POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering 2 Requirements Analysis and Specification Document

Author(s): **Ballabio Giacomo - 10769576**
Benelle Francesco - 10727489
Cavallotti Alberto - 10721275

Academic Year: 2023-2024
Version: 1.0
Release date: 22/12/2023

Contents

1 | Introduction

Online coding challenges and platforms have become an essential resource for programmers and developers in the modern tech landscape. These platforms provide a versatile means to enhance coding skills, offer practical learning experiences, and promote a competitive and engaging approach to problem-solving. Additionally, participation in such platforms can prepare individuals for tech industry job interviews, as many companies utilize similar coding challenges during their recruitment processes.
Indeed, online coding challenge platforms are proving to be invaluable tools. They not only facilitate skill development and community engagement, but also foster professional growth in the continuously evolving field of programming.

What makes the CodeKataBattle platform even more compelling is the involvement of experienced Educators who create coding Battles. These experts design challenges that are not only instructive but also thought-provoking, ensuring a rich and educational experience for participants.

Moreover, these platforms often facilitate the creation of groups, enabling collaborative problem-solving and enhancing team working skills. Users can form teams, tackle challenges together, and learn from one another's approaches. This group dynamic adds an extra layer of motivation and shared learning experiences, enhancing the value of these platforms for participants.

1.1. Purpose

The purpose of this document is to present a detailed description of CodeKataBattle. It is addressed to the developers who have to implement the requirements and could be used as an agreement between the customer and the contractors.
The document is also intended to provide the customer with a clear and unambiguous description of the system's functionalities and constraints, allowing the customer to validate the requirements and to verify if the system meets the expectations.

1.1.1. Goals

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

ID	Description
G1	Allow an ED to create a Tournament
G2	Allow a ST to subscribe to a specific Tournament before the registration deadline
G3	Allow an ED to create a Battle
G4	Allow any User to visualize the profile of other Users and their Badges
G5	Allow a ST to subscribe to a specific Battle before the registration deadline
G6	Allow STs to submit their code before the submission deadline
G7	Allow an ED to set Badges for deserving STs
G8	Allow any User to view the result and the progress of a Tournament
G9	Allow any User to view the results of a Battle

Table 1.1: Goals table.

1.2. Scope

CodeKataBattle (CKB) is a new platform that aims to help Students to improve their software development skills by participating in Tournaments and Battles, in which they will train on code katas, programming exercises that contain some test cases to be passed. Educators will create Tournaments and Battles in order to challenge the Students that will be asked to create groups and compete with their code.

1.2.1. World Phenomena

ID	Description
WP1	The ED wants to create a Tournament
WP2	The ST wants to join a Tournament
WP3	The ED wants to create a Battle
WP4	The ST wants to join a Battle
WP5	The ST wants to create a STG for a Battle
WP6	The ST wants to fork the repository on GH
WP7	The ST wants to push their code on GH
WP8	The ST wants to search another User profile
WP9	The ED wants to search another User profile
WP10	The ST wants to search a Tournament
WP11	The ED wants to search a Tournament
WP12	The ST wants to check the rankings of a Battle
WP13	The ED wants to check the rankings of a Battle
WP14	The ST wants to check the rankings of a Tournament
WP15	The ED wants to check the rankings of a Tournament

Table 1.2: World Phenomenas.

1.2.2. Shared phenomena

ID	Description	Controller	Observer
SP1	The ED creates an account in the CKB system	ED	CKB
SP2	The ED logs in his account in the CKB system	ED	CKB
SP3	The ED logs out of his account from the CKB system	ED	CKB
SP4	The ED creates a Tournament	ED	CKB
SP5	CKB adds the Tournament to the ED's Tournament list	CKB	ED
SP6	The ED checks his Tournament list	ED	CKB
SP7	The ED grants other EDs the permission to create Battles within a Tournament	ED	CKB
SP8	The ED creates a Battle in a specific Tournament	ED	CKB
SP9	CBK adds the Battle to the ED's Battle list	CKB	ED
SP10	The ED checks his Battle list	ED	CKB
SP11	The ED uploads the code kata in the Battle	ED	CKB
SP12	The ED sets the minimum and the maximum number of STs per group for the Battle	ED	CKB
SP13	The ED sets a registration deadline to the Battle	ED	CKB
SP14	The ED sets a final submission deadline to the Battle	ED	CKB
SP15	The ED sets additional configurations for scoring in the Battle	ED	CKB
SP16	The ED sets the Badges that STs could be awarded in the Battle	ED	CKB
SP17	The ED sets the parameters of the Badges that STs could be awarded in the Battle	ED	CKB
SP18	The ST creates an account in the CKB system	ST	CKB
SP19	The ST logs in his account in the CKB system	ST	CKB

SP20	The ST logs out from his account out of the CKB system	ED	CKB
SP21	CKB notifies STs that a Tournament has been created	CKB	ST
SP22	The ST subscribes to a specific Tournament before the registration deadline	ST	CKB
SP23	CKB notifies STs that a Battle has been created	CKB	ST
SP24	The ST subscribes to a specific Battle before the registration deadline	ST	CKB
SP25	CKB adds the Tournament to the ST's Battle list	CKB	ST
SP26	The ST creates a STG for the Battle	ST	CKB
SP27	The ST invites other STs to join his STG for a Battle	ST	CKB
SP28	CKB sends a notification to the ST when he receives ain invitation to join a STG	CKB	ST
SP29	The ST accepts other ST's invitations to join their STG for a Battle	ST	CKB
SP30	CKB adds the STs to the STG	CKB	ST
SP31	CKB creates a GH repository containing the code kata and sends the link to all STs registered in the Battle	CKB	ST
SP32	The ST commits a new version of his code	ST	CKB
SP33	CKB runs the tests on the source code	CKB	ST
SP34	CKB updates the score of the STG on the Battle's Leaderboard	CKB	ST
SP35	CKB updates the score of the ST on the Tournament's Leaderboard	CKB	ST
SP36	STs can view the current ranks evolving during the Battle	ST	CKB
SP37	EDs can view the current ranks evolving during the Battle	ED	CKB
SP38	During the consolidation stage, EDs can manually modify the scores of the STG	ED	CKB

SP39	CKB notifies all STs when the final Battle ranks are available	CKB	ST
SP40	The ED closes the Tournament	ED	CKB
SP41	CKB notifies all the STs involved in the Tournament when the final ranks of the Tournament are available	CKB	ST
SP42	STs can view the current ranks evolving during the Tournament	ST	CKB
SP43	EDs can view the current ranks evolving during the Tournament	ED	CKB
SP44	CKB assigns the Badges to the STs	CKB	ST
SP45	The ST can visualize the profile of other ST or ED	ST	CKB
SP46	The ED can visualize the profile of other ST or ED	ED	CKB

Table 1.3: Shared Phenomenas.

1.3. Definition, Acronyms, Abbreviations

Acronyms	Definition
RASD	Requirements Analysis & Specification Document
ST	Student
ED	Educator
STG	Student Group
CKB	CodeKataBattle
GH	GitHub
User	All STs and EDs
API	Application Programming Interface
DAX	Domain Assumption X
SPX	Shared Phenomena X
WPX	World Phenomena X
RX	Requirement X

Table 1.4: Acronyms used in the document.

1.4. Revision history

- **Version 1.0** - 20/12/2023
- In section 3.2.3 added the Create a new Badge and Modify an Existing Badge use cases and the related sequence diagrams (also the Create Torunament is been modified for this scope).
- In section 3.2.3 added in Create a Battle the creation of github repository.
- In section 3.2.3 some little changes into the Login, Evaluate a code, Join Tournament and Join Battle sequence diagrams and Use case.

1.5. Reference Documents

- Specification Document Assignment
- IEEE Standard Documentation For RASD

1.6. Document Structure

The document is divided into six sections, each with its unique focus, as outlined below.

Introduction: In the first section, we lay out the project's objectives, purposes, and offer a concise examination of global and shared phenomena. This section also includes a compilation of abbreviations and definitions that are essential for comprehending the problem.

Overall Description: The second section provides a comprehensive overview of the problem. It delves into further details about the domain and various scenarios involved, in addition to discussing product and User characteristics, assumptions, dependencies and constraints.

Specific Requirements: The third section is dedicated to an in-depth analysis of the specific requirements. It offers detailed insights into external interface requirements, functional requirements, and performance requirements.

Formal Analysis Using Alloy: The fourth section employs Alloy to conduct a formal analysis. This chapter's primary purpose is to validate the accuracy of the model described in the preceding sections. It focuses on presenting the results of the conducted checks and meaningful assertions.

Effort Spent: Section five outlines the individual efforts contributed by each group member to compose this document.

References: The final section serves as a bibliography, listing the references and additional resources used in the creation of this document.

2 | Overall Description

2.1. Product perspesctive

2.1.1. Scenarios

Scenario 1: Unregistered ST creates an account. Bhristian Ciffi wants to improve his coding skills, so he decides to join CKB. First he opens the website and clicks on the "Sign-Up" button, so he proceeds to insert his name, surname, nickame, eMail and password in order to create a new account. Then, Bhristian will receive a confirmation mail to activate the account. Once the creation of the account is completed he'll be free to join his first competition.

Scenario 2: ED creates a new Tournament. DumbTurkey, who is a Software Engineering teacher and an ED for CKB in his free time, wants to test STs with his new Tournament regarding the various phases of the creation of a calculator that resolves differential equations in Java, so he creates a new Tournament by clicking in his home page on the "Create a Tournament" button. Then he inserts the name of the Tournament, the students' subscription deadline, grants to Gianbrambo Bambo, one of his friends, who is also an ED, the permissions to create Battles within the Tournament by inserting his eMail in the creation Tournament form and he clicks on the "Create Tournament" button. As soon as the button is clicked a notification is sent by CKB to all STs in order to inform them about the creation of the Tournament and another notification is sent to Gianbrambo Bambo to alert him that he has permission in the Tournament.

Scenario 3: ST joins a Tournament. Ilbo Glions wants to join a Tournament that he noticed in the "Open Tournament" section on his Homepage and clicks on the Tournament's name. Opening the Tournament Page, he decides to participate in the Tournament, so he clicks on the "Join Tournament" button.

Scenario 4: ED creates a Battle. Bamba Filo, an ED enter in his Tournament Page in order to create a new Battle with the title "Time Series Forecasting Challenge", so

he inserts the title in the apposite form, sets the minimum and maximum numbers of participants per each group to be respectively 3 and 5, the registration deadline to 4 days from that moment, the final submission deadline to 15 days and uploads the code kata with a brief description of what the STs have to do. Finally he clicks on the “Confirm” button. Simultaneously a notification is sent to every ST that joined the Tournament.

Scenario 5: ST joins a Battle. Mateo Retegui wants to participate in the “Time Series Forecasting Challenge”, a Battle created by an ED. After receiving the notification from the ED, he selects the Battle from the link in the notification and he clicks on the “Join the Battle” button. Then he is asked to join or create a Group in order to participate, so he clicks on the “Create a Group" button, so he can set the Group’s name to be “TheCEOs” and sends the invitations to some of his friends. Finally, once the Group is full he clicks on “Confirm the Group”. Once the registration deadline expires, he will see the GitHub Repository link on the main page of the Battle, so he proceeds to fork the repository and starts coding.

2.1.2. Class diagrams

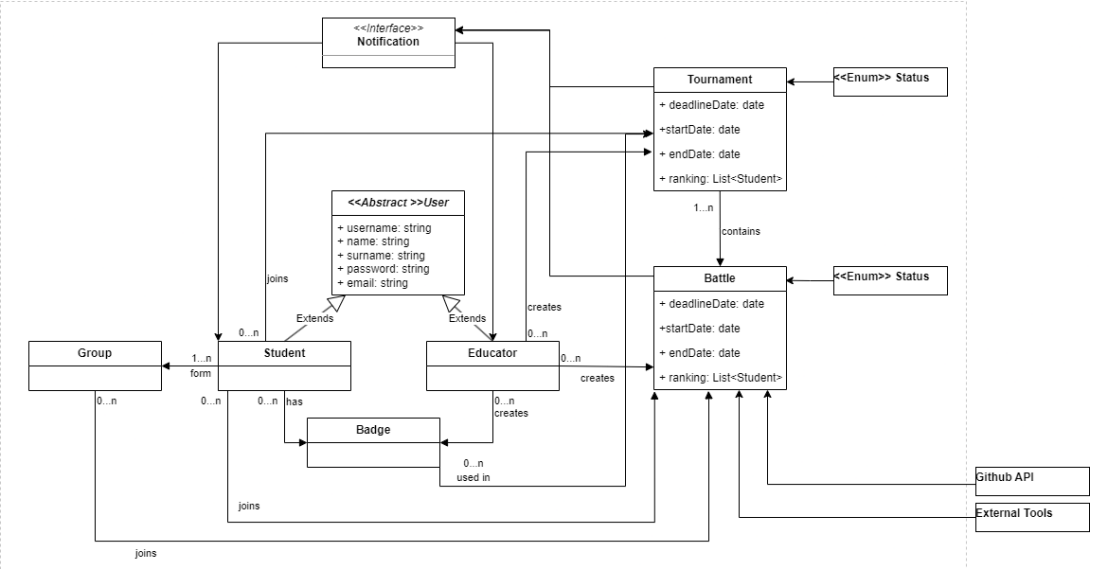

Figure 2.1: High level Class Diagram.

Student and Educator Classes are implemented by extending the abstract Class 'User', since their attributes are almost the same, in fact the Username, name, surname, eMail and password strings are in common for both the classes.
EDs may create new instances of the Tournament Class, which contains the Battles. There are two different enumerations (one for Tournament, the other for Battle) in order to implement and to keep record of the current status of the Tournament or the Battle. The current rankings are saved using List<Student> in the correct classes.
EDs can also set one or more Badges to be awarded to deserving STs. Battle Class may interact with the external Tools (like the GitHub API and the Tools for the code testing). The Notification Interface is used to send different kinds of notification to STs and EDs when a new Tournament or Battle is created, or during the various phases of the competition, or when a ST invites other STs to join a STG.

2.1.3. State diagrams

In this section are presented the State Diagrams of the CKB system representing all the possibile operation that a User can perform.

Signup. When a User wishes to register on CKB, they are required to input their credentials, including their name, surname, nickname, eMail and password in a registration

form. If the provided credentials are deemed valid (ensuring that the eMail or nickname is not already in use, and the password meets the criteria of being at least 8 characters long, containing at least one capital letter, a number, and a special character), CKB will send an eMail to the User. Upon the User confirming the registration through the provided eMail link, the new account is successfully created. If the credentials are not correct CKB shows an error message to the User and redirects him to the signup form page.

Furthermore, if the User selects the option "Register as a new Educator" during the registration process, CKB will enhance the account with additional permissions. These permissions may include the possibility to create Tournaments or Battles.

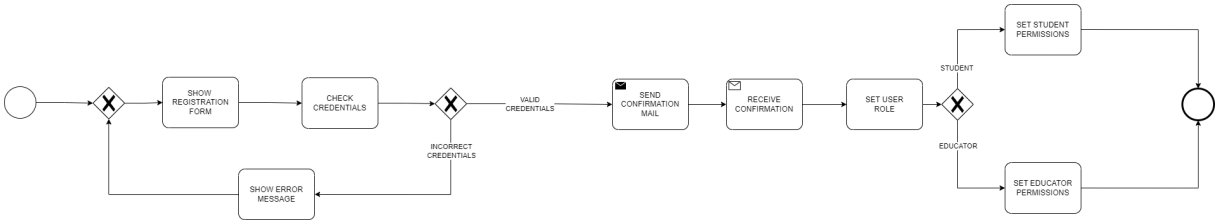

Figure 2.2: Signup state diagram.

Login. When a registered User attempts to log in to their CKB account, they must enter their credentials (eMail and password) into a login form. If the provided credentials are accurate —matching those of a registered User in the CKB database— CKB displays the User's homepage, showcasing Tournaments and ongoing Battles in which the User is participating. On the other hand, if the entered credentials are invalid, CKB presents an error message to the User and redirects them back to the login form page.

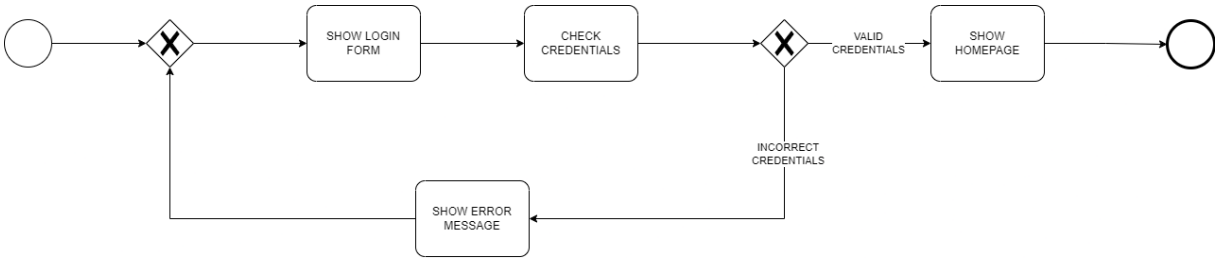

Figure 2.3: Login state diagram.

Create a Tournament. When an ED intends to create a new Tournament on CKB, he is required to input various parameters in the create Tournament form. These parameters include the Tournament name, the registration deadline for the STs and the rules to acquire the Badges. In addition to these parameters, the ED can invite other EDs to join this Tournament, through a notification sent by CKB, to help him create new Battles. If any of these parameters are deemed unacceptable by the system, CKB displays an error message to the ED and redirects him back to the create Tournament form page.

Conversely, if all parameters meet the system's criteria, CKB proceeds to create the Tournament. The newly created Tournament is then added to the ED's Tournaments list, and a notification is sent to all STs to inform them of the availability of a new Tournament.

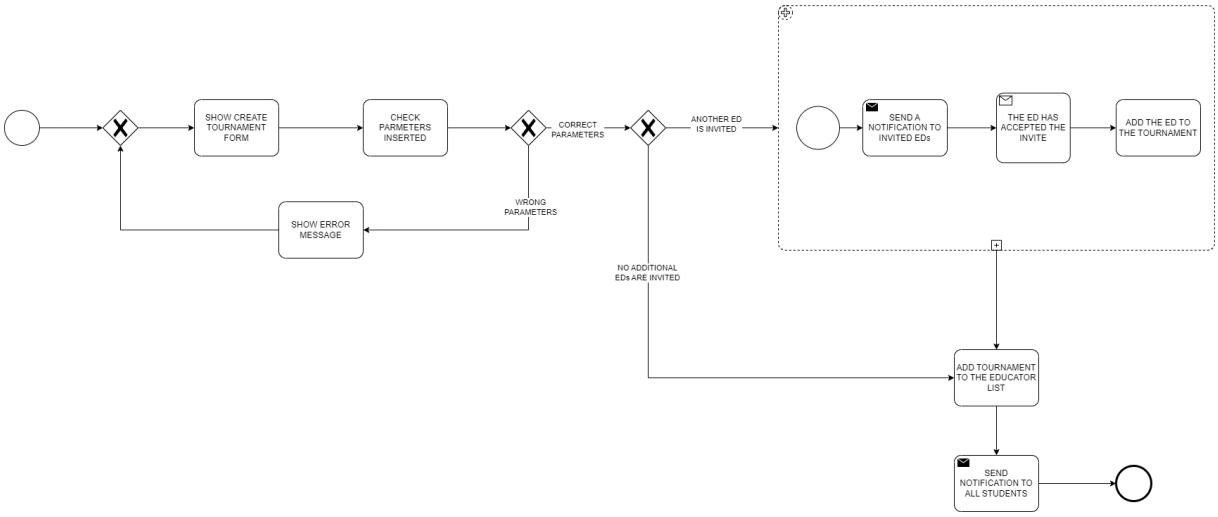

Figure 2.4: Create a Tournament state diagram.

Create a Battle. Once a Tournament is created, an ED has the option to create a new Battle within it. To do so, the ED is required to input various parameters in the create Battle form. These parameters encompass the Battle name, code kata, registration deadline, submission deadline, minimum and maximum members per STG, and any additional rules for evaluation. In the event that any of these parameters are considered unacceptable by the system, CKB displays an error message to the ED and redirects them back to the create Tournament form page.

Conversely, if all parameters meet the system's criteria, CKB proceeds to create the Battle. The newly created Battle is then added to the ED's list of Battles, and a notification is sent to all STs registered to the Tournament where the Battle resides, informing them that a new Battle is now available.

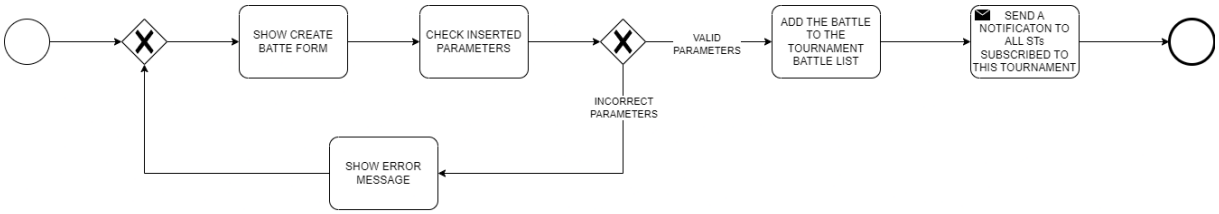

Figure 2.5: Create a Battle state diagram.

Join a Tournament. When a new Tournament is created, a ST receives a notification or discovers it on the homepage. At this point, the ST can choose to join the Tournament. If the ST is not yet registered for that specific Tournament, CKB not only adds the Tournament to the ST's Tournament list but also includes the ST in the participant list of the newly created Tournament. This ensures that the ST is officially registered as a participant, allowing him to engage in the Battles.

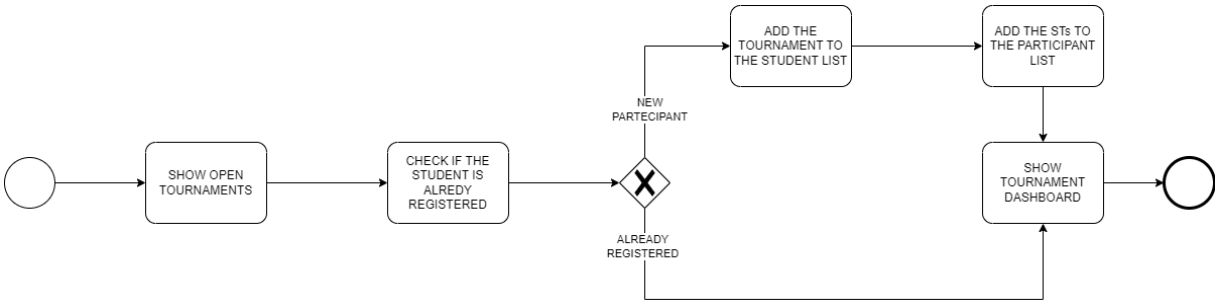

Figure 2.6: Join a Tournament state diagram.

Join a Battle. Once a ST has registered for a Tournament, and a new Battle is created within it, the ST has the option to join a specific Battle. CKB performs a check to verify if the registration deadline for the chosen Battle has not expired. If the ST is within the allowed timeframe, CKB adds the ST to the list of participants for that Battle and directs them to the create group page where they can further engage in the competition.

However, if the registration deadline has already expired, CKB displays an error message to the ST, notifying them of the expiration, and redirects them to the Tournament dashboard with the respective Battle. This ensures that participants are aware of the registration timeline and can take timely actions to join the Battles of their choice.

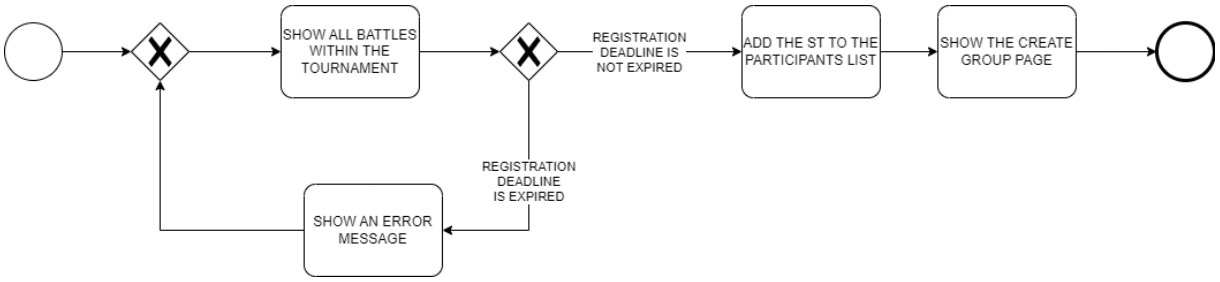

Figure 2.7: Join a Battle state diagram.

Create a Group. The possibility to create a group within a Battle is granted to STs once they have joined a Battle, and the registration deadline has expired. At this point, they are redirected to the create group form, where they write de STG name and can invite other STs to join their STG. The process involves the ST entering a nickname, and upon verification by CKB that the ST with that nickname has also joined the same Battle, a notification is sent to the invited ST, enabling them to join the STG.

Upon receiving a notification, a ST has the option to respond, and upon acceptance, the ST is added to the STG, becoming a participant in the Battle. This invitation process is replicated for each ST that the group creator wishes to invite. Once the group is formed, the ST who initiated the STG can confirm the group, but only if the minimum and maximum number of group members are adhered to. Once confirmed, the STG is ready to actively participate in the Battle.

However, if an invited ST is not part of the Battle, a ST accepts the invitation after the deadline, or the group creation deadline has expired, CKB displays an error message to the respective ST, informing them of the issue and guiding them accordingly. This ensures that the group creation and participation adhere to the specified deadlines and criteria.

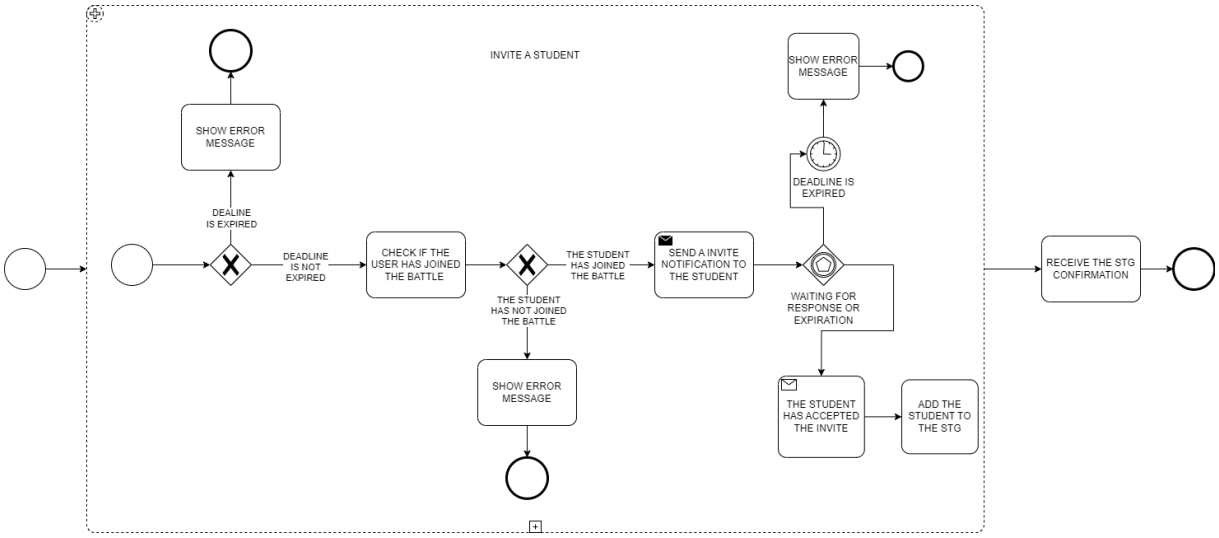

Figure 2.8: Create a Group state diagram.

Evaluate the code. During the consolidation stage, once all commits have been completed and no further changes are possible, an ED has the option to manually inspect a STG code. The ED can download the code from the Battle dashboard for thorough examination. After reviewing the code, the ED can then modify the score assigned to the STG, which was previously assessed by external tools automatically during each commit.

To facilitate this, CKB provides a form that allows the ED to input the adjusted score. However, before updating the score, CKB checks whether the new score adheres to predefined boundaries, such as being non-negative or not exceeding 100. If the new score meets these criteria, CKB successfully updates the score for the STG.

In the event that the new score falls outside the specified boundaries, CKB displays an error message to the ED and redirects them to the evaluate code form.

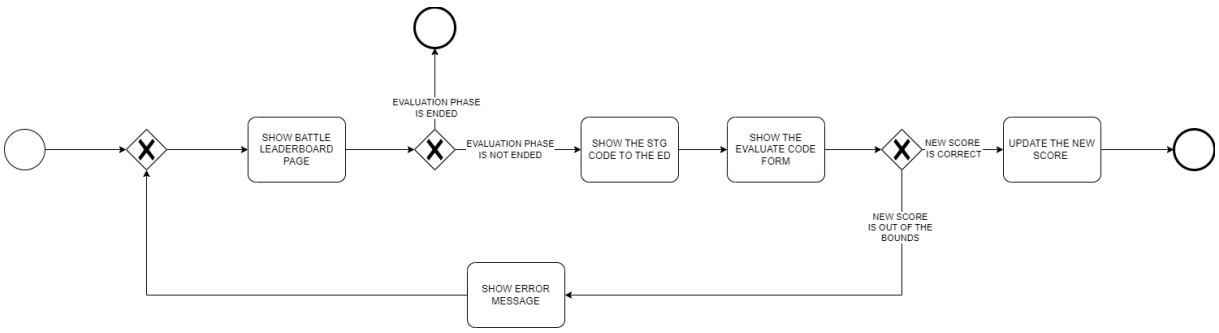

Figure 2.9: Evaluate the code state diagram.

Open a profile. If a User wishes to view another User's profile, including Badges and other relevant information, they can do so by clicking on the User's nickname in various dashboards, such as the Tournament dashboard or the group composition. Upon clicking the nickname, CKB displays the profile of the selected User to provide the desired information. This feature allows Users to easily access and view the profiles of others within the platform.

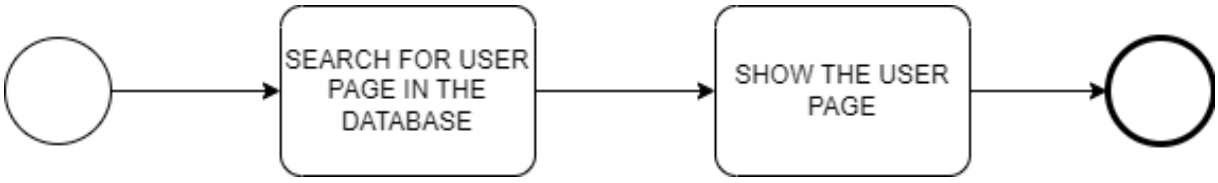

Figure 2.10: Open a profile state diagram.

Search for a profile. In addition to opening a User profile by clicking the nickname in a dashboard, Users also have the option to use the search bar. By entering a nickname or a keyword in the search bar, CKB will display a list of Users whose nickname contain the entered keyword. Users can then click on a specific User from the search results to open and view their profile, providing a convenient and flexible way to access User information within the platform.

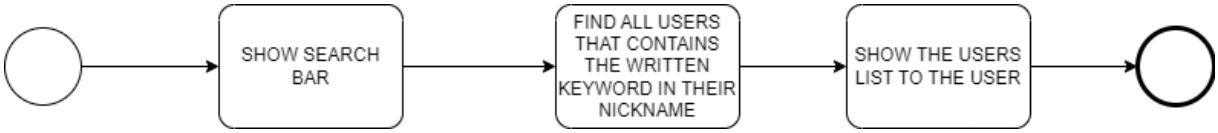

Figure 2.11: Search for a profile state diagram.

Search for a Tournament. Users can utilize the search bar to look for Tournaments by entering a name or a keyword. When a search query is entered, CKB will present a list of Tournaments whose names contain the specified keyword. Users can then click on a particular Tournament from the search results, and CKB will redirect them to the Tournament dashboard for detailed information and engagement with the selected Tournament.

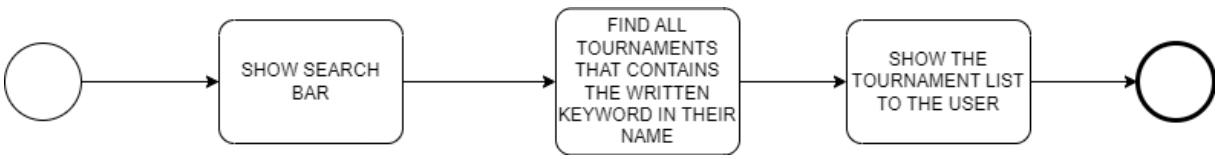

Figure 2.12: Search for a Tournament state diagram.

Logout. Users can log out from their account at any time by clicking the "Logout" button. Upon clicking the button, CKB terminates the User's session, ensuring that they are securely logged out. Following this action, the User is then redirected to the login page, providing them with the opportunity to rejoin the CKB system if they wish to do so.

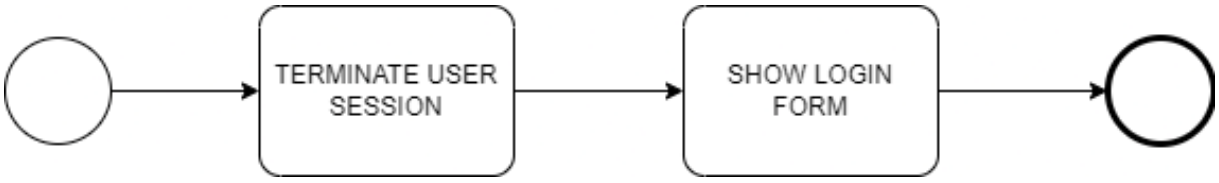

Figure 2.13: Logout state diagram.

2.2. Product functions

Here are listed a summary of the main functions of the CKB system:

Signup: The User submits his name, surname, nickname, eMail address and a new password. A confirmation mail is required to complete the process.

Login: The User sign in after typing his eMail and password in the login form.

Create a Tournament: The ED creates a new Tournament and the system sends a notification to all the STs.

Join a Tournament: A ST can join a Tournament by clicking on the "Join Tournament" button on the Tournament's page.

Create a Battle: An ED can create multiple Battles within a Tournament to test the STs. When a new Battle is created CKB sends a notification to all the STs that have joined the Tournament.

Join a Battle: A ST can join a Battle by clicking on the "Join Battle" button on the Battle's page.

Create a Group: After joining a Battle the ST may create a new group and invite other STs or accept an invitation to join another STG. It is a choice of the ST to play alone or to create a group with STs, whether this is made possible by the minimum num-

ber of STs per STG decided by the ED.

Evaluate Code: During the Consolidation phase, an ED can manually download and evaluate the code of a STG within a Battle.

Create a Badge: An ED can create Badges to be awarded to worthy STs and define their rules.

Search for a Profile: Users can write a nickname or a keyword in the search bar to retrieve another User's profile.

Open a Profile: Users can click on a nickname in a Leaderboard to open another User's profile.

Search for a Tournament: Users can write a Tournament's name or a keyword in the search bar to retrieve the information about a Tournament.

Logout: A User session will be closed if the User clicks on "Logout" button. Next time the User opens CKB he will need to log in again.

2.3. User characteristic

There are two types of registered Users in CKB: Students (STs) and Educators (EDs). Here are briefly explained their main characteristics:

- ST: Students join CodeKataBattle in order to prove and improve their skills in coding. They need to have a device with an internet connection and an account in order to have access to CKB, in which they can join Tournaments and Battles to compete with other STs.
- ED: Educators join CodeKataBattle, where they can create Tournaments in which STs can compete in different Battles and evaluate the STs' works.

Both the EDs and the STs need to register an account in order to access CKB. During the registration process their both asked to provide an eMail account.

2.4. Assumptions, dependencies and constraints

2.4.1. Domain assumptions

ID	Description
DA1	The EDs need to have a valid eMail address.
DA2	The STs need to have a valid eMail address.
DA3	The EDs need to have a device and a reliable internet connection.
DA4	The STs need to have a device and a reliable internet connection.
DA5	The EDs need to have a GitHub account.
DA6	The STs need to have a GitHub account.
DA7	STs need to know how to fork a GitHub Repository.
DA8	STs need to know how to push their code on GitHub.
DA9	CKB needs to communicate with GH through its API.
DA10	CKB needs to communicate with the external Static Analysis Tools in order to compute the scores.
DA11	CKB needs to communicate correctly with the eMail system.

Table 2.1: Domain assumptions.

3 | Specific Requirements

This section is devoted to a specific description of every kind of requirement the system has to deal with in order to achieve all the functionalities described.

3.1. External interface requirements

3.1.1. User interfaces

The CodeKataBattle's User interface will be a website, developed in order to be used both by STs and EDs, available to everybody who has a device with an Internet Browser and a reliable Internet connection.

3.1.2. Hardware interfaces

The system will be accessible from every device with an Internet Browser to access the website and a reliable Internet connection.
The User is free to choose his device like a computer, a tablet or a smartphone, despite that, it is suggested to use a computer, which will make it easier to deal with the writing of the code.

3.1.3. Software interfaces

The system requires the GitHub APIs and the External Tools in order to work properly. Also, it will use a mailing system to send confirmation eMails to the Users during the registration process.

3.1.4. Communication interfaces

The communication Interfaces needed by the system are the HTTPS Protocol and the Mail System Transfer Protocol (SMTP).

3.2. Functional requirements

3.2.1. Requirements

ID	Description
R1	CKB allows unregistered Users to sign up
R2	CKB allows registered EDs to login
R3	CKB allows registered STs to login
R4	CKB allows EDs to create Tournaments
R5	CKB allows EDs to grant the permissions of a Tournament to other EDs
R6	CKB allows EDs to create Battles
R7	CKB allows EDs to uploads the code kata of a Battle
R8	CKB allows EDs to set the minimum and the maximum number of STs per group of a Battle
R9	CKB allows EDs to set a registration deadline of a Battle
R10	CKB allows EDs to set a submission deadline of a Battle
R11	CKB allows EDs to set additional configuration for the scoring system of a Battle
R12	CKB allows EDs to set functional aspects for the scoring system of a Battle
R13	CKB allows EDs to create new Badges
R14	CKB allows EDs to choose the rules related to the awarding of Badges
R15	CKB allows EDs to choose which Badges to award in a certain Tournament
R16	CKB allows EDs to assign a score manually during the consolidation stage
R17	CKB allows EDs to close a Tournament
R18	CKB allows EDs to visualize the profile of another User
R19	CKB allows STs to visualize the profile of another User
R20	CKB allows STs to join a Tournament
R21	CKB allows STs to join a Battle
R22	CKB allows STs to create a new STG
R23	CKB allows STs to join a STG
R24	CKB allows STs to invite other STs in their STG
R25	CKB stores the informations about the Users
R26	CKB shall ensure security of data
R27	CKB sends notifications to every ST when a new Tournament is created
R28	CKB sends notifications when a new Battle is created to every ST which is participating in the Tournament that the Battle is part of

R29	CKB sends notifications to a ST when he receives an invitation to be part of STG
R30	CKB creates a GH repository of the code kata when the registration deadline for the Battle expires
R31	CKB sends the link of the GH repository to every STG that participates in the Battle
R32	CKB evaluates the STG's work every time a push is made on GH and calculates Battle score for the STG
R33	CKB updates the Battle Leaderboard once a new score is registered
R34	CKB updates the Tournament Leaderboard once a new score is registered
R35	CKB allows STs to check the Leaderboard of a Battle
R36	CKB allows EDs to check the Leaderboard of a Battle
R37	CKB allows EDs to analyze the code of a STG
R38	CKB sends notifications to every STs participating in the Battle once the consolidation stage ends
R39	CKB allows STs to check the list of ongoing Tournaments
R40	CKB allows EDs to check the list of ongoing Tournaments
R41	CKB allows STs to check the Leaderboard of a Tournament
R42	CKB allows EDs to check the Leaderboard of a Tournament
R43	CKB sends notification to every ST involved in a Tournament when the Tournament is closed and the final ranks are available
R44	CKB shall communicate with the GH API in order to calculate a new score every time a push action is made by a STG
R45	CKB shall communicate with the external tool in order to calculate the score of a STG
R46	CKB shall communicate with the mailing system in order to allow Users to register their account
R47	STs need to fork the GH repository of the Battle they are participating in
R48	STs need to push their code in the GH repository in order to have their code evaluated
R49	CKB shall assign the Badges to all STs that fullfill their requirements
R50	CKB sends notifications to the ED when he receives the permission to create Battles in a Tournament

Table 3.1: Requirements.

3.2.2. Use case diagrams

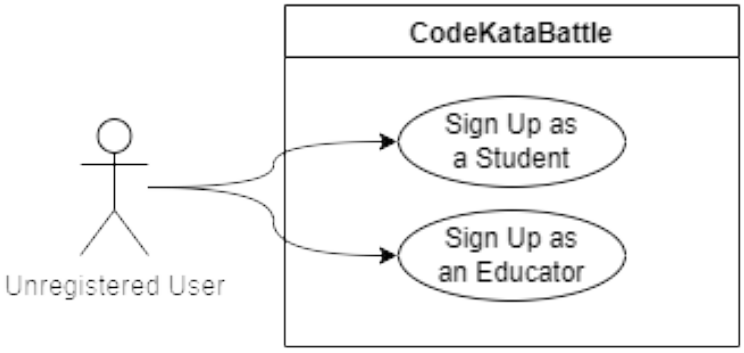

Figure 3.1: Use Cases Diagram for Unregistered Users.

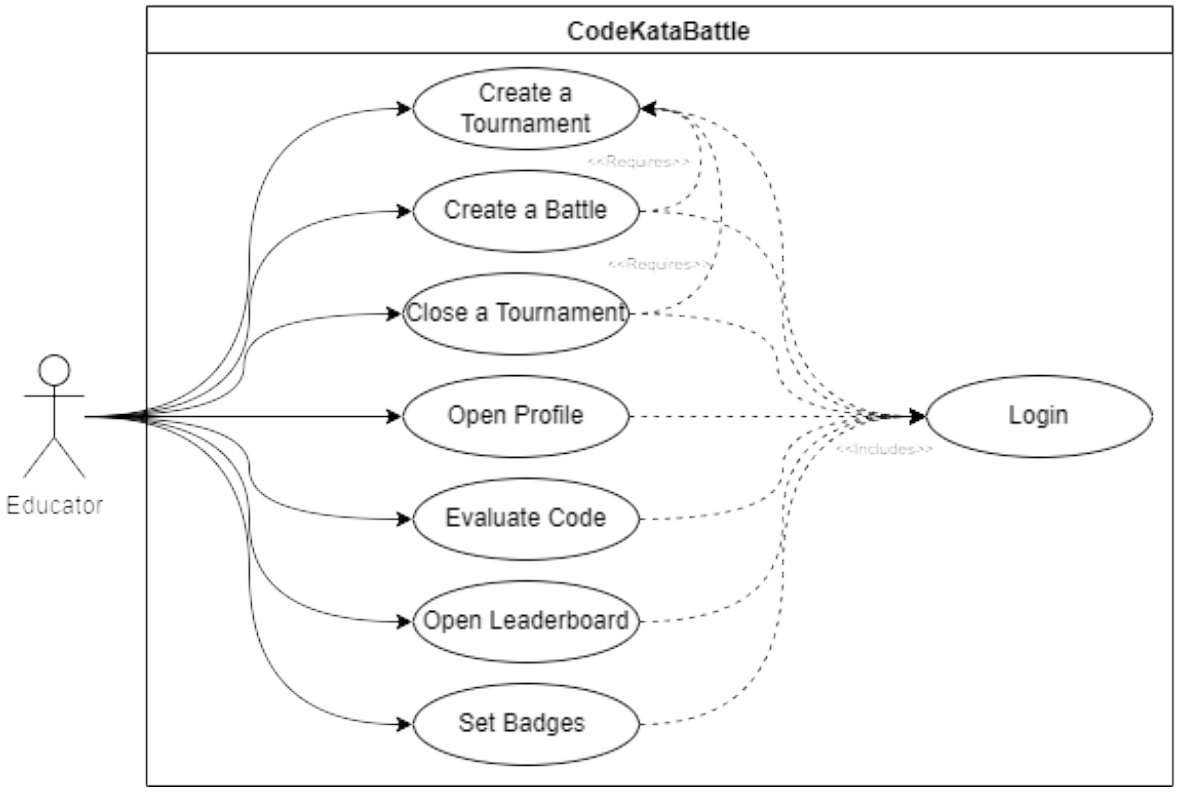

Figure 3.2: Use Cases Diagram for Educators.

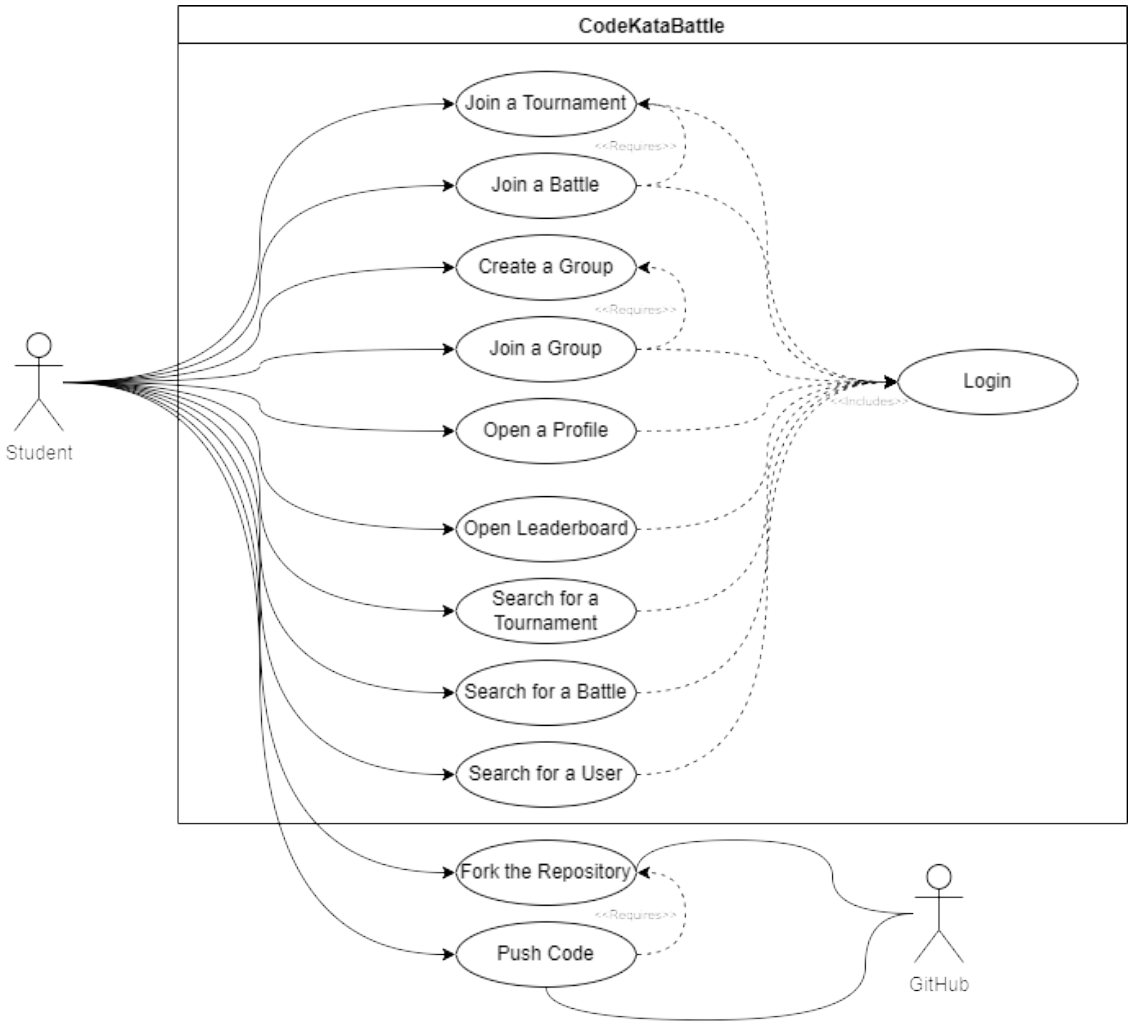

Figure 3.3: Use Cases Diagram for Students.

3.2.3. Use cases

In this section, they are explained and represented the main identified use cases.
There is a table with entry conditions, event flow, exit conditions and exception for each of them, and a sequence diagram that shows the messages exchanged between the entities and the called functions.

UC1. Signup as ED

Actor	ED, eMail provider
Entry conditions	The ED is not already registered in CKB and has to search the CKB URL in the browser search bar
Event Flow	1 - CKB shows the login form. 2 - The ED clicks on "Create an Account" button. 3 - CKB shows the signup form. 4 - The ED inserts his name, surname, nickname, eMail and password in the form and also ticks on the "Signup as Educator" checkbox. 5 - The ED clicks on the "Register" button. 6 - CKB checks all the credentials. 7 - If credentials are correct CKB sends a confirmation eMail to the ED through the eMail provider. 8 - The ED clicks on the confirmation link.
Exit condition	CKB allows the ED to access to the CKB system.
Exceptions	• The eMail address is already linked to an account. In this case an error message is shown and the ED is redirected to the profile creation settings. • Invalid password if it is shorter than 8 characters, if it doesn't have at least 1 number and/or 1 capital letter and/or a special character. In this case an error message is shown and the ED is redirected to the profile creation settings. • The nickname is already used. An error is shown and the ED is redirected to the profile creation settings.

Table 3.2: Signup as ED use case.

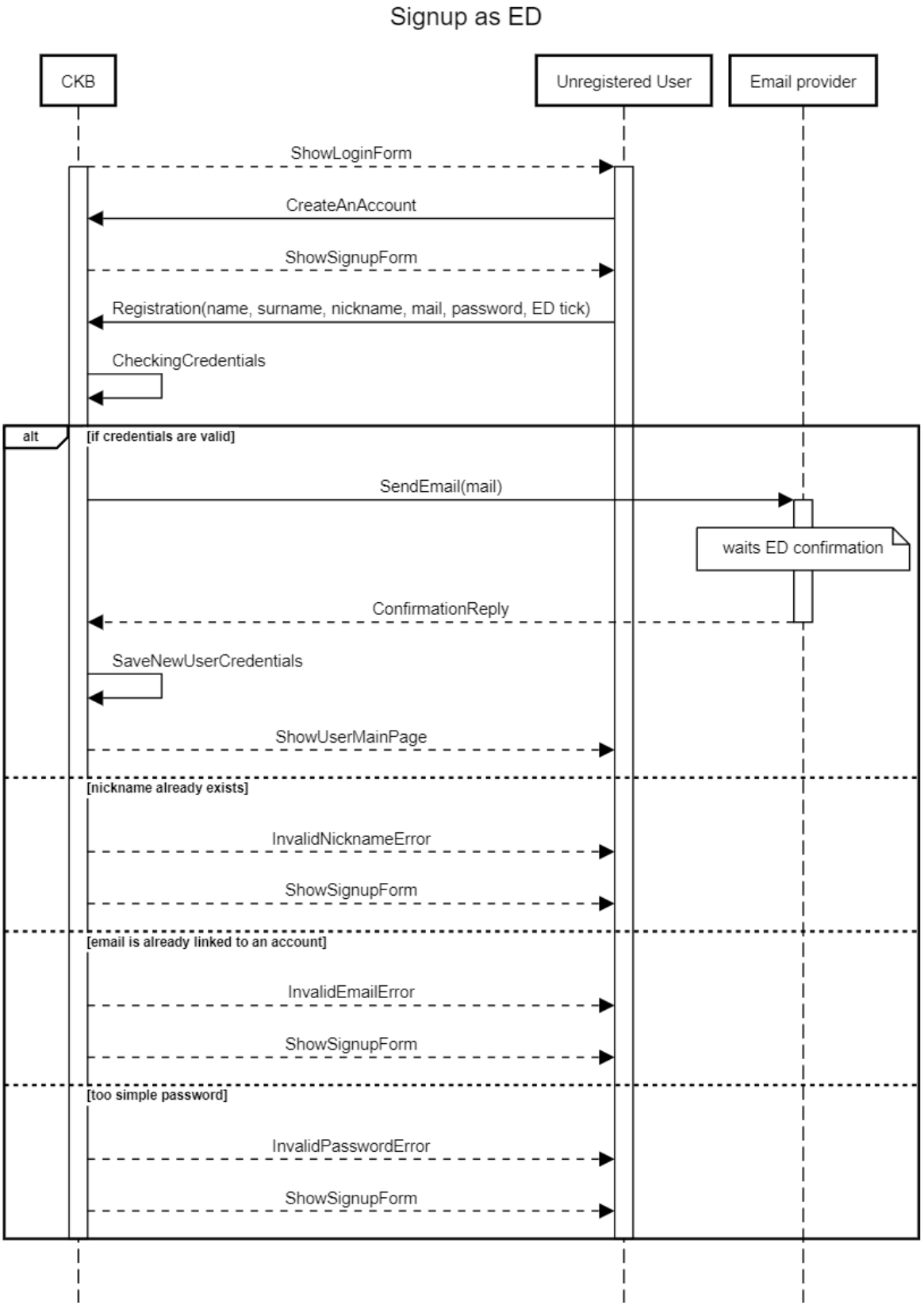

Figure 3.4: Signup as ED sequence diagram.

UC2. Signup as ST

Actor	ST, eMail provider
Entry conditions	The ST is not already registered in CKB and has to search the CKB URL in the browser search bar
Event Flow	1 - CKB shows the login form. 2 - The ST clicks on "create an account" button. 3 - CKB shows the signup form. 4 - The ST inserts his name,surname, nickname and password in the form. 5 - The ST clicks on the "Register" button. 6 - CKB checks all the credentials. 7 - If credentials are correct CKB sends a confirmation eMail to the ST through an eMail provider. 8 - The ST clicks on the confirmation link.
Exit condition	CKB allows the ST to access to the CKB system.
Exceptions	• The eMail address is already linked to an account. In this case an error message is shown and the ST is redirected to the profile creation settings. • Invalid password if it is shorter than 8 characters, if it doesn't have at least 1 number and/or 1 capital letter and/or a special character. In this case an error message is shown and the ST is redirected to the profile creation settings. • The nickname is already used. An error is shown and the ST is redirected to the profile creation settings.

Table 3.3: Signup as ST use case.

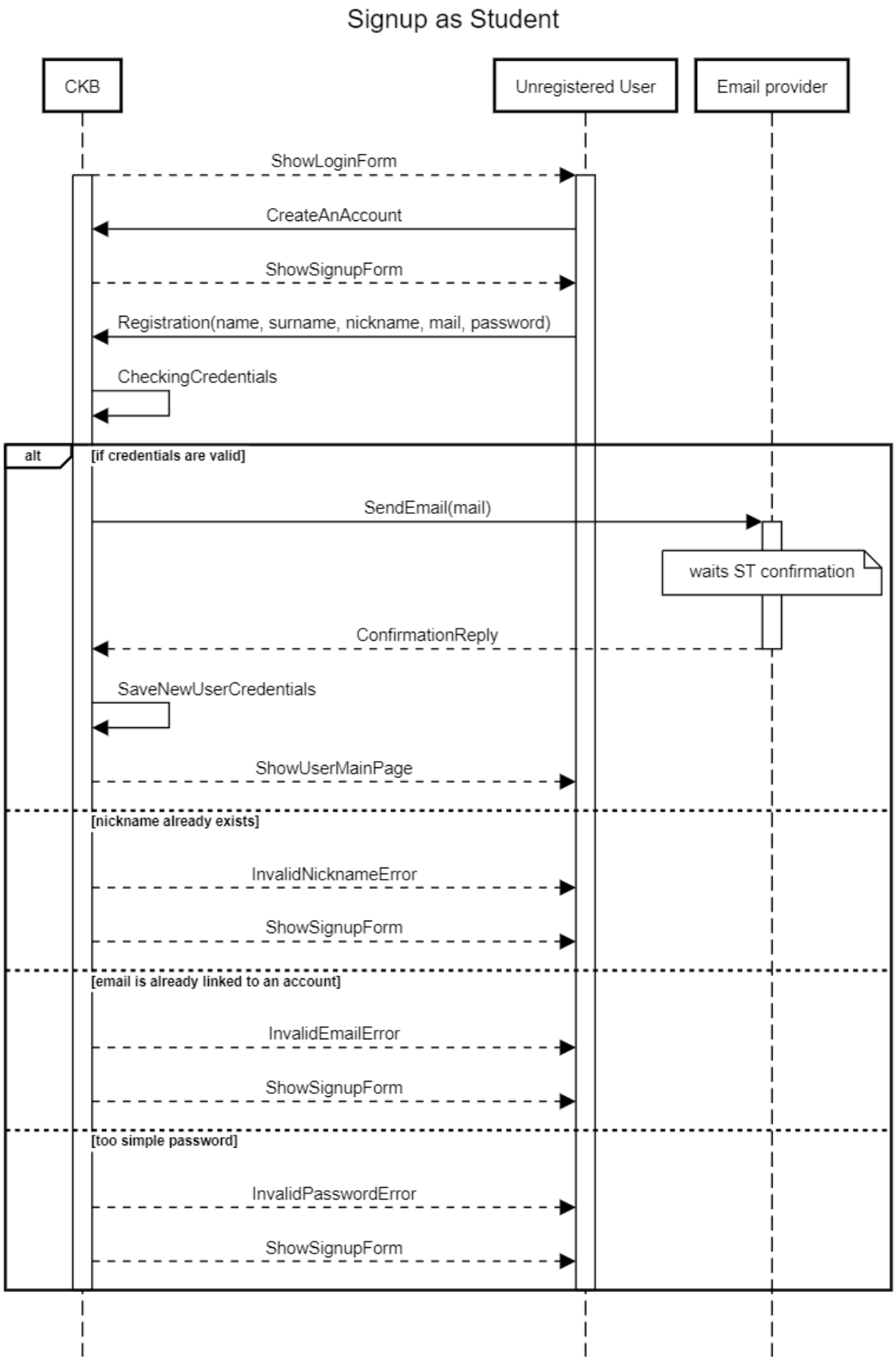

Figure 3.5: Signup as ST sequence diagram.

UC3. Login

Actor	Users
Entry conditions	The User should be registered in CKB and has to search the CKB URL in the browser search bar
Event Flow	1- CKB shows the login form. 2- The User inserts his nickname and password in the form. 3- The User clicks on the "Login" button. 4- CKB checks the credentials.
Exit condition	CKB allows the User to access to the CKB system.
Exceptions	Incorrect eMail or password. An error message is shown and the User is redirected back to the Login page.

Table 3.4: Login use case.

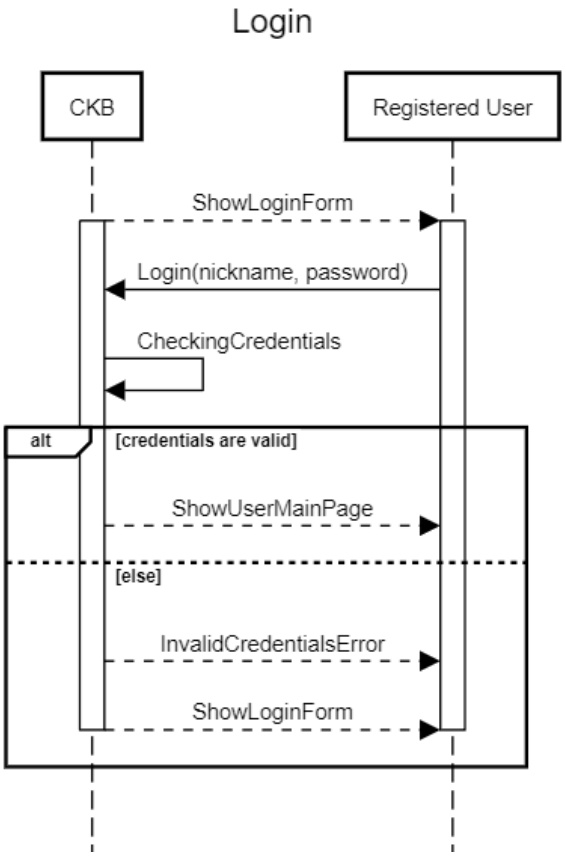

Figure 3.6: Login sequence diagram.

UC4. Create a Tournament

Actor	ED
Entry conditions	The ED is correctly logged in. The ED has decided to create a Tournament. The ED is on his profile page.
Event Flow	1- The ED clicks on "Create a Tournament" button. 2- CKB returns the create Tournament form. 3- The ED writes the Tournament name, sets the deadline for STs to subscribe and chooses and writes the other EDs nicknames to grant them permissions to create new Battles. 4- CKB stores the data of the Tournament. 5- CKB sends a notification to the chosen EDs. 6- CKB notifies all the STs of the creation of the Tournament. 7- CKB checks if the tick on the create badge is true and if it is so CKB retrives all the ED's previously used Badges and redirects the ED to the Badges requirement page. 8- If that's not the case CKB return the Tournament main page.
Exit condition	The Tournament is correctly created and a confirmation message is shown to the ED.
Exceptions	• The Tournament's name is null or already exists. An error message is shown and the ED is redirected back to create Tournament settings. • The EDs chosen are non-existent, or the creator chooses other ED that have already permission in this Tournament. An error message is shown to the ED and the ED is redirected to create Tournament settings. • The registration deadline is a date before the day the Tournament is created. An error message is shown to the ED and a new deadline has to be chosen and the ED is redirected to create Tournament settings.

Table 3.5: Create a Tournament use case.

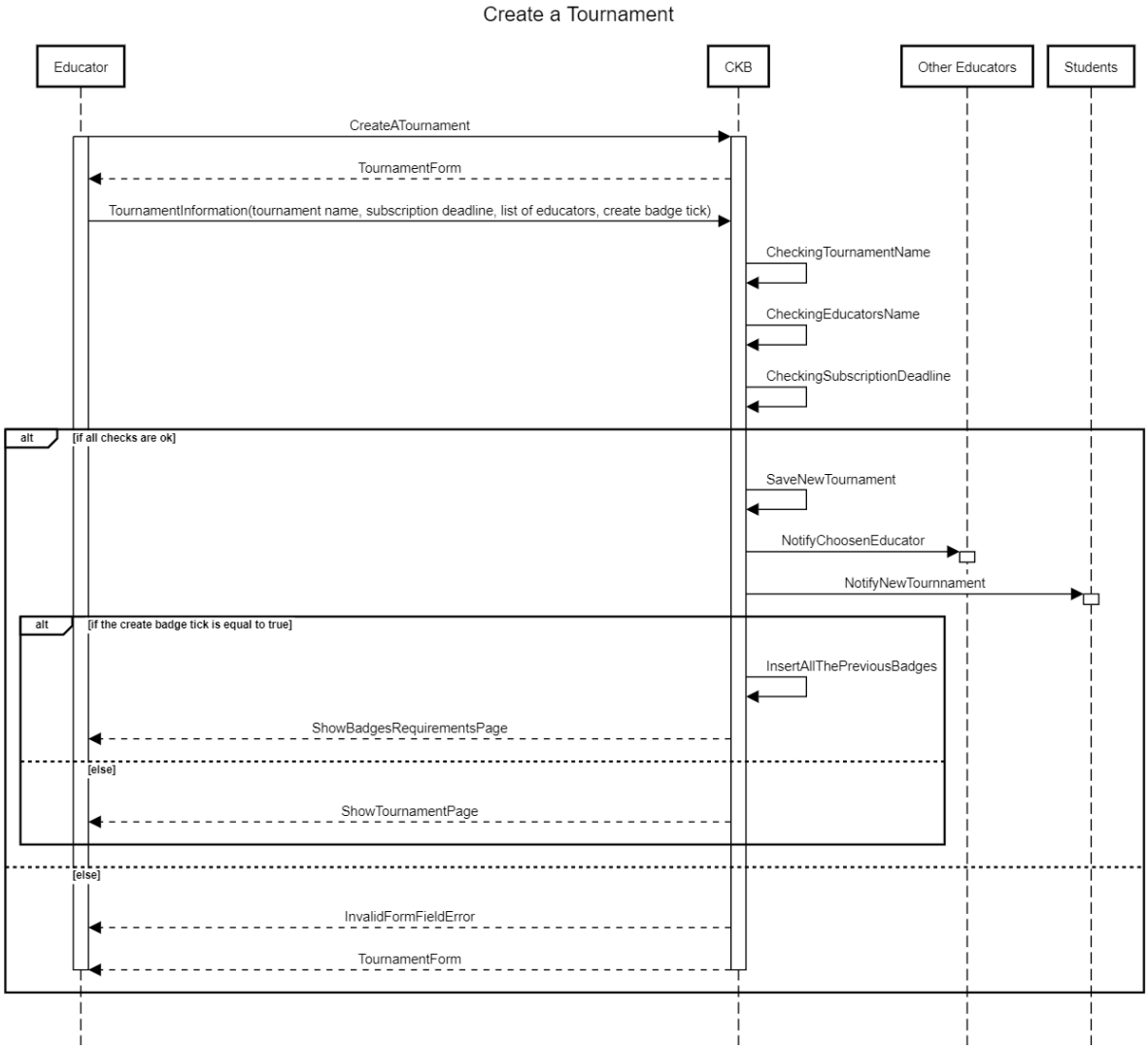

Figure 3.7: Create a Tournament diagram.

UC5. Join a Tournament

Actor	ST
Entry conditions	The ST is correctly logged in. The ST has decided to join a Tournament.
Event Flow	1- The ST search the keyword of a Tournament into the search box 2- CKB returns the result from the search. 3- The ST clicks on the new Tournament's name. 4- CKB redirects the Student to the main page of the Tournament. 5- The ST clicks on the "Join Tournament" button. 6- CKB saves the ST as one of the competitors of the Tournament. 7- CKB shows a confirm message and redirects the ST to the page that contains the current Leaderboard of the Tournament (the Tournament main page).
Exit condition	The ST is now part of the Tournament and will be notified whenever a new Battle will be available.
Exceptions	• Nothing is found from the research. In this case the ST will be redirectd back to the home page. • The Tournament already ended. In this case is shown an error message and the ST is redirected to the final Leaderboard of the Tournament (the Tournament main page). • The ST already joined the Tournament. In this case CKB will not show any error messages, but will redirect the ST to the current Leaderboard of the Tournament (the Tournament main page). • The Deadline already expired. An error message is shown and the ST is redirected back to the Home Page.

Table 3.6: Join a Tournament use case.

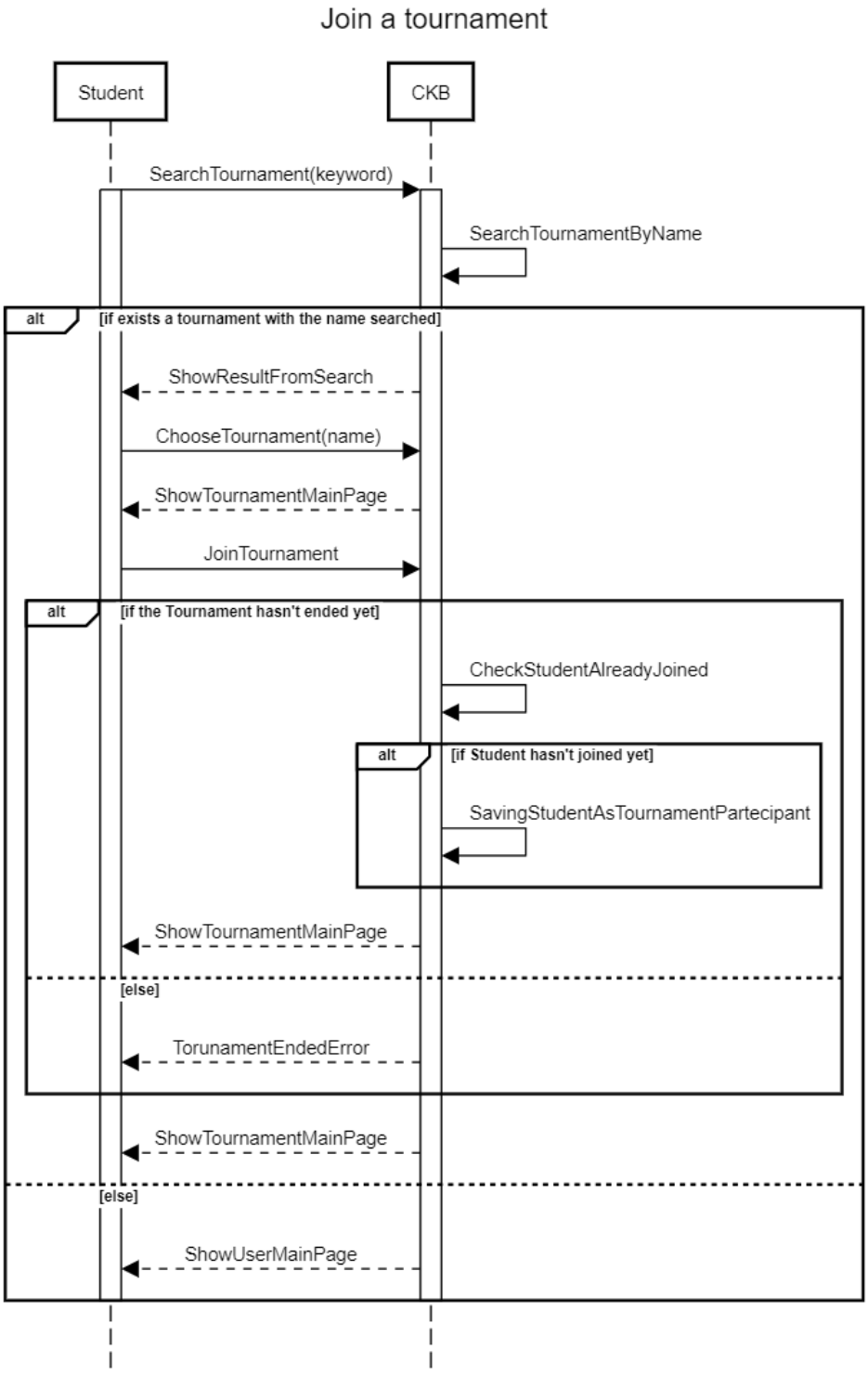

Figure 3.8: Join a Tournament diagram.

UC6. Create a Battle

Actor	ED
Entry conditions	The ED is logged in. The ED is in the Tournament page. The ED has decided to create a Battle.
Event Flow	1- ED clicks on "Create new Battle" button. 2- CKB redirects ED to the create Battle form. 3- The ED uploads the code kata. 4- The ED writes the Battle name. 5- The ED sets the minimum and maximum number of STs per group. 6- The ED sets the registration deadline. 7- The ED sets the final submission deadline. 8- The ED sets additional configurations for scoring. 9- CBK checks the parameters inserted by the ED. 10- CKB create a new github repository 11- CKB creates a new Battle and redirect the ED to the new Battle main page. 12- CKB sends a notification to all STs subscribed to the Tournament
Exit condition	The Battle is correctly created and the ED is redirected to the Battle main page.

Exceptions	• The Battle's name is null or already exists. An error message is shown, the ED is redirected back to the create Battle settings. • The maximum number of students per group is lesser than the minimum. An error message is shown to the ED and the ED is redirected to the create Battle settings. • The minimum number of students per group is lesser or equal than zero. An error message is shown to the ED and the ED is redirected to the create Battle. • The registration deadline is a date before the day the Battle is created. An error message is shown to the ED and a new deadline has to be chosen and the ED is redirected to create Battle settings. • The final submission deadline is a date before the registration deadline. An error message is shown to the ED and a new deadline has to be chosen and the ED is redirected to create Battle settings. • The code kata is not valid. An error message is shown to the ED and the ED is redirected to create Battle settings.

Table 3.7: Create a Battle use case.

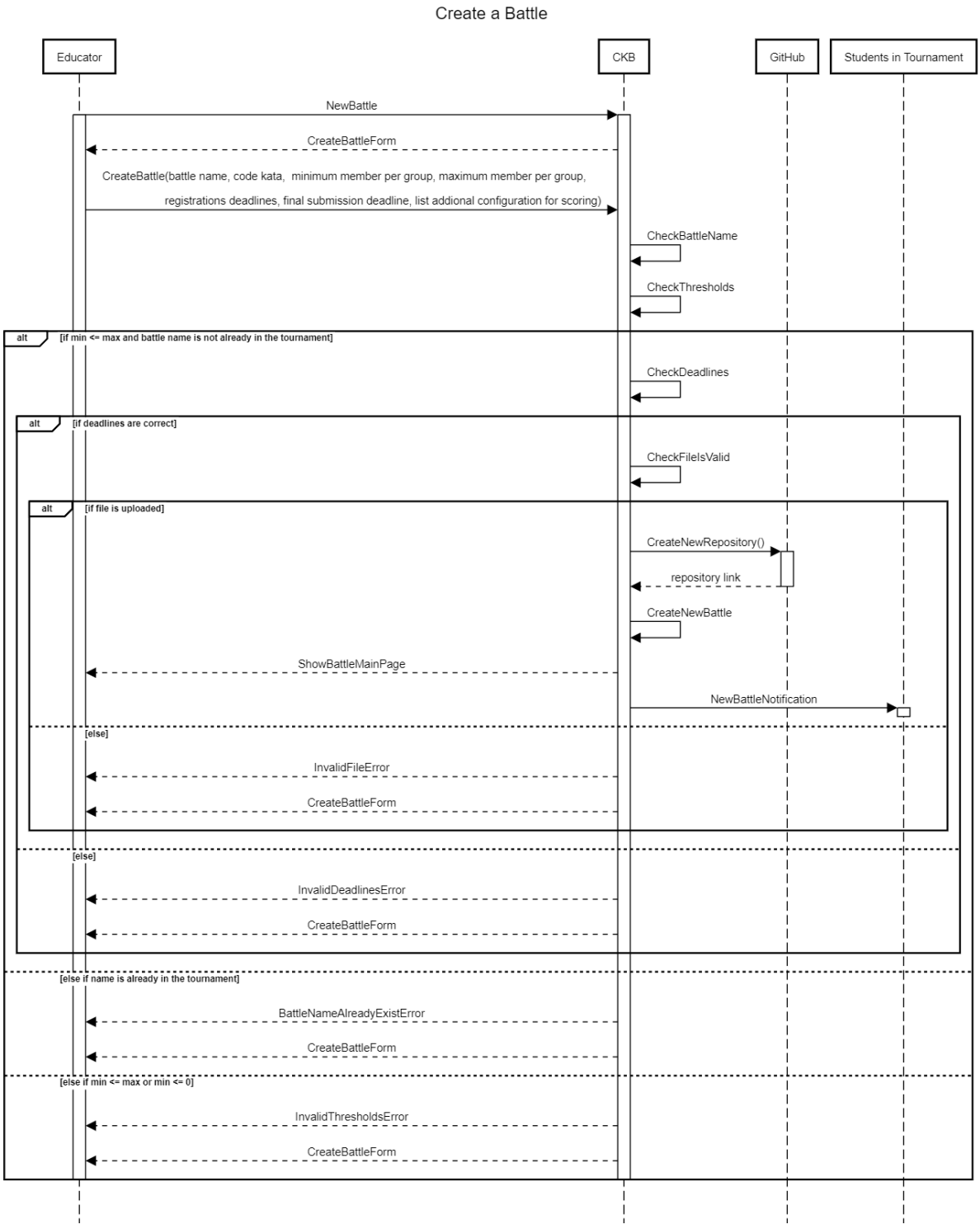

Figure 3.9: Create a Battle sequence diagram.

UC7. Join a Battle

Actor	ST
Entry conditions	The ST is correctly logged in and it is in the Tournament main page. The ST has decided to join a Battle.
Event Flow	1- The ST clicks on a tournament 2- CKB redirects the ST to the page that has the list of all the Battles of the Tournament (Tournament main page). 3- The ST clicks on the "Join Battle" button near the Battle name. 4- CKB checks that the deadline has not expired. 5- CKB checks that the ST joined the Tournament that contains this Battle. 6- CKB redirects the ST to the CreateGroup page of the Battle.
Exit condition	The ST, after the subscription deadline end, will be able to create a group and then confirm the participation to the Battle
Exceptions	• The ST isn't in the Tournament that this Battle is part of. In this case CKB shows an error message, the ST is redirected back to the Tournament main page. • The Battle already ended or already started. In this case CKB shows an error message, the ST is redirected to the final Leaderboard of the Battle. • The ST already joined the Battle. In this case CKB will not show any error messages, but will redirect the ST to the CreateGroup page.

Table 3.8: Join a Battle use case.

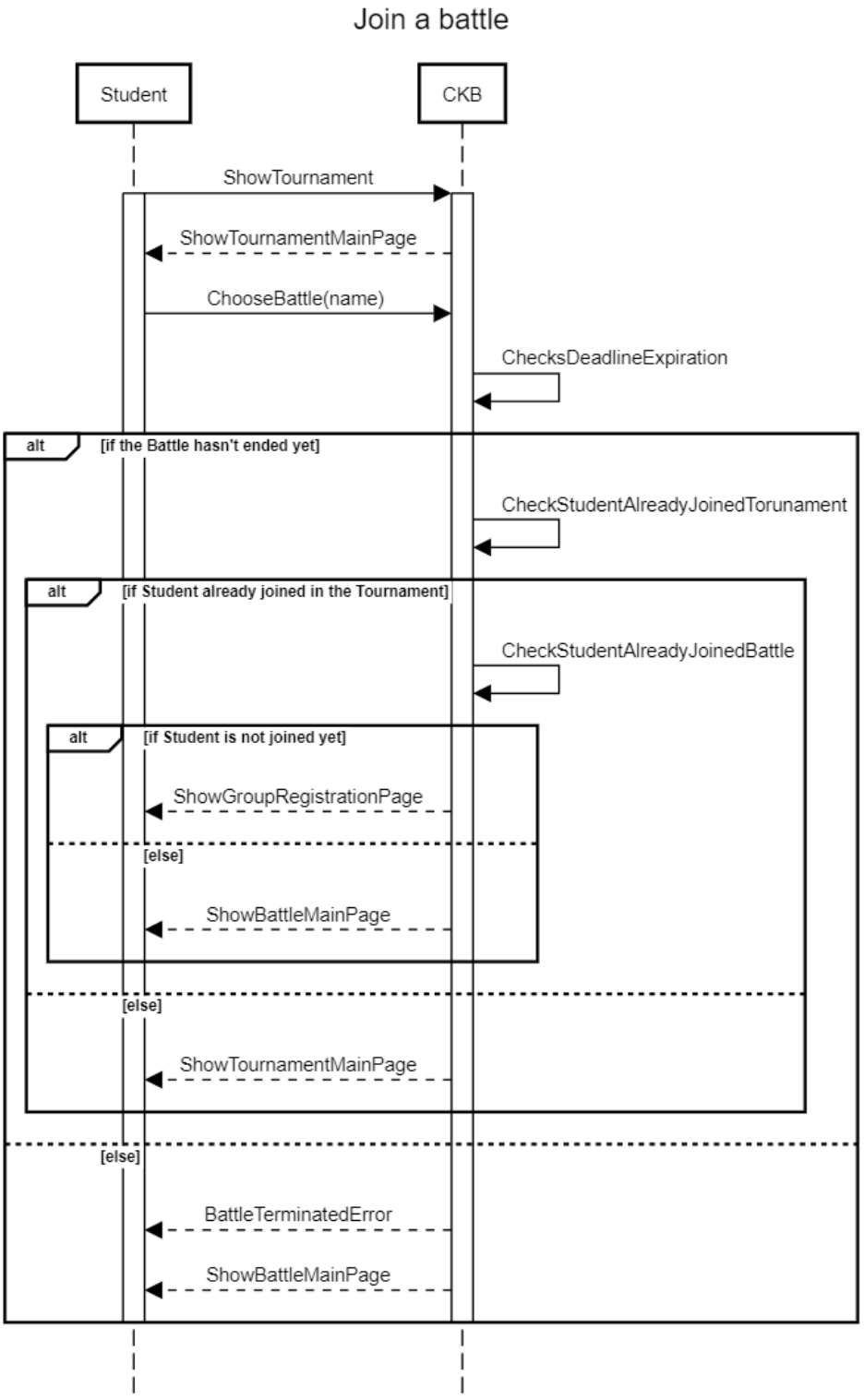

Figure 3.10: Join a battle sequence diagram.

UC8. Create a group and Battle confirmation

Actor	ST
Entry conditions	The ST is correctly logged in and it is in the CreateGroup page. The ST has decided to join a Battle.
Event Flow	1- The ST inserts the STG name. 2- The ST inserts another ST's nickname and clicks on the "Invite" button and then. 3- CBK checks if the ST joined the competition and notifies other students of an invitation. 4- The other STs can decide to accept or reject the invitation clicking on the "Accept" or "Reject" button inside the notification. 5- CKB shows to the ST that has created the STG the different state of the various invitation requests (3 state : accepted, rejected, pending)(only the first n-invited STs that accept the invite are considered part of the group)(where n is the maximum number decided by the ED in the create Battle form). 6- The ST can click on the "Confirm Group" button. 7- CKB checks if the number of members respects the limits given by ED during the Battle creation. 8- CKB saves the ST and the STG as one of the competitors of the Battle. 9- CKB shows a confirmation message and redirects the ST to the main page of the Battle.
Exit condition	The ST is now a competitor in the Battle and he can starts committing after the creation group deadline expires

Exceptions	• The STG name already exists in the Battle. In this case CKB shows an error message and the ST is redirected back to the CreateGroup page. • The STG already joined the Battle. In this case CKB will not show any error messages, but will redirect the ST to the current Leaderboard of the Battle. • The number of STs in the group is wrong. CKB will show an error and redirect the ST to the CreateGroup page. • If the creation group deadline expires, all the STs without group or all the STG not confirmed will be kicked out from the Battle. • If the invited ST has not already joined in the competition or if the nickname does not exist, CKB sends an error and redirects the ST to the CreateGroup page.

Table 3.9: Create a group and Battle confirmation use case.

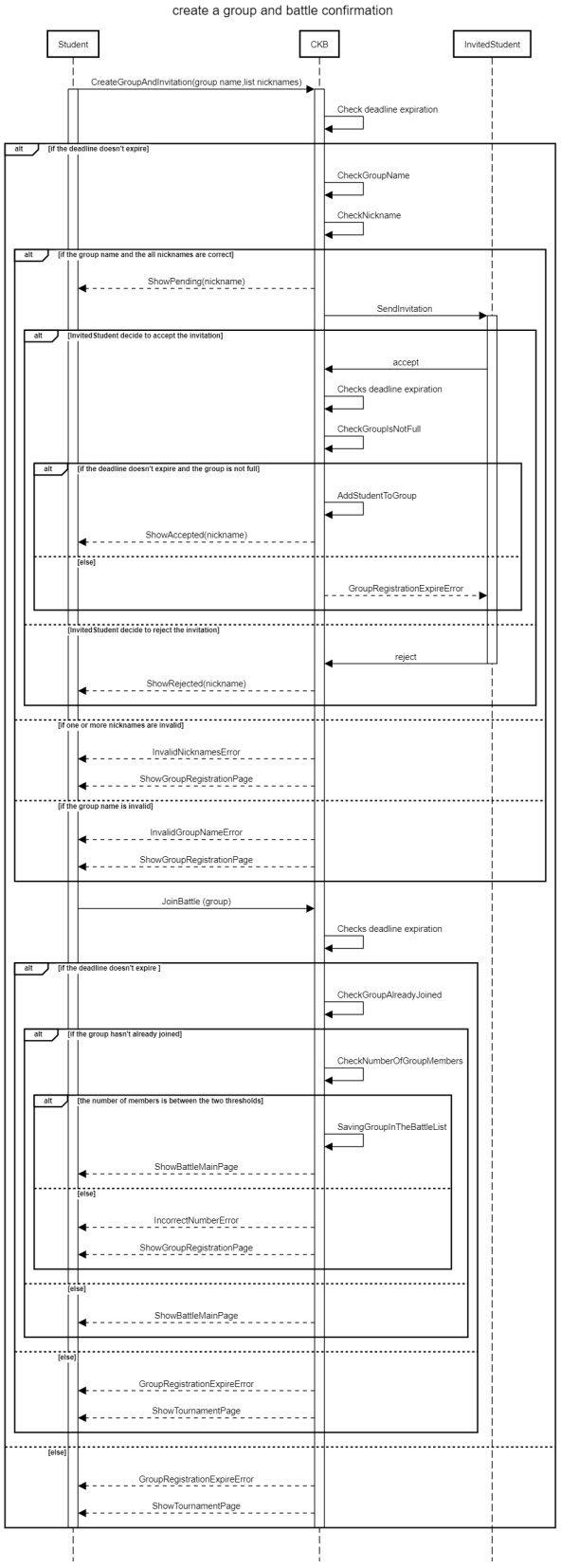

Figure 3.11: Create a group and Battle confirmation sequence diagram.

UC9. Open a profile

Actor	ST or ED
Entry conditions	The User is logged in. The User should be in a Leaderboard (both Battle or Tournament) page for retreiving a ST profile from it or an ED profile from the owner name of the Tournament (or from the list of all the EDs)
Event Flow	1- User clicks on a ST's nickname (from the Leaderboard) or ED's nickname (from the Tournament owner or from the list of all the EDs). 2- CKB returns the profile page of the selected User.
Exit condition	User is able to know particular information of the User that he had selected.
Exceptions	The User is not found. In this case the User will be redirected back to the previous page.

Table 3.10: Open a profile use case.

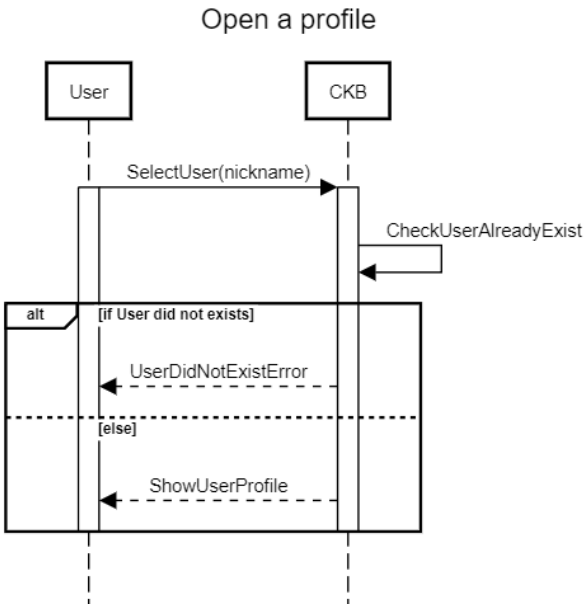

Figure 3.12: Open a profile sequence diagram.

UC10. Search for a profile

Actor	ST or ED
Entry conditions	The User is logged in
Event Flow	1- User clicks on the search bar. 2- User writes another User's nickname or a keyword of it. 3- CKB returns the list of the Users that contain the written keyword in their nickname.
Exit condition	User is able to find the User that he was searching.
Exceptions	None is found from the research. In this case the User will be redirectd back to the previous page.

Table 3.11: Search for a profile use case.

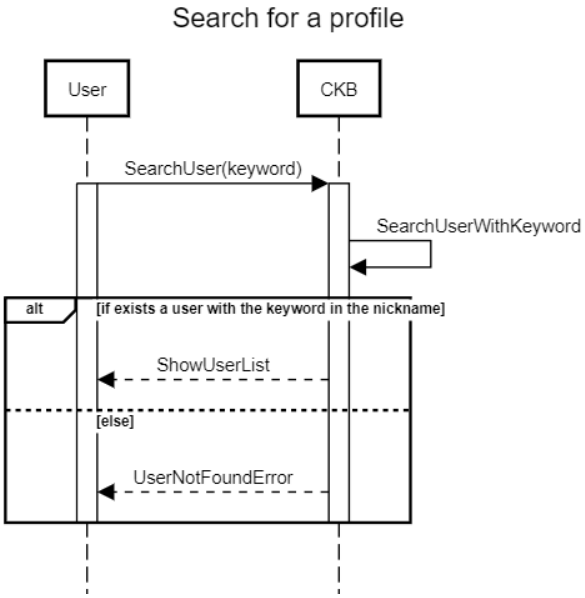

Figure 3.13: Search for a profile sequence diagram.

UC11. Search for a Tournament

Actor	ST or ED
Entry conditions	The User is logged in
Event Flow	1- User clicks on the search bar. 2- User writes a Tournament's name or a keyword of it. 3- CKB returns the list of the Tournaments that contain the written keyword in their name.
Exit condition	User is able to find the Tournament that he was searching.
Exceptions	Nothing is found from the research. In this case the User will be redirected back to the previous page.

Table 3.12: Search for a Tournament use case.

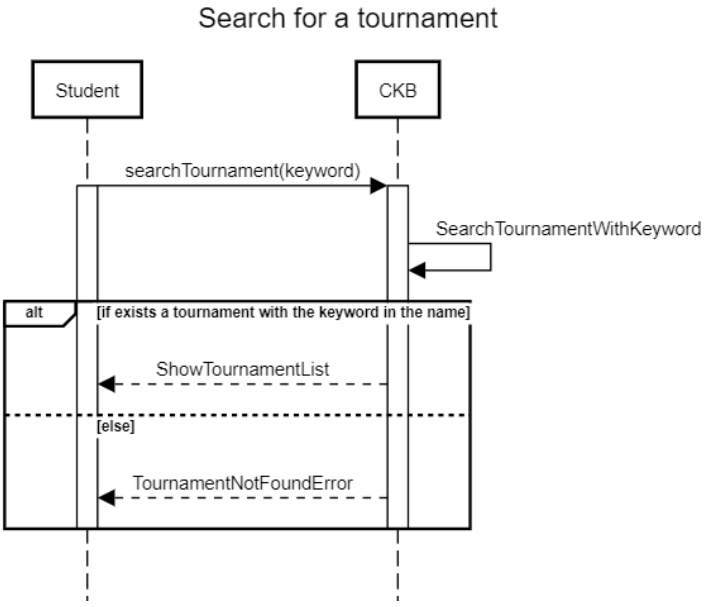

Figure 3.14: Search for a Tournament sequence diagram.

UC12. Evaluate a code

Actor	ED
Entry conditions	The ED is correctly logged. During the Consolidation Stage the Educator decides to check a STG's code.
Event Flow	1- The ED clicks on a battle in the Leaderboard (in Tournament main page) 2- CKB shows the Battle Main page. 3- The ED clicks the "Analyze source code" button that corresponds to a STG. 4- CKB shows the source code to the ED. 5- The ED manually checks the code. 6- The ED assign a score to the STG code.
Exit condition	CKB updates the new score to the STG code
Exceptions	• The Consolidation Stage already ended, an error message is shown and the ED is redirected to the battle Leaderboard page. • The ED awards the STG a improper score, like a score beyond the minimum and maximum bounds decided by the ED during the Creation Phase. An error message is shown and the ED is redirected to the battle Leaderboard page.

Table 3.13: Evaluate a code use case.

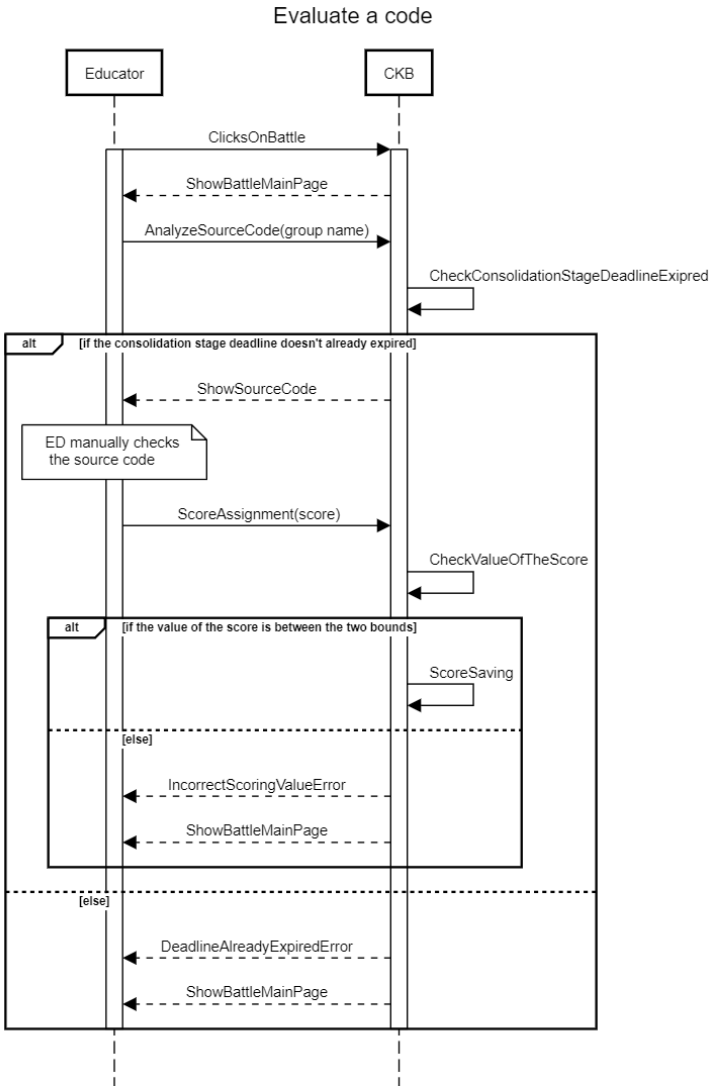

Figure 3.15: Evaluate a code sequence diagram.

UC13. Create a new Badge

Actor	ED
Entry conditions	The ED is correctly logged. During the Torunament creation the ED has ticked the "Create Badge" setting.
Event Flow	1- The ED inserts the Badge name, description and the rules to obtain a new Badge. 2- The ED clicks on the "Create Badge" button. 3- CKB checks the Badge's details. 4- CKB saves the Badge's details. 5- CKB return the Tournament main page.
Exit condition	CKB adds the Badge to the Tournament information.
Exceptions	• The Badge's name is null or already exists. An error message is shown and the ED is redirected back to the create Badge settings. • The Badge's description is null. An error message is shown and the ED is redirected back to the create Badge settings. • The Badge's rules are null. An error message is shown and the ED is redirected back to the create Badge settings.

Table 3.14: Create a new Badge use case.

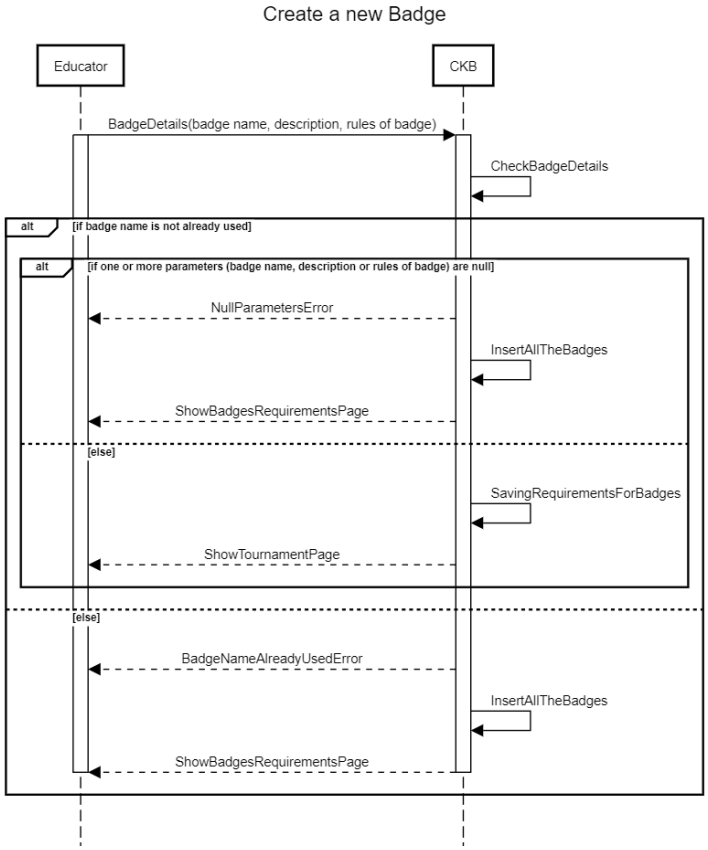

Figure 3.16: Create a new Badge sequence diagram.

UC14. Modify an existing Badge

Actor	ED
Entry conditions	The ED is correctly logged. During the Torunament creation the ED has ticked the "Create Badge" setting.
Event Flow	1- The ED clicks on a previously created Badge name. 2- CKB shows to the ED the "Modify Badge" form 3- The ED modifies the Badge name, description or the rules to obtain this Badge. 4- The ED clicks on the "Modify Badge" button. 5- CKB checks the Badge's details. 6- CKB saves the Badge's details.
Exit condition	CKB adds the modified Badge to the Tournament information.
Exceptions	• The modified Badge's name is null. An error message is shown and the ED is redirected back to the create Badge settings. • The modified Badge's description is null. An error message is shown and the ED is redirected back to the create Badge settings. • The modified Badge's rules are null. An error message is shown and the ED is redirected back to the create Badge settings. • The modified Badge's name does not belongs to the ED.An error message is shown and the ED is redirected back to the create Badge settings.

Table 3.15: Modify an existing Badge use case.

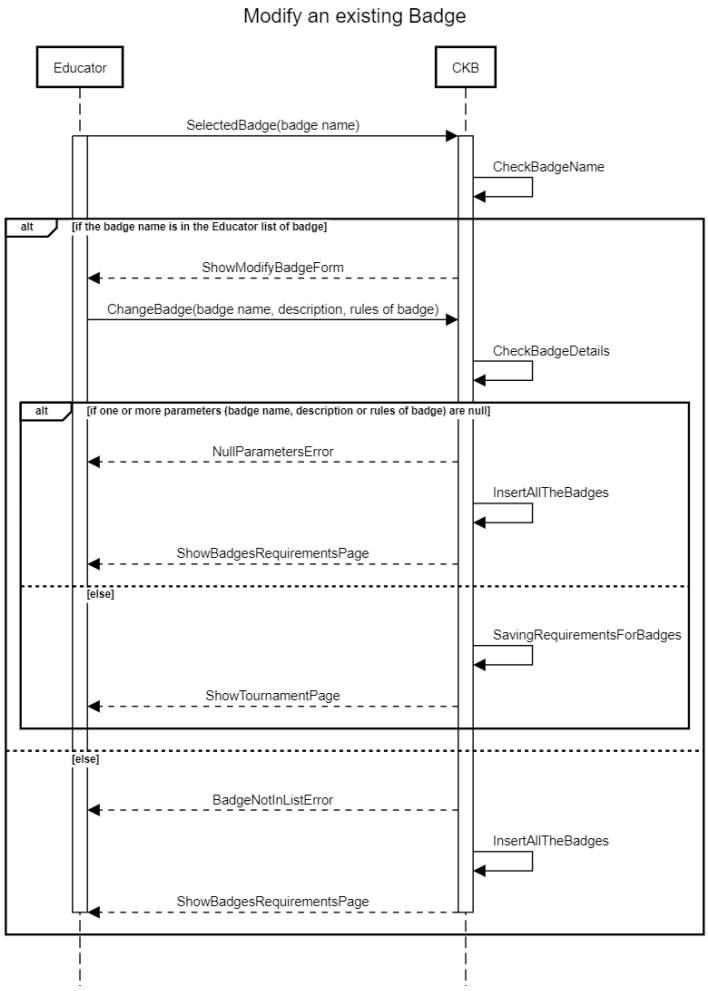

Figure 3.17: Modify an existing Badge sequence diagram.

3.2.4. Mapping on goals

G1 - Allow an ED to create a Tournament.

- R1: CKB allows unregistered Users to sign up.
- R2: CKB allows registered EDs to login.
- R4: CKB allows EDs to create Tournaments.
- R5: CKB allows EDs to grant the permissions of a Tournament to other EDs.
- R15: CKB allows EDs to choose which Badges to award in a certain Tournament.
- R17: CKB allows EDs to close a Tournament.
- R27: CKB sends notifications to every ST when a new Tournament is created.
- R50: CKB sends notifications to the ED when he receives the permission to create Battles in a Tournament.

G2 - Allow a ST to subscribe to a specific Tournament before the registration deadline.

- R1: CKB allows unregistered Users to sign up.
- R3: CKB allows registered STs to login.
- R20: CKB allows STs to join a Tournament.
- R39: CKB allows STs to check the list of ongoing Tournaments.
- R43: CKB sends notification to every ST involved in a Tournament when the Tournament is closed and the final ranks are available.
- R46: CKB shall communicate with the mailing system in order to allow Users to register their account.

G3 - Allow an ED to create a Battle.

- R6: CKB allows EDs to create Battles.
- R7: CKB allows EDs to uploads the code kata of a Battle.

- R8: CKB allows EDs to set the minimum and the maximum number of STs per group of a Battle.
- R9: CKB allows EDs to set a registration deadline of a Battle.
- R10: CKB allows EDs to set a submission deadline of a Battle.
- R11: CKB allows EDs to set additional configuration for the scoring system of a Battle.
- R12: CKB allows EDs to set functional aspects for the scoring system of a Battle.
- R16: CKB allows EDs to assign a score manually during the consolidation stage.
- R28: CKB sends notifications when a new Battle is created to every ST which is participating in the Tournament that the Battle is part of.
- R30: CKB creates a GH repository of the code kata when the registration deadline for the Battle expires.
- R31: CKB sends the link of the GH repository to every STG that participates in the Battle.
- R32: CKB evaluates the STG's work every time a push is made on GH and calculates Battle score for the STG.
- R33: CKB updates the Battle Leaderboard once a new score is registered.
- R46: CKB shall communicate with the mailing system in order to allow Users to register their account.

G4 - Allow any User to visualize the profile of other Users and their Badges.

- R18: CKB allows EDs to visualize the profile of another User.
- R19: CKB allows STs to visualize the profile of another User.
- R25: CKB stores the information about the Users.
- R26: CKB shall ensure security of data.

G5 - Allow a ST to subscribe to a specific Battle before the registration dead-

line.

- R21: CKB allows STs to join a Battle.
- R22: CKB allows STs to create a new STG.
- R23: CKB allows STs to join a STG.
- R24: CKB allows STs to invite other STs in their STG.
- R29: CKB sends notifications to a ST when he receives an invitation to be part of STG.
- R31: CKB sends the link of the GH repository to every STG that participates in the Battle.

G6 - Allow STs to submit their code before the submission deadline.

- R16: CKB allows EDs to assign a score manually during the consolidation stage.
- R32: CKB evaluates the STG's work every time a push is made on GH and calculates Battle score for the STG.
- R33: CKB updates the Battle Leaderboard once a new score is registered.
- R34: CKB updates the Tournament Leaderboard once a new score is registered.
- R37: CKB allows EDs to analyze the code of a STG.
- R38: CKB sends notification to every ST involved in a Tournament when the Tournament is closed and the final ranks are available.
- R44: CKB shall communicate with the GH API in order to calculate a new score every time a push action is made by a STG.
- R45: CKB shall communicate with the external tool in order to calculate the score of a STG.
- R47: STs need to fork the GH repository of the Battle they are participating in.
- R48: STs need to push their code in the GH repository in order to have their code evaluated.

G7 - Allow an ED to set Badges for deserving STs.

- R13: CKB allows EDs to create new Badges.
- R14: CKB allows EDs to choose the rules related to the awarding of Badges.
- R15: CKB allows EDs to choose which Badges to award in a certain Tournament.
- R49: CKB shall assign the Badges to all STs that fulfill their requirements.

G8 - Allow any User to view the result and the progress of a Tournament.

- R34: CKB updates the Tournament Leaderboard once a new score is registered.
- R40: CKB allows EDs to check the list of ongoing Tournaments.
- R41: CKB allows STs to check the Leaderboard of a Tournaments.
- R42: CKB allows EDs to check the Leaderboard of a Tournaments.
- R43: CKB sends notification to every ST involved in a Tournament when the Tournament is closed and the final ranks are available.

G9 - Allow any User to view the results of a Battle.

- R33: CKB updates the Battle Leaderboard once a new score is registered.
- R35: CKB allows STs to check the Leaderboard of a Battle.
- R36: CKB allows EDs to check the Leaderboard of a Battle.

Requirements	G1	G2	G3	G4	G5	G6	G7	G8	G9
R1	✓	✓							
R2	✓								
R3		✓							
R4	✓								
R5	✓								
R6			✓						

R7			✓						
R8			✓						
R9			✓						
R10			✓						
R11			✓						
R12			✓						
R13							✓		
R14							✓		
R15	✓						✓		
R16			✓			✓			
R17	✓								
R18				✓					
R19				✓					
R20		✓							
R21					✓				
R22					✓				
R23					✓				
R24					✓				
R25				✓					
R26				✓					
R27	✓								
R28			✓						
R29					✓				
R30			✓						
R31			✓		✓				
R32			✓			✓			
R33			✓			✓			✓
R34						✓		✓	
R35									✓
R36									✓
R37						✓			
R38						✓			
R39		✓							
R40								✓	
R41								✓	

R42								✓	
R43		✓						✓	
R44						✓			
R45						✓			
R46		✓	✓						
R47						✓			
R48						✓			
R49							✓		
R50	✓								

Table 3.16: Traceability Matrix for Goals and Requirements

3.3. Performance requirements

Number of concurrent Users: According to recent researches, websites with a similar goal of CKB, has about 1.8 million Users. We want our website to attract at least 25% of those Users, which means CKB should be able to handle up to 500,000 Users simultaneously. This is important for making sure our website works well for a good number of people, giving them a smooth and enjoyable experience.

Data storage: CKB needs to save and manage all the details about both STs and EDs. Additionally, it has to store information about all the Tournaments, including the associated Battles. When a Tournament is closed all the data about it won't be canceled and a reference of it and of the Battles remains in the ED's page .

Time response: Every operation that is directly executed by CKB, i.e. register, login, create, join and evaluate, should be in the domain of milliseconds. Other operations such as the ones that involve GH, the mailing system and the External Tools cannot be guaranteed by CKB itself.

3.4. Design constraints

3.4.1. Standard compliance

The system must be compliant to the EU's GDPR (General Data Protection Regulation), a set of regulations that is designed in order to protect the personal data, the privacy and security of the EU's citizens.

3.4.2. Hardware limitations

The only hardware limitations are the support for a reliable internet connection and for a Web Browser.

3.5. Software system attributes

3.5.1. Reliability

The system has to be fault tolerant in order to prevent the propagation of errors and to guarantee a continuous usability of the system.

3.5.2. Availability

The system must be available the most time possible, with a minimum value of 99.9% (three-nines) of time. In this way the system will be unavailable for only 8.76 hours a year.
It shall be prevented a case scenario in which a maintenance break occurs near to Battle's end, therefore there must be as few maintenance breaks as possible, with them possibly at nightime.

3.5.3. Security

The system must control the access rights of the Users. The system shall grant both authentication, verifying the identity of the Users that attempt to login and authorization, verifying the permission of the already logged Users to perform certain requested actions. Measures to protect the database will be adopted, such as defense against query injections, and password and Users' personal data stored will be encrypted.

3.5.4. Maintainability

The system must be designed using scalable and reusable models in order to permit future addition of features with minimum effort.
Ordinary maintenance has to be scheduled at nightime, in order to keep the services available when the User traffic is high.

3.5.5. Portability

The system must be accessible by the Users from every kind of Web Browser.
There are no particular portability requirements server side.

4 | Formal Analysis Using Alloy

Here is an alloy specification of the system described in the RASD:

```
abstract sig User{
   name: Name,
   surname: Surname,
   nickname: Nickname,
   email: Email,
   tournaments: set Tournament,
   battles: set Battle,
   password: Password
}


sig Student extends User{
   badges: set Badge,
   groups: set Group
}


sig Educator extends User{
}


sig Group{
   name: Name,
   members: set Student,
   highestScore: Score,
   battle: Battle,
```

```
    repo: one GitHubRepo,
}


sig Tournament{
    name: Name,
    creator: Educator,
    mods: set Educator,
    deadlineSub: Deadline,
    partecipants: set Student,
    rankings: set Student,
    stage: TournamentStage,
    battles: set Battle
}


sig Battle{
    name: Name,
    creator: Educator,
    belongsTo: Tournament,
    deadlineSubscribe: Deadline,
    deadlineSubmission: Deadline,
    manualEvaluationEnabled: Bool,
    partecipants: set Student,
    rankings: set Group,
    kata: Kata,
    availableBadges: set Badge,
    stage: BattleStage,
    mainRepo: OriginalRepo,
    forkedRepos: set GitHubRepo,
    minPerGroup: Min,
    maxPerGroup: Max,
    automatedRules: AutomatedRules,
}


sig OriginalRepo{
    link: Link,
    battle: Battle
```

```
}


sig GitHubRepo{
   forkedFrom: OriginalRepo,
   group: Group,
   link: Link
}


sig AutomatedRules{
   securityCheckOn: Bool,
   reliabilityCheckOn: Bool,
   maintainabilityCheckOn: Bool,
   percentageTestCasesToPass: Percentage
}


sig Badge{
   name: Name,
   variables: set Variable,
   rule: BadgeRule
}



abstract sig Name {}
abstract sig Surname {}
abstract sig Nickname {}
abstract sig Email {}
abstract sig Password {}

sig Kata{}
abstract sig Link {}
abstract sig Score {}
abstract sig Deadline {}
abstract sig Percentage {}
abstract sig Max {}
abstract sig Min {}
```

```
abstract sig TournamentStage{}
sig Subscribe extends TournamentStage{}
sig Open extends TournamentStage{}
sig Ended extends TournamentStage{}

abstract sig BattleStage{}
sig SubscribeTime extends BattleStage{}
sig GroupsCreation extends BattleStage{}
sig OpenStage extends BattleStage{}
sig EndedBattleStage extends BattleStage{}
sig Consolidation extends BattleStage{}

abstract sig Bool {}
sig True extends Bool {}
sig False extends Bool {}

sig Variable{}
sig BadgeRule{}



-----------------------------------------------------------------
-- FACTS
-----------------------------------------------------------------



-- A Student is a participant of the battle if and only if
-- he is listed in the set of participants of the battle .
fact StudInBattle{
   all s: Student |
      all b : Battle |
         s in b.partecipants
            iff
         b in s.battles
}
```

```
-- Every Tournament that is present in the Educator's tournaments set,
-- must have the Educator as a mod or as the creator.
fact EdTournamentsCorrespondence{
   all t: Tournament |
      all e: Educator |
         t in e.tournaments
            iff
         (
            e in t.mods or e = t.creator
         )
}


-- Every Tournament of the list of turnaments that the student joined,
-- must have the same Student in the participants list of the Tournament
fact allSubscribedStudentsArePartecipants{
   all t: Tournament |
      all s: Student |
         s in t.partecipants
            iff
         t in s.tournaments
}


-- Every Battle present in the Educators' battles set must have the same Educator
-- as the creator. The Educator must also be in the set of mods or the creator of
-- the Tournament that the Battle belongs to.
fact EdBattleCorrespondence{
   all b: Battle |
      all e: Educator |
         b in e.battles
            iff
         (
            e = b.creator
               iff
            (e in b.belongsTo.creator or e in b.belongsTo.mods)
```

```
        )
}
```

```
-- Every battle that is part of a certain tournament, must have the belongsTo
-- field set to that Tournament and the creator of the Battle must be the
-- creator or a moderator of the Tournament
fact BattlesInTournament{
   all b: Battle |
      all t: Tournament |
         b.belongsTo = t
            iff b in t.battles iff
         (
            b.creator in t.mods or   b.creator in t.creator
         )
}
```

```
-- All the Battles that are part of a Tournament must be listed in the set of
-- Battles of the Tournament.
fact BattlesInTournament{
   all b: Battle |
      all t: Tournament |
         b.belongsTo = t
            iff
         b in t.battles
}
```

```
-- There must be no Users using the same email or password
fact noDuplicatedUsers{
   no disj u1, u2: User |
      u1.nickname = u2.nickname
         or
      u1.email = u2.email
}
```

```
-- There must be no groups with the same name within a battle
fact noDuplicatedGroups{
   no disj g1, g2: Group |
      g1.battle = g2.battle
         and
      g1.name = g2.name
}
```

```
-- Every repository used by any group partecipating to a battle
-- must be a different fork of the same initial repository created for the battle .
fact RepoConstraint{
   all disj f1, f2: GitHubRepo |
      f1.forkedFrom = f2.forkedFrom
         iff (
            f1.group != f2.group
               and
            f1.link != f2.link
               and
            f1.group.battle = f2.group.battle
               and
            f1.group.battle = f1.forkedFrom.battle
               and
            f1.link != f1.forkedFrom.link
         )
}
```

```
-- There must be no battles with the same name within a Tournament
fact noDuplicatedBattles{
   no disj b1, b2: Battle |
      b1.name = b2.name
         and
      b1.belongsTo = b2.belongsTo
```

```
}


-- There must be no Original GitHub Repositories that share the same link.
--There must also be only one Repository for each Battle.
fact noDuplicatedOriginalRepos{
   no disj r1, r2: OriginalRepo|
      r1.battle = r2.battle
         or
      (r1.battle.mainRepo=r2)
         or
      (r1.battle.mainRepo.link=r2.link)
}


-- There must be no Tournaments with the same name
fact noDuplicatedTournaments{
   no disj t1, t2: Tournament |
      t1.name = t2.name
}


-- There must be no duplicated Forked GitHub Repository
fact noDoubleRepoLinks{
   no disj r1, r2: GitHubRepo|
      r1.link=r2.link
}


-- There must be no Groups that share the same Forked GitHub Repository
fact noMultipleGroupsShareRepo{
   no disj r1, r2: GitHubRepo|
      r1.group=r2.group
}
```

```
-- There must be no Original GitHub Repositories that share the same link
fact noOriginalReposShareLinks{
   no disj r1, r2: OriginalRepo|
      r1.link=r2.link
}
```

```
-- There must be no Forked Repository that has the same link of any Original Repository.
fact noDoubleLinksBetweenForkedOrigina{
   all r1: OriginalRepo |
      all r2: GitHubRepo |
         r1.link != r2.link
}
```

```
-- If a Student is a member of a Group, then the Battle that the group is subscribed
-- to must be one of the battles that the Student had previously joined.
fact GroupStudentBattlesCorrespondence{
   all g: Group |
      all s: Student |
         s in g.members implies g.battle in s.battles
}
```

```
-- There must be no Groups that have the same Forked Repository.
fact GroupsForkedReposCorrespondence{
   no disj g1, g2: Group |
      g1.repo = g2.repo
}
```

```
-- Groups that are subscribed to the same Battle must have no Students in common.
fact noCommonStudentsInDifferentGroups{
   all disj g1, g2: Group |
      g1.battle = g2.battle
         implies
      #(g1.members & g2.members) = 0
}


-- The correlation between Students and Groups are expressed both in the
-- members set in Group and in the groups set in Student
fact s_in_g{
   all g: Group |
      all s: Student |
         g in s.groups iff s in g.members
}



-- The correlation between Students and Tournaments are expressed both in the
-- tournaments set in Student and in the participants set in Tournament
fact s_in_T{
   all s: Student |
      all t:Tournament |
         s in t.partecipants iff t in s.tournaments
}



-- The Students that participate to a Battle must be also
-- participants of the Tournamnet that the Battle belongs to .
fact s_in_B_T{
   all s: Student |
      all b: Battle |
         b in s.battles implies s in b.belongsTo.partecipants
}
```

```
-- All the Groups that participate to a Battle
-- must be in the Rankings of the same Battle.
fact GroupInRankings{
   all g: Group |
      all b: Battle |
         g.battle = b iff g in b.rankings
}



-- There must be no group without members.
fact NoEmptyGroups{
   all g: Group |
      #(g.members) >0
}



-- The instances of the Group and their repository must be correctly linked.
fact GroupRepoConsistency{
   all g: Group |
      all r: GitHubRepo |
         g.repo = r iff r.group = g
}



-- The Forked Repositories mist be linked to the same Battle
-- that their original Repositories are linked to.
fact ForkedOriginalConsistency{
   all g: GitHubRepo |
      all o: OriginalRepo |
         g.forkedFrom = o
            implies
         g in o.battle.forkedRepos
}
```

```
-- Every Student must be in a Group for every Battle he is in.
fact BattleGroupsCardinality{
   all s: Student | #s.battles = #s.groups
}



-- Every Student must be in a Group for every Battle he is in.
fact BattleGroupsConsistency{
   all s: Student |
      all b: Battle |
         b in s.battles
            iff
         one g: Group |
            b = g.battle and s in g.members
}



-- Every Battle in the Student's battles set
-- must have the Student in the partecipant set.
fact allSubscribedStudentsArePartecipantsBattle{
   all b: Battle |
      all s: Student |
         s in b.rankings.members
            iff
         b in s.battles
            iff
         #(b.rankings & s.groups) = 1
}



-- Every Group's Repository must be a Forked Repository of the Battle's Main Repository.
fact groupReposConsistency{
   all g: Group |
      g.repo in g.battle.forkedRepos
         and
      g.repo.forkedFrom = g.battle.mainRepo
```

```
}


-- Every Group of a Student must have him listed in the set of its members.
fact StudentGroupsMembersConsistency{
   all s: Student |
      all g: Group |
         g in s.groups
            iff
         ( s in g.members )
}


-- Every Badge that is awarded to a Student must be one of the Badges that
-- could be awarded in the Battles that he previously partecipated to
fact StudentsBadges{
   all s: Student |
      all b: Badge |
         b in s.badges implies b in s.battles.availableBadges
}


-- Every Forked Repository must be linked to and only to the same Battle
-- that its Original Repository is linked to.
fact RepoLinkedToBattle{
   all r: GitHubRepo |
      one b: Battle |
         r in b.forkedRepos
            and
         b = r.forkedFrom.battle
}
```

```
-- No Battle can have the same Forked Repositories as another Battle.
fact NoCommonForkedRepos{
   no disj b1, b2: Battle |
      #(b1.forkedRepos & b2.forkedRepos) >0
}
```

```
-----------------------------------------------------------
-- ASSERTIONS
-----------------------------------------------------------
```

```
assert assertion1{
   all s: Student |
      all b: Battle |
        b in s. battles
           implies
        ( s in b.partecipants
           and
          s in b.belongsTo.partecipants
           and
          b.belongsTo in s.tournaments
           and
         (# (s.groups & b.rankings)=1)
        )
}
check assertion1 for 5
```

```
assert assertion2{
   all s: Student |
      all g: Group |
        s in g.members
           implies
        g in s.groups
```

```
        and
    g.battle in s.battles
        and
    g in g.battle.rankings
        and
    s in g.battle.partecipants
        and
    #(g.battle.rankings & s.groups)=1
        and
    g.repo in g.battle.forkedRepos
        and
    g.repo.forkedFrom = g.battle.mainRepo
        and
    s in g.battle.belongsTo.partecipants
        and
    g.battle.belongsTo in s.tournaments
}
check assertion2 for 5



-- PREDICATES


-- In the baseWorld there is only one Tournament, one Group, two Battles,
-- two Students and one Educator. This is one of the simpliest possible
-- worlds that can be generated.
pred baseWorld{
  #Tournament = 1
  # Group = 1
  # Student = 2
  # Battle = 1
  # Educator = 1
}
run baseWorld for 5
```

```
-- In this World there is still only one Tournament,
-- but there are three Groups of Students and there are two different Battles.
pred World{
   #Tournament=1
   #Group = 3
   #Student>2
   #Educator>1
   #Battle = 2
   #Badge >4
   #AutomatedRules = #Battle
}
run World for 5
```

```
-- In this World there are two Tournaments, each of them having at least one Battle.
pred World2{
   #Tournament=2
      all t: Tournament | #(t.battles) >0
   #Student=4
   #Battle =2
}
run World2 for 5
```

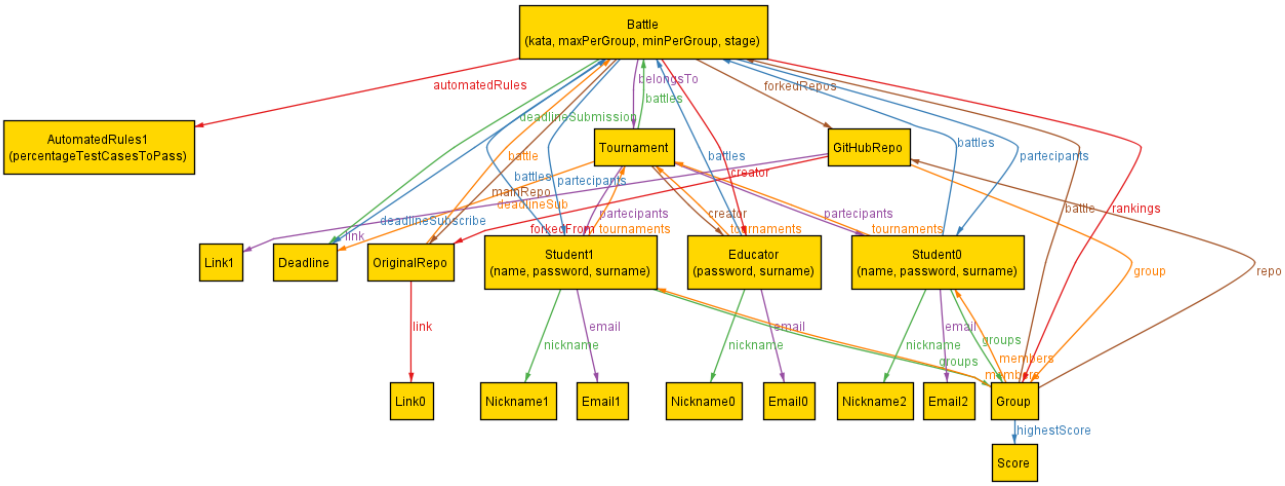

Figure 4.1: Instance for *baseWorld*.

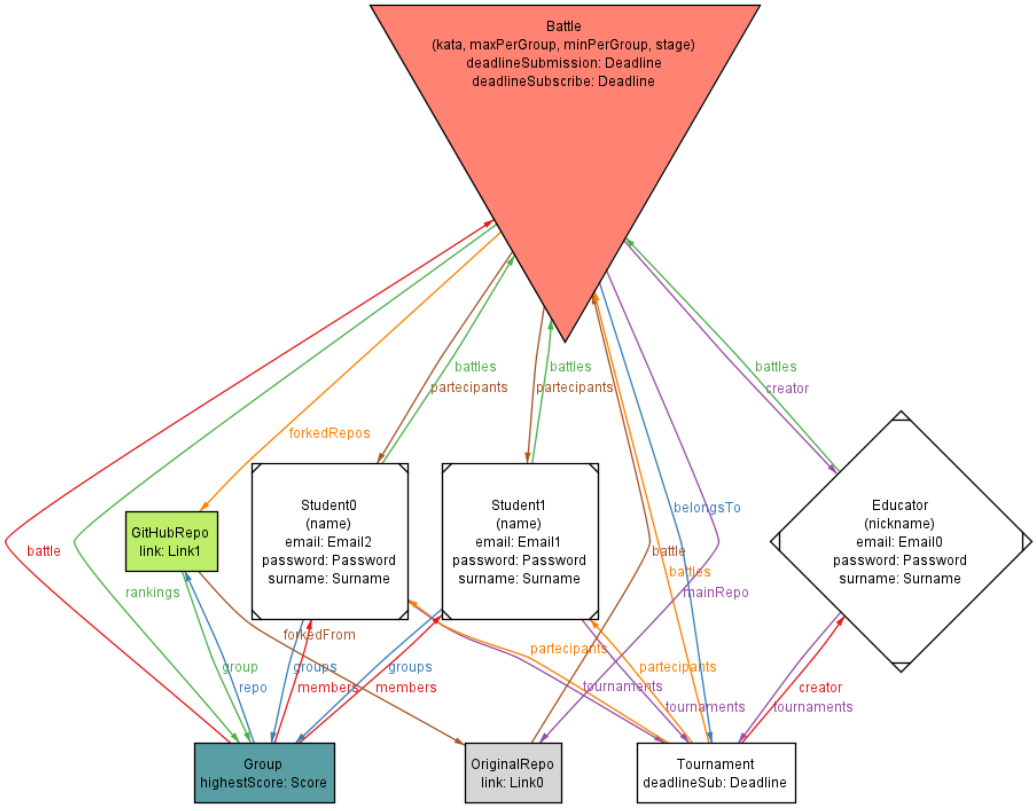

Figure 4.2: Simplified Instance for *baseWorld*.

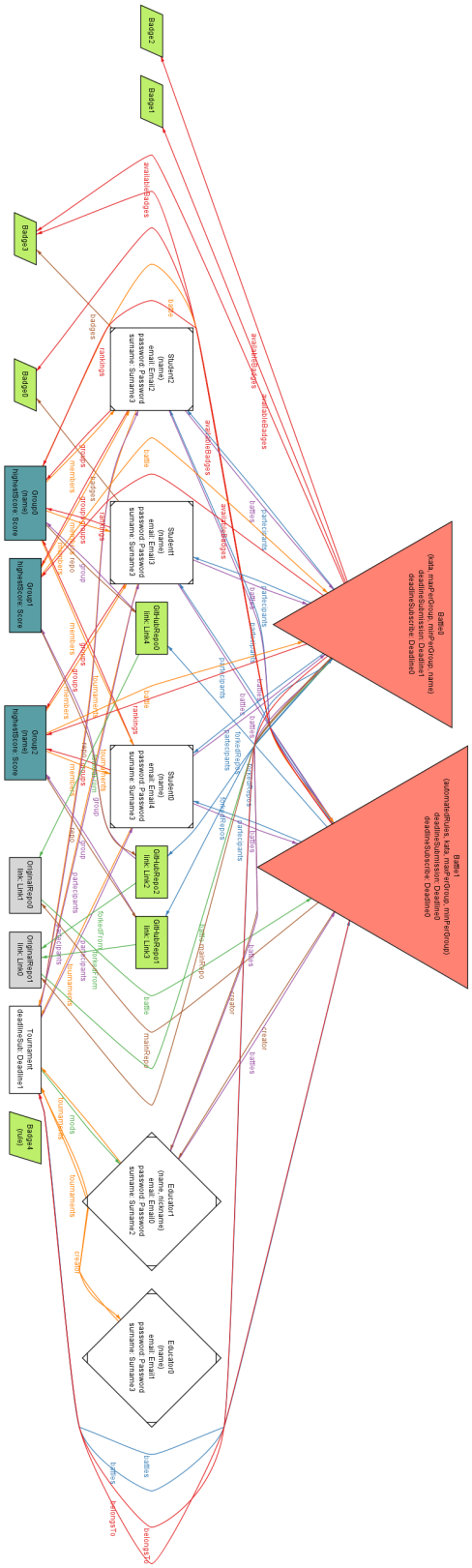

Figure 4.3: Simplified Instance for *World*.

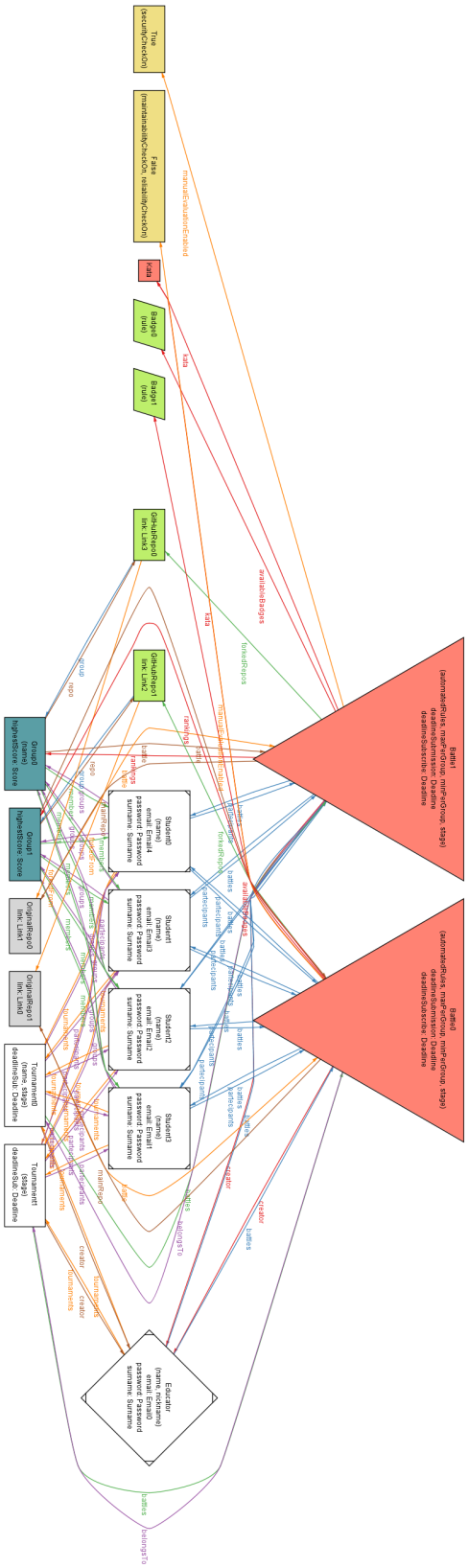

Figure 4.4: Simplified Instance for *World2*.

5 | Effort Spent

Member of group	Effort spent	
Ballabio Giacomo	Introduction	1.5h
	Overall description	10h
	Specific requirements	11h
	Formal analysis	1.5h
	Reasoning	3h
Benelle Francesco	Introduction	2h
	Overall description	6.5h
	Specific requirements	9h
	Formal analysis	15h
	Reasoning	3.5h
Cavallotti Alberto	Introduction	2h
	Overall description	6h
	Specific requirements	16h
	Formal analysis	1.5h
	Reasoning	3.5h

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

6 References

6.1. References

- ISO/IEC/IEEE 29148:2018 - Systems and software engineering - Life cycle processes - Requirements engineering.
- The Requirement Engineering and Design Project specification document A.Y. 2023–2024.

6.2. Used Tools

- GitHub for project versioning and sharing.
- LaTeX and *Visual Studio Code* as editor for writing this document.
- *sequencediagram.org* for the sequence diagrams' design.
- *draw.io* for the other diagrams' design.
- *Alloy* for formal analysis.
- *Google Documents* for collaborative writing, notes and reasoning.

List of Figures

List of Tables