CodeKataBattle

Diegoli Tommaso, Tagliani Fabio

15/12/2023

Contents

1 Introduction

1.1 Purpose

An essential component of every programming course involves ensuring that theoretical explanations are complemented by rigorous practical training, ensuring a thorough comprehension by students. However, a challenge arises, especially in larger classes, where educators find it difficult to individually monitor and provide feedback on each assignment. The CodeKataBattle platform aims to address this issue by providing an interactive space where students can actively practice and assess their programming skills. Through automated evaluation, students can compare their results with their peers, fostering a collaborative learning environment.
The platform consists of tournaments in which one or more educators can insert battles where teams of students can compete. The platform then provides a rank for each student in the tournament and a rank for the teams in a battle.
The subsequent section outlines the specific goals of the platform in a schematic manner.

1.1.1 Goal

G1. Educators and students must be allowed to register and access the CKB platform

G2. Educators must be allowed to create tournaments

G3. Educators must be allowed to create battles in tournaments

G4. Students must be allowed to subscribe to a tournament

G5. Students must be allowed to join battles in teams (even solo teams)

G6. Educators must be allowed to optionally manually evaluate teams' codes

G7. Students' codes must be automatically evaluated by the system

G8. Users must be allowed to see battles teams' ranks

G9. Users must be allowed to see tournaments students' ranks

G10. Students must be notified about relevant events happening in CodeKataBattle

1.2 Scope

To better define the project we need to define the phenomena happening in the world it will be placed. We distinguish between phenomena that don't directly impact the platform (World phenomena); phenomena generated by the platform that will influence the world (Machine-Generated Shared phenomena) and the ones generated by the world that will influence the platform (World-Generated Shared phenomena).

1.2.1 World Phenomena

Real world phenomena that don't directly influence the system. We define the following ones:

W1. Educators want to improve and test students' practical skills

W2. Educators write CodeKatas

W3. Students want to improve programming skills

W4. The adoption of new learning techniques like collaborative learning

W5. Students program on GitHub

1.2.2 World-Generated Shared Phenomena

SW1. An educator creates a tournament

SW2. An educator creates a battle (setting parameters and uploading the CodeKata)

SW3. A student subscribes to a tournament

SW4. A student create a team for a battle

SW5. A student join a team for a battle

SW6. A student pushes a new commit into the main branch of their repository

SW7. An educator evaluates the work done by teams of students

1.2.3 Machine-generated Shared Phenomena

SM1. CKB shows all the tournaments and for the subscribed ones all the available battles

SM2. CKB sends a notifications to students once they register

SM3. CKB sends a notifications to students once a new tournament is created

SM4. CKB sends a notification to students subscribed to a tournament when a new battle is available in that tournament

SM5. Once registration deadline to a battle is over, CKB creates a GitHub repository with the CodeKata

SM6. CKB sends a notification to all students subscribed to the battle when the final rank is available

SM7. CKB shows the teams' ranking in battles

SM8. CKB shows the students' ranking in tournaments

1.3 Definitions, Acronyms, Abbreviations

The following terms are used in the document:

- CodeKata: a programming exercise in a chosen language. It includes a text description, build automation scripts and test cases that must be passed to complete the challenge.
- CKB: CodeKataBattle
- SSO: Single-Sign on

1.4 Revision History

15/12/2023 Version 1.0

1.5 Reference Documents

Assignment Document: Assignment RDD AY 2023-2024.pdf

1.6 Document Structure

The document is composed of 6 main sections.

In the opening section, rationale behind constructing the platform is provided, accompanied by a delineation of its goals. Following this, the document introduces the phenomena occurring in the world where the platform is intended for use. The document proceeds to elucidate key terms for better understanding. Lastly, the revision history and the referenced document are defined, contributing to a comprehensive overview for the reader.

In section two there is an overall description of the platform and the main product functions are defined. Then the different users that will use the system are depicted and the main scenarios in which they'll use it are explained. Lastly assumptions about the platform's domain are illustrated. Here it's included also a class diagram.

In section three there is the description of use cases represented also through a diagram and sequence diagrams.
Then the platform's requirements are introduced, in particular external interface requirements, functional requirements and non functional ones (for example performance requirements). This is followed by the mapping between platform's goals, use cases and functional requirements.

In section four a formal analysis of the model proposed is defined. This analysis is performed with the help of Alloy by modelling our world and running it with the defined assertions.

The documents ends with sections five and six that respectively shows the effort spent by each team's member in the creation of the document and the references to resources that have been used.

2 Overall Description

2.1 Product Perspective

Here we introduce some real world examples to show how the platform will be used.

2.1.1 Scenarios

Scenario 1: Registration to CodeKataBattle

1.A: Registration by Educator
Educator Paolo, professor of Computer Science, wants to introduce some programming challenges to his students. He asks his colleagues the best way to do that. They recommend him the CodeKataBattle web-app. He then proceeds to register by using university credentials.

1.B: Registration by Student
Student Marco, that attends the course of educator Paolo, wants to start the challenges that were introduced during the last lecture. So, he proceeds to visit the recommended website CodeKataBattle and register with his university credentials.

Scenario 2: Creation of a tournament
Educator Paolo, logged in the CKB web-app, wants to create a tournament to have a place in where students can compete through challenges. So, he starts the tournament creation process choosing the name and the subscription deadline. After the creation he can choose to grant other registered colleagues to post challenges in that tournament.

Scenario 3: Creation of a battle
Professor Luca, who was allowed by his collegue to post challenges in a specific tournament, wants to assign to his students an homework for the weekend. So, he proceeds to create a battle: he uploads the CodeKata, he sets minimum and maximum number of students per team, he sets "Java" as programming language, he sets a registration deadline by Friday midnight and a final submission deadline by Monday and he chooses the aspects on which the students' codes should be evaluated.

Scenario 4: Subscription to a tournament
Student Cesare, registered to the CKB platform, receives an email notification about the creation of a new tournament by professor Luca. He sees that the registration deadline is until Friday midnight and so he decide to immediately subscribe to that tournament following the procedure explained in the email.

Scenario 5: Joining battles
Students Cesare, Alessandro, Matteo and Mattia are all subscribed to Luca's tournament, for which they received some notifications about new available battles, want to start doing some of them because they have some free time. So, they visit the CKB platform and look for some of them.

5.A: Joining solo
On Friday 11 PM, student Cesare couldn't find someone to work with. Hence, he decides to simply join one of available battles by creating a solo team without inviting anyone.

5.B: Creating a team

On the WhatsApp group of his class, Alessandro managed to find some colleagues to work with for some of the available battles. He noticed that the platform offers the possibility to collaborate with other registered students when joining battles by sharing an invite code and so he sends it to Matteo and Mattia to become part of his team.

5.C: Joining a team

Matteo and Mattia receives the code from Alessandro so they use it to join a battle as a team and start working together.

Scenario 6: Manually evaluation of codes

When a battle he created ended, educator Luca wants to give his personal score in addition to the automatic one. So, he visits the page of the battle and he go through all the the sources produced by each team and assigns a score.

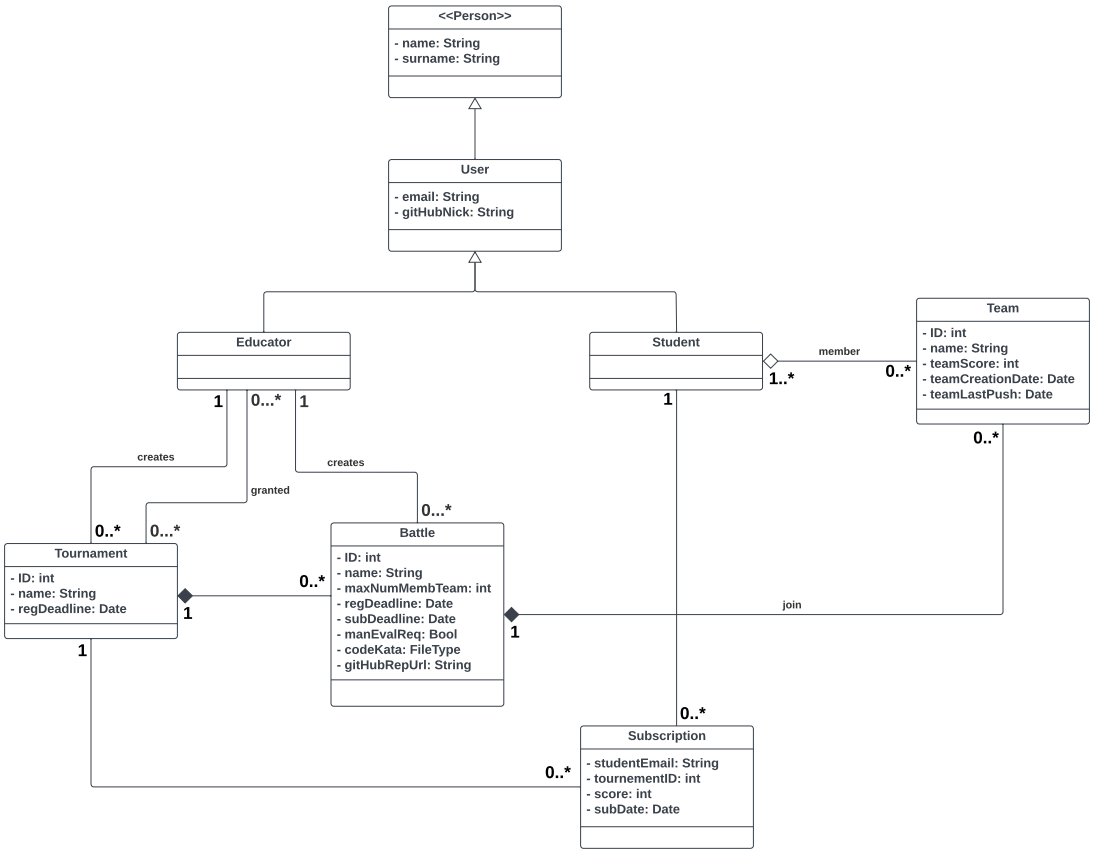
Scenario 7: Browsing battle's ranking

On Sunday, the Luca's battle is still ongoing and Alessandro, whose team already pushed the final solution, wants to check the partial ranking of the battle to check how his team is performing. So, he visits the battle's page on the CKB platform and checks the ranking.

Scenario 8: Browsing tournament's ranking

On Monday afternoon, Alessandro receives an email notification about the availability of the final ranking of Lucas's weekend battle. So, he visits the platform and firstly he checks in the battle's page what score was given to his team and then he goes back to the tournament page to check the updated students' ranking to check if he gained or lost some positions.

2.1.2 Class diagram



Here it's represented the class diagram. It's quite simple and it shows the distinction between the two different types of users, educators and students, that have different roles inside the platform. The class subscription is introduced to manage the different scores of the players in different tournaments. There could be a similar class also between teams and battles but it's not really needed as each team competes in only one battle.

2.2 Product Functions

Here the different functions provided by the platform to the different actors are illustrated.

2.2.1 Unregistered person

Registering to CKB platform The platform can only be used by registered users, so it requires to sign up at the first access. There are different options to do so:

1. **Single sign-on**
   If the school/university has an unified platform to access to online services and that platform is supported by the single sign-on external service implemented by CKB, a student/educator can directly sign up by simply selecting their school and being redirected to the school's platform. In this way CKB can automatically retrieve the relevant information about the person (name, surname, mail, if the user is a student or an educator) from the external service. In the last step of the registration the student has to provide also its GitHub nickname.

2. **Classic registration**
   The CKB platform offers also the classic registration, where an educator/student can sign up by providing their name, surname, mail, GitHub nickname, password and selecting if they are an educator. Students registrations are automatically approved while educator ones are manually reviewed by CKB staff.

2.2.2 Registered educator

Creation of a tournament
The system offers the possibility to create tournaments to all educators correctly registered and verified. The system only requires to insert the tournament's name and the subscription deadline. It will generate an unique ID for that tournament and it will send automatically an email notification to all students subscribed to the CKB platform.

Management of a tournament
The educator that created a tournament is able to manage its settings from the tournament management's page:

- enabling the possibility to other colleagues to post battles
- deleting the tournament
- removing subscribed students

Creation of a battle
The system allows educators that were granted permissions to post in a tournament to create battles inside the tournament and will assign them an unique ID. It requires to insert some information:

- the battle's name
- the CodeKata
- minimum and maximum number of students per team
- battle's programming language
- registration deadline
- final submission deadline
- set additional configurations for scoring (selecting the aspects to be automatically evaluated and if manual evaluation from educator is needed).

Management of a battle
The system allows the creator of a battle to manage battle's settings:

- deleting a battle
- changing maximum number of students per team
- modifying registration deadline
- changing final submission deadline
- removing a team

Manually evaluation of codes
Once a battle is finished, if its settings require a manual evaluation then the system allows the battle's creator to manually review all the teams' codes. Hence, the educator can attribute a personal score.

2.2.3 Registered student

Subscription to a tournament
The system allows registered students to see all the available tournaments and subscribe to them.

Joining a battle
The systems allows registered students subscribed to a specific tournament to see all the battles inside it and join them in different ways. When a user requests to join a battle, the system requires to input the number of team's members that will compete in that battle and the team's name. If the number is greater than 1, then the system will provide a code to be shared with the other colleagues. Instead, if a user requests to join a team, the system asks for the invite code.

2.2.4 Registered user

These are functions available to both registered student and educators.

Login
The system must allow registered students/educator to access the platform by providing the email and the password chosen at the registering phase. It will then shows options on the platform based on the user's role

View ranks
The system must allow users to check students' rankings in tournaments. It also must allow students subscribed to a tournament and educators that were granted permissions for a tournament to see teams' rankings in the battles of that tournament. Rankings are automatically built by the system by sorting scores.

2.3 User Characteristics

Unregistered Person
This is either a student or an educator that has some interests in using CKB platform. To start using the platform services they will have to go through the registration process.

Registered Student

This is a student that has completed the registration process and can access the platform and starts joining tournaments and battles to try to improve their programming skills and their position in rankings.

Registered Educator
This is an educator that has completed the registration process and that has been verified successfully by the staff. They can start to create and manage tournaments and battles to train and test their students.

2.4 Assumptions, Dependencies and Constraints

2.4.1 Domain assumptions

DA1 Both students and educators must have an internet connection

DA2 There exists an external service that CKB can use to send email notifications to users

DA3 There exists a SSO service that allows users to sign in with schools' credentials

DA4 Students must have a GitHub account

DA5 There is a GitHub API to create and pull repositories

DA6 Students will create a GitHub action to inform CKB about new pushes

DA7 Educators can change battles and tournaments' names only by deleting and recreating them.

DA8 Once a battle has been created its programming language cannot be changed

DA9 The student will use the same GitHub account that they have set in the registration process to work for battles

DA10 Students will program using the same programming language set in the specific battle

DA11 There exists an external service with APIs that CKB can use to do static analysis of codes

2.4.2 Mapping on domain assumptions

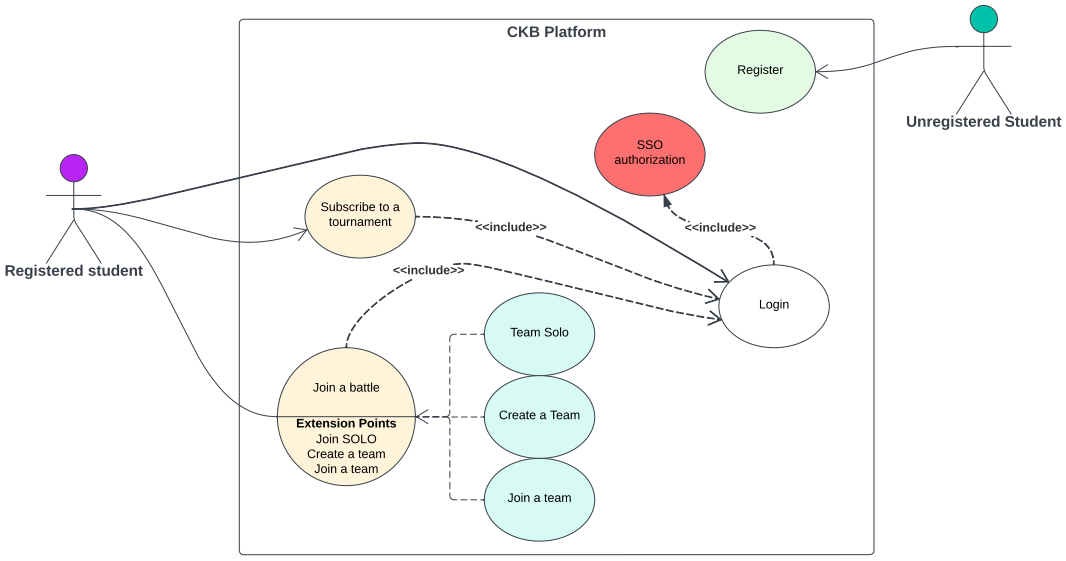
Here there is the mapping between goals and the domain assumptions that were just introduced.

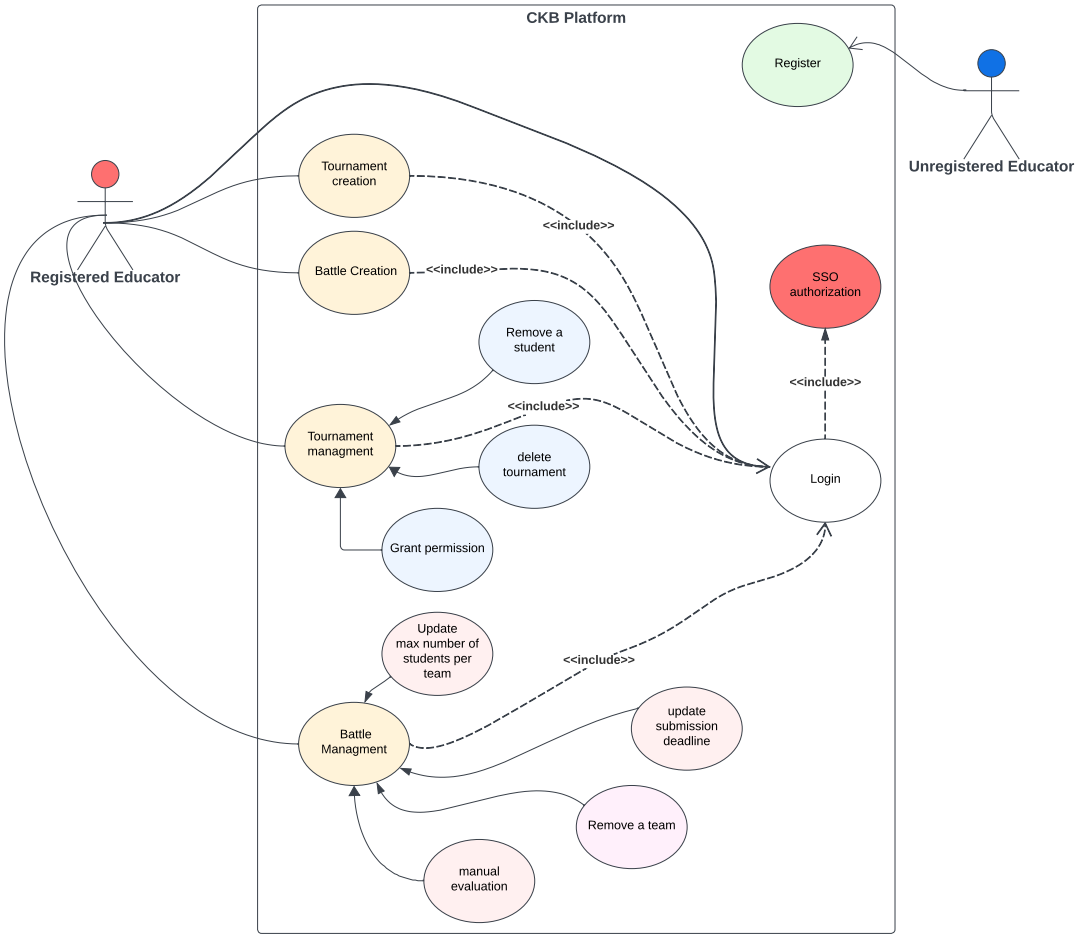
Goal	Domain assumptions
G1	DA1, DA3, DA4
G2	DA1, DA7
G3	DA1, DA5, DA7, DA8
G4	DA1
G5	DA1, DA9
G6	DA1
G7	DA1, DA6, DA9, DA10, DA11
G8	DA1
G9	DA1
G10	DA1, DA2

3 Specific Requirements

3.1 Use case diagram

Here firstly there is the use case diagram, an high level representation of what the system does and how the main actors use it:




CKB Platform
Register
Unregistered Educator
Registered Educator
Tournament creation
Battle Creation
<<include>>
<<include>>
SSO authorization
Remove a student
<<include>>
Tournament managment
delete tournament
Login
Grant permission
Update max number of students per team
<<include>>
Battle Managment
update submission deadline
Remove a team
manual evaluation

3.1.1 Use Cases

For every suggested use case, there is an underlying assumption that the platform dynamically presents users only with options they are permitted to access. Consequently, the exceptions don't include disallowed operations (e.g, creation of a team if the user is already in a team for the specific battle, the deletion of a tournament by an user that's not the creator, etc). Also, making sure that the inputs follow the specified pattern is managed from the client side (e.g. score should be between 0 and 100, deadline dates shouldnt be before current date, missing inputs etc.) and if the users manage to send a forged request with non coherent infos it will be discarded.

Table 1: **UC1: Educator registration**

Actors	Unregistered educator
Entry Condition	The educator opens the CKB platform
Event flow	1. The educator clicks on the button to register 2. The educator inserts the requested information (name, surname, mail, GitHub nickname and password) 3. The educator selects the "I'm an educator" flag 4. The educator confirms by clicking the confirmation button
Exit condition	The system processes the information and send them to the CKB staff in order to be approved. The system shows a waiting for approval success message and a confirmation email is sent to the educator through an external service
Exception	1. The Email provided is already in use. The system shows an email already registered error 2. The GitHub Nickname provided is already in use. The system shows a GitHub Nickname already registered error

Table 2: **UC2: Student Registration**

Actors	Unregistered Student
Entry Condition	The student opens the CKB platform
Event flow	1. The student clicks on the button to register 2. The student inserts the requested information (name, surname, mail, GitHub nickname and password) 3. The student confirms by clicking the confirmation button
Exit condition	The system processes the information, create the user. The system shows a success message and a confirmation email is sent to the student through an external service
Exception	1. The Email provided is already in use. The system shows an email already registered error 2. The GitHub Nickname provided is already in use. The system shows a GitHub Nickname already registered error

Table 3: **UC3: User login**

Actors	Registered Educator/Student
Entry Condition	The user opens the CKB platform
Event flow	1. The user clicks on the button to login 2. The user inserts login credentials 3. The user clicks on the confirmation button
Exit condition	The system checks the credentials. The system shows a success message and logs in the user
Exception	1. Either the mail or the password provided are wrong. The system shows an error message

Table 4: **UC4: User Single sign-on access (SSO)**

Actors	Educator/Student
Entry Condition	The educator opens the CKB platform
Event flow	1. The user clicks on the button to access the SSO service 2. The user completes the access steps on the external service in which they are redirected by the system 3. If it's the first access of the user they insert the GitHub username and confirm
Exit condition	The system checks the response from the external service and register the user if it's the first access. Otherwise the system logs in the user and shows a success message
Exception	1. The SSO login fails, the system shows an error message 2. The Github username is already in use. The system shows an error message

Table 5: **UC5: Tournament creation**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the main page with the tournaments' list
Event flow	1. The educator clicks on the button to create a new tournament 2. The educator inserts the tournament's name 3. The educator inserts the tournament's subscription deadline 4. The educator clicks on the button to submit
Exit condition	The system checks the tournament name. The tournament create the tournament (assigning a tournament ID) and shows a success message. An email is sent to the students through an external service to inform them about the new tournament.
Exception	The tournament's name inserted already exists. The system shows an existing name error

Table 6: **UC6: Battle creation**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of a tournament in which they are allowed to post battles
Event flow	1. The educator clicks on the button to create a new battle 2. The educator fills the form with the battle's name, code kata, the minimum and maximum number of students per team, the battle's programming language, the registration deadline, the final submission deadline and set additional configurations for scoring (selecting the flags on the aspects of teams' codes to be automatically evaluated and if manual evaluation from the educator is needed) 3. The educator submits the form
Exit condition	The system checks all the information and shows a success message. The system create the battle (assigning a battle ID). An email is sent to the students subscribed to a tournament through an external service to inform them about the new battle.
Exception	The battle's name inserted already exists. The system shows an existing name error and the educator has to insert a new one

Table 7: **UC7: Management of a tournament - Removing a student**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of a tournament they created
Event flow	1. The educator selects the option to remove subscribed students from the tournament 2. The educator selects one or more students from the subscribed users' list prompted by the system 3. The educator clicks on the confirm button
Exit condition	The system removes the selected student/s from the tournament, from the rankings and from all the battles' teams The system shows a success message to the educator and through an external mail provider notifies the removed students
Exception	No exceptions

Table 8: **UC8: Management of a tournament - Deleting the tournament**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of a tournament they created
Event flow	1. The educator selects and confirms the option to delete the tournament
Exit condition	The system deletes the tournament and all the related battles/teams/rankings. The system shows a success message to the educator and through an external mail provider notifies the students that were subscribed to the tournament
Exception	No exceptions

Table 9: **UC9: Management of a tournament - enabling other colleagues to post battles**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of a tournament they created
Event flow	1. The educator selects the option to grant the option to other colleagues to post battles 2. The educator inserts the other colleague's email 3. The educator submits the request
Exit condition	The system checks the inserted email and enables the inserted educator to post battles in that tournament. The system shows a success message to the educator and through an external mail provider notifies the new allowed educator
Exception	1. The new educator's email is not registered. The system shows a messages indicating that the email provided doesn't refer to an exiting registered educator 2. The selected educator is already enabled to post battles in that tournament. The system shows a messages indicating that the new educator is already allowed to post battles

Table 10: **UC10: Management of a battle - deleting a battle**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is either in a battle they created or in a battle of a tournament they created
Event flow	1. The educator clicks on the button to delete the battle confirms the choice
Exit condition	The system deletes the battle and all the related teams/rankings and shows a success message to the educator. The system through an external mail provider notifies the students that were in a team in the deleted battle
Exception	No exceptions

Table 11: **UC11: Management of a battle - changing minimum and maximum number of students per team**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of one of their battles
Event flow	1. The educator selects the button to change battle's settings 2. The educator modifies the current minimum and maximum number of students per team 3. The educator confirms the update
Exit condition	The system updates the new minimum and maximum number of students per team for that battle and shows a success message to the educator. The system through an external mail provider notifies the students that were in a team in the updated battle
Exception	1. There is at least one team in the battle with number of members greater than the new maximum number. The system shows a message to remove first the team/s and retry. 2. There is at least one team in the battle with number of members less than the new minimum number. The system shows a message to remove first the team/s and retry.

Table 12: **UC12: Management of a battle - modifying registration deadline**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of one of their battles
Event flow	1. The educator selects the button to change battle's settings 2. The educator modifies the current registration deadline 3. The educator confirms the update
Exit condition	he system updates the new registration deadline for that battle and shows a success message to the educator. The system through an external mail provider notifies the students that were in a team in the updated battle
Exception	No exceptions

Table 13: **UC13: Management of a battle - modifying final submission deadline**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of one of their battles
Event flow	1. The educator selects the button to change battle's settings 2. The educator changes the current final submission deadline date 3. The educator confirms the update
Exit condition	The system updates the new final submission deadline for that battle and shows a success message to the educator. The system through an external mail provider notifies the students that were in a team in the updated battle
Exception	No exceptions

Table 14: **UC14: Management of a battle - Removing a team**

Actors	Registered Educator
Entry Condition	The educator is logged into the CKB platform and is on the page of one of their battles
Event flow	1. The educator selects the button to change battle's settings 2. The educator selects the option to remove a subscribed team 3. The educator selects one or more teams from the subscribed teams' list prompted by the system 4. The educator confirms the update
Exit condition	The system removes the selected team/s from the battle and from the the battle's rank and shows a success message to the educator. The system through an external mail provider notifies the students that were in the deleted team
Exception	No exceptions

Table 15: **UC15: Subscribing to a tournament**

Actors	Registered Student
Entry Condition	The student is logged into the CKB platform and is on the tournaments' list page
Event flow	1. The students selects the tournament they want to subscribe to 2. The student clicks the button to subscribe to that tournament
Exit condition	The system subscribes the student to the selected tournament and shows a success message to the student. The system through an external mail provider notifies the student about the successful subscription
Exception	The subscription deadline has already passed. The system shows a a subscription period is over message

Table 16: **UC16: Joining a battle - Creating a solo team**

Actors	Registered Student
Entry Condition	The student is logged into the CKB platform and is on a subscribed tournament's page seeing the list of all available battles
Event flow	1. The student selects the battle they want to join 2. The student clicks the button to join the battle 3. The student inserts the team's name 4. The student inserts "1" as the number of team's members 5. The student confirms
Exit condition	The system checks the information and adds the team (assigning an ID) to the selected battle and shows a success message. The system through an external mail provider notifies the student about the successful team's creation
Exception	1. The team's name already exists. The system shows an already existing team's name error message 2. The registration deadline has already passed. The system shows a a registration period is over message

Table 17: **UC17: Joining a battle - Creating a Team**

Actors	Registered student
Entry Condition	The student is logged into the CKB platform and is on a subscribed tournament's page seeing the list of all available battles
Event flow	1. The student selects the battle they want to join 2. The student clicks the button to join the battle 3. The student inserts the team's name 4. The student inserts the number of team's members 5. The student confirms
Exit condition	The system checks the information and adds the team (assigning it an ID) to the selected battle. The system shows a success message with a generated invite code to share with other colleagues The system through an external mail provider notifies the student about the successful team's creation and include the invite's code in the email
Exception	1. The team's name already exists. The system shows an already existing team's name error message 2. The registration deadline has already passed. The system shows a a registration period is over message

Table 18: **UC18: Joining a battle - Joining via an invite code**

Actors	Registered student
Entry Condition	The student has an invite code and is logged into the CKB platform and is on a tournament page seeing the list of all available battles
Event flow	1. The student selects the battle they want to join 2. The student clicks the button to join the battle 3. The student inserts the team's code 4. The student confirms
Exit condition	The system checks the information and adds the student to the corresponding team and shows a success message to the student. The system through an external mail provider notifies the student about the successful team's join
Exception	1. The maximum number of team's members have been already reached. The systems shows a number of members exceed the allowed limit error 2. The invite code is wrong. The system shows a invite code error

Table 19: **UC19: Manually evaluation of codes**

Actors	Registered Educator
Entry Condition	1. A battle has just finished and for it it's enabled the option to manually evaluate the code by educators 2. Educator is logged in and is on the battle's page
Event flow	1. The educator selects the button to manually evaluate the teams' codes 2. The educator set a score in the field next to the code of each team 3. The educator confirms
Exit condition	The system processes and adds the scores to the related teams for the temporary team's ranking. The system shows a success message to the educator
Exception	No exceptions

Table 20: **UC20: View tournament's rank**

Actors	Registered Student/Educator
Entry Condition	The educator/student is logged in and is on a tournament's page
Event flow	1. The educator/student clicks on the button to view the tournament's rank
Exit condition	The system shows the students' rank for that tournament
Exception	The tournament doesn't have any completed battle. The system shows a rank not available error

Table 21: **UC21: View battle's rank**

Actors	Registered Student/Educator
Entry Condition	The educator/student is logged in and is on a battle's page
Event flow	1. The educator/student clicks on the button to view the battle's rank
Exit condition	The system shows the teams' rank for that battle
Exception	None of the teams has submitted a solution. The system shows a rank not available error

3.2 Use case sequence diagrams

In this section there is the mapping of each use case with an use case sequence diagrams. Some different use cases are condensed into a single sequence diagrams because the action performed is basically the same but it was split for a better comprehensibility.

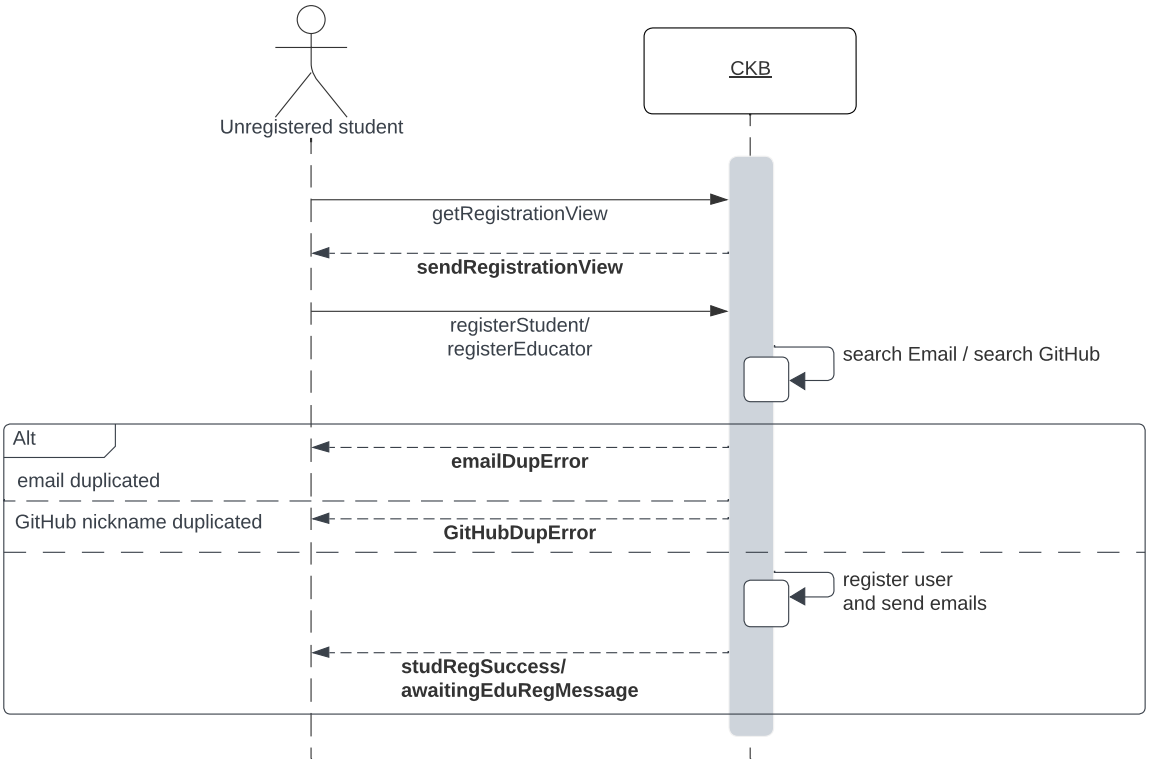


Figure 1: Sequence diagram for UC1 / UC2

Same process for the registration of students/educators. The difference is that the educator send the eduator flag to true and that the system in the educator registration must send a mail to the staff.

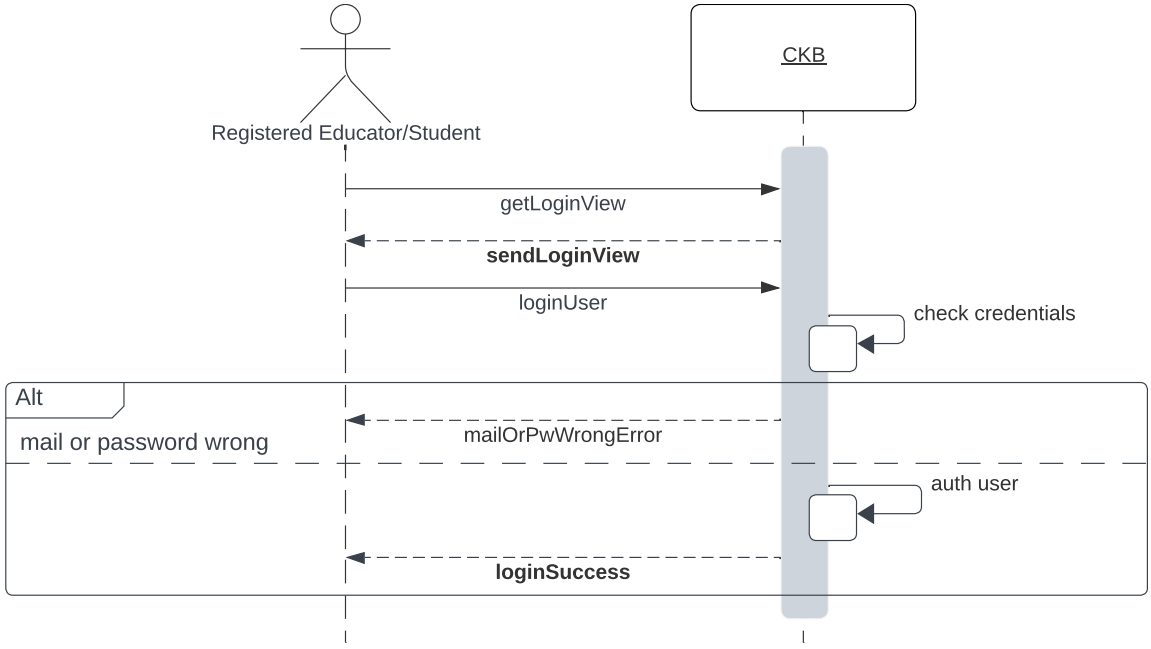


Figure 2: Sequence diagram for UC3: User login

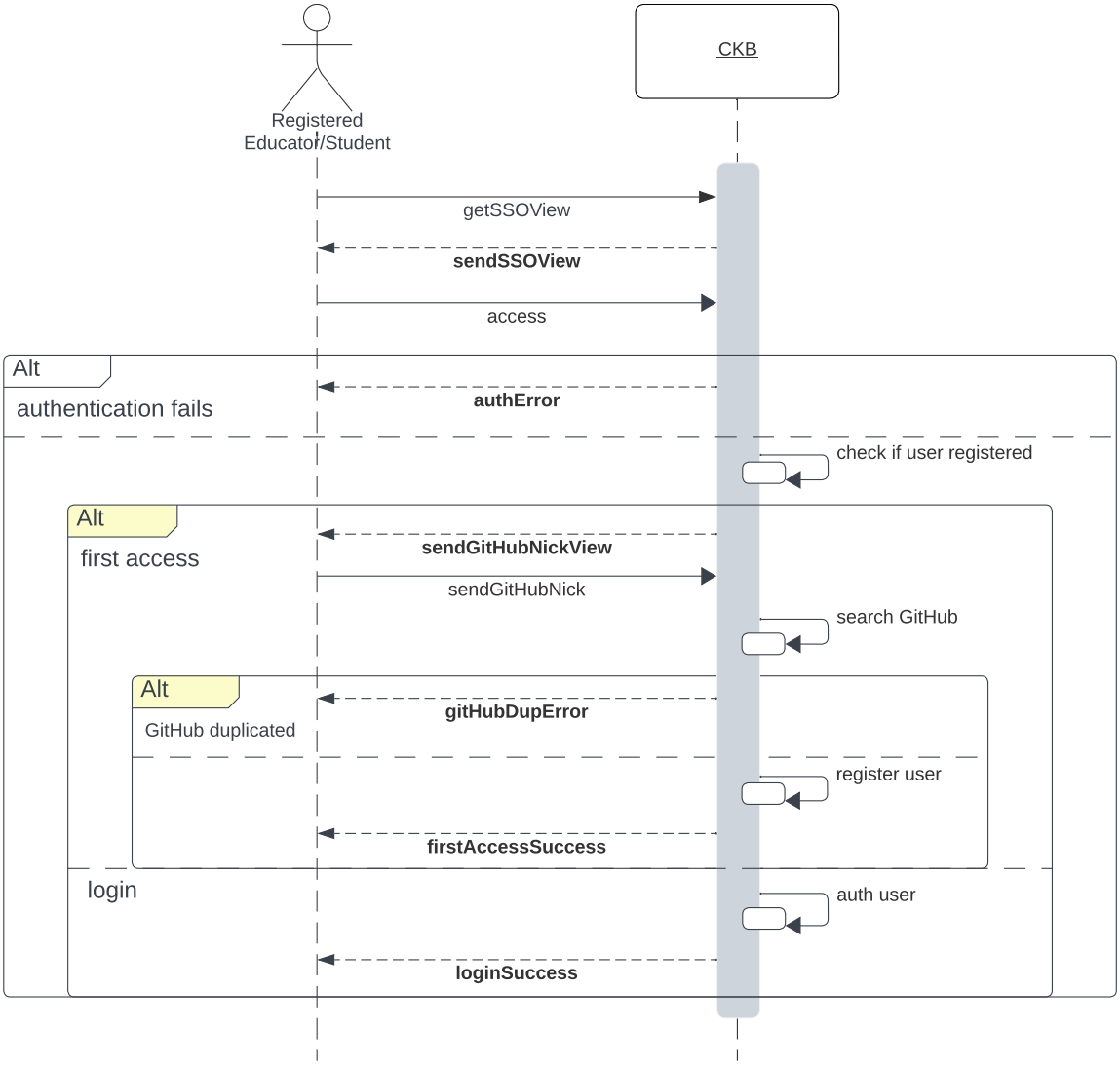


Figure 3: Sequence diagram for UC4: User Single sign-on access (SSO)

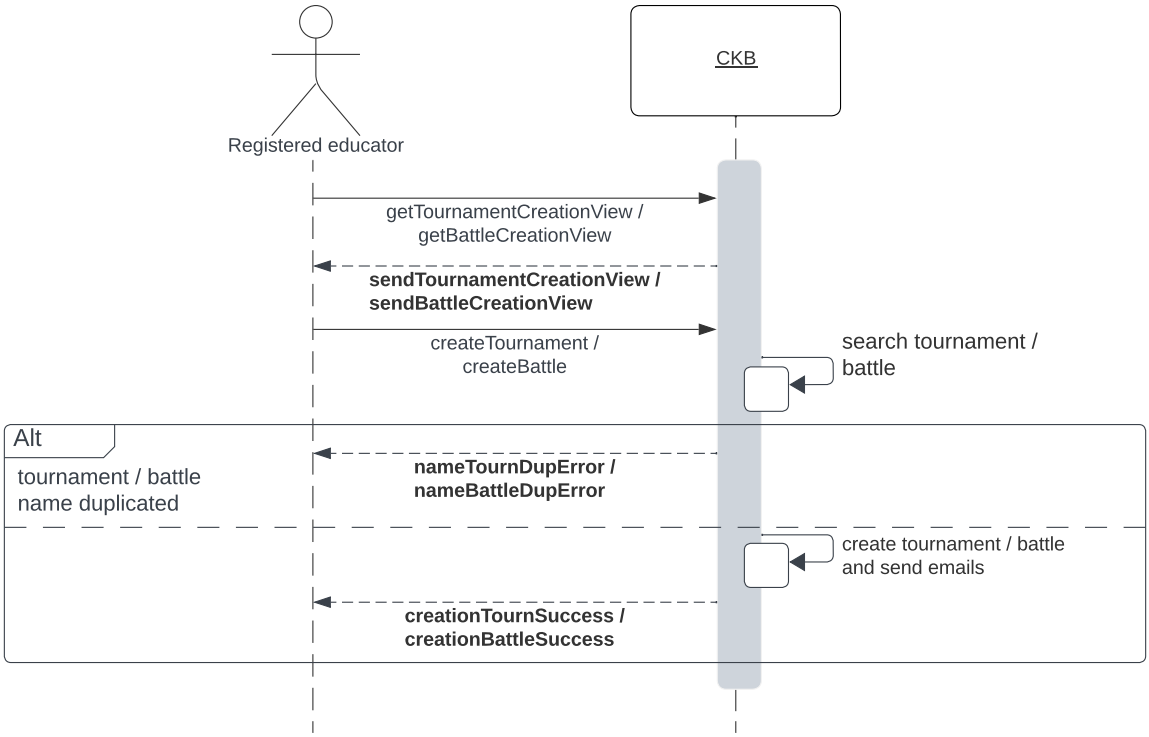


Figure 4: Sequence diagram for UC5 / UC6

Same processes for the creation of tournaments and battles (different requests and data to be sent)

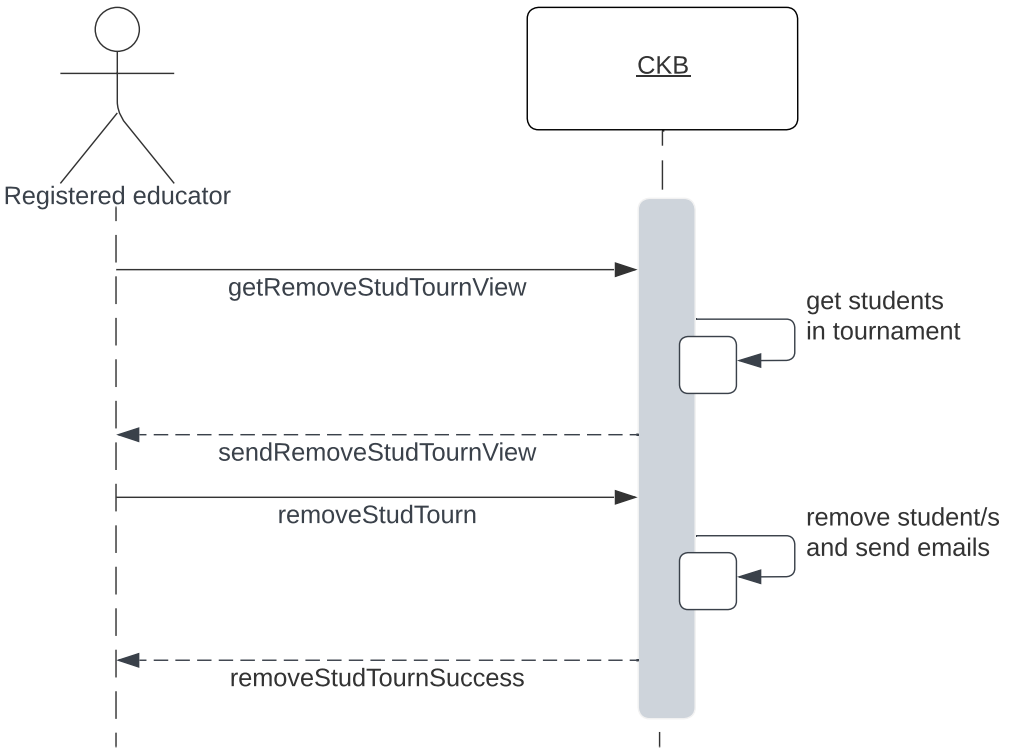


Figure 5: Sequence diagram for UC7: Management of a tournament - Removing a student

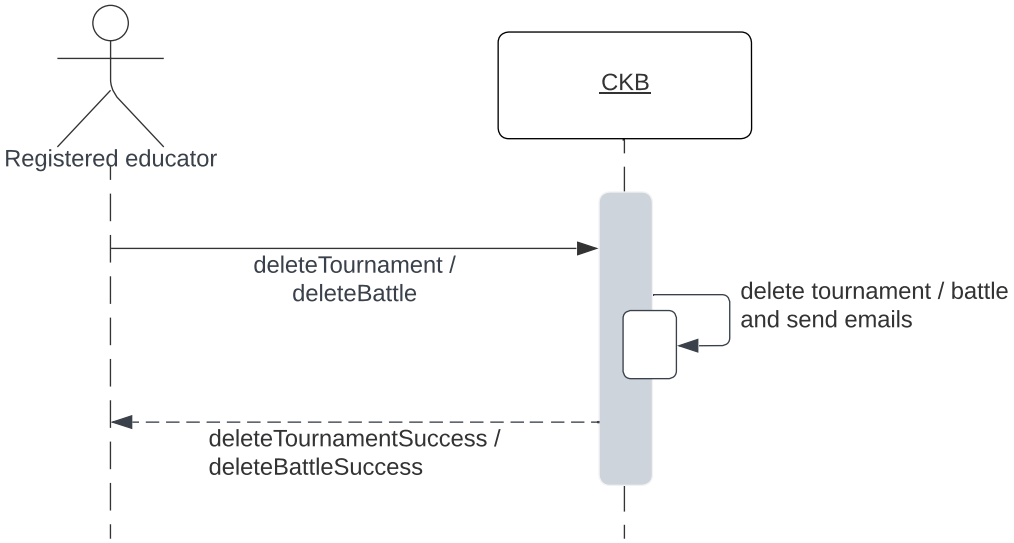


Figure 6: Sequence diagram for UC8 / UC10

Same processes for the deletion of tournaments and battles (different requests and data to be sent)

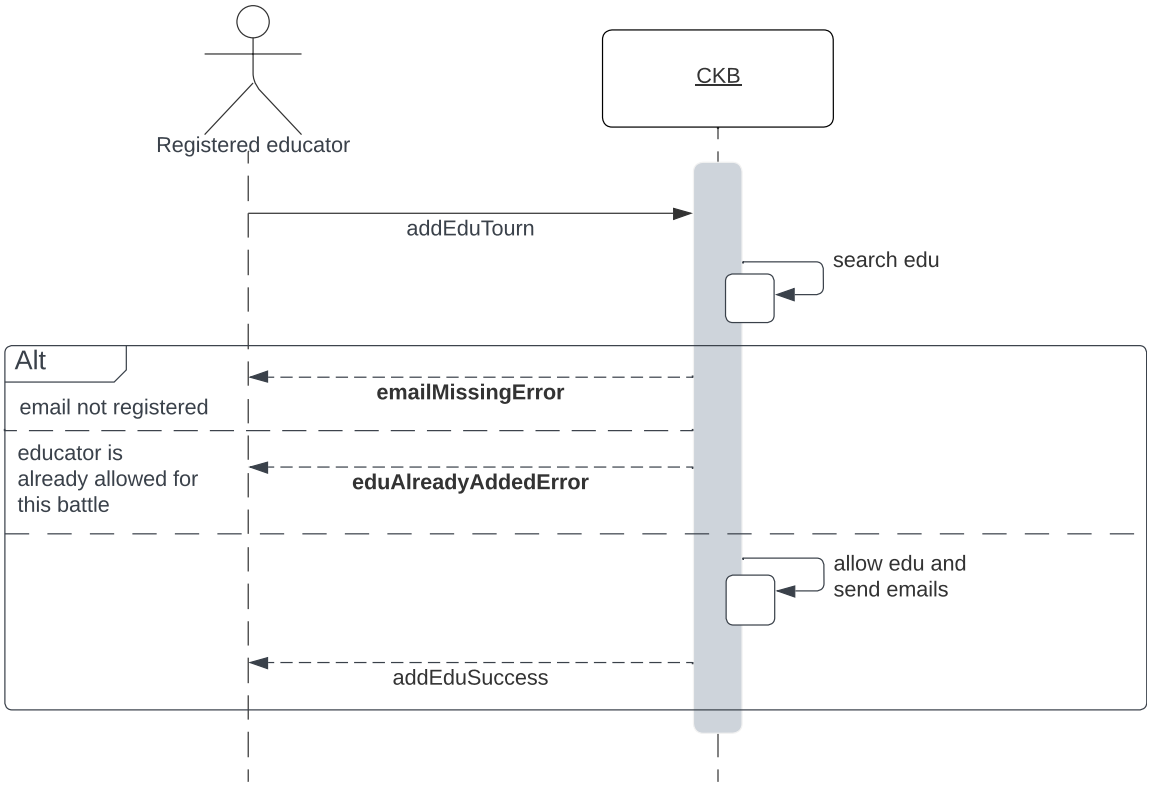


Figure 7: Sequence diagram for UC9: Management of a tournament - enabling other colleagues to post battles

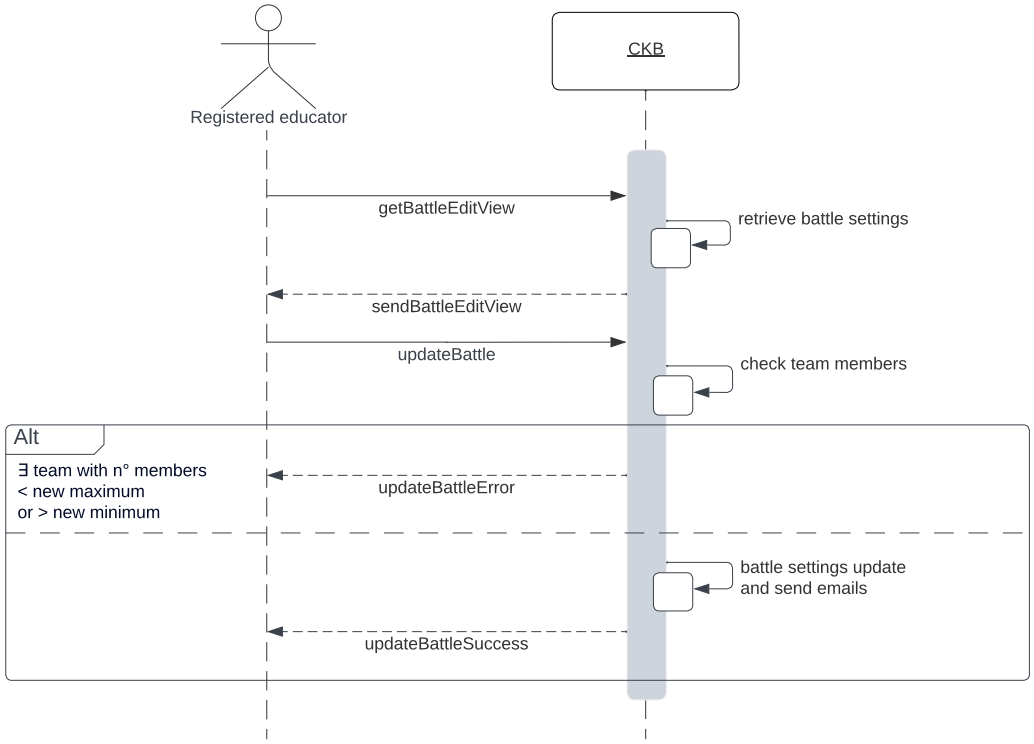


Figure 8: Sequence diagram for UC11 - UC12 - UC13

Here (figure 8) this single sequence diagram was split in 3 uses cases for comprehensibility but the action to be seen in the diagram is the same as changing the min/max number of students per team or modifying the final submission / registration deadline involves the same request to the platform. The only difference is that the exception is meant only for the UC11 use case.

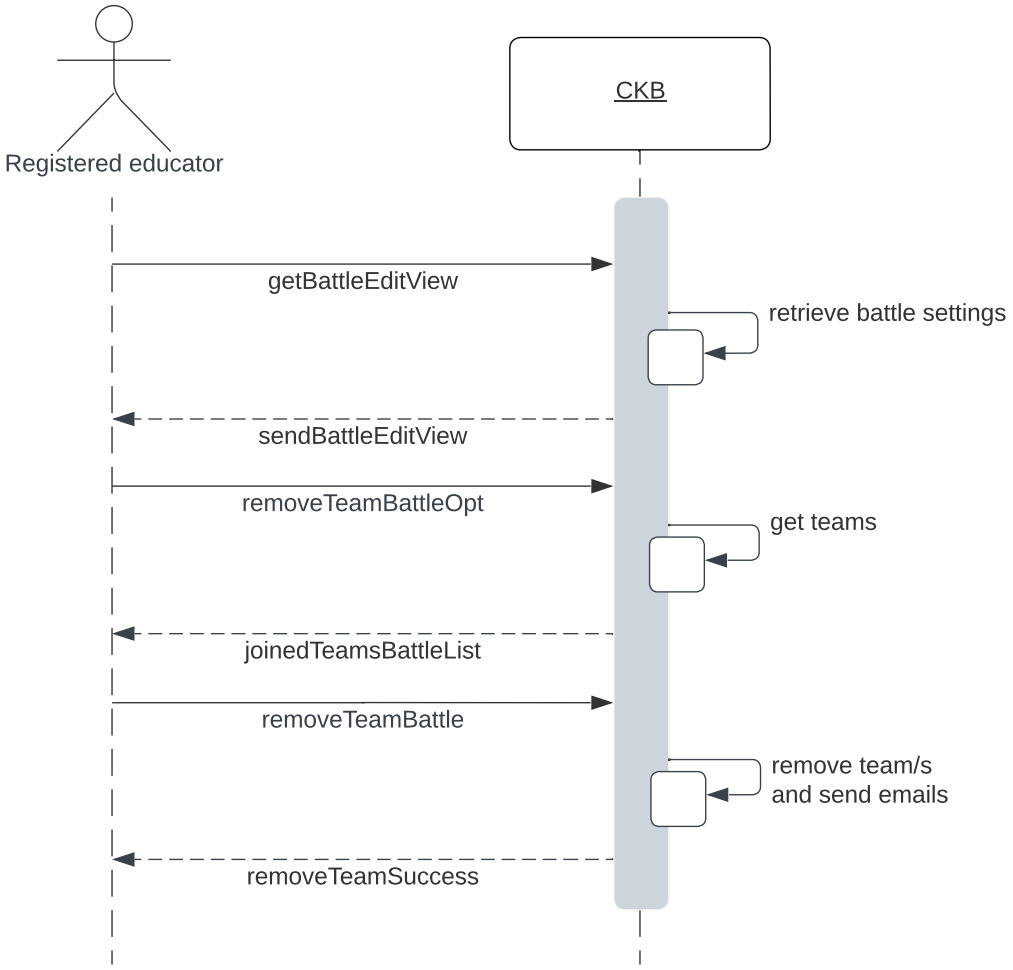


Figure 9: Sequence diagram for UC14: Management of a battle - Removing a team

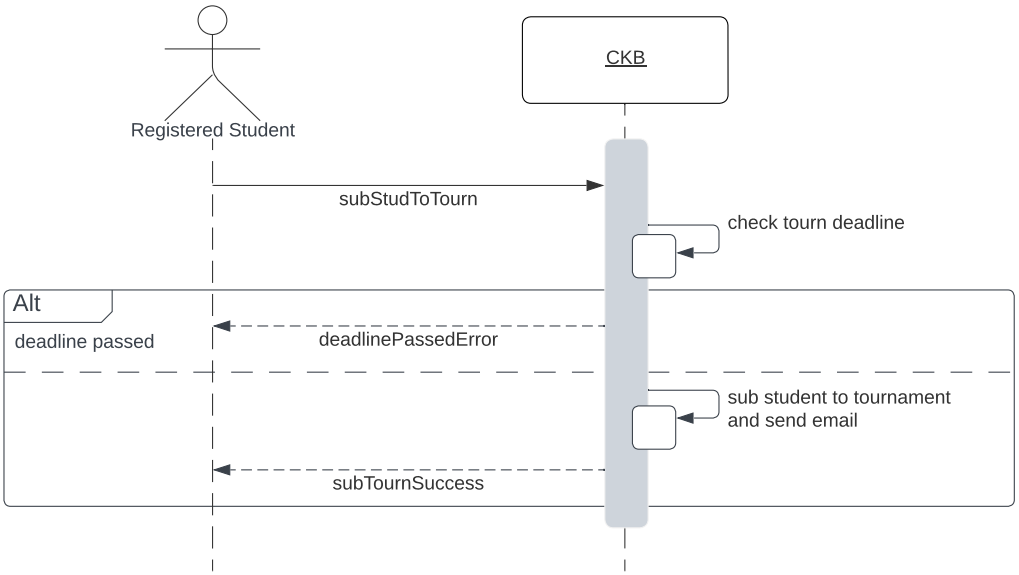


Figure 10: Sequence diagram for UC15: Subscribing to a tournament

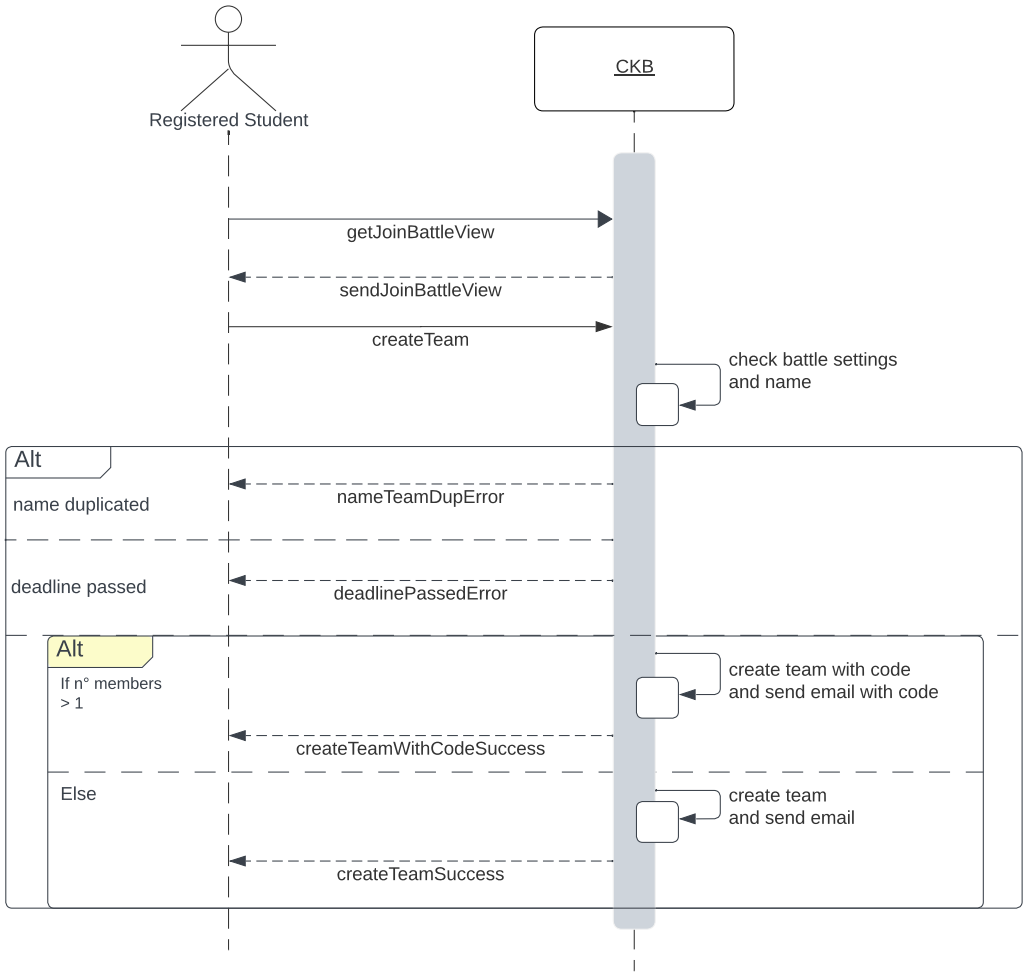


Figure 11: Sequence diagram for UC16 - UC17

Here (11) the UC16 and UC17 are merged into the same sequence diagram as both use cases represent the action of creating a team and involve the same request to the platform. The difference is in the internal alt where if it's not a solo team there is also to introduce the invite code.

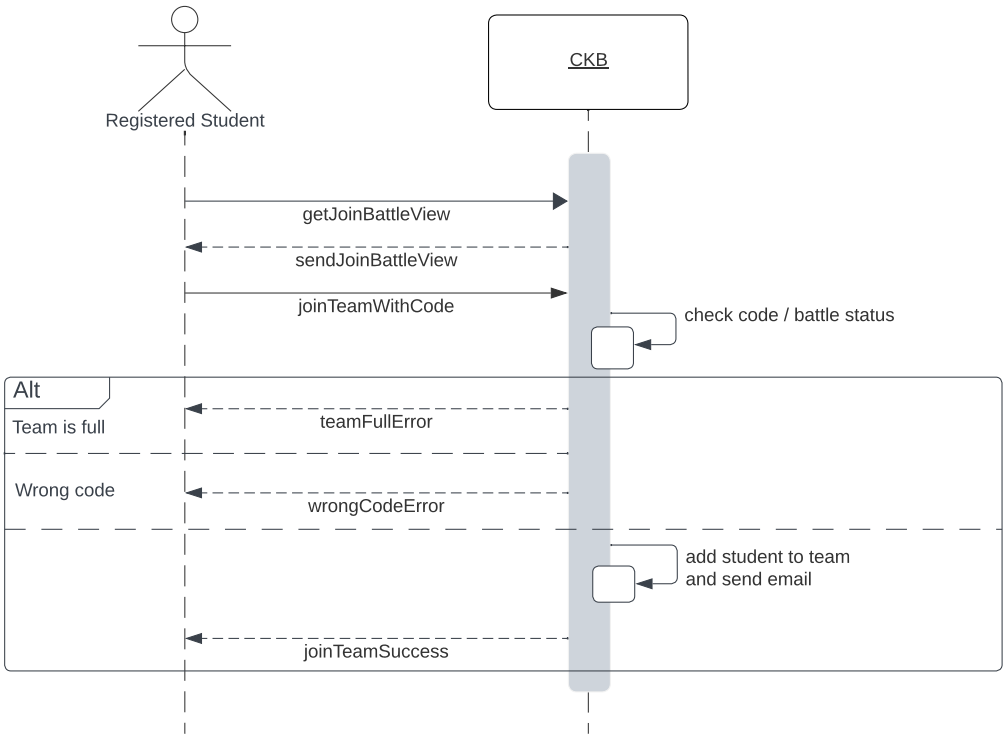


Figure 12: Sequence diagram for UC18: Joining a battle - Joining via an invite code

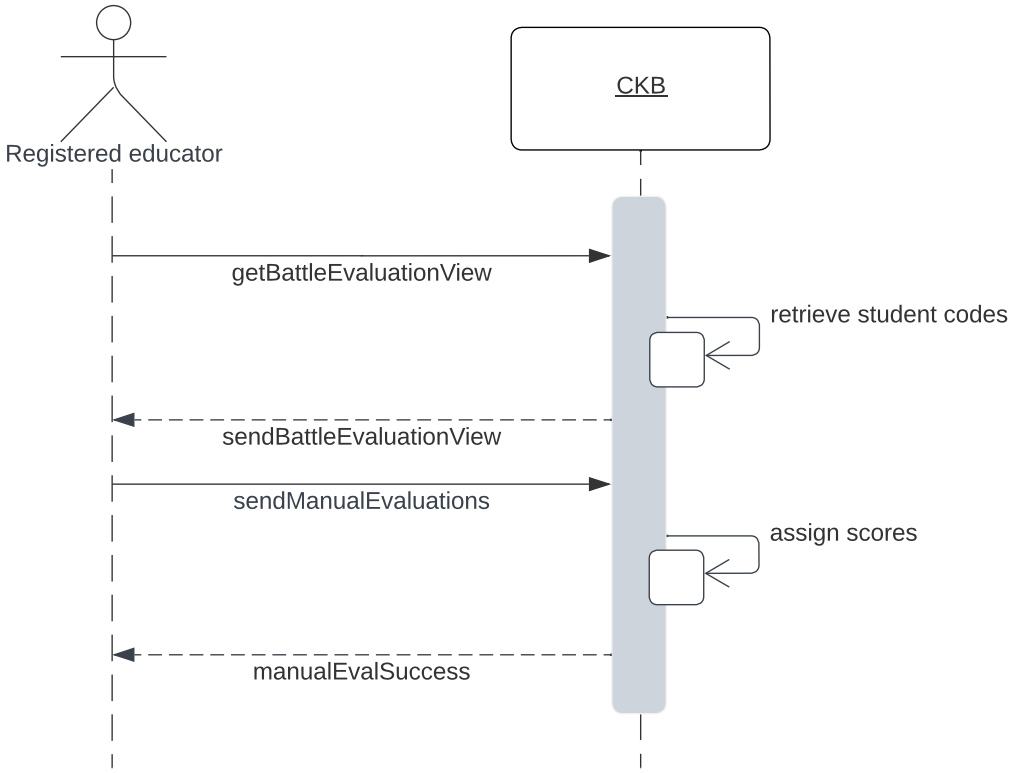


Figure 13: Sequence diagram for UC19: Manually evaluation of codes

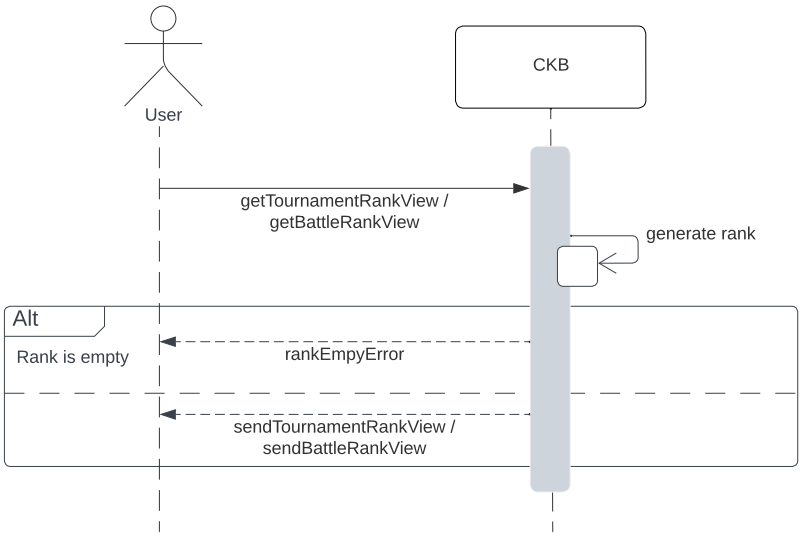


Figure 14: Sequence diagram for UC20 - UC21

Here (figure 14) UC20 and UC21 are merged because the logic is the same, the only difference is in the name of the requests.

3.3 Functional requirements

R1 The system must allow unregistered student to sign up

R2 The system must allow unregistered educator to sign up

R3 The system must notify the user about the successful registration via an external mail service

R4 The system must allow educators/students to access the platform via an external SSO service

R5 The system must allow registered educator to login

R6 The system must allow registered educator to login

R7 The system must allow educators to create tournaments (setting name and subscription deadline).

R8 The system must allow the creator of a tournament to delete it and to remove students from it, students affected must be notified via an external mail service

R9 The system must allow the creator of a specific tournament to invite other colleagues to post battles in it and notify them via an external mail service

R10 The system must allow users to see all the available tournaments

R11 The system must allow users to see tournaments' ranks

R12 The system must notify registered students about the creation of a new tournament via an external mail service

R13 The system must allow registered students to subscribe to a tournament. Students must be notified of the successful subscription via an external email service

R14 The system must allow the creator of a tournament or the educators that were granted the permission to post in it, to create battles in it (this includes setting name, battle's deadline, the aspects of the codes to be evaluated automatically, the min and max members per team and uploading the CodeKata)

R15 The system must be able to create a GitHub repository for each battle

R16 The system must allow the creator of a specific battle to delete it, to change minimum and maximum number of students per team, to modify the registration deadline, to change the final submission deadline, to remove a team. Students affected must be notified via an external mail service

R17 The system must allow subscribed students to a tournament and educators that were granted permissions for the tournament to see all the available battles in that tournament

R18 The system must notify students about the creation of a new battle in a tournament they are subscribed for via an external mail service

R19 The system must allow subscribed students to a tournament and educators that were granted permissions for the tournament to see ranks of battles in that tournament

R20 The system must allow subscribed students to a tournament to create a team for a battle in that tournament. Student must be notified of the successful join via an external email service

R21 The system must allow subscribed students to a tournament to join a team for a battle in that tournament by providing the team's invite code. Student must be notified of the successful join via an external email service

R22 The system must allow the creator of a battle to evaluate team's codes for that battle if the battle's settings expect it

R23 The system must be able to pull repositories from GitHub

R24 The system must be able to automatically evaluates teams' codes after each push on GitHub before the submission deadline based on the number of test cases they pass, the timeliness with respect to the deadlines and the set quality aspects with static analysis tools

R25 The system must have an API to be able to be informed by a GitHub Action of new students' pushes

R26 The system must notify students that joined a battle that the final rank for the battle is available via an external mail service

3.3.1 Mapping use cases

Use case	Requirements
UC1	R2, R3
UC2	R1, R3
UC3	R5, R6
UC4	R4
UC5	R7, R12
UC6	R14, R18
UC7	R8
UC8	R8
UC9	R9
UC10	R16
UC11	R16
UC12	R16
UC13	R16
UC14	R16
UC15	R13
UC16	R20
UC17	R20
UC18	R21
UC19	R22
UC20	R11
UC21	R19

3.3.2 Mapping goals

Goal	Requirements
G1	R1, R2, R4, R5, R6
G2	R7, R8, R9
G3	R14, R15, R16
G4	R10, R13
G5	R17, R20, R21
G6	R22, R23
G7	R23, R24, R25
G8	R10, R11
G9	R17, R19
G10	R3, R12, R18, R26

3.4 External Interface Requirements

3.4.1 User Interfaces

Even if the platform is built having in mind a target of programming students the interface of CKB should be as simple and clear as possible to offer them an immediate visualization of their progresses. Educators will find a well-organized UI that allows them to efficiently create and manage tournaments. The system offers also a simple yet powerful battle creation tool. Educators can easily set a wide range of parameters including evaluation criteria and upload the Code Kata. The goal is to facilitate the creation of diverse and engaging programming challenges for students. Since the platform will be used on a browser it is compatible with all devices as it easily adapts to different screens sizes.

3.4.2 Hardware Interfaces

CKB is designed to simply access and manage challenges and asses progresses so, even if a computer is recommended during the programming on GitHub, for CKB even a simple smartphone with internet access is enough.

3.4.3 Software Interfaces

Since CKB implements the single sign-on option, it will interacts with external services that provide this feature to sign-up and login using a secure and unified authentication process. The system will need to be able to interact with an email service API, in order to send all the needed notifications. It also will need to interact with GitHub APIs to retrieve the students' sources and with an API that make static analysis to evaluate them.
In addition, CKB will offer a simple API in order to be able to receive updates about students' pushes through GitHub Actions.

3.4.4 Communication Interfaces

The CKB exploits the internet connection for the communication to and from users' devices and to and from the external services. The communication from CKB to GitHub APIs can be asynchronous, because CKB will retrieve the updated source code only when it receives a notification from the defined GitHub Action when a new code update is pushed to a specific repository. It is a more efficient way to handle updates and reduce unnecessary resource usage.

3.5 Performance Requirements

There are no specific measurable performance requirements for the system to deliver its services correctly to end users. However, a system that responds more promptly to user queries is always preferred over a slower one. There shouldn't be noticeable delays to users' requests as none of the direct ones involve a communication with an external API.

3.6 Design Constraints

3.6.1 Standards Compliance

- GDPR for end user data protection in CKB system
- HTTP/HTTPS with RESTful Design for communications between GitHub and CKB API

3.6.2 Hardware Limitations

It's a web based platform with no particular computations requirements so the only requirement is a device with internet connection enabled.

3.7 Software System Attributes

3.7.1 Reliability

To ensure a positive user experience, the system must exhibit reliability in 99.90% of cases. CKB's API exposed to GitHub must receive all updates about pushes that are crucial for the correctness of the platform. Achieving this reliability can involve implementing redundancy in components, building process resilience, and conducting regular maintenance.

3.7.2 Availability

The system is anticipated to operate around the clock, similar to standard websites. The aim is to minimize average downtime, as any unavailability during a user's need for subscribe to a tournament / join a battle could potentially strand them without the means to complete the challenge. Additionally, the system's availability is contingent on the accessibility of external API, emphasizing the interconnected nature of the system with these external services. Also the internal API unavailability used by GitHub could potentially cause a loss of pushes.

3.7.3 Security

The system stores personal data such as end users' emails, name, surname and GitHub nickname. It is crucial for the entire system to manage data securely, such as storing hash-encrypted passwords for both students and educators in the Database Management Systems (DBMSs). For what concerns the internal API exposed, in order to guarantee a certain level of security is used:

- HTTPS: Ensure that the API is accessible over HTTPS rather than HTTP.

3.7.4 Maintainability

The system must be easily maintainable and adaptable for future updates and extensions. To ensure this, the system's architecture should prioritize modularity, minimizing excessive coupling between modules. Incorporating software engineering patterns is a proactive approach to meeting this requirement from the outset.

3.7.5 Portability

The CKB application will be exclusively accessible from a browser. This browser-based accessibility enhances the application's versatility, allowing users to seamlessly interact with the platform regardless of their device or operating system. This approach eliminates the need for specific installations, providing users with the convenience of accessing CKB from any web-enabled device, be it a desktop computer, laptop, tablet, or even a smartphone.

4 Formal analysis

4.1 Alloy code

In this section, we provide a comprehensive analysis of the system utilizing Alloy. The aim is to underscore specific constraints that need to be enforced. Moreover, we aim to demonstrate the satisfiability of the system model.

The following code represents our modeling of the CKB system, encompassing every class in accordance with the class diagram's structure. Through this model, we formally validate the robustness and coherence of our system construction approach. This signifies that the relationships between classes and the imposed constraints yield a rational and beneficial outcome. This validation serves to justify the modeling efforts undertaken.

```
sig Email {}
sig TournamentName {}
sig BattleName{}
sig GitHubNick{}


// date
sig Date {
  date : one Int } {
  date >= 0
}

sig Text {}

// abstract signature.
abstract sig Person {
  name: one Text,
  surname: one Text
}

// An abstract signature extending Person with attributes
// Every User must have an email and a gitHubNick
abstract sig User extends Person{
  email : one Email,
  gitHubNick : one GitHubNick,
}

// Signatures extending User
sig Educator extends User {}
sig Student extends User {
  subscribed: set SubscriptionToTournament
}
```

```
sig Tournament{
    name: one TournamentName,
    creator: one Educator,
    granted: set Educator,
    subscribed: set SubscriptionToTournament,
    regDeadline: one Date
}

sig Battle{
  name: one BattleName,
  tournament: one Tournament,
  creator: one Educator,
  regDeadline: one Date,
  subDeadline: one Date,
  maxTeamMember: Int,
  minTeamMember: Int
}{
  maxTeamMember >= minTeamMember
}

// With the constraint that the creator of a team is a member itself,
// members must be students, creation of a team (and so joining a battle)
// must happen before the battle registration deadline, and
// the team's last push must occur before the battle submission deadline
sig Team{
  creator: one Student,
  members: some Student,
  battle: one Battle,
  teamCreationDate: one Date,
  teamLastPush: one Date
}{
  battle.regDeadline.date > teamCreationDate.date
  teamLastPush.date < battle.subDeadline.date
}

// Student subscription must occur before the registration
//deadline for a tournament
sig SubscriptionToTournament{
  tournament: one Tournament,
  student: one Student,
  subDate: one Date
}{
  tournament.regDeadline.date > subDate.date
}
```

```
// SubscriptionToTournament must always be present when
//a user subscribes to a tournament
fact subscriptionToTournamentMustExist {
  all t: Tournament, s: Student, d: Date |
    (s in t.subscribed.student && d =
      s.subscribed.tournament.regDeadline) implies
    one st: SubscriptionToTournament |
      st.tournament = t && st.student = s && st.subDate = d
}

// No users with the same email
fact uniqueEmails{
   all disj u1, u2: User | u1.email != u2.email
}

// No users with the same GitHub nickname
fact uniqueGitHubNick{
  no disjoint u1, u2: User | u1.gitHubNick = u2.gitHubNick
}

// No tournaments with the same name
fact uniqueTournName{
  no disjoint u1, u2: Tournament | u1.name = u2.name
}

// No battles with the same name
fact uniqueBattleName{
  no disjoint u1, u2: Battle | u1.name = u2.name
}

// All emails are connected to users
fact allEmailConnected{
  all e: Email | one u: User | e = u.email
}

// All GitHub nicknames are connected to users
fact allGitHubConnected{
  all e: GitHubNick | one u: User | e = u.gitHubNick
}

// Creators of a battle must be either the creator or
// an educator with granted permission
fact creatorsInGranted {
  all b: Battle | b.creator in (b.tournament.granted + b.creator)
}
```

```
// Every tournament must have a creator
fact everyTournamentHasCreator {
  all t: Tournament | one c: Educator | c = t.creator
}

// A student cannot be part of a team or create it if
//not subscribed to the related tournament
fact studentMustBeSubscribed {
  all t: Team |
    let s = t.creator + t.members|
      s in t.battle.tournament.subscribed.student
}

fact studentMustBeSubscribedForJoiningBattle {
  all t: Team |
    let s = t.members |
      s in t.battle.tournament.subscribed.student
}

// Every battle must be associated with a tournament
fact everyBattleHasTournament {
  all b: Battle | one t: Tournament | t = b.tournament
}

// The number of students per team must be at most the maxTeamMember
fact maxStudentsPerTeam {
  all t: Team | #t.members <= t.battle.maxTeamMember + 1
}

// The number of students per team must be at least the minTeamMember
fact maxStudentsPerTeam {
  all t: Team | #t.members >= t.battle.minTeamMember + 1
}

// A student (including the creator) is not in more than
//1 team for a given battle for a given tournament
fact oneStudentOneTeamPerBattlePerTournament {
  all t: Tournament, b: Battle |
    all s: Student |
      lone team: Team | (s in team.members or s = team.creator)
        && team.battle = b && b.tournament = t
}

// the team's creator must be a member itself
assert creatorIsMember {
  all t: Team | t.creator in t.members
```

```
}


pred show() {}

pred world1 {
  #Student = 3
  #Educator = 1
  #Tournament = 1
  #Battle = 1
  #Team = 2

}

run world1 for 10

// check if exists a team with two members
pred checkTwoStudentsInSameTeam {
  some t: Team | some s1, s2: Student | s1 in t.members && s2
    in t.members && s1 != s2
}

run checkTwoStudentsInSameTeam for 3 but exactly 2 Student, 1 Team
```

4.2 Resulting world

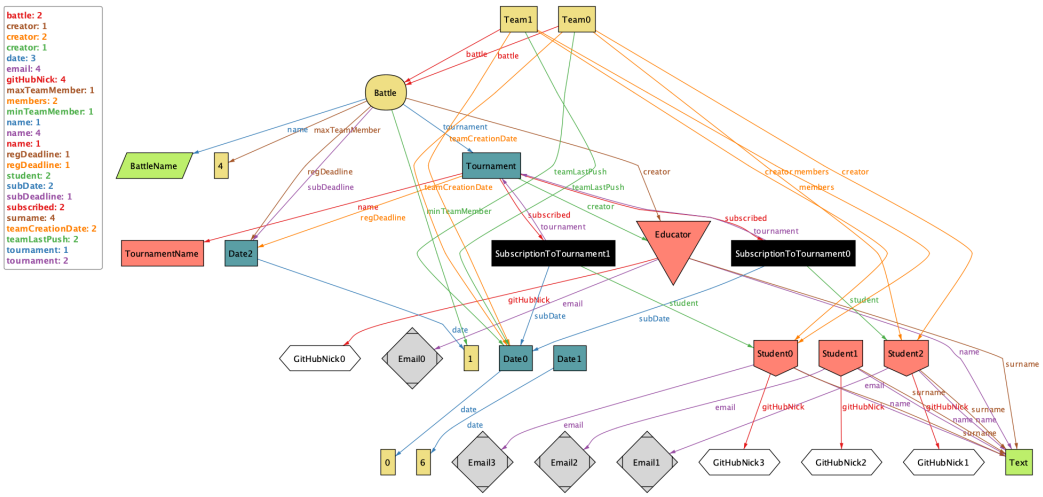


Figure 15: Resulting world1 - Alloy

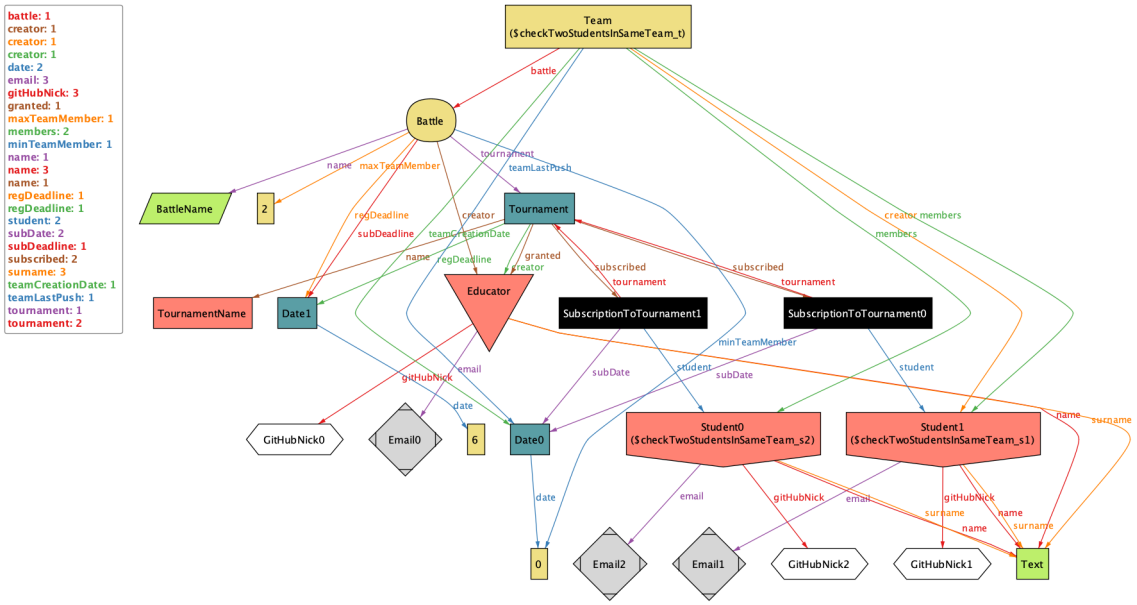


Figure 16: checkTwoStudentsInSameTeam - Alloy

5 Effort

The timetables provided below are only an estimate of the time dedicated to writing and discussing each specific chapter of this document by the team. These durations were not formally tracked during the document's creation and are solely based on the team members' personal perceptions of the time invested.

Diegoli Tommaso

Section	Effort (in hours)
1	4
2	8
3	20
4	2
TOT:	**34**

Tagliani Fabio

Section	Effort (in hours)
1	4
2	7
3	13
4	9
TOT:	**33**

6 References

- SSO: https://www.techtarget.com/searchsecurity/definition/single-sign-on
- Alloy: https://alloytools.org/tutorials/online/
- GitHub actions: https://docs.github.com/actions
- GitHub documentation: https://docs.github.com/en
- Static Analysis: https://www.perforce.com/blog/sca/what-static-analysis
- Similar platform: Codewars: https://www.codewars.com/