to explain the results giving some feedback</p>	
<p>[D1]: Educators possess comprehensive knowledge in software development and are skilled in crafting coding challenges that mirror real-world problems, encouraging practical learning</p> <p>[D4]: All users, including students and educators, are committed to maintaining a constructive and respectful environment, adhering to a community code of conduct that encourages positive interactions</p> <p>[D6]: Integration with external platforms, particularly GitHub, functions seamlessly, allowing for efficient code management, submission, and retrieval without technical glitches</p> <p>[D8]: Users must have a stable internet connection</p> <p>[D9]: An Educator User has to actually be an Educator</p> <p>[D11]: The User data has to be correct and up-to-date</p> <p>[D14]: Students adhere to registration and submission deadlines set by educators</p>	<p>[R7]: The platform allows users to search for other profiles as well as their own</p> <p>[R8]: The system allows users to visit other profiles and their own too</p> <p>[R12]: The system lets educators manually score a group's performance in a submission if they wish to</p> <p>[R23]: The system allows student groups to hand in several submissions for each battle, although only the last one to be submitted will be scored</p> <p>[R24]: The system allows student groups to delete submissions if they wish to, although it is strongly discouraged</p> <p>[R30]: CKB will notify students when an educator/the system has rated his/her group submission to a specific battle</p>

<p>[G10]: Enable measures to prevent hacking, protect data by for example adding authentication methods, ensure fault tolerance and data-recovery protocols, creating in this way a robust as well as secure platform</p>	
<p>[D4]: All users, including students and educators, are committed to maintaining a constructive and respectful environment, adhering to a community code of conduct that encourages positive interactions</p> <p>[D6]: Integration with external platforms, particularly GitHub, functions seamlessly, allowing for efficient code management, submission, and retrieval without technical glitches</p> <p>[D12]: Users need to know their credentials in order to log in</p>	<p>[R3]: CKB allows all users that already have an account to log in.</p> <p>[R4]: CKB allows all users to log out</p> <p>[R10]: CKB will allow users to delete their account under certain conditions specified in the Class Diagram if the user is a student and without restrictions if it is an educator</p> <p>[R31]: The system will store the users' personal information</p>

<p>[G11]: Allow communication between students and educators through several ways like e-mail, forums or direct messages</p>	
<p>[D4]: All users, including students and educators, are committed to maintaining a constructive and respectful environment, adhering to a community code of conduct that encourages positive interactions</p> <p>[D6]: Integration with external platforms, particularly GitHub, functions seamlessly, allowing for efficient code management, submission, and retrieval without technical glitches</p>	<p>[R7]: The platform allows users to search for other profiles as well as their own</p> <p>[R8]: The system allows users to visit other profiles and their own too</p> <p>[R30]: CKB will notify students when an educator/the system has rated his/her group submission to a specific battle</p>

[D7]: Timely delivery of notifications and alerts is ensured in less than 3 minutes, keeping users informed about challenge updates, submission deadlines, feedback, and other relevant information

[D8]: Users must have a stable internet connection

[D9]: An Educator User has to actually be an Educator

[D15]: A Student user has to actually be a student

[R33]: The platform will allow users to send direct messages amongst them

[R34]: The platform will allow users to publish messages in a forum

[R35]: The platform will allow users to send emails to other users

[G12]: Include a section explaining how the several features of the platform work (gamification, sending messages, joining a group, creating a tournament, etc) with an index so users can directly check whatever they need

[D5]: The user interface of CKB is designed to be intuitive and user-friendly, accommodating users with different levels of technical expertise and providing an efficient learning and navigation experience

[R32]: Users can check FAQs to understand how the platform works in case they are not sure how they can do something

3.2.3 Use Case Diagrams

Profile Management

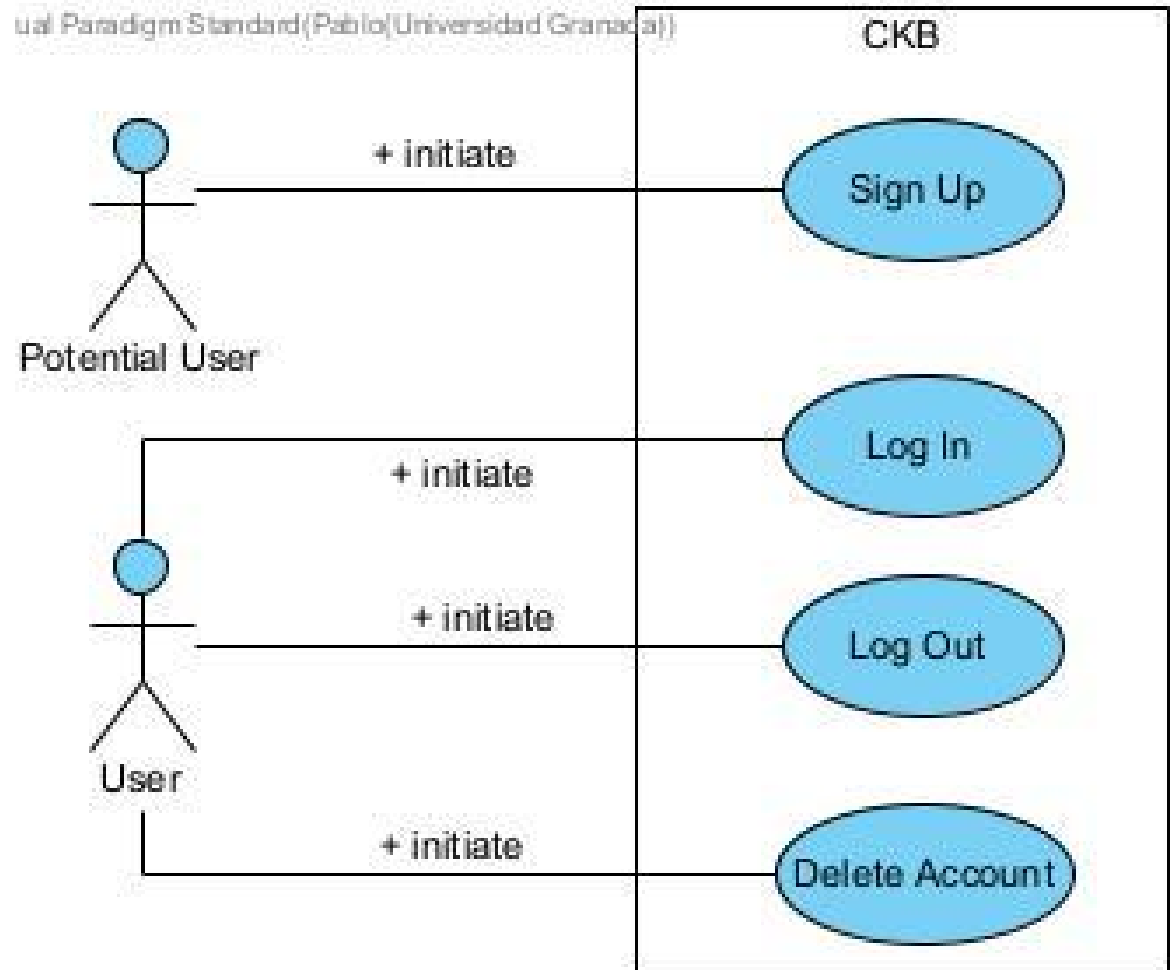


Figure 3.1

Educator Features I

Visual Paradigm Standard (Pablo(Universidad Granada))

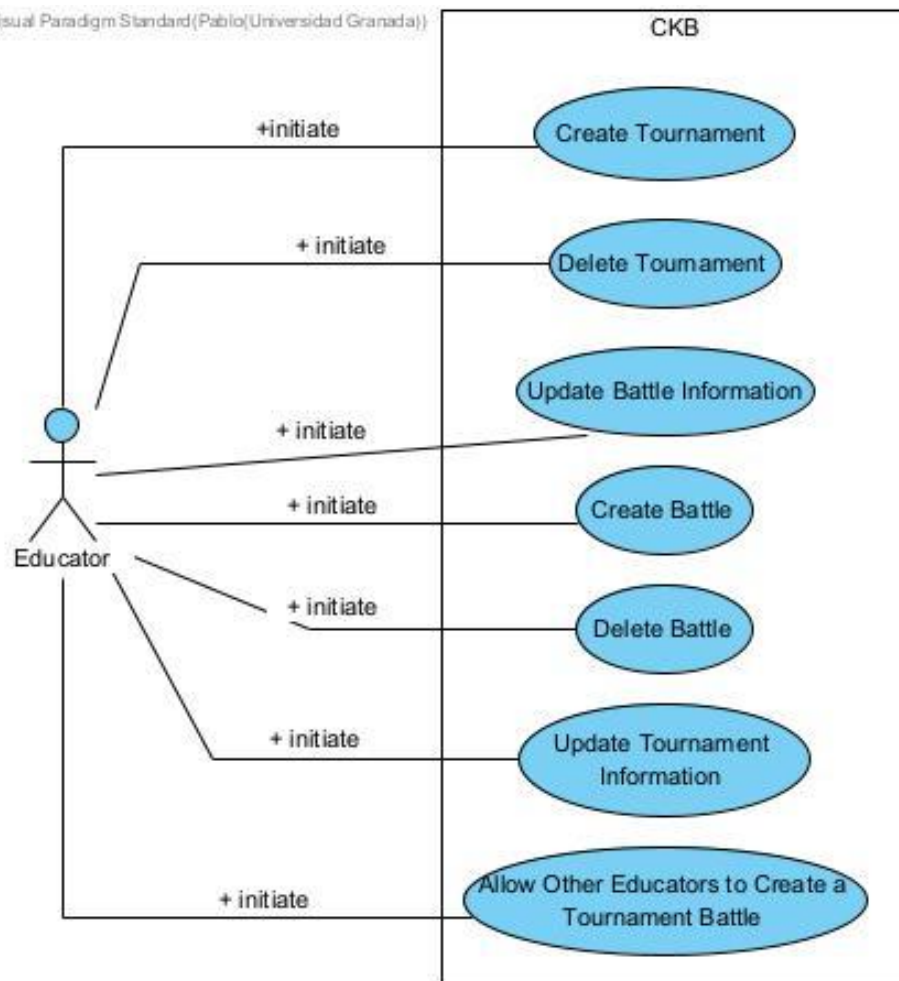


Figure 3.2

Educator Features II

Visual Paradigm Standard (Pablo(Universidad Granada))

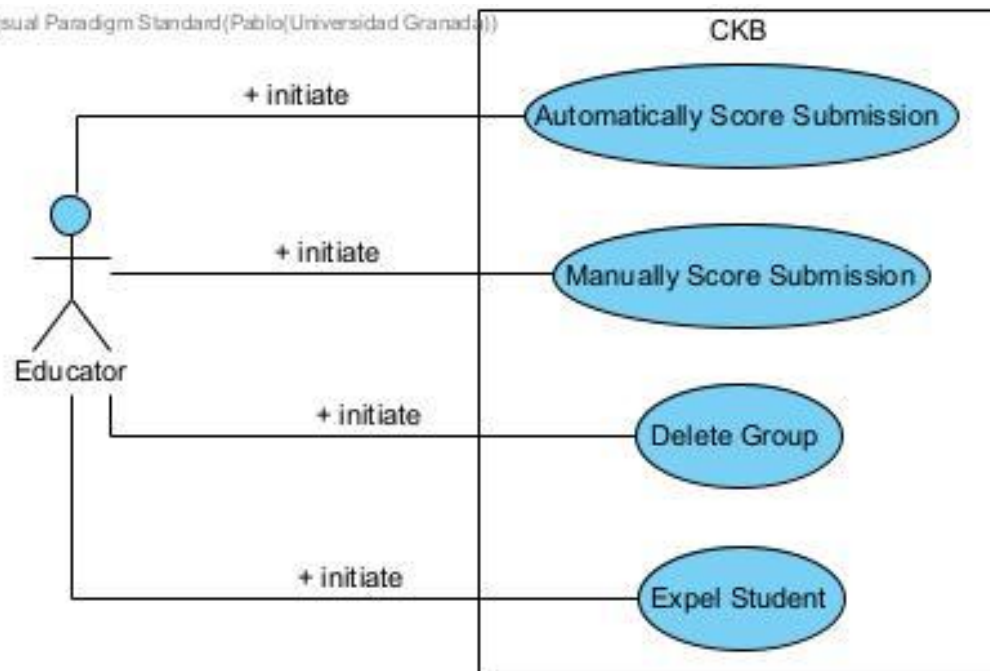


Figure 3.3

Student Features

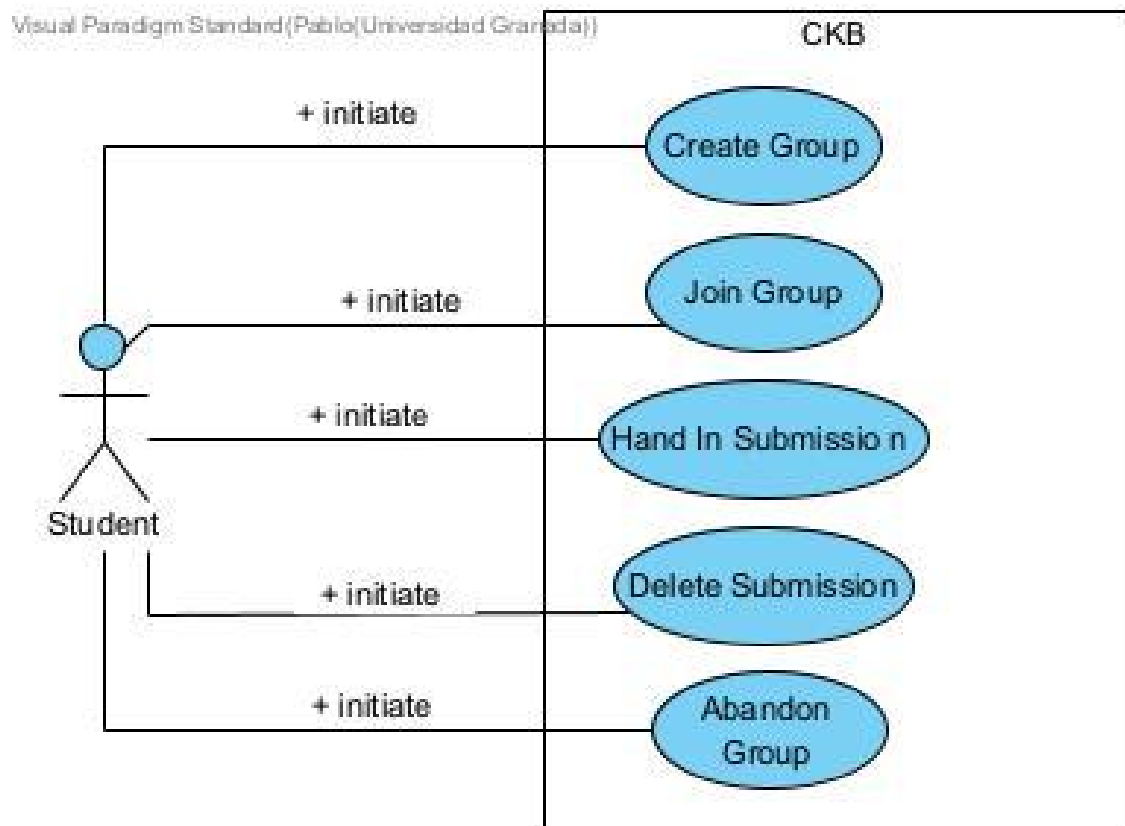


Figure 3.4

Common Features

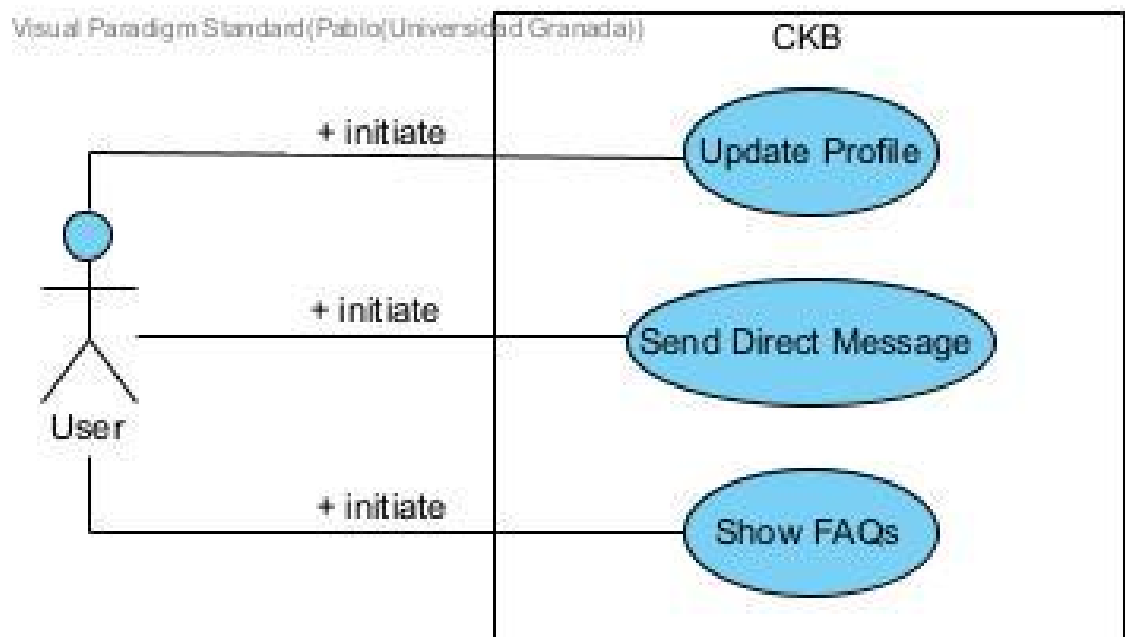


Figure 3.5

Gamification Features

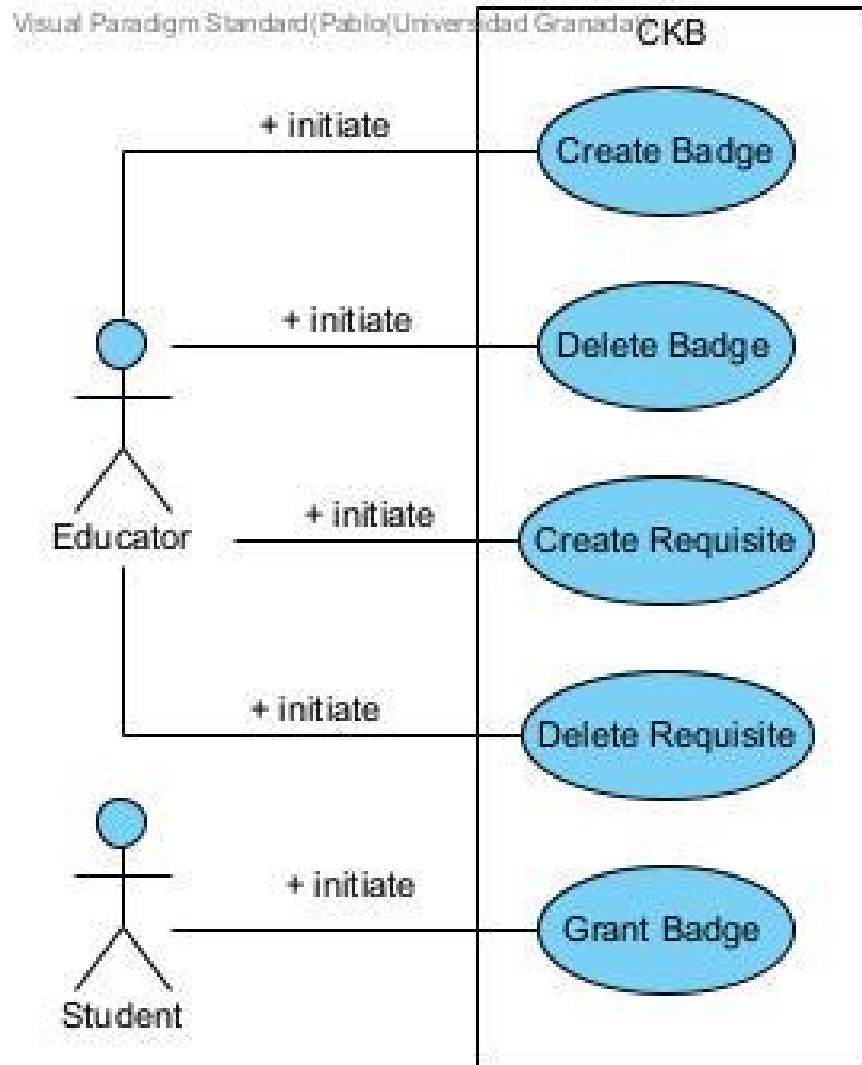


Figure 3.6

3.2.4 Use Cases

Name	Sign Up	[UC1]
Actors	Potential User	
Entry Condition	The potential User opens the system and there is not an user session open	
Event Flow	1 - Potential User enters his/her personal data (email, name, surname, nickname, a password, the password confirmation)	

	2 - Potential User clicks on the “Create Account” button 3 - CKB checks email domain and assigns a type of user (student/educator) 4 - CKB checks passwords coincide 5 - CKB checks nickname does not contain an offensive message 6 - CKB creates a user profile with the information provided
Exit Condition	A new account is created in the system and the Potential User becomes either a Student User o Educator User
Exception	<p>The email domain does not belong to a I.I.S.S. Bertrand Russel member and/or the nickname contains an offensive message: In this case the system sends the following error message: “Nickname and/or email not valid. Please, try again”</p> <p>The password and password confirmation do not coincide: In this case the system sends the following error message: “The passwords do not coincide. Please, try again”</p>

Name	Log In	[UC2]
Actors	User (Student/Educator)	
Entry Condition	The system has not kept his/her last session open and the user already has an account	
Event Flow	1 - User enters his/her email and password 2 - The system allows user to access his/her profile	
Exit Condition	The user has logged in and can now do several things in the platform	
Exception	The password/email is not correct: In this case the platform will display the following system on the screen “Password/email invalid. Please, try again” and it will wait for the user to enter the credentials again. The user can do that or click on “I forgot my password” in that case the system will ask the user to enter his/her email and will send a change of password email afterwards	

Name	Log Out	[UC3]
Actors	User (Student/Educator)	

Entry Condition	The User is logged in
Event Flow	1 - User goes to his/her profile settings and clicks on “Log Out” 2 - The system will display the following message: “Are you sure you want to log out?” and two options namely “Confirm Log Out” and “Cancel” 3 - The user clicks on “Yes” 4 - The system will close the current session
Exit Condition	It is now impossible to access the user profile or make use of the platform until the user Logs In again
Alternative	The user clicks on the “Cancel” button: In that case the system will not do anything and the platform will be like if nothing had happened

Name	Delete Account	[UC4]
Actors	User (Student/Educator)	
Entry Condition	The user is logged in	
Event Flow	1 - User goes to his/her profile settings and click on “Delete Account” 2 - The system will display the following message: “Are you sure you want to delete your account? All your progress will be lost for good” and two options namely “Yes, delete everything” and “Cancel” 3 - The user clicks on “Yes, delete everything” 4 - The system checks that the users fulfils the requirements to delete the account 5 - The system deletes the account	
Exit Condition	The account will disappear from the platform forever, that is the badges earned, the personal data, the submissions, the account will not be able to be reached anymore, etc...	
Exception	The system checks that the user is a student that is participating in a battle/tournament: In that case the system will display the following message “You cannot delete your account while participating in a battle/tournament. Please, try again later”. After that the system will be as it was before the Use Case even started.	

Name	Create Tournament	[UC5]
Actors	Educator	

Entry Condition	The educator user is logged in
Event Flow	<p>1 - The educator clicks on “Create Tournament” in the management section of the platform</p> <p>2 - The system will ask the educator to insert the starting and expiration dates and write a description of up to 6000 characters explaining the tournament purpose and other details.</p> <p>3 - The system will create the tournament and will assign an id to it</p> <p>4 - The system will notify students that a tournament is about to take place in the following days</p>
Exit Condition	A new tournament is created in the system
Exception	The educator enters the wrong information, that is, the dates already happened, starting date is later than expiration date and/or the description exceeds 6000 characters. In that case the system will display the following message on screen: “The dates are wrong and/or description is too long (Up to 6000 characters). Please, try again” and it will let the educator fix the information and try again.

Name	Create Battle	[UC6]
Actors	Educator	
Entry Condition	At least one tournament is already taking place or is about to start	
Event Flow	<p>1 - The educator goes to the tournament in which he/she want to create the battle and clicks on “Schedule a new battle”</p> <p>2 - The system will kindly ask the educator to enter the battle information</p> <p>3 - The educator enters the starting and expiration date of the battle, a description of up to 4500 characters and a GitHub link to the repository that contains the code and all necessary files for student groups to participate in the battle</p> <p>4 - The platform will check whether the dates, description and GitHub link are correct</p> <p>5 - A new battle will be created and the system will assign an id to it</p> <p>6 - The system will notify students that a new battle is about to take place</p>	
Exit Condition	A new tournament battle exists now in the tournament and users will be able to interact with it	
Exception	The information entered by the educator is wrong, that is the starting date is later that expiration date or one of them has already happened, the description contains more than 4500 characters and/or the GitHub link is not a valid one. In such a case, the system will display the following message on screen “The dates, description and/or GitHub link are	

	incorrect. Please, try again” and it will let the educator fix the information and try again
--	--

Name	Delete Tournament	[UC7]
Actors	Educator	
Entry Condition	The tournament that wants to be deleted, must both exist and not have already finished	
Event Flow	1 - The educator goes the tournament page and clicks on “Delete Tournament” 2 - The system asks the educator if he/she is sure to delete the tournament and displays two buttons: “Indeed” & “I changed my mind” 3 - The educator clicks on “Indeed” 4 - The system deletes the tournament	
Exit Condition	The tournament is deleted from the system for good	
Alternative	The educator clicks on “I changed my mind”, in that case the system will not delete anything	

Name	Delete Battle	[UC8]
Actors	Educator	
Entry Condition	The battle that wants to be deleted, must both exist and not have already finished	
Event Flow	1 - The educator goes the battle page and clicks on “Delete Battle” 2 - The system asks the educator if he/she is sure to delete the battle and displays two buttons: “Indeed” & “I changed my mind” 3 - The educator clicks on “Indeed” 4 - The system deletes the battle	
Exit Condition	The battle is deleted from the tournament for good	
Alternative	The educator clicks on “I changed my mind”, in that case the system will not delete anything	

Name	Create a Badge	[UC9]
Actors	Educator	

Entry Condition	The user needs to be logged in
Event Flow	1 - In the management page the educator clicks on “Create Badge” 2 - The system will ask the educator to add a name without an offensive message and up to 50 characters and a description of up to 1000 characters 3 - The user will correctly enter the information 4 - The system will ask the educator to select the requisites the will conform the badge 5 - The educator will select the requisites needed to earn the badge among the ones that already exist 6 - The system will create the badge and will assign an id to it
Exit Condition	There is a new badge created in the system
Exception	The educator introduced incorrect data. The system will display the following message on screen: “The name and the description must be up to 50 and 1000 characters respectively. Please, try again” and then it will let the educator to reintroduce the information

Name	Create a Requisite	[UC10]
Actors	Educator	
Entry Condition	The user needs to be logged in	
Event Flow	1 - In the management page the educator clicks on “Create Requisite” 2 - The system will ask the educator to add a description of up to 300 characters 3 - The user will correctly enter the information 4 - The system will create the requisite and will assign an id to it	
Exit Condition	There is a new requisite created in the system	
Exception	The educator introduced incorrect data. The system will display the following message on screen: “The description must be up to 300 characters. Please, try again” and then it will let the educator to shorten the description	

Name	Allow other educator to create a tournament battle	[UC11]
Actors	Educator 1, Educator 2	
Entry Condition	Educator 1 must be logged in	

Event Flow	1 - Educator 1 goes to the tournament page and clicks on “Allow Battle Creation” 2 - The system will show a list of educators 3 - Educator 1 looks for the id of Educator 2 and selects it 4 - The system will grant the permission to Educator 2 to create battles
Exit Condition	Educator 2 can now create battles in that tournament as well

Name	Delete Badge	[UC12]
Actors	Educator	
Entry Condition	The badge that wants to be deleted must exist	
Event Flow	1 - The educator goes the management page and clicks on “Delete Badge” 2 - The system will show all the badges that Educator has created 3 - Educator will select the badge/s that he/she wants to delete 4 - The system asks now the educator if he/she is sure about deleting those badges and displays two buttons: “Indeed” & “I changed my mind” 5 - The educator clicks on “Indeed” 6 - The system deletes the badge/s	
Exit Condition	The badge/s has/have disappeared from the system and can no longer be accessed	
Alternative	Educator clicks on “I changed my mind”, in that case the system will not delete anything	

Name	Delete Requisite	[UC13]
Actors	Educator	
Entry Condition	The requisite that want to be deleted must exist	
Event Flow	1 - The educator goes the management page and clicks on “Delete Requisite” 2 - The system will show all the requisites that Educator has created 3 - Educator will select the requisite/s that he/she wants to delete 4 - The system asks now the educator if he/she is sure about deleting those requisites and displays two buttons: “Indeed” & “I changed my mind” 5 - The educator clicks on “Indeed” 6 - The system deletes the requisite/s and will remove the requisite from all the badges that contain such a requisite	

Exit Condition	The requisite/s has/have disappeared from the system and can no longer be accessed
Alternative	Educator clicks on “I changed my mind”, in that case the system will not delete anything and will not update any badge

Name	Create Group	[UC14]
Actors	Student	
Entry Condition	Student need to be logged in	
Event Flow	1 - Student goes to the homework page and clicks on “Create New Group” 2 - The system will ask the student to introduce the name of the group (up to 15 characters) and the size of it (up to 5) 3 - The system will check that the information created is correct and will create a group that can join tournaments, battles in those tournaments, etc	
Exit Condition	There is a new group in the system that students can request to join	
Exception	The student introduced incorrect data. The system will display the following message on screen: “The group name must be up to 15 characters and not include offensive messages. Besides, the group size must not exceed 5 members. Please, try again” and then it will let the student reintroduce the information	

Name	Join Group	[UC15]
Actors	Student, Student Group	
Entry Condition	Student must be logged in and there must be at least one group available	
Event Flow	1 - Student goes to the homework page and clicks on “Join Group” 2 - The system will show to them the list of groups that are available to join 3 - The student can select as many as he/she wishes 4 - The system will send every join request to each group 5 - A group accepts the request 6 - The system will add the Student to the group members	
Exit Condition	The group has increase its number by one and the student is now a member of the group	
Alternative	Not a single group accepts the request, in that case the system will not let the student join a group and will need to send other requests	

Name	Grant Badge	[UC16]
Actors	Student	
Entry Condition	Student must be logged in	
Event Flow	1 - Student fulfils one of the requisites for a badge 2 - The system will automatically check if there are not more requisites left and will grant the badge to the student 3 - The system will display the following message on the top left corner of the screen: "Congratulations, you won the badge <name of the badge>" 4 - The badge is now available to be seen in the student's profile	
Exit Condition	The student has another badge in his/her collection	
Alternative	There are still requisites to complete in that badge. In that case the system will not do anything	

Name	Hand In Submission	[UC17]
Actors	Student, Educator	
Entry Condition	The student group must be enrolled in the battle and one of the members must be logged in	
Event Flow	1 - A member of the group goes to the battle page and clicks on "Hand In Submission" 2 - The system will ask the student to enter the GitHub link and a name up to 20 characters 3 - The system will check the information entered 4 - The system will assign to the submission the tournament id, the battle id, an id and a date 5 - The system will send the submission to the educator that created the battle	
Exit Condition	The submission is now created and the educator has received it	
Exception	The student introduced incorrect data. The system will display the following message on screen: "The submission name must be up to 20 characters and not include offensive messages and the GitHub link could be incorrect. Please, try again" and then it will let the student reintroduce the information	

Name	Delete Submission	[UC18]
Actors	Student	

Entry Condition	A member of the group must be logged in and the group must at least have already handed in a submission
Event Flow	1 - The student goes to the battle page and clicks on “Delete Submission” 2 - The system will show him/her all the submissions that can be deleted 3 - The student selects the submission/s to be deleted 4 - The system deletes them
Exit Condition	The submissions disappear from the system and can no longer be accessed

Name	Manually Score Submission	[UC19]
Actors	Educator	
Entry Condition	A student group has to at least handed in one submission by the time the deadline arrives	
Event Flow	1 - The educator goes to the battle page and turns on the option “Manual Scoring” 2 - The educator selects the group that wants to score at that moment 3 - The system will show to him/her the submission and will ask him/her to set up a score from 0 to 30 4 - Educator thoroughly analyses the submission and rates it from 0 to 30 5 - System assigns score to the group	
Exit Condition	The Group has now an score assigned based on the submission performance	
Alternative	An educator that was gonna manually score a group cannot do it anymore, maybe because he/she does not have time so he/she can turn the “Manual Scoring” option off and then the system will automatically call the UC20	

Name	Automatically Score Submission	[UC20]
Actors	Educator	
Entry Condition	The option “Manual Scoring” must be turned off	
Event Flow	1 - A Student Group hand in a submission 2 - The system will automatically score it by the time the deadline arrives and will send the results to the educator in charge	
Exit Condition	The group has the automatic scored assigned and the Educator can also see it	

Name	Show FAQs	[UC21]
Actors	User (Student/Educator)	
Entry Condition	The user must be logged in	
Event Flow	1 - In the home page, the user clicks on the “FAQs” button 2 - The system will show to him/her a list with FAQs so users can learn about how the platform works	
Exit Condition	The system shows the FAQs page to the user	

Name	Send Direct Message	[UC22]
Actors	User 1 (Student/Educator), User 2 (Student/Educator)	
Entry Condition	User 1 must logged in	
Event Flow	1 - User goes to the communication page and clicks on “Send DM” 2 - The system will show him/her a list of available users (both students and educators) to send a direct message 3 - User selects the user that he/she wants to contact, writes the message and confirms it 4 - The system will send the direct message to the other user	
Exit Condition	Now User 2 has receive a direct message whose author is User 1	
Exception	The internet connection is unstable. The system will ask the user to try to send the message later when the situation improves	

Name	Update Profile	[UC23]
Actors	User (Student/Educator)	
Entry Condition	The user must be logged in	
Event Flow	1 - User goes to his/her profile page and clicks on “Update Profile” 2 - The system will allow the user to modify the information 3 - The user will modify the profile 4 - The system will update the user’s profile	
Exit Condition	The profile is successfully updated	
Exception	The new information written is not correct. More characters than possible, wrong mail, inexistent department (in the case of an educator), then the	

	system will display the following message: “The new data cannot be added” and will let the user to fix it
--	---

Name	Expel Student from a Group	[UC24]
Actors	Educator, Student, Student Group	
Entry Condition	Educator must be logged in and the student that is going to be expelled must belong to the group	
Event Flow	1 - Educator goes to the Group Page and clicks on “Select Group” 2 - The system shows a list of the groups that he/she is in charge of 3 - Educator chooses a group 4 - The system will show the group information 5 - Educator clicks on the “Expel Student” button 6 - The system will show a list of the group members 7 - Educator will click on the student to be expelled from the group 8 - The system will expel that student	
Exit Condition	The group number is decreased by one and that student is no longer a member of the group	

Name	Update Battle Information	[UC25]
Actors	Educator	
Entry Condition	Educator must be logged in and the battle to be updated has to actually exist	
Event Flow	1 - Educator goes to the battle page and clicks on “Update Information” 2 - The system asks the user to update the information, that is the GitHub Link, the description and/or the winner group 3 - Educator enters the new information 4 - The system checks it and updates the battle	
Exit Condition	The battle is updated with new information	
Exception	The educator introduced incorrect data. The system will display the following message on screen: “The description must be up to 4500 characters, the GitHub link must be a valid one and the winner group must exist. Please, try again” and then it will let the educator rewrite the information	

Name	Update Tournament Information	[UC26]
Actors	Educator	
Entry Condition	Educator must be logged in and the tournament to be updated has to actually exist	
Event Flow	1 - Educator goes to the tournament page and clicks on "Update Information" 2 - The system asks the user to update the information, that is the description and/or the winner group 3 - Educator enters the new information 4 - The system checks it and updates the tournament	
Exit Condition	The tournament is updated with new information	
Exception	The educator introduced incorrect data. The system will display the following message on screen: "The description must be up to 6000 characters and the winner group must exist. Please, try again" and then it will let the educator rewrite the information	

Name	Delete Group	[UC27]
Actors	Educator	
Entry Condition	The student group that wants to be deleted must exist	
Event Flow	1 - The educator goes to the groups page and clicks on "Delete Group" 2 - The system will show him/her the list of groups he/she is in charge of 3 - The educator will select the group/s that are wished to be deleted 4 - The system will delete those groups	
Exit Condition	The groups deleted do not exist in the system anymore	

Name	Abandon Group	[UC28]
Actors	Student	
Entry Condition	The student that wants to leave the group must belong to it	
Event Flow	1 - The student goes to the Groups page and clicks on "Abandon Group" 2 - The system will show to him/her the group/s that he/she has joined and that can be selected 3 - The student selects the group/s that he/she wants to leave 4 - The system will drop him/her out of the group/s	

Exit Condition

The groups selected have now a smaller number of integrants (decreased by one) and those whose number is 0 are deleted

3.2.5 Sequence Diagrams

Sign Up

Qual Paradigm Standard (Pablo/Universidad Granada)

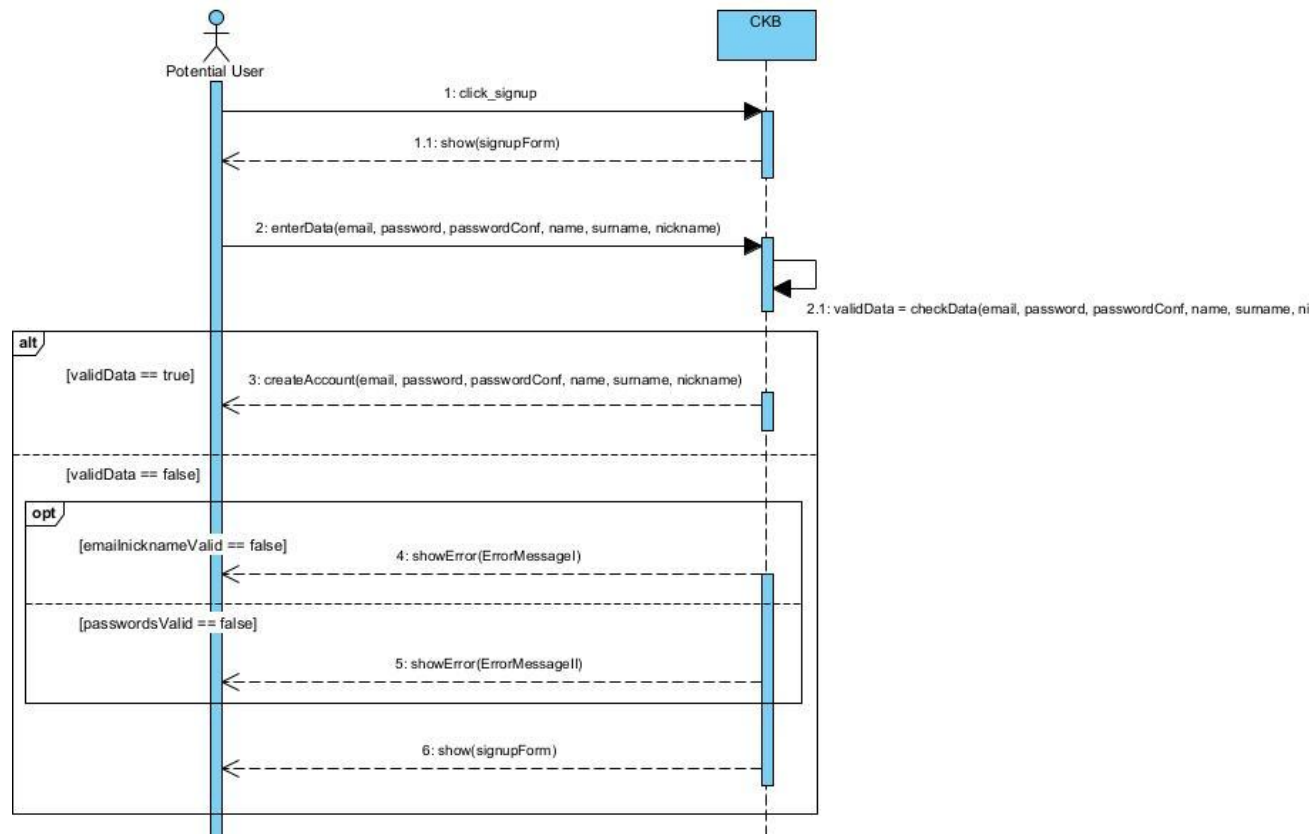


Figure 3.7

Log In

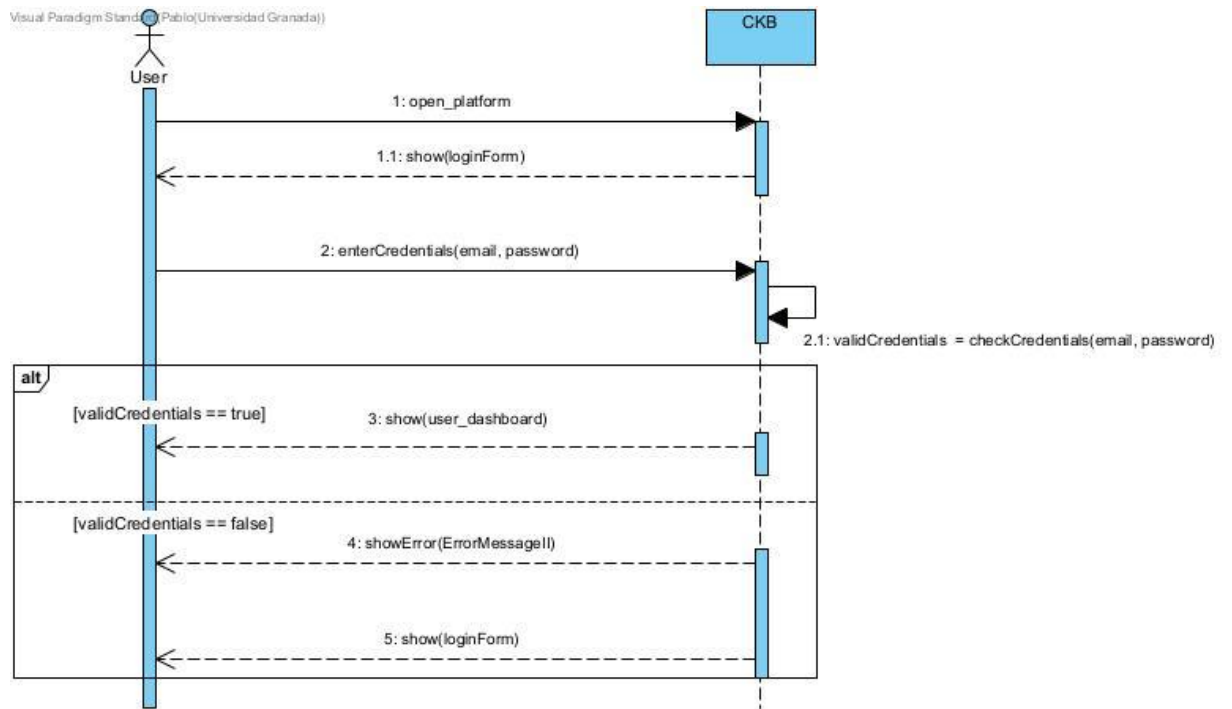


Figure 3.8

Log Out

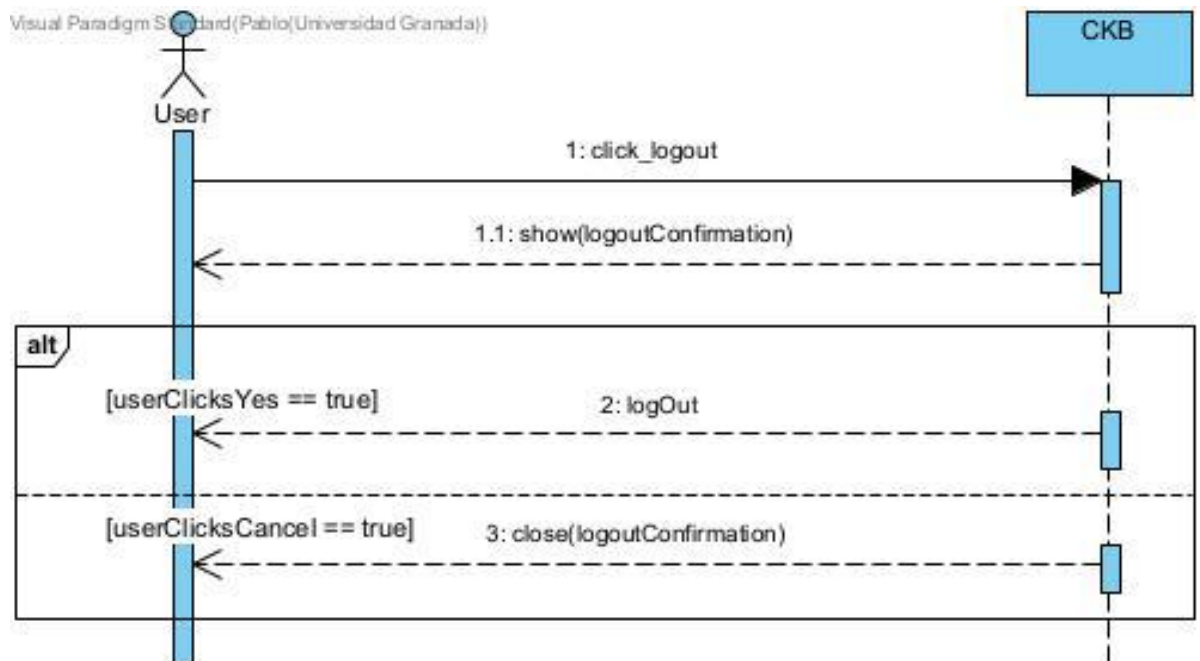


Figure 3.9

Delete Account

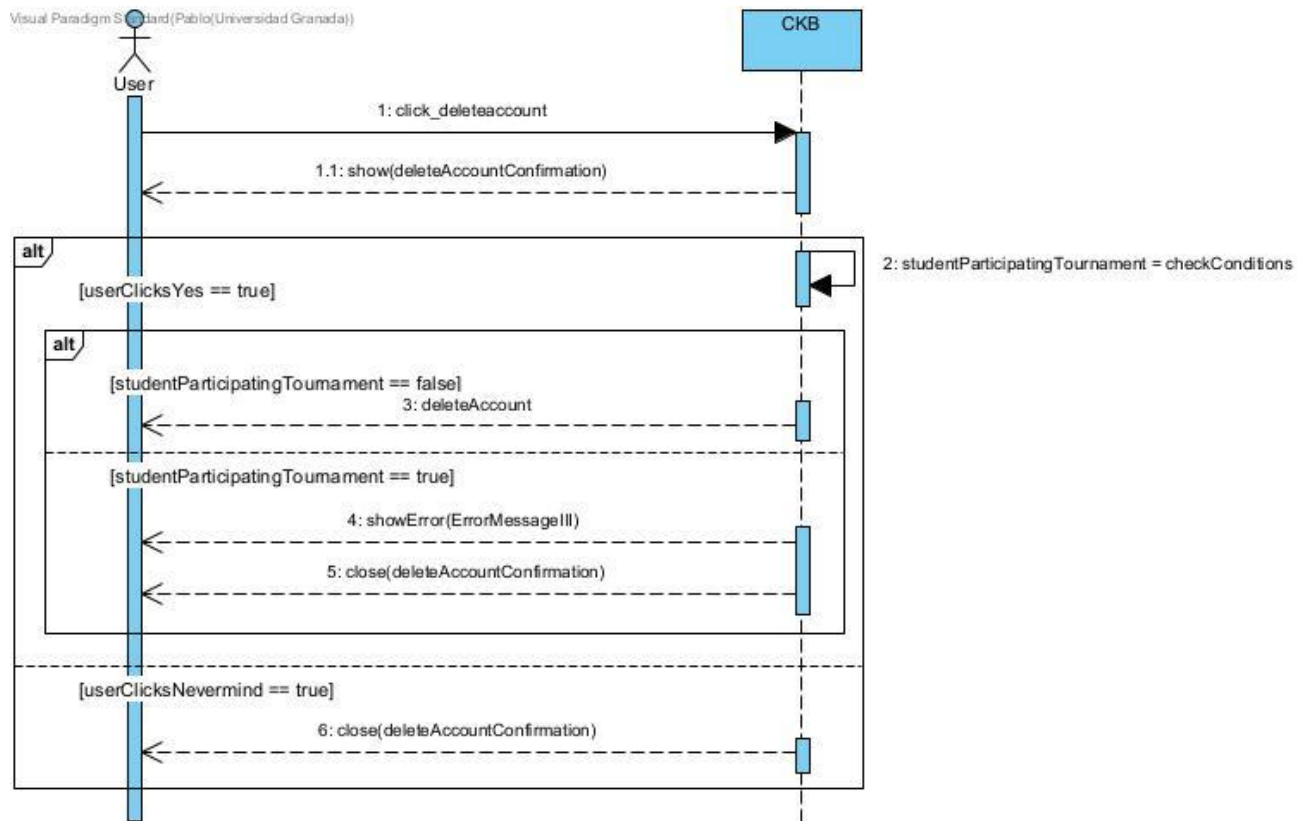


Figure 3.10

Create Tournament

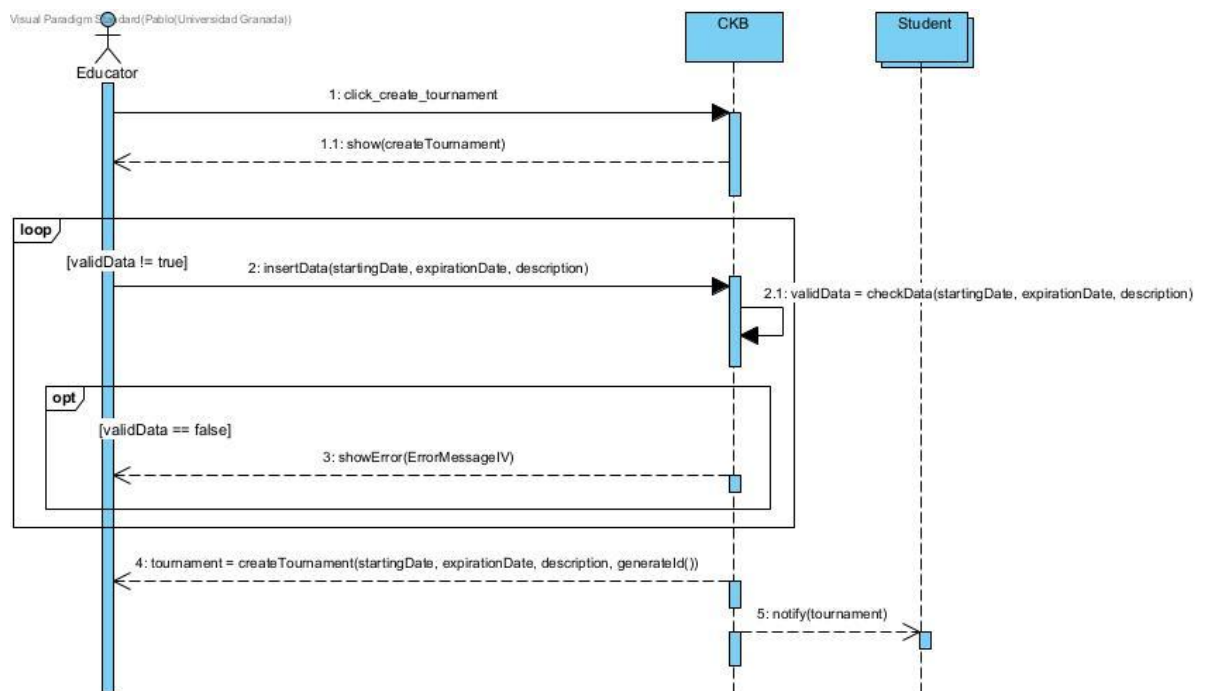


Figure 3.11

Create Battle

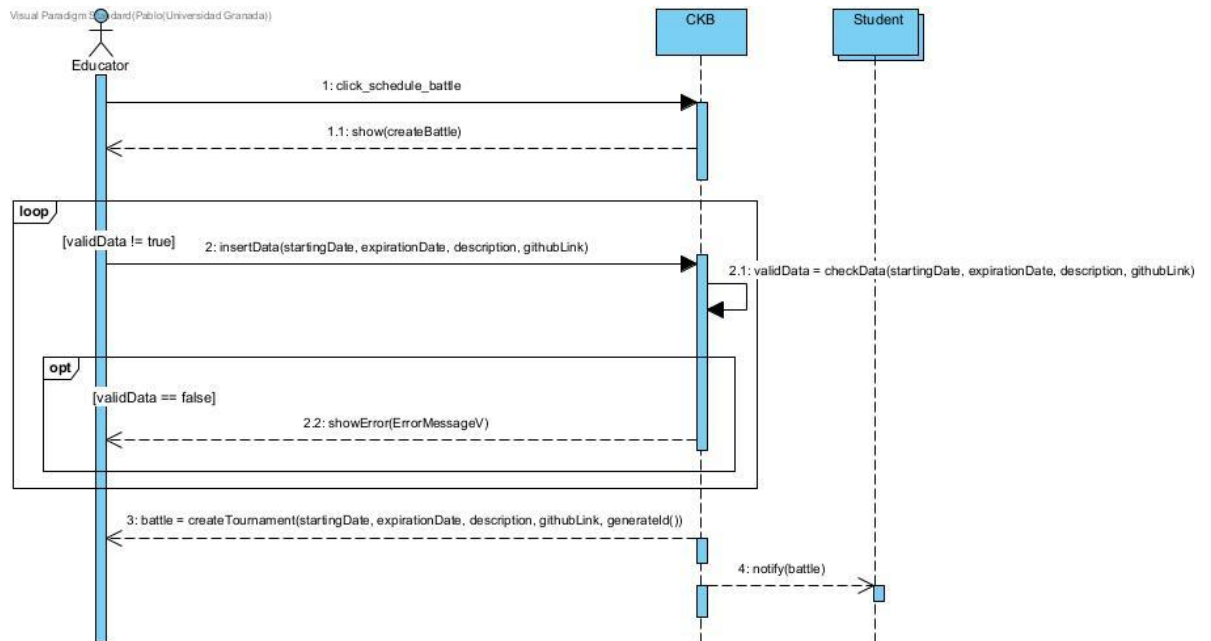


Figure 3.12

Delete Tournament

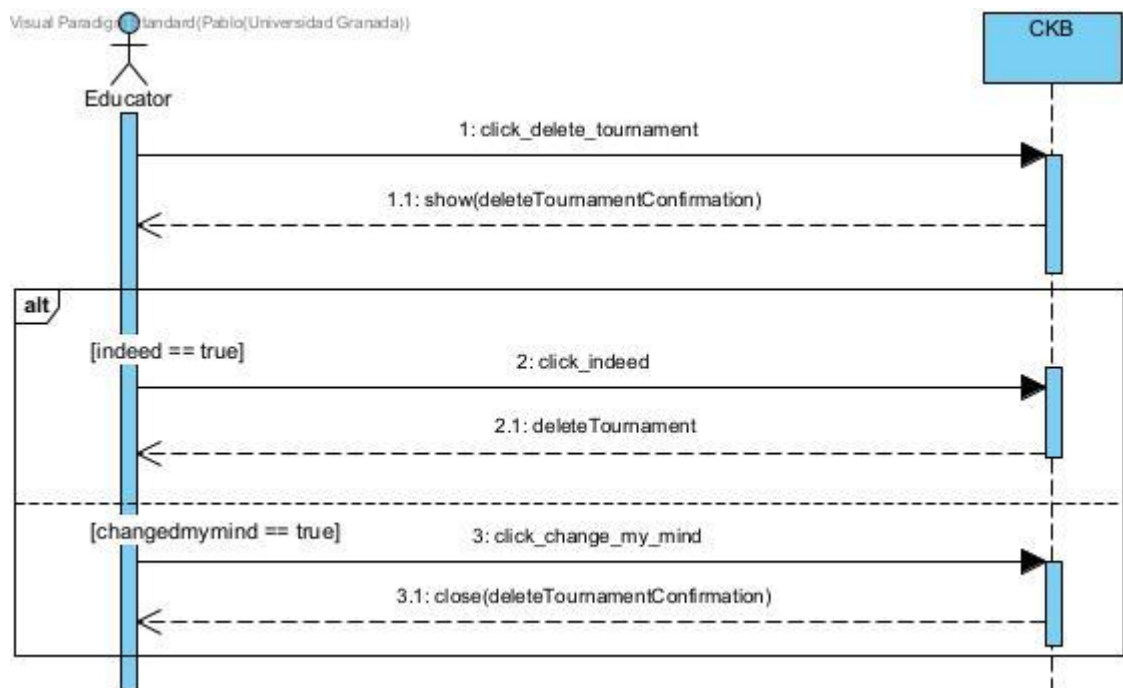


Figure 3.13

Delete Battle

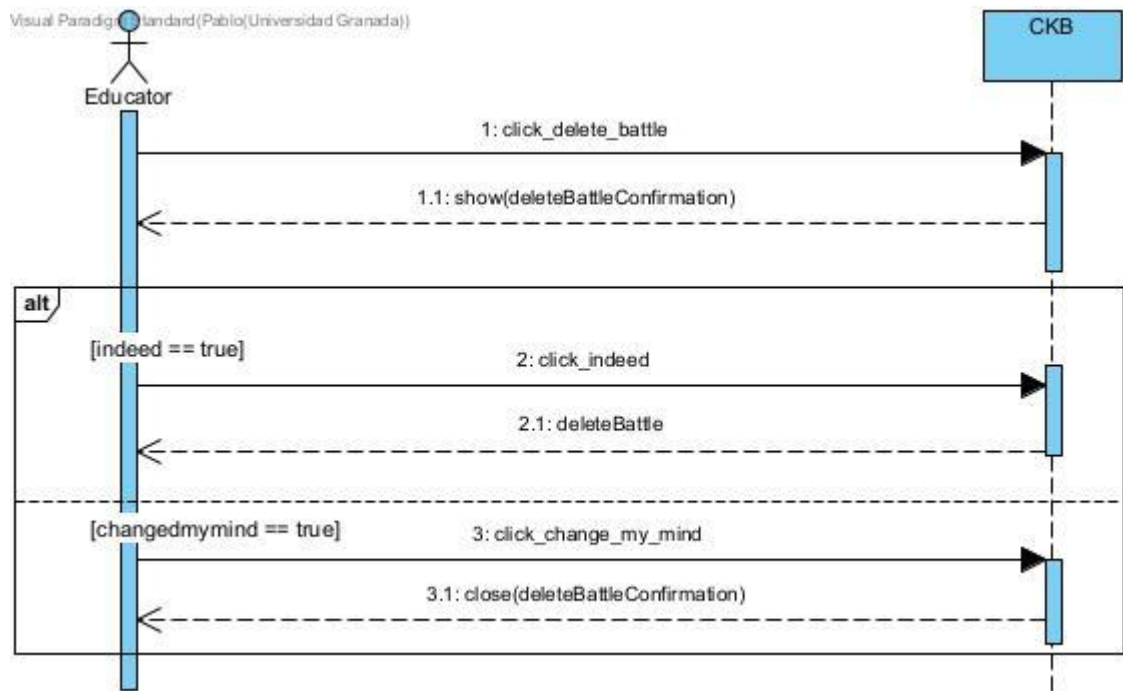


Figure 3.14

Create Badge

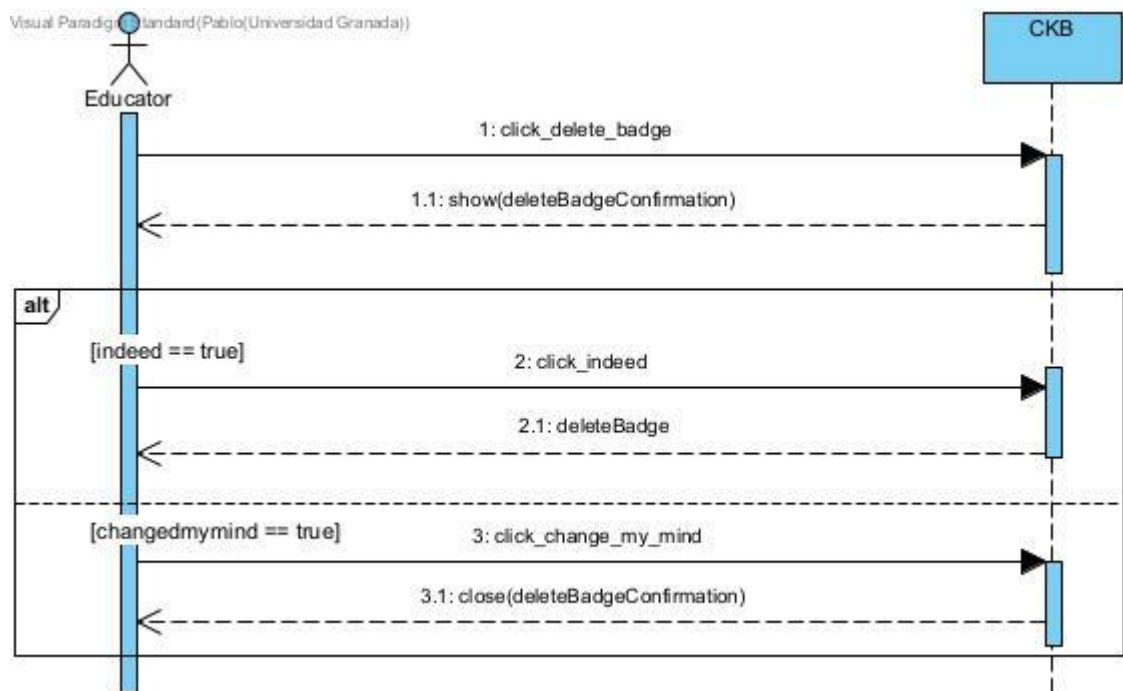


Figure 3.15

Create Requisite

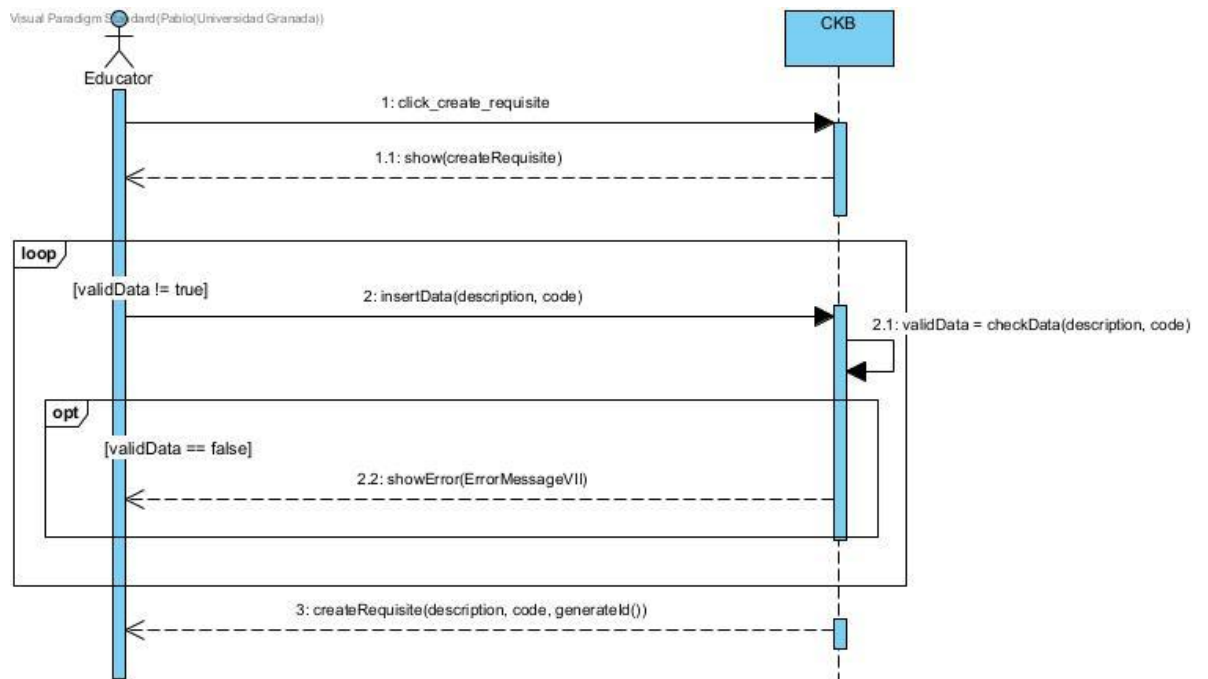


Figure 3.16

Allow Other Educator to Create Battles

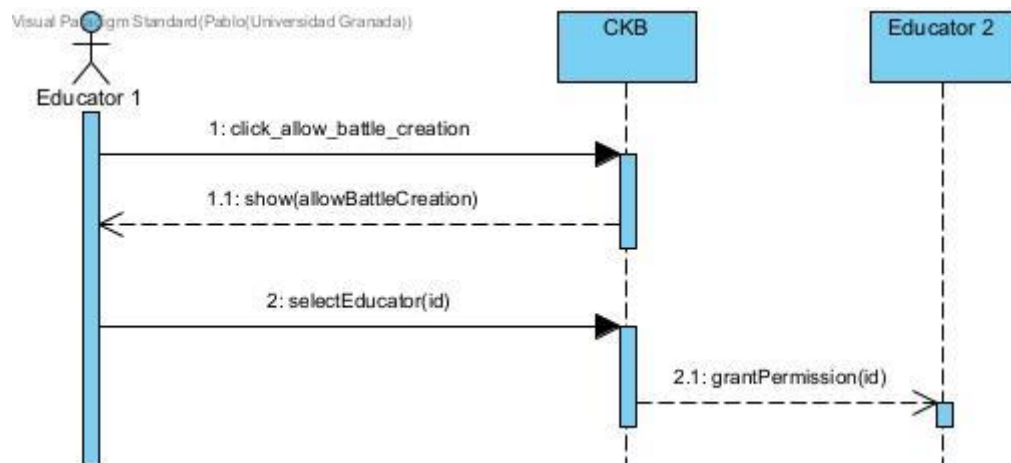


Figure 3.17

Delete Badge

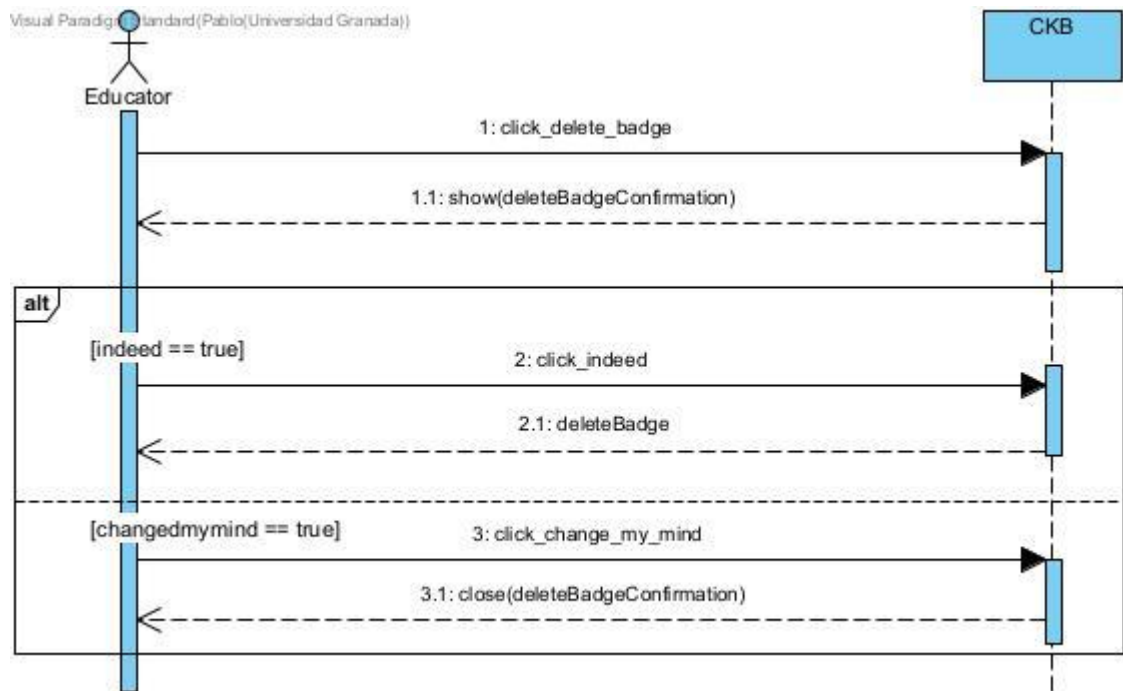


Figure 3.18

Delete Requisite

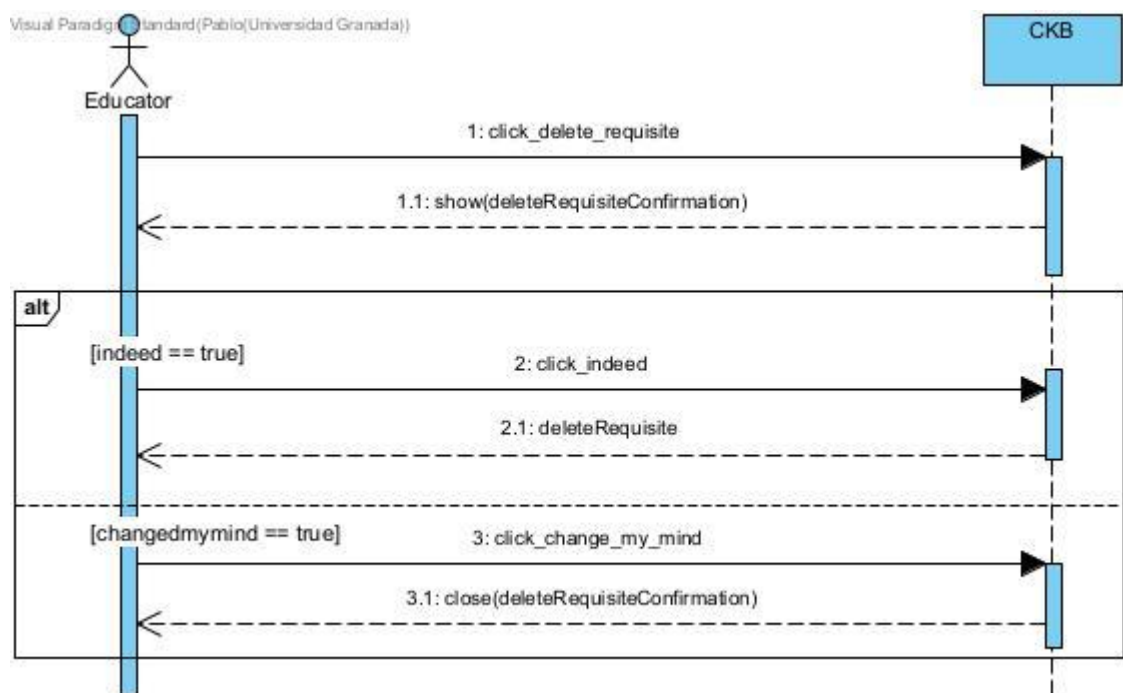


Figure 3.19

Create Group

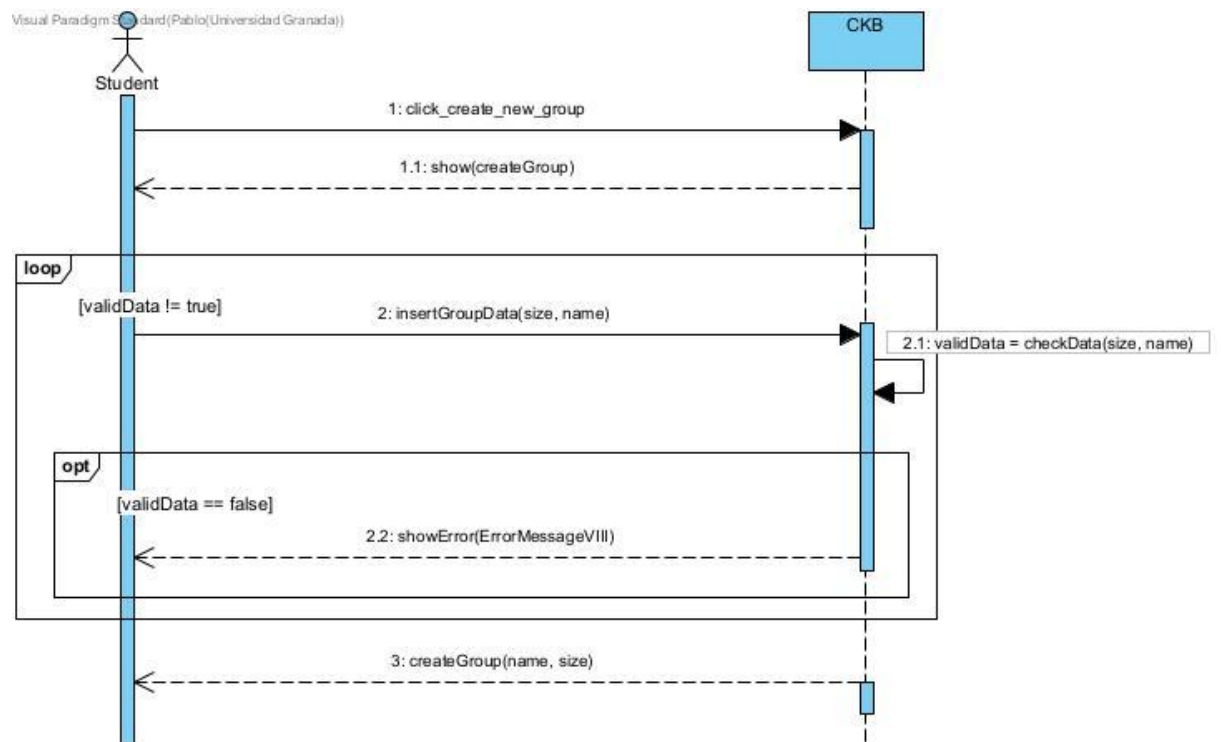


Figure 3.20

Join Group

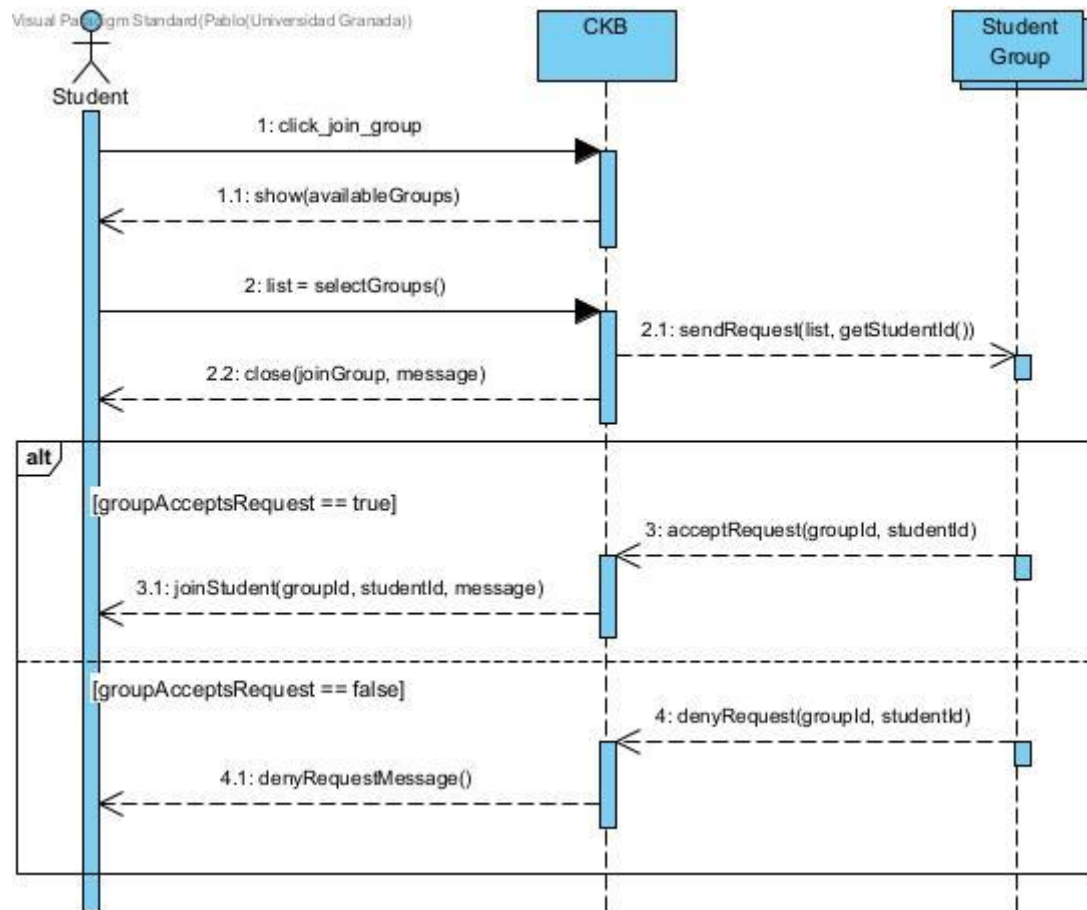


Figure 3.21

Grant Badge

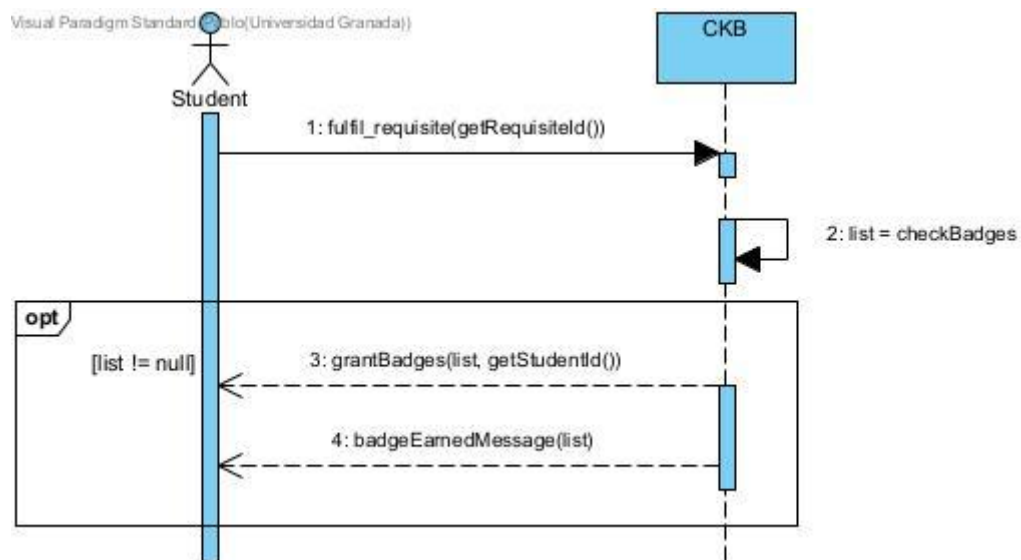


Figure 3.22

Hand In Submission

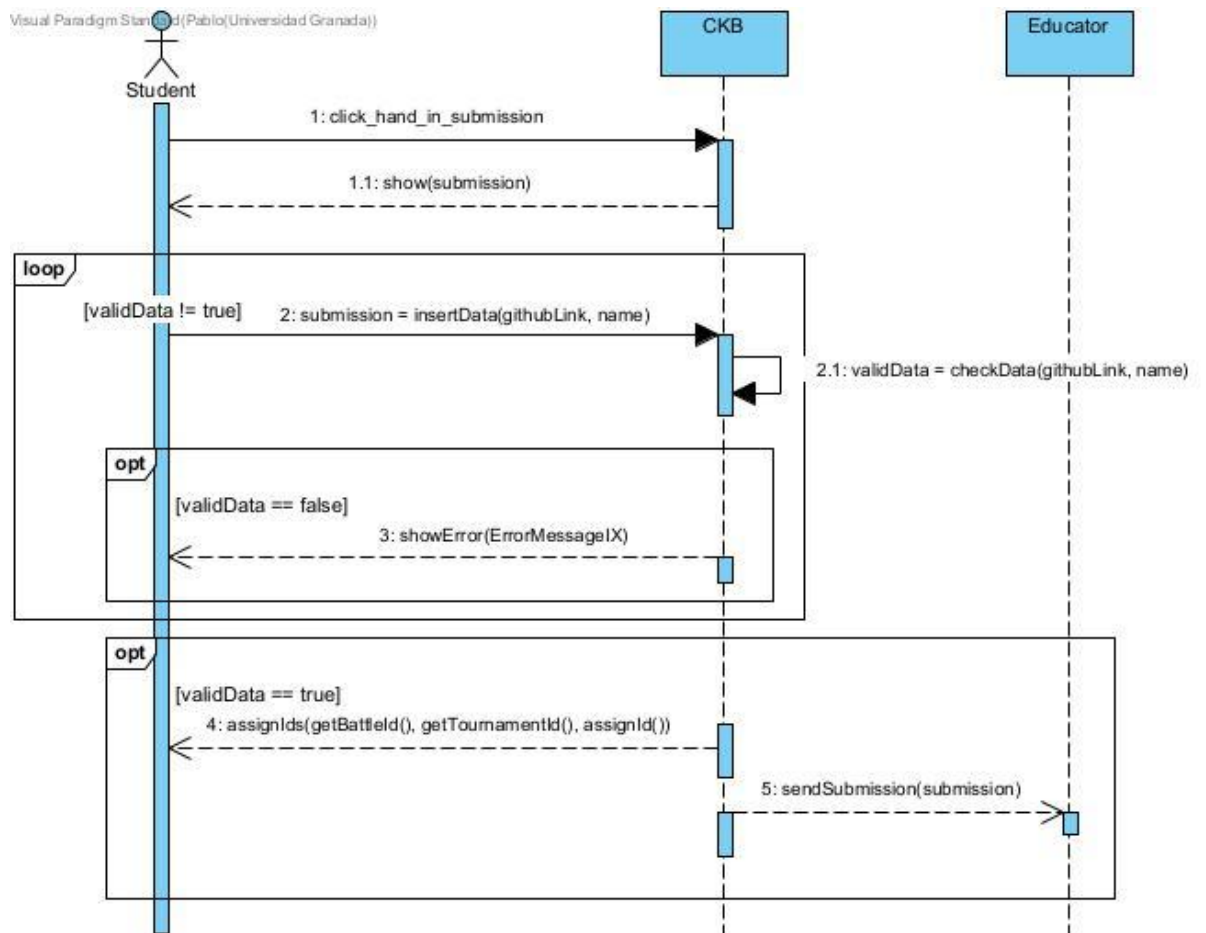


Figure 3.23

Delete Submission

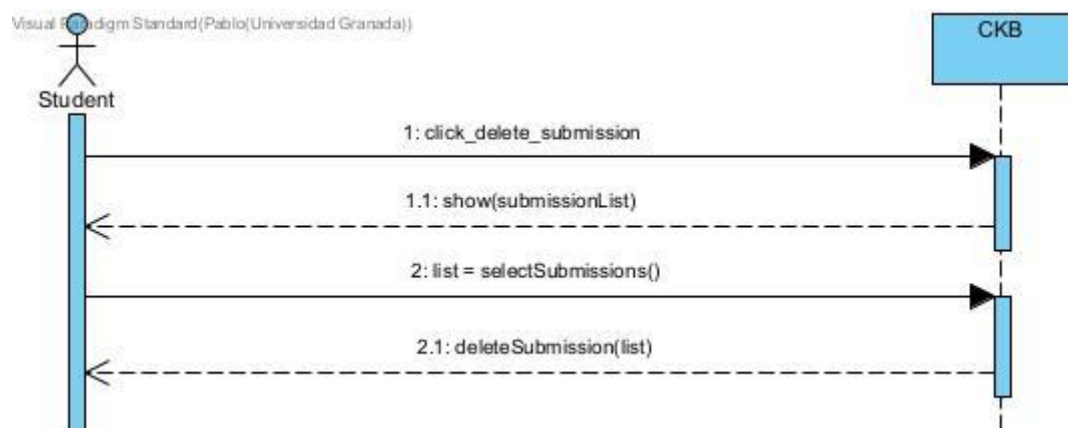


Figure 3.24

Manually Score Submission

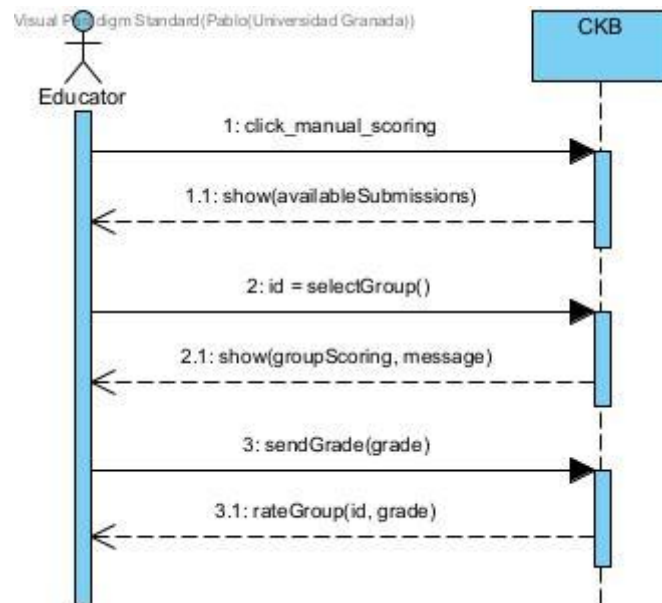


Figure 3.25

Automatically Score Submission

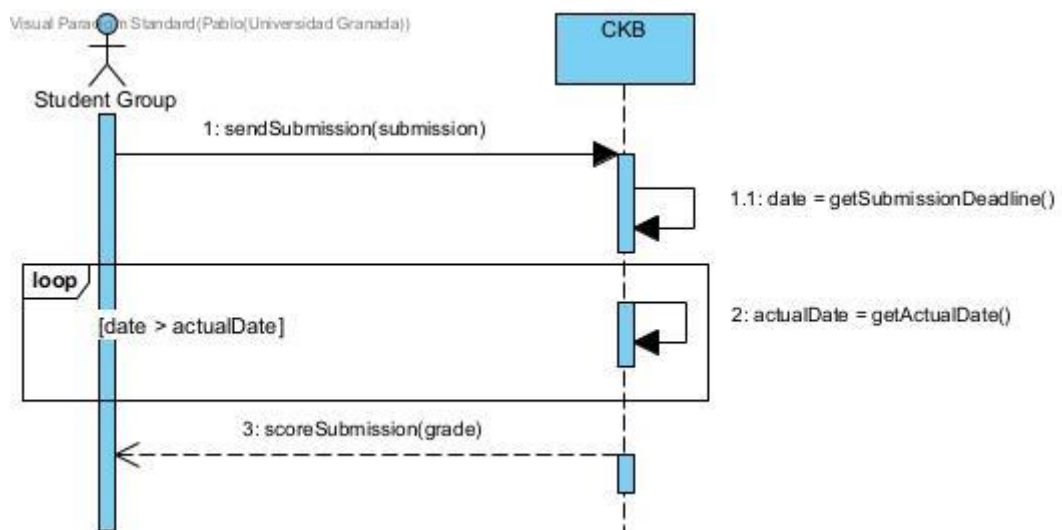


Figure 3.26

Show FAQs

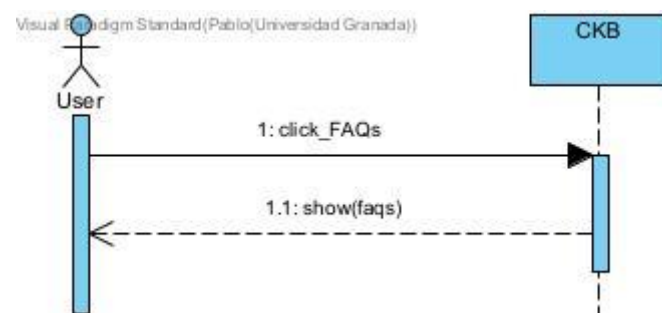


Figure 3.27

Send Direct Message

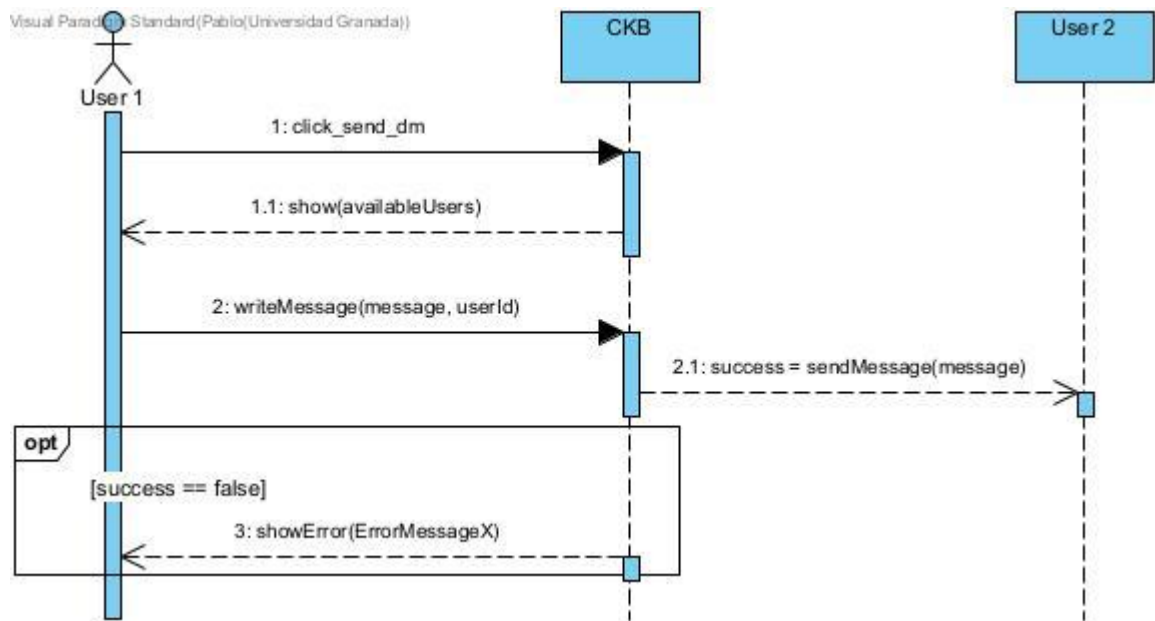


Figure 3.28

Update Profile

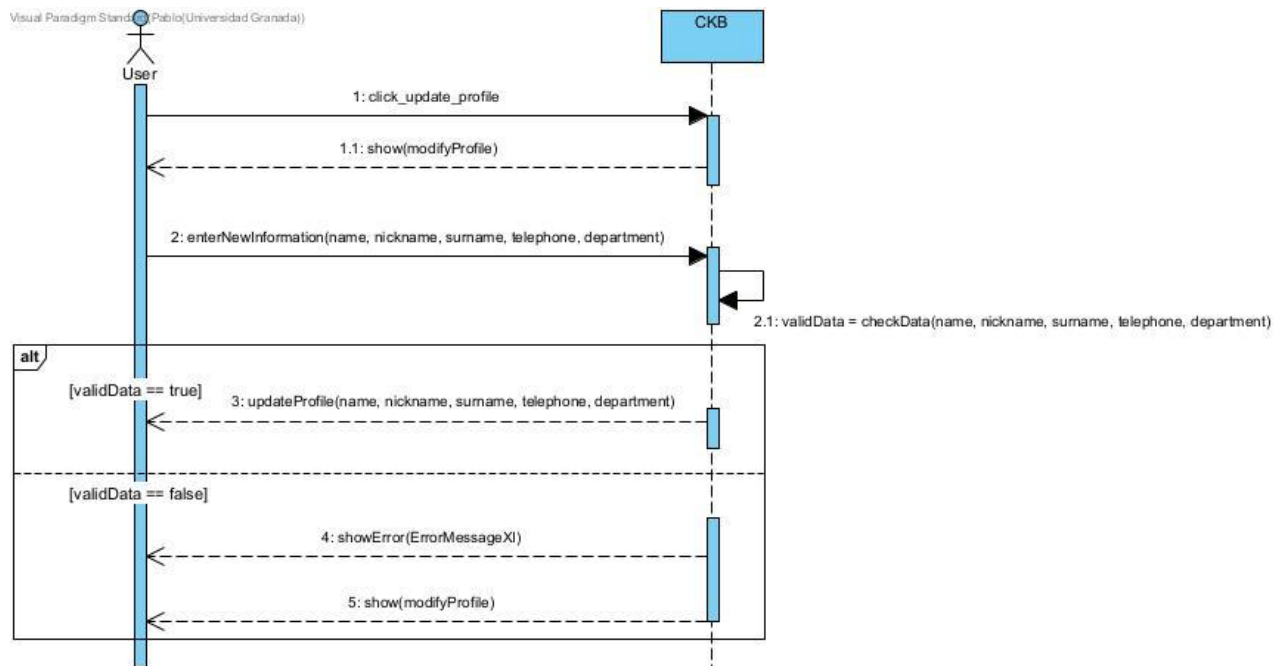


Figure 3.29

Expel Student From A Group

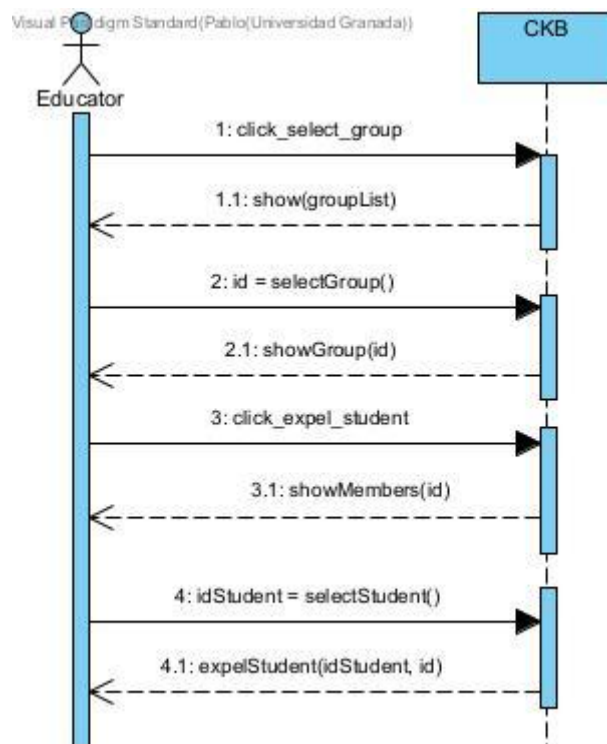


Figure 3.30

Update Battle Information

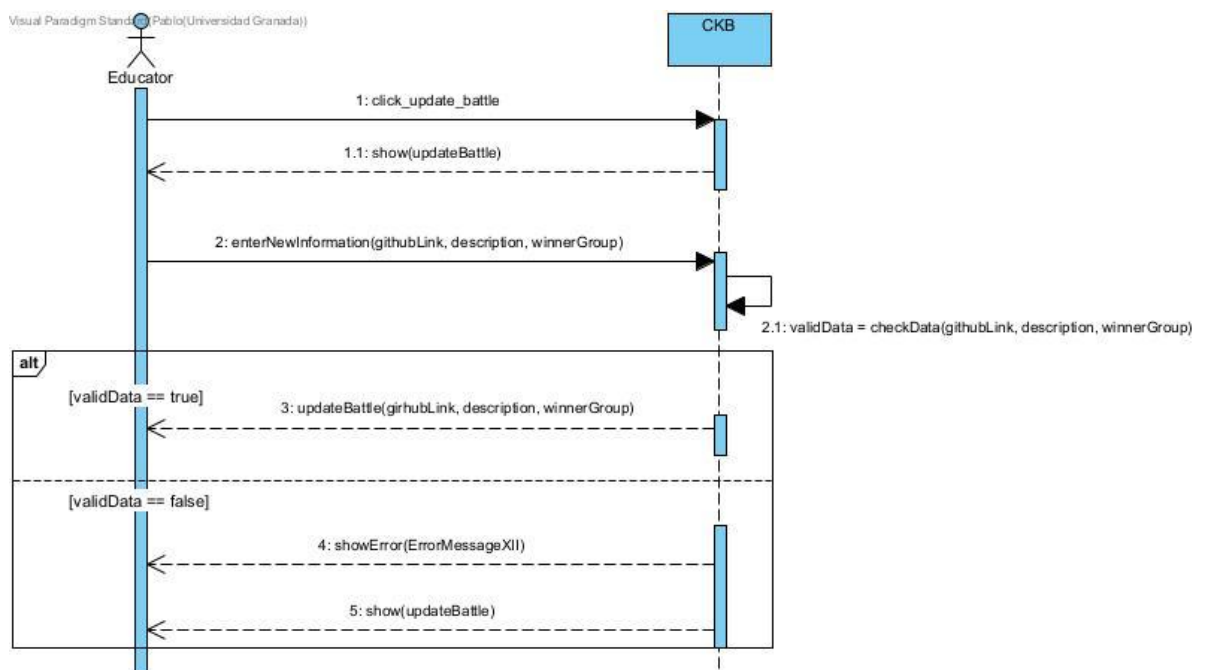


Figure 3.31

Update Tournament Information

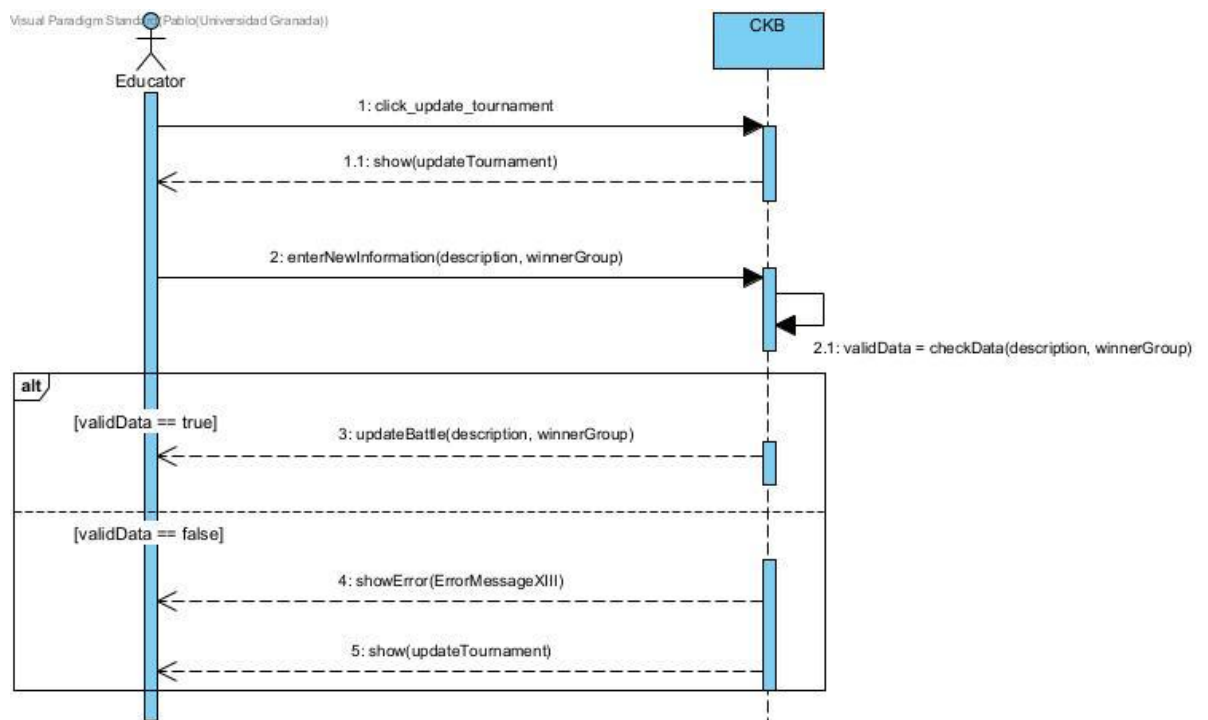


Figure 3.32

Delete Group

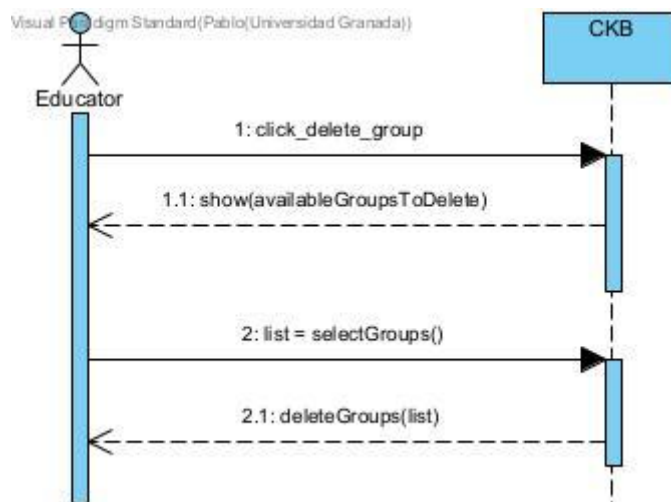


Figure 3.33

Abandon Group

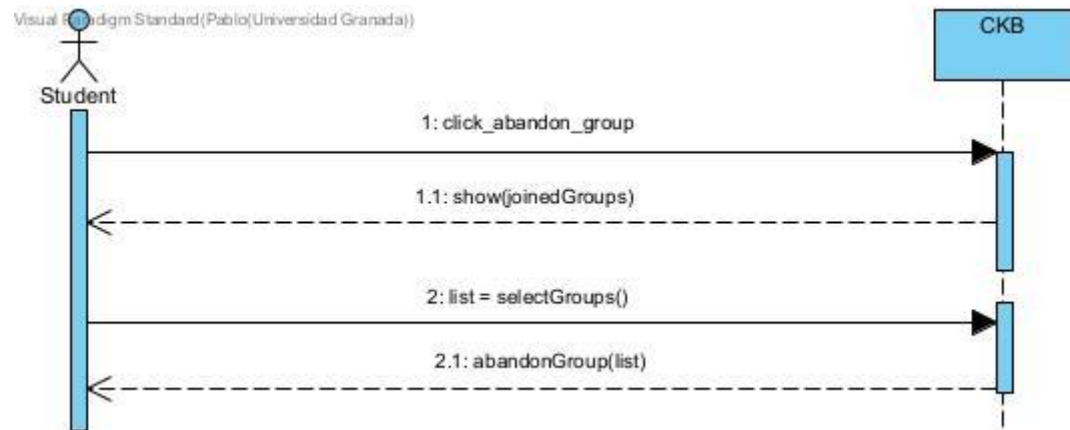


Figure 3.34

3.3 Performance Requirements

3.3.1 Response Time

- Requirement: The CKB platform should aim to provide rapid response times for user actions.

- Explanation: Users interacting with the platform, especially during coding challenges, expect near-instantaneous responses when running code tests or accessing their GitHub repositories. Sluggish response times can negatively impact the user experience.

- Measurement: Response times for typical user actions, such as loading a coding challenge, submitting code, running tests, or accessing a GitHub repository, should be kept under 1 second to ensure a smooth experience.

3.3.2 Scalability

- Requirement: The platform must be able to scale horizontally to handle a growing number of users and challenges without significant performance degradation.

- Explanation: As the user base expands and more coding challenges are introduced, the system needs to adapt to increased demand while maintaining performance. Scalability ensures that the platform remains responsive even as user and challenge numbers increase.

- Measurement: The system should be capable of handling a tenfold increase in users and challenges without a substantial decrease in response time, ensuring that scalability goals are met.

3.3.3 Throughput

- Requirement: The platform should support a high throughput of code submissions and test executions, especially during coding challenges.

- Explanation: During coding challenges, multiple users may simultaneously submit code for testing, requiring the system to handle a high volume of requests efficiently. Adequate throughput ensures that user submissions are processed promptly.

- Measurement: The system should support a throughput of at least 100 code submissions and test executions per minute during peak usage to meet user expectations.

3.3.4 Resource Utilization

- Requirement: The platform should optimize resource utilization to ensure efficient server performance.

- Explanation: Efficient utilization of server resources is vital to minimize operational costs and maintain responsiveness. Striking a balance in resource usage contributes to cost-effectiveness and system stability.

- Measurement: Regular monitoring of CPU, memory, and storage usage should ensure resource utilization remains below predefined thresholds, allowing for efficient server operation.

3.4 Design Constraints

3.4.1 Standards Compliance

- Requirement: The platform must adhere to industry-standard coding and security practices.

- Explanation: Adherence to established coding and security standards is crucial to ensure code quality, security, and compatibility with external services like GitHub. Conformance to these standards facilitates collaboration, code quality assurance, and system security.

- Measurement: Regular code reviews, audits, and security scans should confirm compliance with coding standards and identify potential vulnerabilities, which are subsequently addressed.

3.4.2 Hardware Limitations

- Requirement: The platform should be designed to operate on standard web server hardware.

- Explanation: To ensure accessibility and minimize hosting costs, the platform should not depend on specialized or high-end hardware configurations. Compatibility with common web server hardware contributes to the platform's reach.

- Measurement: Regular testing on standard web server configurations should confirm compatibility, ensuring that hardware requirements do not pose limitations for users.

3.4.3 Other Constraints

- Requirement: The platform should support multiple modern web browsers.
- Explanation: The platform's user base may use different web browsers, and the system should be compatible with popular options. This compatibility promotes accessibility and user satisfaction.
- Measurement: Regular testing should confirm compatibility with browsers like Chrome, Firefox, Edge, and Safari, ensuring a consistent experience across different browser environments.

3.5 Software System Attributes

3.5.1 Reliability

- Requirement: The system must prioritize reliability, minimizing downtime and data loss.
- Explanation: Users rely on the platform for coding challenges, and any interruption in service can disrupt their learning or participation. Reliability ensures uninterrupted access and data integrity.
- Measurement: The platform should aim for at least 99.9% uptime, with automated backups and disaster recovery procedures in place to minimize data loss and downtime in case of unexpected events.

3.5.2 Availability

- Requirement: The platform should be available 24/7, with minimal planned maintenance windows.
- Explanation: Users from various time zones and schedules should have uninterrupted access to the platform. Scheduled maintenance windows should be minimized to reduce service interruptions.
- Measurement: Planned maintenance windows should be infrequent, scheduled during low-traffic times, and communicated to users well in advance to minimize disruption.

3.5.3 Security

- Requirement: The platform must implement robust security measures to protect user data, code, and infrastructure.
- Explanation: Security is critical to safeguard user data and maintain user trust. Adequate security measures, including encryption, authentication, and authorization, are essential to protect sensitive information.

- Measurement: Regular security audits, penetration testing, and adherence to security best practices should ensure a high level of protection against common security threats, ensuring the security of user data and the platform.

3.5.4 Maintainability

- Requirement: The platform should be designed for ease of maintenance and updates.

- Explanation: To keep the system current and secure, maintenance should not be overly complex or time-consuming. Code modularity, automation of deployment processes, and version control contribute to maintainability

- Measurement: Regular updates, automation of deployment, and well-documented processes should ensure that maintenance tasks are efficient and do not disrupt the platform's availability.

3.5.5 Portability

- Requirement: The platform should be accessible from various devices and web browsers.

- Explanation: Users may access the platform from different devices and locations. Ensuring cross-device and cross-browser compatibility promotes accessibility and user engagement.

- Measurement: Regular cross-browser testing and responsive design should ensure portability, allowing users to access the platform from smartphones, tablets, laptops, and different web browsers while maintaining a consistent user experience.

4.

Pablo García Alvarado	INTRODUCTION	7h
	OVERALL DESCRIPTION	8h
	SPECIFIC REQUIREMENTS	13h
	FORMAL ANALYSIS USING ALLOY	6h
	EFFORT SPENT	5 min
	REFERENCES	5 min
Adrian Valica	INTRODUCTION	6h
	OVERALL DESCRIPTION	7h
	SPECIFIC REQUIREMENTS	11h
	FORMAL ANALYSIS USING ALLOY	8h
	EFFORT SPENT	5 min
	REFERENCES	5 min
Rishabh Tiwari	INTRODUCTION	8h
	OVERALL DESCRIPTION	7h
	SPECIFIC REQUIREMENTS	10h
	FORMAL ANALYSIS USING ALLOY	9h
	EFFORT SPENT	5 min
	REFERENCES	5 min

Rishabh :

2.1 Product Perspective

Detailed Scenarios for CodeKataBattle (CKB)

User takes part into a Code Kata Battle:

Alfredo wants to sharpen his skills in Software engineering. In order to do this, he goes on CodeKataBattle to find a challenge. He searches the challenges related to his coding language, and can set some more filters. After timeFrame, the system shows up all the possible challenges and he can browse through the list.

Alfredo clicks on the “Take part” button of the challenge he chose. The systems create a Github repository, and shortly after (timeframe), it shows the link to the repository on the screen. Alfredo can click it and access it.

On the page of the challenge, some runnable tests snippets appear. When clicking on any of the “Run” buttons, or on the “Run all” button, the system proceed to launch all the tests on the code of the repository. As soon as it is done (timeFrame), the system shows some results (errors, or output of the program), and if missed tests are highlighted in red.

Alfredo and his team can therefore Alfredo, aiming to sharpen his skills in software engineering, navigates to CodeKataBattle in search of a challenge. He specifies his preferred programming language and fine-tunes the search results by applying

filters for difficulty level, topic, and expected duration. Within moments, the system displays a curated list of challenges that perfectly align with Alfredo's criteria.

From the list, Alfredo selects a challenge that catches his interest and clicks the "Take part" button. The system immediately generates a private GitHub repository for the selected challenge and promptly provides Alfredo with a direct link to access the newly created repository.

On the challenge page, Alfredo finds runnable test snippets. To evaluate his initial code, he can click on "Run" for individual tests or "Run all" to execute the full suite of tests against his repository code. The system quickly provides detailed feedback, displaying the output or errors for each test case. Any failing tests are clearly marked in red, allowing Alfredo to easily identify areas that need attention.

Throughout the duration of the challenge, Alfredo and his team engage in collaborative coding, iterating on their solution. The system supports this process by offering real-time feedback and performance metrics, and they can see their score on the global leaderboard at every submission. It is updated at each push of code on the repository.

At any time, Alfredo can submit his final solution. The code is directly taken from the Github repository and is saved as the final result of his participation.

User checks the results of the competition

Pierluca, eager to see how he fared, receives an email and in-app notification from CodeKataBattle indicating that the results of the challenge he recently participated in are now available. The message comes after a meticulous consolidating phase where organizers grade and verify all submissions.

He promptly logs into CodeKataBattle and navigates to 'My Challenges' on his dashboard, which efficiently organizes his participations in reverse chronological order. Selecting the challenge in question, Pierluca clicks to uncover a detailed report of the event.

The platform provides him with an interactive scorecard, clearly displaying his overall score out of 100. It delves into his performance, offering a nuanced breakdown across essential factors:

- Functional Correctness: Displayed as the ratio of passed test cases to the total, showcasing his solution's efficacy.
- Timeliness: Measured by the interval between the registration deadline and his last commit, indicating his efficiency.
- Code Quality: Evaluated through static analysis tools that assess various code aspects pre-selected by the challenge creator—such as security, reliability, and maintainability—reflecting the sophistication of his work.

Pierluca is then presented with the final leaderboard, a dynamic display of top-scoring solutions and a comprehensive list of rankings. His own submission is distinctively highlighted, allowing him to instantly locate his standing.

To encourage a culture of learning and improvement, the platform encourages exploration. Pierluca can view profiles and submissions of top-ranking participants, glean insights from their approaches and potentially enhancing his own software engineering acumen for future challenges.

User subscribes to an upcoming Challenge

Pablo wants to have some programming exercise and decides to join a new CodeKataBattle challenge. He signs in to the CodeKataBattle platform and employs filters to select challenges that align with his expertise and preferences. His attention is captured by a challenge scheduled for the upcoming weekend which perfectly fits into his schedule. The interface conveniently displays the time remaining until the challenge begins and prompts him with an "Enroll" option.

Upon clicking "Enroll," Pablo's participation is confirmed, and his account is seamlessly added to the participant list for the challenge. The system acknowledges his enrollment and ensures he's informed about the essential details.

In anticipation of the challenge start date, the platform proactively generates a private GitHub repository for each contestant. This strategic move is designed to distribute system load and prevent delays. Pablo, along with the other participants, is notified as the challenge kicks off, marking the beginning of an exciting coding journey.

At the start of the challenge, Pablo receives an alert, along with a link to his exclusive GitHub repository, which had been kept secret until now. With a click, he accesses his repository and is all set to dive into the coding problems presented by the CodeKataBattle challenge, ready to code his way to solutions.

User cancels his participation before the challenge starts

Mario, a student and an aspiring programmer, had enthusiastically registered for an upcoming CodeKataBattle weekend challenge. However, as the week progresses, he realizes that his video game project is likely to require more of his time than he initially thought. He makes the practical decision to cancel his participation in the challenge.

He logs into his account on the CodeKataBattle platform and heads straight to the 'My Challenges' section. There, amongst his list of activities, he easily spots the upcoming challenge. Right next to the challenge details, the 'Withdraw' button catches his eye, indicating a clear option for opting out.

With a purposeful click on 'Withdraw,' a confirmation prompt appears, ensuring that he doesn't accidentally cancel his participation. Mario reaffirms his decision, and clicks 'Confirm.' His action is immediately processed by the system. In response, his account is removed from the registered account for the challenge is cancelled, his dedicated private GitHub repository for the challenge is erased, and the challenge itself vanishes from his 'My Challenges' list.

Now certain that he will not be interrupted by any notifications related to the challenge, Mario focuses his attention back on his video game project. Despite withdrawing, he can effortlessly re-enroll in the challenge any time before it officially ends.

User cancels his participation during the challenge

Just as the CodeKataBattle challenge begins, Luigi receives a call from his brother, who urgently needs his help over the weekend. Family comes first for Luigi, and he quickly realizes that he won't be able to participate in the challenge. He decides to withdraw to support his brother.

Luigi logs into CodeKataBattle, heads to the challenge page, and immediately spots the 'Withdraw' button. He clicks it, and a confirmation pop-up appears. Without hesitation, Luigi confirms his decision, understanding the implications.

Upon confirmation, the system efficiently processes Luigi's withdrawal. It removes his account from the list of active participants, ensuring his prompt exit from the competition. The system takes further steps to maintain data integrity and privacy: it erases Luigi's submission history, clears any of his entries from the leaderboard, and deletes the GitHub repository that was allocated for his participation in the challenge. As a final step, the challenge itself is removed from Luigi's list of current participations.

While Luigi steps back from the challenge to focus on family matters, he remains aware that he can rejoin the competition at a later time if his schedule allows. However, Luigi knows that if he chooses to re-enter, he will have to start from scratch, as all his previous progress has been completely reset following his withdrawal.

Educator creates a tournament

Dr. Luca, an educator of the computer science class, decides it's time to challenge his students with a practical coding tournament. He plans to use CodeKataBattle to create this exercise session.

He logs into his CKB educator account. His credentials grant him access to an array of tools tailored for educators. Dr. Smith navigates to the 'Create Tournament'

section. He's greeted by a user-friendly form designed to capture all the essentials of a compelling coding tournament.

With careful thought, Dr. Smith begins filling in the details of the tournament. He chooses the title of the tournament, and a detailed description. He considers the skill level of his students and sets the tournament's difficulty level accordingly. He drafts a series of problem statements, each designed to test different aspects of coding, and sets two deadlines: one for the students to register in groups to the tournament, and one for the end of the submissions in the tournament challenges.

Next, Dr. Luca uploads several supporting documents. These include resources to help his students understand the problems better and criteria for automatic scoring that ensure a fair and objective evaluation of the submissions.

Before making the tournament live, Dr. Luca takes a moment to review all the information. Confident with the setup, he clicks the 'Publish Tournament' button. Within moments, the tournament goes live on CKB. Simultaneously, an automated notification is dispatched to all his students, inviting them to participate in this exciting coding adventure.

Educator scores a challenge manually

The deadline has passed, and it's time for Dr. Luca to start checking all the submissions that await his review.

He logs into the CodeKataBattle platform, where he's greeted with a dashboard containing the latest submissions. One particular challenge in the tournament, known for its complexity, requires a more nuanced approach—manual scoring.

Dr. Luca carefully opens each submission, his screen transforming into a canvas of code. He meticulously evaluates each key factors such as efficiency, readability, and correctness for every submission.

Dr. Luca assigns a score to each submission. He also writes a detailed feedback for each student, comments designed to guide, encourage, and enlighten.

Once the scoring is complete, Dr. Luca publishes the scores and feedback on the CodeKataBattle platform. All the students that took part into the tournament are notified that the grades are available. The manual scoring of Dr. Luca and the automatic scoring from CodeKataBattle are combined to provide a final score, with proportions fixed in advance by the educator.

Educator adds a new collaborator

Dr. Calu wants to use the tournament that Dr. Luca created in order to evaluate his students. But he also wants to add some more exercises for his class.

Dr. Luca agrees to this and wants to add him as a collaborator on the tournament management. He logs into the CodeKataBattle account and on the page of his

tournament, he clicks on the button “Add Collaborators”. A pop up appears, and he now enter a username to add. He enters Dr. Calu’s username and confirms his choice.

Dr. Calu gets notified on his account that he is invited to become a collaborator on the tournament. As he accepts the invitation, he is brought to the page of the tournament. The interface is the same as the one of the Dr. Luca’s, he can edit anything and manage the tournament as its owner, with one difference: he cannot delete it.

Dr. Luca can see the list of the members on the top of the screen and decide to restrain one’s access anytime.

User earning a badge

Alice, a diligent and skillful coder, has been a whirlwind of activity on the CodeKataBattle platform, partaking in numerous tournaments. Her dedication to the craft is evident in her consistent top-tier performances, each submission a testament to her growing prowess.

Behind the scenes, the gamification system of CKB acts as a silent auditor of success, meticulously tracking Alice's achievements. It's not just a counter of points; it's a recognizer of effort and skill. As Alice continues to excel, the system notes her eligibility for one of the most coveted acknowledgments on the platform – the 'Coding Master' badge.

The moment Alice's latest successful submission ticks the final checkbox on the list of criteria, the system springs into action. With digital fanfare, the 'Coding Master' badge is automatically awarded to her, its icon taking pride of place on her CKB profile.

Almost immediately, a notification pops up, a digital drumroll announcing her new achievement. Alice clicks on the alert, and her profile blooms with the new badge, its presence a badge of honor and a milestone in her coding journey.

As Alice views her badge, a symbol of her dedication and expertise, it serves as both a reward and a beacon, inspiring her and her peers to continue pushing the boundaries of their coding capabilities on CodeKataBattle.

Users forming a group for a challenge

John and Emma, both keen to leverage their collective problem-solving skills, decide to join forces for a group challenge hosted on CodeKataBattle (CKB). Recognizing the power of collaboration, they are excited to see what they can achieve together.

Taking the first step in their collaborative journey, they log into CKB and set up a group on the platform, seamlessly inviting one another to join. This virtual space is set to become their hub of innovation and teamwork.

With the group established, John and Emma engage in a strategy session. They use the platform's integrated chat feature to communicate in real-time, discussing the challenge's intricacies and plotting their course of action. They divide the tasks according to their strengths, with Emma taking on the algorithmic challenges and John focusing on data structure optimization. For more complex aspects of the project, they turn to external tools that offer real-time collaborative coding capabilities, ensuring that no stone is left unturned.

The challenge is in full swing as they begin to weave their code together. They each contribute parts of the project, merging their individual work into a cohesive whole. After rigorous testing and revision, they are ready to push their combined code to their shared GitHub repository. With a sense of accomplishment, they submit their final solution through CKB, marking the culmination of their joint effort.

As they await the verdict, the CKB system evaluates their submission. Before long, John and Emma receive detailed feedback on their work, along with points that reflect their hard work and creativity. These points are a boon to their individual profiles, contributing to their standings on the platform and opening up new opportunities for learning and growth.

Their successful collaboration on CodeKataBattle not only strengthens their programming abilities but also cements a partnership that blends two coding minds into a formidable force.

These scenarios illustrate the comprehensive functionalities of CKB, from the creation and participation in coding challenges to the integration of gamification and real-time feedback mechanisms. They showcase the platform's ability to facilitate a dynamic and interactive learning environment for both students and educators in the field of software development.

Product Functions:

1. User Authentication and Profile Management:

- Secure login and logout functionality for users.
- Profile creation and management for both students and educators.
- Personalized settings, including notification preferences.

2. Tournament Creation and Management:

- Educators can create coding tournaments with customizable problem statements, difficulty levels, deadlines, and scoring criteria.
- Functionality for editing, deleting, or updating ongoing tournaments.

3. Problem Statement Repository and Custom Challenge Creation:

- A repository of coding problems of varying difficulty levels.
- Educators have the ability to create custom challenges that align with specific learning objectives.

4. Solution Submission and Integration with External Platforms:

- Facility for students to submit their solutions via GitHub integration or direct upload.
- Support for both individual and collaborative (group) submissions.

5. Automatic and Manual Scoring System:

- Automatic scoring of submissions based on predefined criteria.
- Manual scoring and feedback provision by educators for detailed evaluation.

6. Real-Time Leaderboard and Ranking System:

- Dynamic leaderboards that update in real-time based on student submissions and scores.
- Ranking of participants based on performance in individual challenges or tournaments.

7. Gamification and Achievement System:

- Badges and achievements awarded to students based on their performance and participation.
- A system that tracks progress and milestones, encouraging continuous learning.

8. Communication and Collaboration Tools:

- Integrated messaging and discussion forums for communication between students and educators.
- Group functionality to enable collaborative problem-solving and learning.

9. Notifications and Alerts:

- System-generated notifications for new tournaments, submission deadlines, scores, and feedback.
- Customizable alerts for students and educators about relevant events and updates.

10. Administrative and Reporting Tools:

- Dashboard for educators to monitor student progress and tournament participation.
- Reporting tools for analytics on student performance, engagement, and tournament outcomes.

11. Resource Library and Learning Material:

- Access to a library of resources, tutorials, and documentation to assist with coding challenges.
- Educational content to support self-paced learning and skill enhancement.

12. Mobile and Cross-Platform Accessibility:

- Responsive design for access on various devices, including potential future mobile app support.

These functions collectively define the capabilities of the CodeKataBattle system, outlining its comprehensive approach to facilitating an interactive and engaging learning environment for coding skills enhancement. The platform is designed to be user-friendly, scalable, and adaptable to the evolving needs of the software development education community.

User Characteristics of CKB

1. Educators:

- Role and Responsibilities:

- Create and manage coding tournaments and challenges.
- Provide guidance, resources, and feedback to students.
 - Evaluate student submissions, either manually or through the automated scoring system.
- Monitor student progress and adapt teaching methods accordingly.

- Skills and Expertise:

- Proficient in various programming languages and software development methodologies.
- Skilled in creating educational content and problem statements that reflect real-world coding scenarios.
- Capable of assessing code quality, including factors like efficiency, readability, and correctness.

- Motivations and Goals:

- Enhance the coding skills of their students.
- Foster a competitive yet collaborative learning environment.
- Keep track of student progress and adapt teaching strategies to meet diverse learning needs.

- Interaction with CKB:

- Frequent use of the platform for creating and managing coding challenges.
- Active participation in the community through discussions and feedback.

2. Students:

- Role and Responsibilities:

- Participate in coding tournaments and challenges.
- Collaborate with peers in group challenges.
- Actively engage in self-directed learning and skill enhancement.

- Skills and Expertise:

- Range from beginner to advanced coding skills.
- Familiarity with basic concepts of programming and problem-solving.
- As they progress, develop expertise in more complex algorithms and coding practices.

- Motivations and Goals:

- Improve coding skills and theoretical knowledge.
- Gain practical experience by solving real-world coding problems.
- Earn achievements and recognition within the CKB community.

- Interaction with CKB:

- Regularly participate in challenges and tournaments.
- Engage with the community through discussions and collaborative projects.
- Utilise resources and tutorials provided on the platform for learning and problem-solving.

3. Administrators:

- Role and Responsibilities:

- Oversee the overall functioning of the CKB platform.
- Manage user accounts and ensure the smooth operation of the system.
- Address technical issues and implement updates and improvements.

- Skills and Expertise:

- Technical proficiency in managing web-based platforms.
- Skills in user experience design and system maintenance.
- Ability to troubleshoot and resolve technical issues efficiently.

- Motivations and Goals:

- Ensure a seamless and user-friendly experience for all CKB users.
- Maintain system security and data integrity.
- Continuously improve the platform based on user feedback and technological advancements.

- Interaction with CKB:

- Constant monitoring and maintenance of the platform.
- Implementing updates and new features.
- Providing support and assistance to educators and students as needed.

4. Community Members (Optional Future Expansion):

- Role and Responsibilities:

- Engage with CKB as guest users or potential future users.
- Participate in public forums and discussions.
- Explore available resources and challenges.

- Skills and Expertise:

- Varied levels of coding expertise, from novice to expert.
- General interest in software development and coding practices.

- Motivations and Goals:

- Explore the world of coding and software development.
- Seek learning opportunities and potential future engagement with CKB.

- Interaction with CKB:

- Limited access compared to registered students and educators.
- Potential for future engagement as full users based on their experience as community members.

These user characteristics outline the diverse range of individuals who interact with the CodeKataBattle system, each with unique roles, skills, motivations, and ways of engaging with the platform. This diversity underpins the multifaceted nature of CKB,

catering to a wide spectrum of needs and expectations within the realm of coding education.