Accademic Year: 2023-2024

POLITECNICO
MILANO 1863

Requirements Analysis and Specification Document

CodeKataBattle

SOFTWARE ENGINEERING 2

PROFESSOR DI NITTO ELISABETTA

CONTI ALESSANDRO CODICE PERSONA: 10710583 MATRICOLA: 252665

DI PAOLA ANTONIO CODICE PERSONA: 10717589 MATRICOLA: 956038

Contents

1. INTRODUCTION

1.1. Overview

The following document aims at providing an overview of the project CodeKataBattle. This document will guide the reader in understanding the purpose of the project, on which grounds it is set and in which environment it will live. Precisely, it will illustrate both informally and formally which are the goals and how these may be reached by guaranteeing certain functional and nonfunctional requirements, while certain phenomena hold in the world. The document will provide a baseline for the project's planning and estimation and will support the software's evaluation. Finally, it will be the starting point for the software's review and update. The document is addressed to all those who intend to benefit from the service and who may be involved in its development.

1.2. Purpose

The purpose of our project, named CodeKataBattle (abbreviated as CKB), is to provide a digital platform that enables students to develop and refine their skills in software development through a collaborative experience. The CKB platform allows students to practice in a programming language of their choice. The exercises offered on this platform are created by educators and require students to complete the code or parts of it, adding the missing components they deem appropriate. The code is then executed and subjected to specific tests created by educators to verify its correctness.

Exercises are organized into tournaments by educators, within 'battles' are held to evaluate the performance of the students. This process culminates in the creation of performance rankings among all students involved in the battles and tournaments. In this way, CKB encourages friendly competition and the growth of students' skills in the field of software development.

1.2.1. Goal

[G1] Allow registered users as educators to create and manage tournaments and battles to offer enrolled students the opportunity to improve their skills in the field of software development.

[G2] Allow users registered as students to participate in tournaments and battles of their interest to enhance their skills in software development. Student joins a battle in a tournament they are register for and invite some friends to participate with them, thus creates a team for the battle.

[G3] Provide an opportunity for students to receive an evaluation for their implementation concerning the required problem, considering different parameters.

1.3. Scope

Following the definition originally proposed by M. Jackson and P. Zave in 1995, we will try to clarify the scope and the application range of the System by identifying the so-called World and Machine phenomena.

The Machine is the System to be developed, in this case a software, while the World, also known as environment, corresponds to the part of the real world that is affected by the System.

The World and the Machine are associated with different types of events which we are going to describe and detail.

1.3.1. World phenomena

Phenomena events that take place in the real world and that the machine cannot observe.

[WP1]. Educator wants to create a tournament.

[WP2]. Educator wants to create a battle.

[WP3]. Student wants to join a tournament.

[WP4]. Student contacts mates participating in the tournament to form a team.

[WP5]. Students fork the GitHub repository.

[WP6]. Students and Educator wants to subscribe into the platform.

[WP7]. The student, who is participating in a battle that is still ongoing, is working independently on the code he will have to submit for assessment.

1.3.2. Shared phenomena

1.3.2.1. Controlled by the machine and observable by the world

[SP1]. The platform assigns scores to groups to create a competition rank.

[SP2]. Students are notified when a new tournament is created.

[SP3]. All student that are subscribed on a tournament are notified of all upcoming battles creation.

[SP4]. The platform creates a GitHub repository containing the Code Kata and then sends the link to all students who are members of subscribed teams.

[SP5]. When the battle ends the system evaluates the student automatically.

[SP6]. Each push before the deadline triggers the CKB platform, which pulls the latest sources, analyses them, and runs the tests on the corresponding executable to calculate and update the battle score of the team.

1.3.2.2. Controlled by the world and observable by the machine

[SP7]. Students join into a tournament.

[SP8]. Educator creates a tournament.

[SP9]. Educator creates a battle.

[SP10]. Educator gives permission to other ones to create battle in the tournament.

[SP11]. Group of student joins into a battle.

[SP12]. Groups deliver their solution to the platform.

[SP13]. Students set up the automated workflow in GitHub so that informs the CKB as soon as students push a new commit.

[SP14]. At the end of the battle, optionally the educator evaluates the student by overwriting the result that the platform returned.

1.3.3. Machine phenomena

Phenomena that the machine performs.

[MP1]. CKB assigns automatically a grade to a student's code by assigning a grade based on performance.

1.4. Definitions, Acronyms, Abbreviations

1.4.1. Definitions

- *Commit*: Commits are the core building block units of a Git project timeline. Commits can be thought of as snapshots or milestones along the timeline of a Git project. Commits are created with the '*git commit*' command to capture the state of a project at that point in time.
- *Fork*: A fork is a new repository that shares code and visibility settings with the original "upstream" repository.
- *Code Kata*: A code kata contains the description of a battle and the software project on which the student will have to work, including also test cases and build automation scripts.
- *Test Case*: A test case is a singular set of actions or instructions that the Educator wants to perform to verify a specific aspect of the project pushed by the students. If the test fails, the result might be a software defect that students might have not found.
- *Upload*: Upload in which the educator sends to the platform database the code kata for a specific battle.
- *Repository*: A Git repository is the *.git/folder* inside a project. This repository tracks all changes made to files in your project, building a history over time. Meaning, if you delete the *.git/folder*, then you delete your project's history.
- *Branch*: In Git, a branch is a new/separate version of the main repository. Branches allows users to work on different parts of a project without impacting the main branch. That main branch is the one seen as the default one.
- *Push*: The '*git push*' command is used to upload local repository content to a remote one. Pushing is how you transfer commits from your local repository to a remote one.
- *Pull*: The '*git pull*' command is used to fetch and download content from a remote repository and immediately update the local one to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.

1.4.2. Acronyms

- CKB: CodeKataBattle.
- CK: CodeKata, Code Kata.
- GH: GitHub.
- GHA: GitHub Action.
- GDPR: General Data Protection Regulation.

1.4.3. Abbreviations

- [Gn]: the n-th goal of the system.
- [WPn]: the n-th world phenomena.
- [SPn]: the n-th shared phenomena.
- [MPn]: the n-th machine phenomena.
- [Sn]: the n-th scenario.
- [DAn]: the n-th domain assumption.

- [Dn]: the n-th dependency.
- [Cn]: the n-th constraint.
- [Rn]: the n-th functional requirement.
- [UCn]: the n-th use case.
- [SDn]: the n-th sequence diagram.
- Repo: Git repository.

1.5. Revision history

- Version 1 (2023/12/21)

1.6. Reference documents

This document is strictly based on:

- The specification of the RASD and DD assignment of the Software Engineering 2 course, held by professor Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli at the Politecnico di Milano, A.Y. 2022/2023.
- Slides of Software Engineering 2 course on WeBeep.
- ISO/IEC/IEEE 29148 - Systems and software engineering.
- Official link of CodeKata: http://codekata.com/.

1.7. Documents Structure

The rest of the document is organized as follows:

- *Overall Description* (Section 2) contains an in-depth description of the domain under analysis, following the Jackson-Zave approach used in the Scope paragraph in Section 1. Class Diagrams will be hereby provided to formalize all the actors involved in the System and their relations, while their needs and scope as Stakeholders will be analysed later in the section. Key functional requirements will then be listed, together with all the Domain Assumptions that can be used to entail each Goal.
- In the *Specific Requirements* (Section 3) functional requirements are associated to use cases and Sequence Diagrams to clarify processes and interactions between users and the CKB. Nonfunctional requirements are also included here, together with external constraints (mainly regulations) imposed on the System.
  Developers can also find requirements regarding External Interfaces, with a focus on Hardware and Software Interfaces. Finally, a brief overview of the properties that the CKB will have to guarantee is provided.
- *Formal Analysis* (Section 4) uses Alloy to generate a formal model of two distinct parts of CodeKataBattle, the identification and requests.

2. OVERALL DESCRIPTION

2.1. Product perspective

In this section we will explain in more detail the various World, Machine and shared phenomena. We will also provide some of the descriptions with State and Class Diagrams.

2.1.1. Scenarios

[S1]. Educator creates a new tournament

Greg and Paul are two computer science teachers who want to assess their students' programming skills. While Greg was browsing the internet, he discovered that CKB platform could be suitable software for this purpose. As a result, he decided to register on the platform and asked Paul to do the same. To complete the registration, he had to fill out a form with his personal information and received a confirmation via email once the registration was completed.

Subsequently, Greg logged into the platform by entering the email address and password used during registration on the login page. After Greg logged into the application, he created a new tournament, granting Paul permission to create battles within the newly created tournament. After that the platform notifies all user registered as Student that a tournament has been created.

Finally, Greg asked all his students to register on the platform as students and join in the tournament he had created.

[S2]. Educator creates a battle

John has created a tournament and now wants to add a new battle to it. To do this, he has prepared a detailed description of the activities to be performed, has set up a code with some parts to be completed, established evaluation criteria, chosen the group size, and set the registration deadline.

Now, John can access his profile to actually create the challenge on the platform. From the main page, he selected the tournament he just created and filled out the form with the previously prepared information. At this point, the platform will notify all the students registered for the tournament, and those interested have time until the deadline to form a group and register.

[S3]. Student joins a tournament

Ted wants to improve his programming skills, and a friend suggests trying the CKB platform. Ted trusts his friend and decides to register as a student on the platform. Now that he has created his profile, he can access the platform and start searching for a tournament to join.

From his homepage, Ted can view and search for all the available and active tournaments. In case he does not find a tournament that meets his needs or there are no active tournaments, the platform will inform him by sending an email when a new tournament is created.

When Ted finds a tournament suitable for his needs, all he has to do is register before the established deadline, and he can start improving his programming skills.

[S4]. Student joins a battle

Ann, along with her friends Liz and Eddie, received a notification from the CKB platform about the creation of a new battle in the tournament organized by Mr. Smith. They decided to join as a team, and Ann initiated the process. To do so, Ann logged into the platform and, from Mr. Smith's tournament page, found the newly created competition, registered, and invited her two friends to join her to form a group.

Once the registration deadline had passed, they received an email from the platform containing a link to their GitHub repository created specifically for this battle. Eddie, being the first to view and open the email, accessed the GitHub repository and, following the

instructions, forked the repository and modified the GHA so that every new commit made by her or her friends would be visible to the CKB evaluation platform.

[S5]. Student push a commit in the main branch

Mary is participating in a battle, and after making a change to the project in a secondary branch and ensuring that everything works correctly, she decides to merge her branch with the main branch. After confirming that the branch merge was successful, Mary makes a commit that activates the GHA, allowing the CKB platform to evaluate the changes made.

The proposed solution is assessed positively, and this enables Mary to achieve a score of 95/100. If she wants to further improve her score, she can make additional code changes by making additional commits before the battle's deadline.

[S6]. Last battle ends and educator closes the tournament

Tom is an educator who has created a tournament on the CKB platform. He has already assigned six battles, and the seventh has just concluded.

All the students who participated in the seventh battle were automatically notified of its completion and can view a draft of their scores and the related ranking. However, Tom noticed that Alice used a highly maintainable solution by implementing some design patterns. He wishes to reward her with an extra 5 bonus points and proceeds to modify the scores. Once the interval for score modifications has ended, a new email is sent to all participants with their final scores and rankings.

At this point, the results of this battle are added to the overall results of the students in the tournament, thus affecting the general ranking.

Tom decides that seven battles are more than sufficient for this tournament and closes it, allowing the platform to automatically send an email to all registered students to inform them of the tournament's closure along with the final rankings and scores.

2.1.2. Domain class diagram

Figure 1 illustrates the High-Level Class Diagram associated with CKB. This diagram encompasses all the components within the system's domain and illustrates the interactions among these elements. The subsequent section provides detailed descriptions of the key components of the diagram to enhance comprehension of the domain.

- We identify the User class, which represents the users who interact with the application. This class is divided into two subclasses that delineate the two types of users in the system: the Student and the Educator.
- It can be noted that the Educator class is linked to the Tournament class because it is created by the educator, and the Student class is also connected as students actively participate by registering for the tournament.
- The Tournament class is associated with the Battle class since multiple battles between students take place within a tournament. Students enrol in battles as teams, hence the Team class depicted in the diagram.
- Given that each battle involves a task to be completed, code to be filled in, test cases to pass and parameters of particular interest to the Educator, all this information, entered by the educator who created the battle, is stored in the CodeKata class. This class is linked to the Battle class, highlighting the connection between the programming challenge specifications and the actual battle event.
- The evaluation of a student is stored in the Evaluation class, which is linked to the ongoing battle. Each time a new code is added for evaluation, the Battle class retrieves the necessary information from the CodeKata class, assesses the student's work, and saves the evaluation in the Evaluation class.

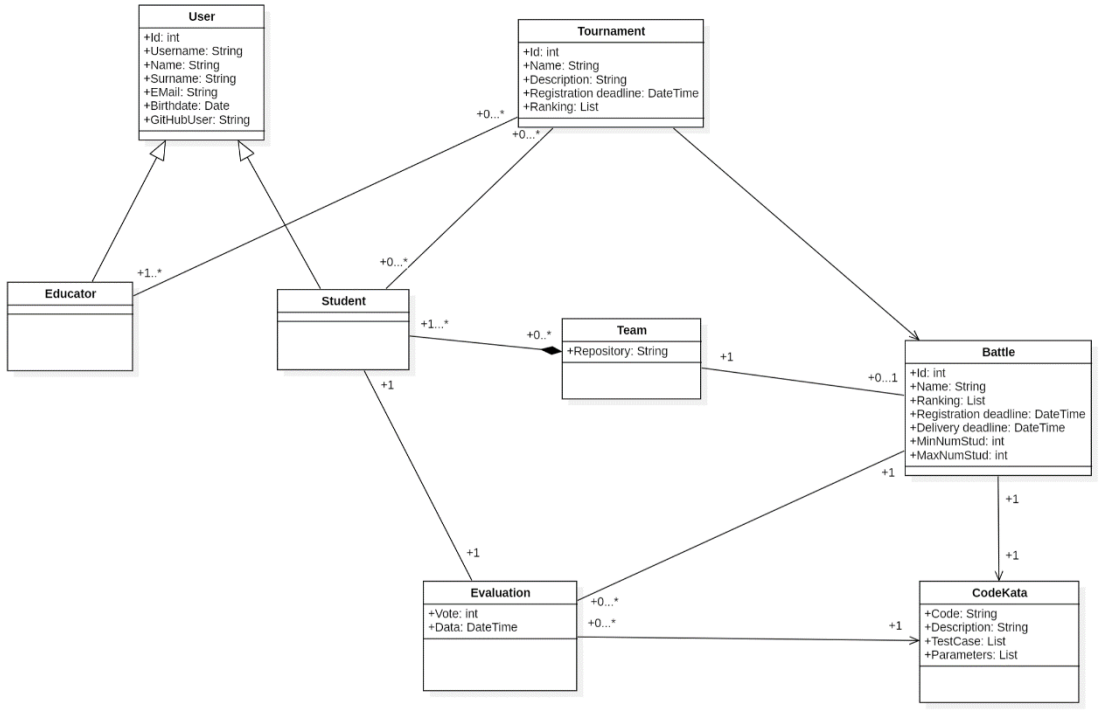


Figure 1 Domain Class Diagram

2.1.3. State Diagrams

2.1.3.1. Tournament state diagram

a. *Registration*: In the registration phase, all the operations that occur from the moment the Educator begins entering the data to create the tournament until the deadline for student enrolment are included.
b. *On Going*: The system enters this state if there is at least one participant in the tournament when the enrolment deadline has expired and it persists until a battle is initiated.
c. *Battle*: This state handles the management of battles; from the moment an educator begins entering data for the first battle until the tournament is completed.
d. *Ended*: This state is responsible for concluding the tournament, which can occur either because no student has enrolled or because an overseeing educator decides to close it.

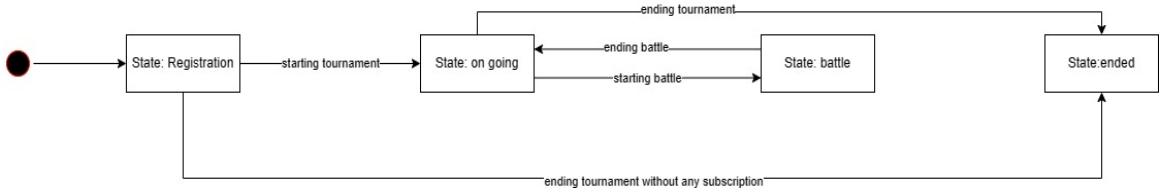


Figure 2 Tournament State Diagram

2.1.3.2. Battle state diagram

a. *Registration*: In the registration phase, all the operations that occur from the moment the Educator begins entering the data to create the battle until the deadline for student enrolment are included.
b. *On Going*: The system enters this state if there is at least one participant in the battle when the enrolment deadline has expired and it persists until the battle is ended.
c. *Ended*: This state is responsible for concluding the battle, which can occur either because no student has enrolled or because the battle deadline occurs.

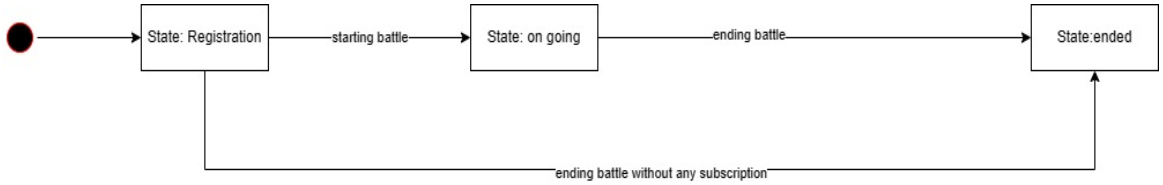


Figure 3 Battle State Diagram

2.2. Product functions

This section is devoted to analysing the Value Proposition that CodeKataBattle provide to its users. The presented functionalities are grouped according to the user they address.

2.2.1. Sign up and Login

These functions will be available to all users.

The registration functionality allows users to sign up for the system. Each user will need to choose whether to register as a student or an educator, and they will need to provide an email address, a password, and some personal information. Once the registration is completed, the system will send a confirmation email, officially enrolling the user in the CKB platform.

The login functionality allows registered users to access the platform and start using it.

2.2.2. Manage a tournament

One of the main functions of the platform is to comprehensively manage tournaments. The platform allows educators to create tournaments and involve their colleagues as collaborators for their management. After creating a tournament and setting the registration deadline, the platform will send an email notification to all registered students, informing them of the creation of a new tournament and offering them the opportunity to register.

Furthermore, when an educator creates a new battle or when the overall ranking is updated, the platform will send notifications to all participating students, keeping them informed about any changes that occur within the tournament to which they are registered.

2.2.3. Manage a battle

One of the main functions of the platform is to comprehensively manage all aspects of battles within a specific tournament. The platform allows educators to create a battle, including a detailed description of the exercise to be completed, a code with parts to be filled in by students, evaluation criteria, the group size, and a registration deadline.

After creating the battle, the platform will send a notification to all students registered for the tournament, allowing them to enrol. When the deadline is exceeded, the platform will automatically create GitHub repositories containing the exercise to be completed. When a student activates the GHA, allowing the platform to receive all the commits made by them, it will evaluate the solutions according to the criteria chosen by the educator.

At the conclusion of the battle, the platform will generate a ranking based on the evaluations of all students and notify the results. Subsequently, within a new deadline, the educator will have the opportunity to modify the scores assigned to individual students. After the deadline for modifying grades has passed, the platform will add the results obtained in this battle to the scores from previous battles, altering the overall tournament leaderboard.

2.3. User characteristics

CKB targets two types of users: Student and Educator.

2.3.1. Student

A Student is a user eager to hone his skills in software development and decides to enrich his education by registering on the CKB application.

After successfully registering on the application, the Student has the opportunity to immerse himself in challenging tournaments. Once registered in these tournaments, he can engage in battles against other Students equally eager to improve their skills. In this competitive environment, the user has the opportunity to compare his results with those of his peers, creating a dynamic and collaborative environment for learning and developing software development skills.

2.3.2. Educator

An Educator is a user who shares their skills in software development with other users eager to improve their skills.

After registering, the Educator has the ability to create tournaments, organising programming battles within them in which registered Students can participate to hone their skills. The Educator oversees the tournament by monitoring the codes submitted by the Students to the platform, providing additional evaluation in addition to the automatic evaluation. This process creates a collaborative environment where students can receive direct and targeted feedback to further improve their software development skills.

2.4. Assumption, Dependencies and Contraints

2.4.1. Domain Assumption

[DA1]. Users must have a valid email address.

[DA2]. Students must have a GH profile.

[DA3]. Students need to know how to access GitHub, how to fork a repository and how to enable GHA.

[DA4]. Each user will create an account.

[DA5]. Personal data given by user during the registration process are assumed truthful.

[DA6]. User must have an internet connection.

[DA7]. Educator must upload the correct CK.

[DA8]. It is assumed that interaction with GH is always possible.

2.4.2. Dependency

[D1]. The CKB will make use of the GHA service offered by GitHub user account.

[D2]. The CKB will access GitHub APIs to create new repositories and add Students.

2.4.3. Constraint

[C1]. The CKB must abide to the GDPR regulation with regards to the treatment of personal data.

[C2]. The CKB must be implemented as a web site.

3. SPECIFIC REQUIREMENTS

3.1. External interface

3.1.1. User interface

In this paragraph, we will provide an overview of the graphical interface of the application.

- This image represents an overview of how we will then go about developing the Home Page of the application. This page allows you to choose whether you want to register or access the application

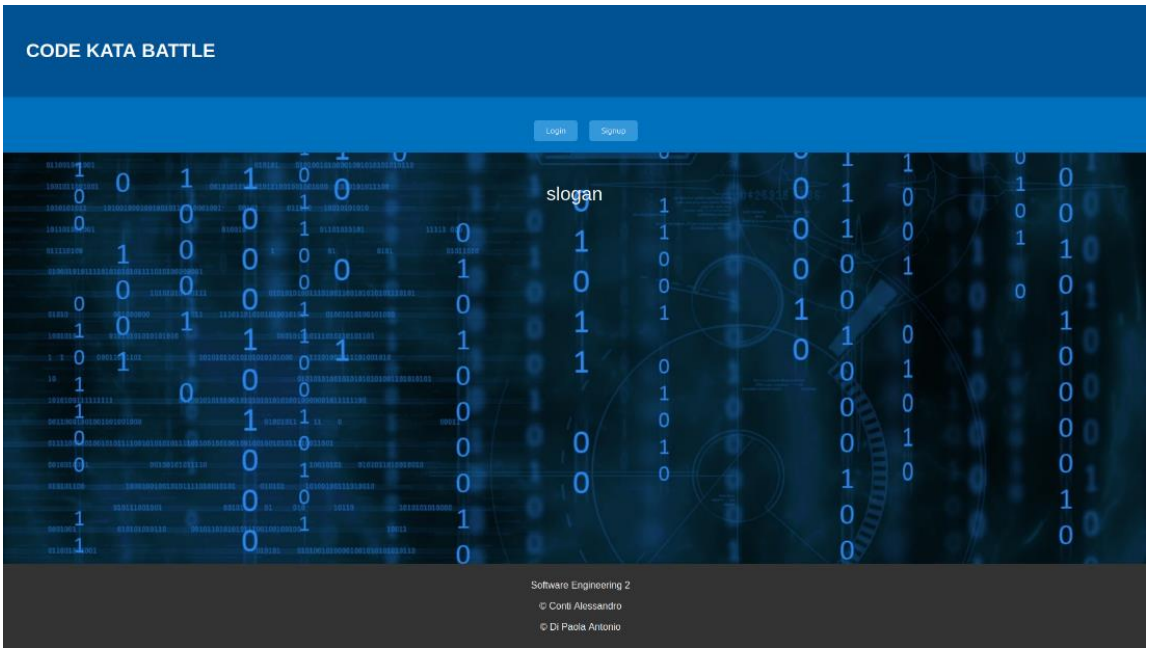


Figure 4 Main Page

- After logging into the application, users will be presented with two similar pages, but with differences based on their roles as Educators or Students. Whether you are an Educator or a Student, the page will always display a search bar, allowing you to search for new tournaments and view all the tournaments you are enrolled in. In the case of a user logging in as an Educator, there is an additional option compared to Students—an extra button that, when clicked, directs them to the page dedicated to creating a new tournament.

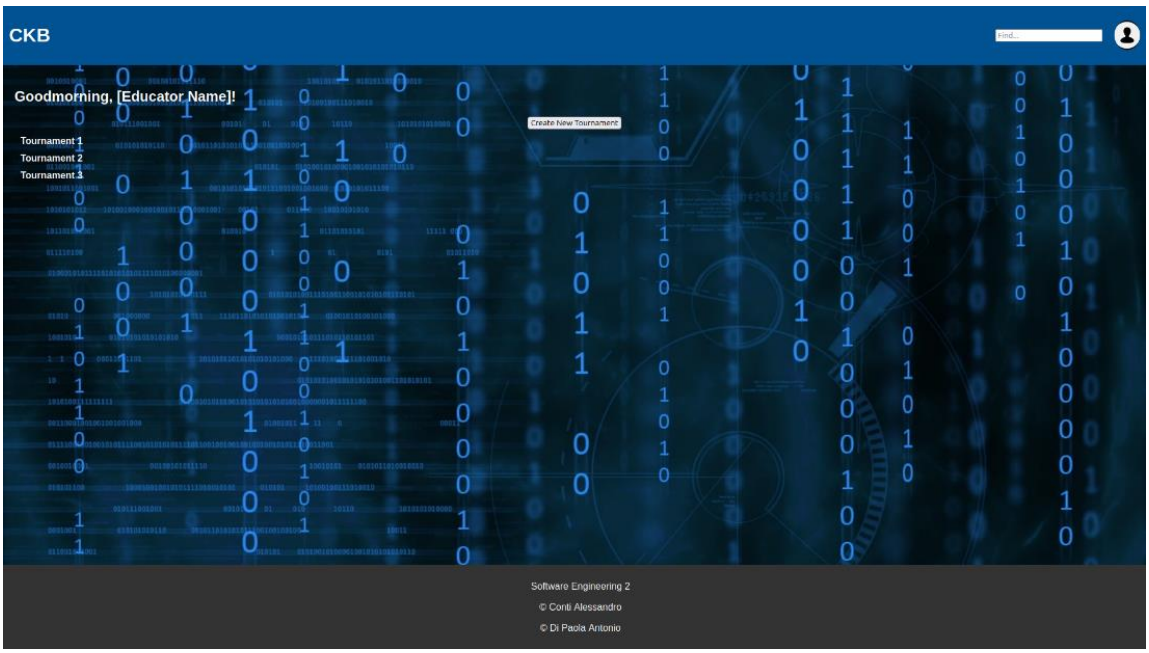


Figure 5 Educator's Home Page

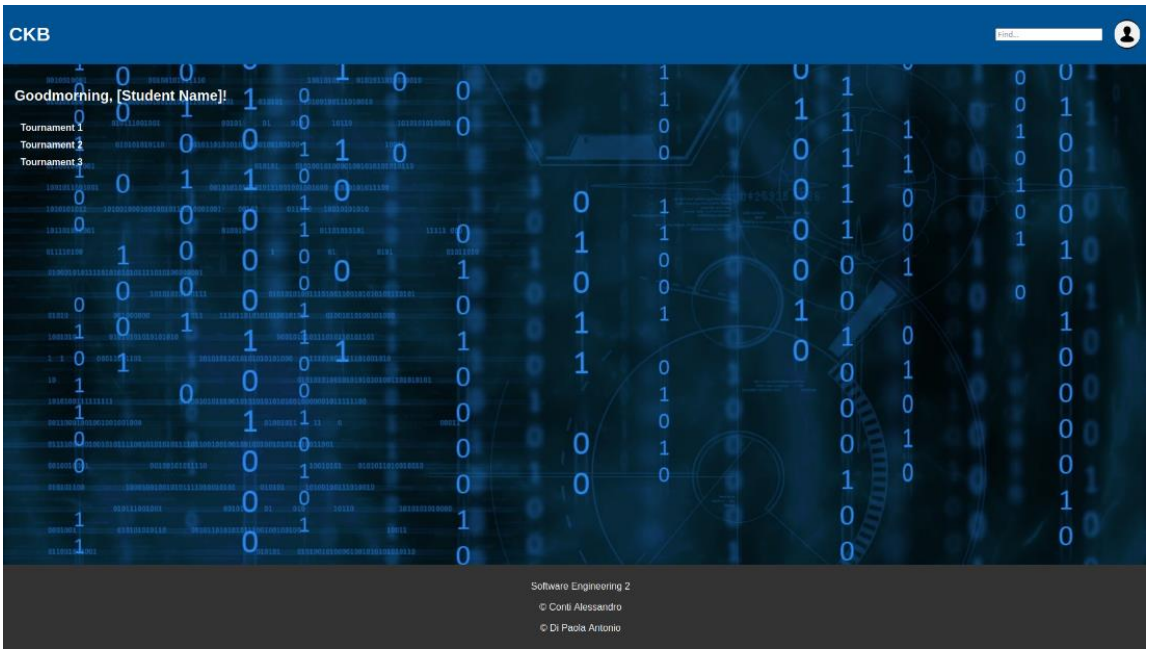


Figure 6 Student's Home Page

- Once a user selects a tournament to view, either through the search bar or via links to tournaments they are enrolled in, they will be redirected to a page displaying all relevant details about that tournament. In the event that the user is an Educator responsible for managing the tournament, their page will include a button enabling the creation of new battles; otherwise, this option will not be available.
  If the user is a Student who has not yet enrolled in the tournament and the registration deadline has not expired, the page will display a button allowing them to sign up.

In other scenarios, the page will only provide details about the tournament without additional options.

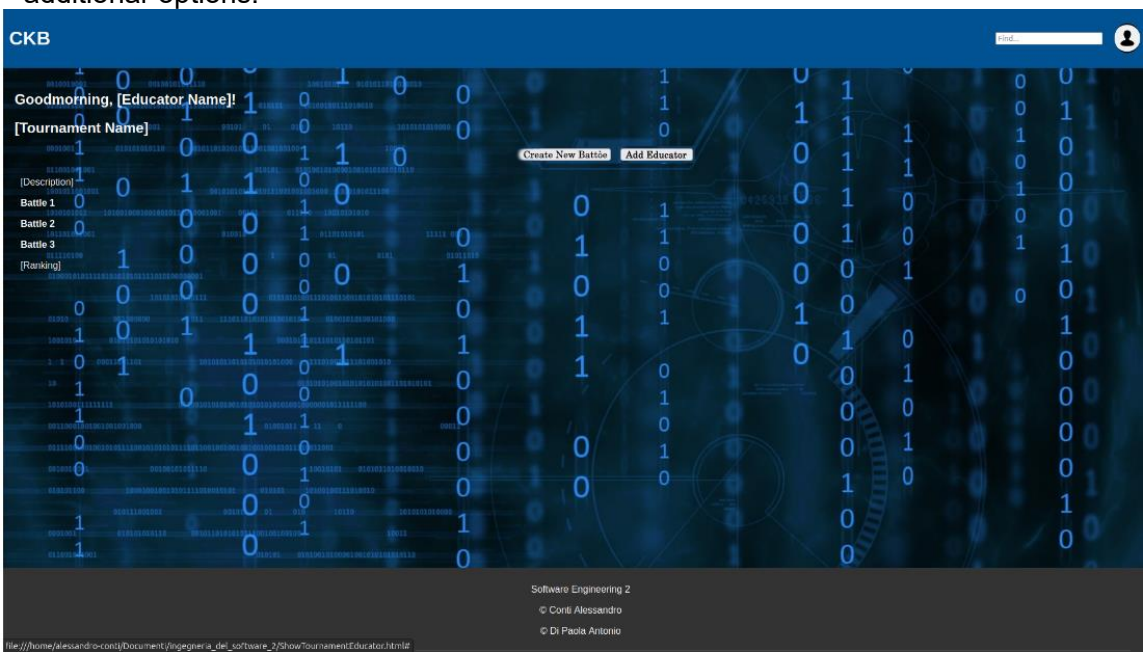


Figure 7 Page that shows a tournament to which the Educator is responsible for managing.

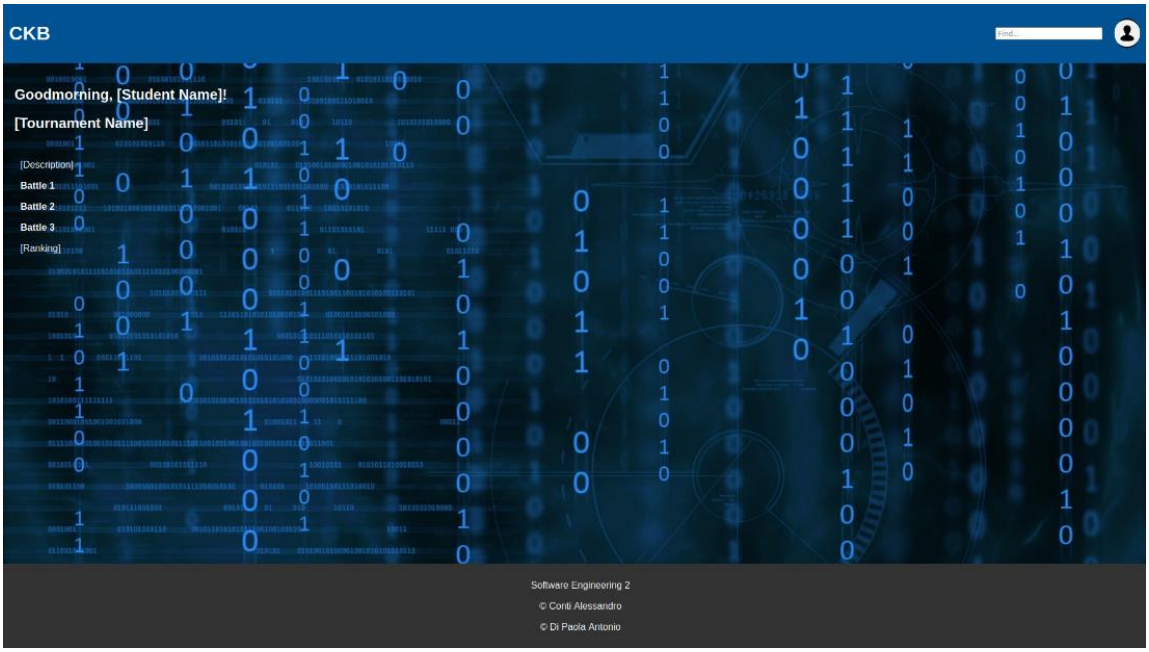


Figure 8 Page that shows a tournament in which the Student is registered.

- Once a user selects a battle to view through its link within their tournament, they will be redirected to a page presenting all the details related to that battle. In the event that the user is a Student enrolled in the tournament but not yet in that specific battle, the page will display the registration button as long as the registration deadline has not passed.
  In other scenarios, the page will only provide details about the battle without additional options.

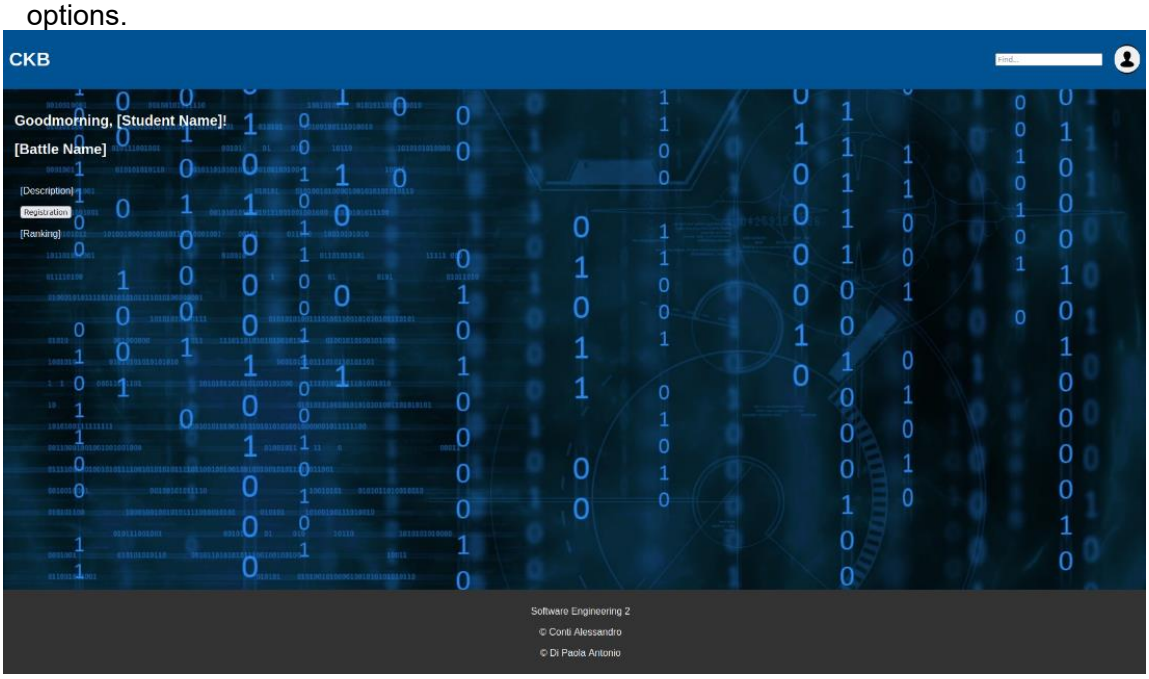


Figure 9 Page showing a battle that the Student can join.

3.1.2. Hardware interface

The application does not require any specific hardware interface as long as you have a computer that connects to an Internet network and a browser that can read and run JavaScript.

3.1.3. Software interface

The application requires two software interfaces to optimize communication management and simplify the evaluation process of student projects.

First, the use of the email API enables direct interaction with mail handlers, allowing automatic notifications to be sent to relevant users without requiring external user intervention. This functionality improves the efficiency of communications and ensures timely dissemination of relevant information.

Interfacing with the GH platform is implemented through GHAs. When properly configured, GHAs allow student projects to be automatically transferred to the CKB application for evaluation. This automated approach simplifies, automates and speeds up the assessment process provided to students.

3.1.4. Communication interface

The communication protocol we use is the HTTPS protocol; it is important to use it so that we have secure communication through the Internet.

3.2. Functional requirements

All the requirements that the system must fulfil are specified below.

In order to work properly, the system should:

[R1]. System allows students to sign up.

[R2]. System allows educators to sign up.

[R3]. System allows students to login.

[R4]. System allows educators to login.

[R5]. System allows educator to create a new tournament and to add all the needed information.

[R6]. System allows educator to grant the permission to other colleagues to create battle in a specific tournament.

[R7]. System allows educator to create a battle inside a particular tournament by adding CK and the deadlines.

[R8]. System allows educator to see rankings of each tournament, which he created or in which he has been nominated collaborator.

[R9]. System allows educator to see ranking of a specific battle.

[R10]. System allows educator to modify the evaluation manually of a specific group in a battle.

[R11]. System allows educator to close a tournament.

[R12]. System notifies students about the creation of a new tournament.

[R13]. System notifies students about the creation of a new battle in a specific tournament in which they have subscribed.

[R14]. System notifies students about publication of final ranking of a specific tournament.

[R15]. System notifies students about publication of final ranking of a specific battle.

[R16]. System allows students to join a tournament.

[R17]. System allows students to join a battle.

[R18]. System allows students to see rankings of each tournament, which they are subscripted.

[R19]. System allows students to see ranking of a specific battle.

[R20]. System, at end of the registration period of a battle, create a GH repo and send invitation to all member of the group.

[R21]. System automatically evaluates the code in the main branch of the repo after each commit in it.

[R22]. System allows students to search tournament by key words.

[R23]. System allows interaction with GH platform.

3.2.1. Forms

This section explains the form prompts present within the CKB application.

3.2.1.1. Sign in

- Role
- Name
- Surname
- Birthdate
- Username
- Email
- Password with confirmation
- Confirm password
- User GitHub
- Acceptance of privacy policy.

3.2.1.2. Login

- Username or email
- Password

3.2.1.3. Create a tournament

- Name
- Description
- Registration deadline

3.2.1.4. Create a battle

- Name
- Description
- Registration deadline
- Submission deadline
- CodeKata
- Test case
- Minimum number of students in a team
- Maximum number of students in a team

3.2.2. Use cases diagram

The diagram of use cases derived from the scenarios described above is shown below. These diagrams are useful to show the actors interacting with the system and to understand their roles.

3.2.2.1. Student use case diagram

Figure 10 Student Use Case Diagram

3.2.2.2. Educator use case diagram

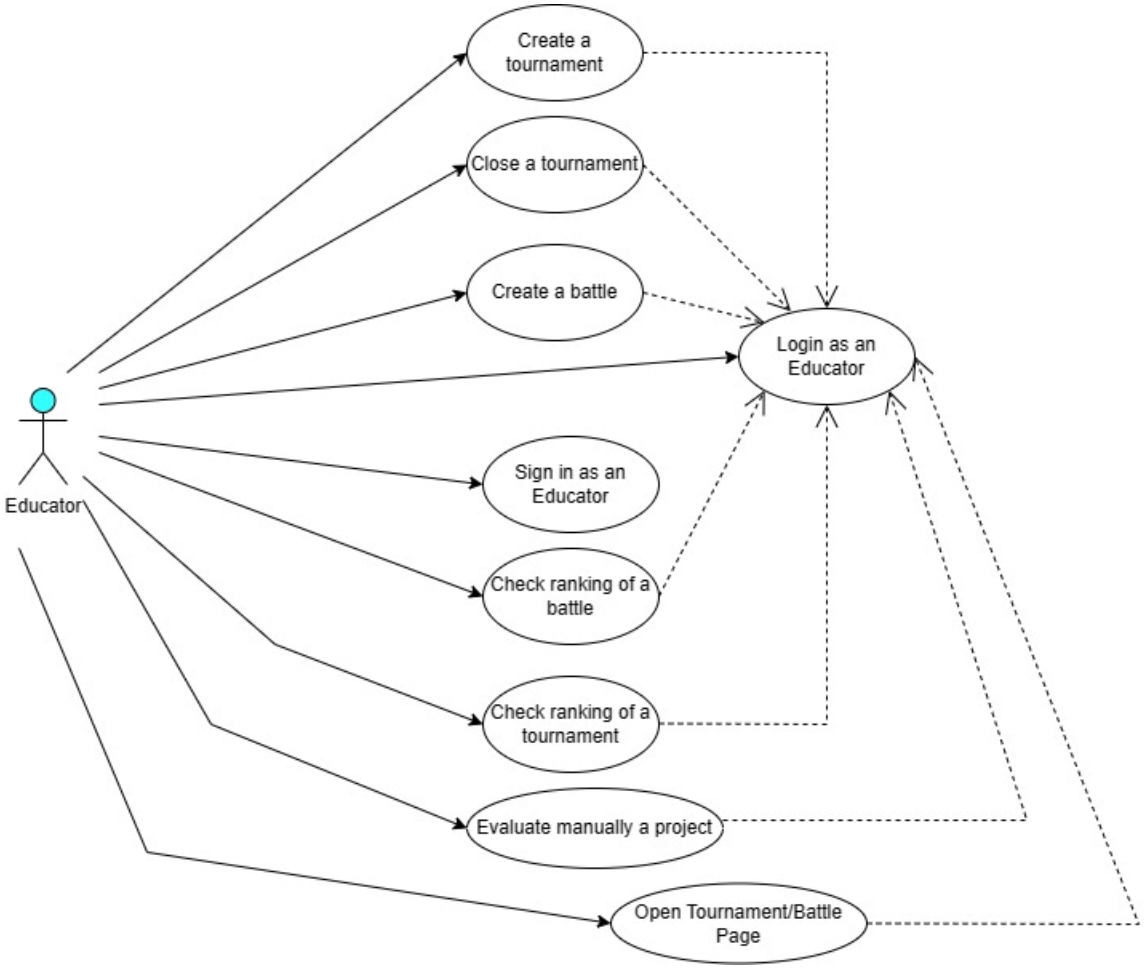


Figure 11 Educator Use Case

3.2.3. Use cases

[UC1]. Sign in of a new user

Name	Sign in of a new user
Actors	User
Entry Condition	A user opens the site
Event Flow	1. User opens the site 2. System shows the home page 3. User clicks the “Sign in” button 4. System shows the sign in page 5. User inserts their personal data 6. User clicks the confirmation button 7. System checks if the credential inserted in the previous step are not already used by another account 8. System sends a confirmation email to the user 9. User confirms the registration by clicking the link received by email
Exit Condition	Registration has been successful. The student data is stored into the system’s database. The student then will be able to login in the system by using their credentials
Exception	7. The username or the email are already used. The system comes back to point 3 adding an error message

[UC2]. Login of a user

Name	Login of a user
Actors	User
Entry Condition	A user opens the site
Event Flow	1. User opens the site 2. System shows the home page 3. User clicks the “Log in” button 4. System shows the login page 5. User inserts their credential 6. User clicks the “Enter” button 7. System checks the credentials 8. System shows the user’s home page
Exit Condition	User has accessed to his personal home page
Exception	7. User inserted invalid credentials. The system comes back to point 3 adding an error message

[UC3]. Creation of a tournament

Name	Creation of a tournament
Actors	Educator and student
Entry Condition	Educator opens the site
Event Flow	1. Educators open the site 2. System shows the home page 3. Educator logs in 4. System shows the educator's home page 5. Educator fills the form for the creation of a new tournament 6. Educator clicks the "Create a new tournament" button 7. System saves the tournament 8. System shows the tournament page 9. System notifies all students subscribed to CKB about the creation of the tournament 10. Educator clicks the "Adds another educator as a collaborator" button 11. System shows a form to fill 12. Educator fills the form with the username of the educator to add 13. System saves the modification at the tournament
Exit Condition	Tournament has been successful created. The tournament data is stored into the system's database. Now student can join the tournament whenever they want until the registration deadline expires.
Exception	9. If the educator does not want to add collaborators. The steps 10 through 13 are skipped

[UC4]. Creation of a battle

Name	Creation of a battle
Actors	Educator and student
Entry Condition	An educator opens the site
Event Flow	1. Educator opens the site 2. System shows the home page 3. Educator logs in 4. System shows the educator's home page 5. Educator clicks on the tournament in which he wants to create a battle 6. System shows the tournament page 7. Educator press the "Create a new battle" button 8. System shows the page for creating the battle 9. Educator fills the form 10. Educator clicks the "Create" button 11. System saves the battle 12. System notifies all students that are subscribed to the tournament that a new battle has been created
Exit Condition	Battle has been successful created. The battle data is stored into the system's database. Now student can join the battle whenever they want until the registration deadline expires.

[UC5]. Student joins a tournament

Name	Student joins a tournament
Actors	Student
Entry Condition	A student opens the site
Event Flow	1. Student opens the site 2. System shows the home page 3. Student logs in 4. System shows the student's home page 5. Student searches for a tournament they want to enter 6. System shows details about selected tournament 7. Student clicks on a specific tournament 8. System shows the tournament page 9. Student click the "Join tournament" button
Exit Condition	Registration has been successful. The subscription is stored into the system's database. Now student will have to wait until the registration deadline expires and the educator creates a new battle.

[UC6]. Student joins a battle

Name	Student joins a battle
Actors	Student
Entry Condition	A student opens the site
Event Flow	1. Student opens the site 2. System shows the home page 3. Student logs in 4. System shows the student's home page 5. Student searches for a tournament in which they are registered 6. Student clicks on specific tournament 7. System shows the tournament page 8. Student click on the battle which he is interested too 9. System shows the battle page 10. Student click the "Join battle" button 11. Student selects if they participate alone or with a group 12. Student invites all mates to the groups 13. System sends a notification to all students belonging to the group, informing them that it is open for them to join it. 14. System waits until registration deadline expires 15. System creates a repository for the group 16. Student forks the repository and activate GHA
Exit Condition	Registration has been successful. The subscription is stored into the system's database. Now student is able to modify the project and receive an evaluation by committing the modification into the main branch of the repo.
Exception	9. If the deadline for tournament registration has passed. The steps 10 through 15 are skipped. 11. If the student decides to participate alone The step 12 and 13 is skipped

[UC7]. Closing a tournament

Name	Closing a tournament
Actors	Educator and student
Entry Condition	And educator opens the site
Event Flow	1. Educator opens the site 2. System shows the home page 3. Educator logs in 4. System shows the educator's home page 5. Educators clicks on the tournament which he wants to close 6. System shows the tournament page 7. Educator clicks the "Close the tournament" button 8. System publishes final ranking of the tournament 9. System notifies all students subscribed in the tournament the closing of the tournament
Exit Condition	Tournament has been successful closed. The data is stored into the system's database.

[UC8]. Evaluation of a code

Name	Evaluation of a code
Actors	Student, GHA and educator
Entry Condition	A student pushes a commit into the main branch of the repo
Event Flow	1. Student pushes a commit into the main branch of the repo 2. GHA notify the system about the push 3. System clones the project in a protected environment 4. System tests the code based on the test case and the properties chosen by the educator 5. Educator manually evaluates the code and modifies the evaluation made by the system
Exit Condition	Evaluation has been successful. New data and new ranking are stored in the system database. Now everyone can check the updated ranking
Exception	5. This step may not happen because is an optional step.

[UC9]. Check ranking of a tournament

Name	Check ranking of a tournament
Actors	User
Entry Condition	A user opens the site
Event Flow	1. User opens the site 2. System shows the home page 3. User logs in 4. System shows the user's home page 5. User searches for the tournament he is interested 6. User clicks the interested tournament 7. System shows the tournament page 8. User clicks the "Ranking" button 9. System shows ranking of the tournament
Exit Condition	Now user can check the ranking of the tournament

[UC10]. Check ranking of a battle

Name	Check ranking of a battle
Actors	User
Entry Condition	A user opens the site
Event Flow	1. User opens the site 2. System shows the home page 3. User logs in 4. System shows the user's home page 5. User searches for the tournament he is interested 6. User clicks the interested tournament 7. System shows the tournament page 8. User clicks the battle in the tournament that they are interested too 9. System shows the battle page 10. User click the "ranking" button 11. System shows the ranking of the battle
Exit Condition	Now user can check the ranking of the battle

[UC11]. Show tournament page

Name	Show tournament page
Actors	User
Entry Condition	A user opens the site
Event Flow	1. User opens the site 2. System shows the home page 3. User logs in 4. System shows the user home page 5. User searches for the tournament they are interested 6. User clicks the tournament they are interested 7. System shows the tournament page
Exit Condition	Now user has access to all tournament information and functionality

[UC12]. Show battle page

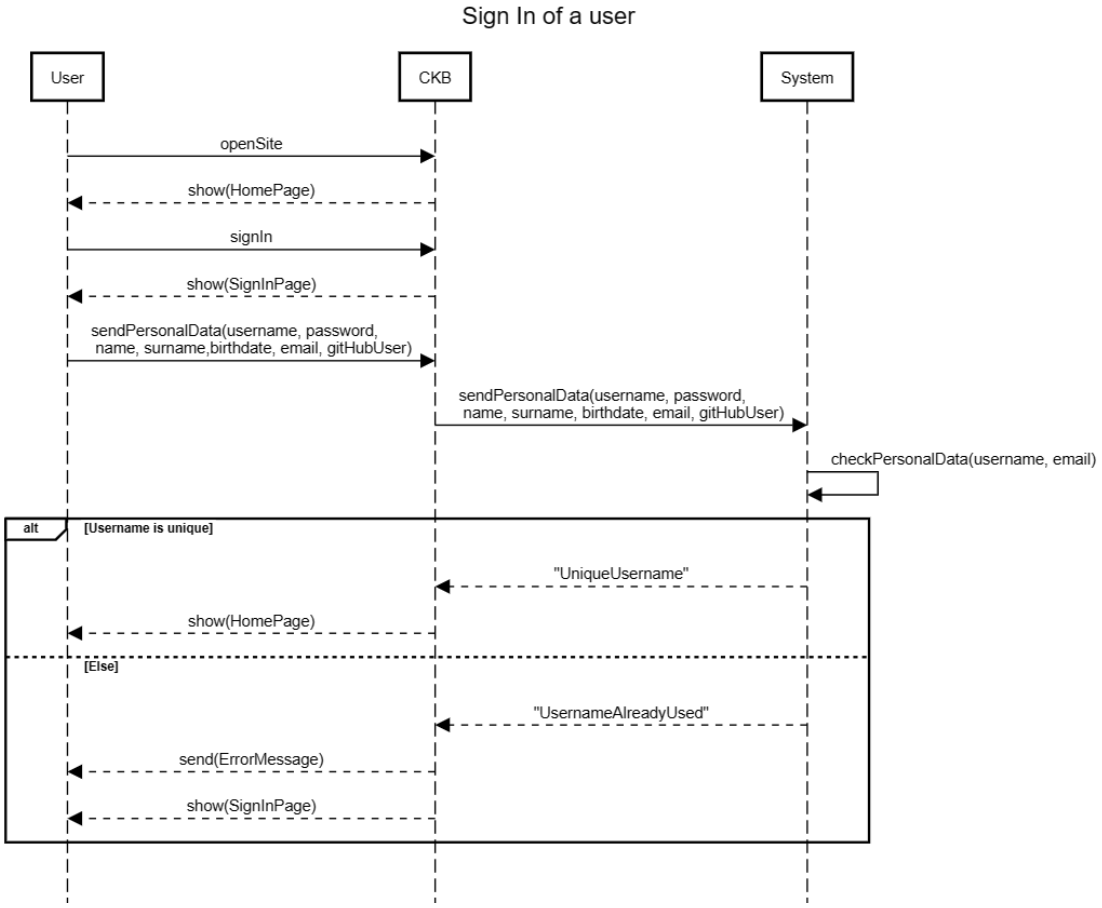
Name	Show battle page
Actors	User
Entry Condition	A user opens the site
Event Flow	1. User opens the site 2. System shows the home page 3. User logs in 4. System shows the user home page 5. User searches for the tournament they are interested 6. User clicks the tournament they are interested 7. System shows the tournament page 8. User clicks the battle in the tournament that they are interested too 9. System shows the battle page
Exit Condition	Now user has access to all battle information and functionality

[UC13]. Student accepts an invite for a battle

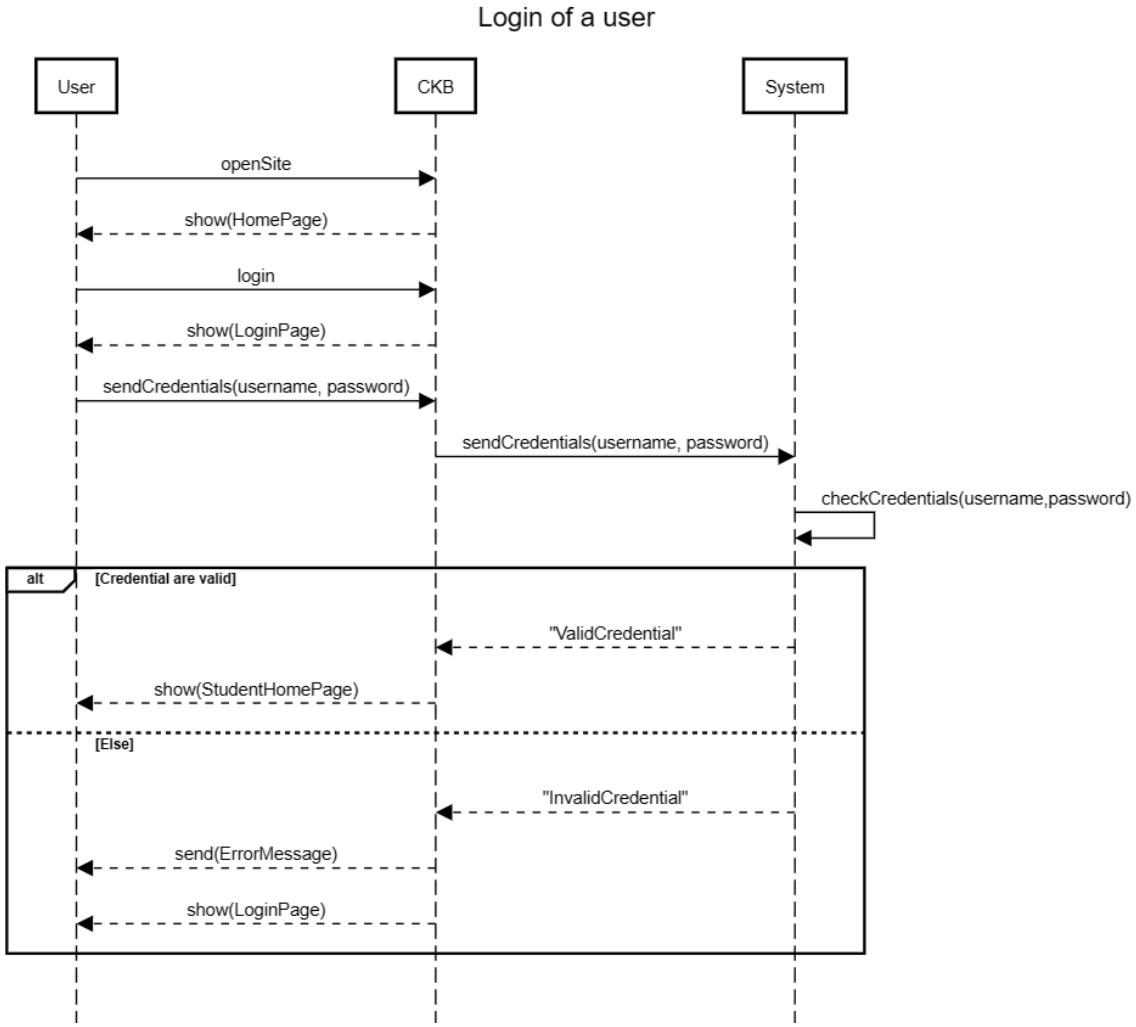
Name	Student accepts an invite for a battle
Actors	Student
Entry Condition	A student opens the site
Event Flow	1. Student opens the site 2. System shows the home page 3. Student logs in 4. System shows the student home page 5. Student search for a tournament 6. Student clicks on a specific tournament 7. System shows the tournament page 8. Student clicks on the battle which he is interested too 9. System shows the battle page 10. Student clicks on the invitation for the battle 11. System shows all the invitation received 12. Student clicks “Join with this group” button 13. System waits until registration deadline expires 14. System creates a repository for the group 15. Student fork the repository and activates GHA
Exit Condition	Registration has been successful. The subscription is stored into the system’s database. Now student is able to modify the project and receive an evaluation by committing the modification into the main branch of the repo.
Exception	11. System shows an empty page in case they have not received any invite

3.2.4. Sequence diagram

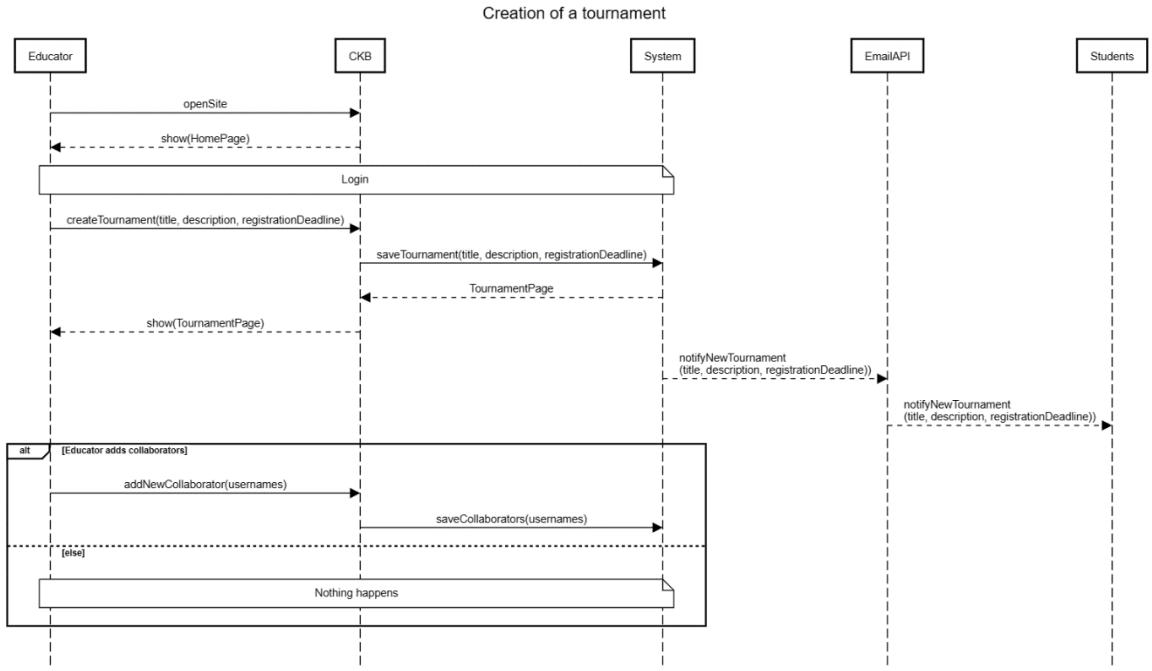
[SD1]. Sign in of a new user



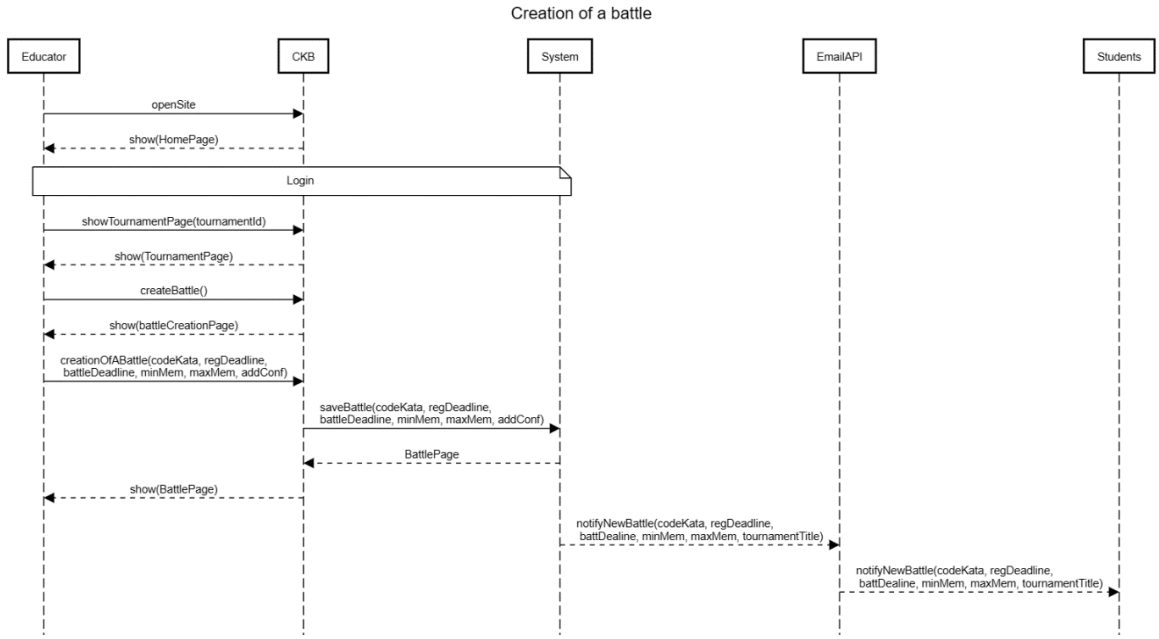
[SD2]. Login of a user



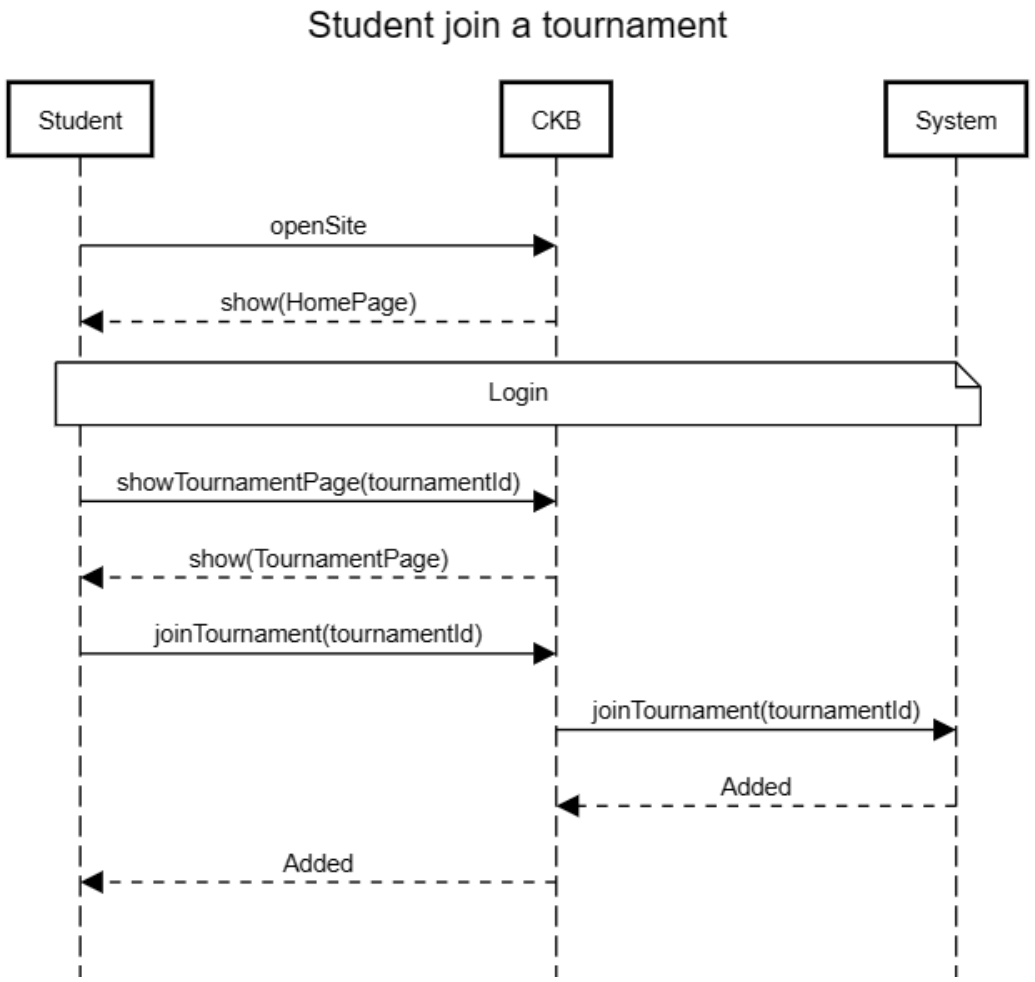
[SD3]. Creation of a tournament



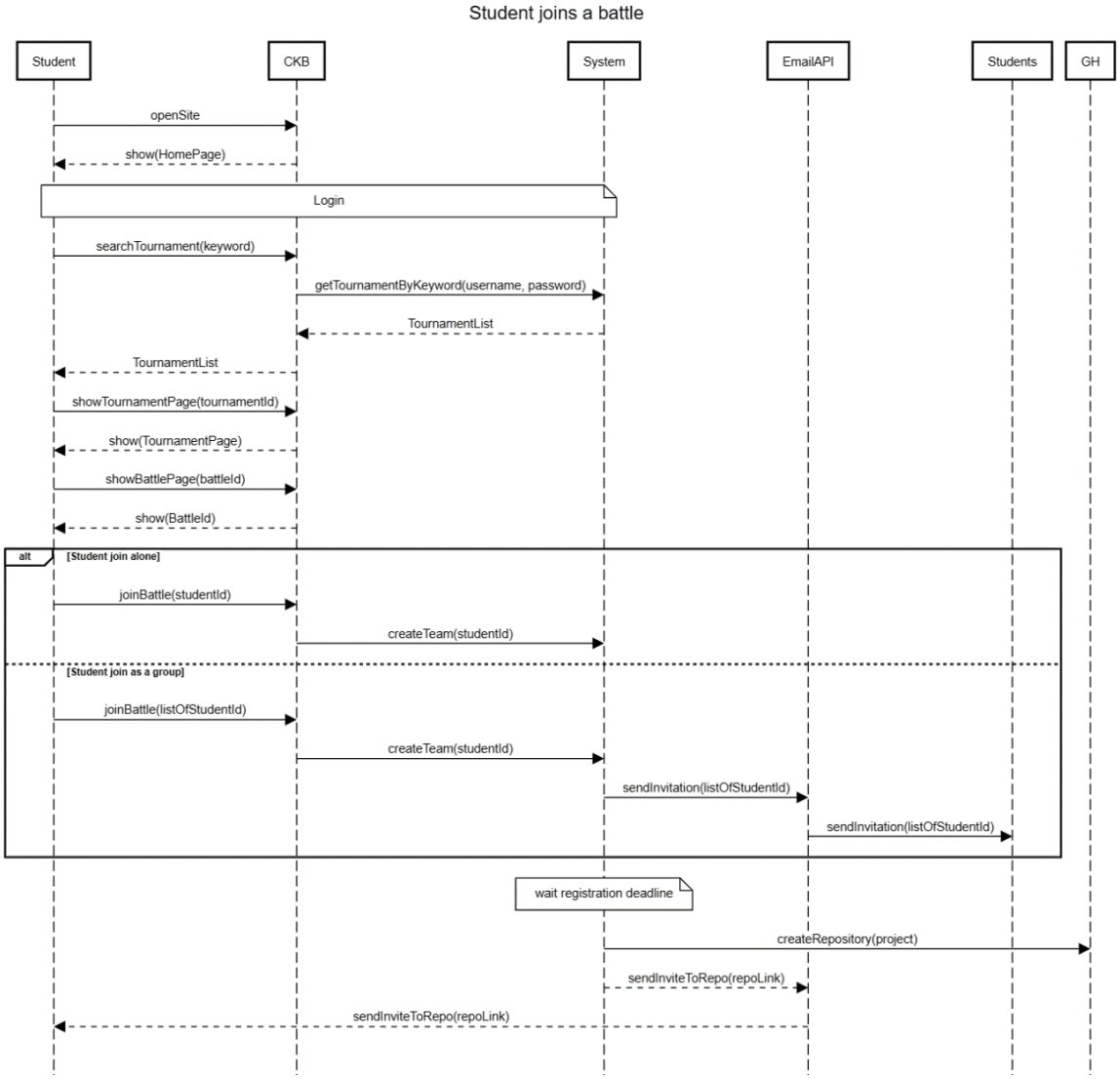
[SD4]. Creation of a battle



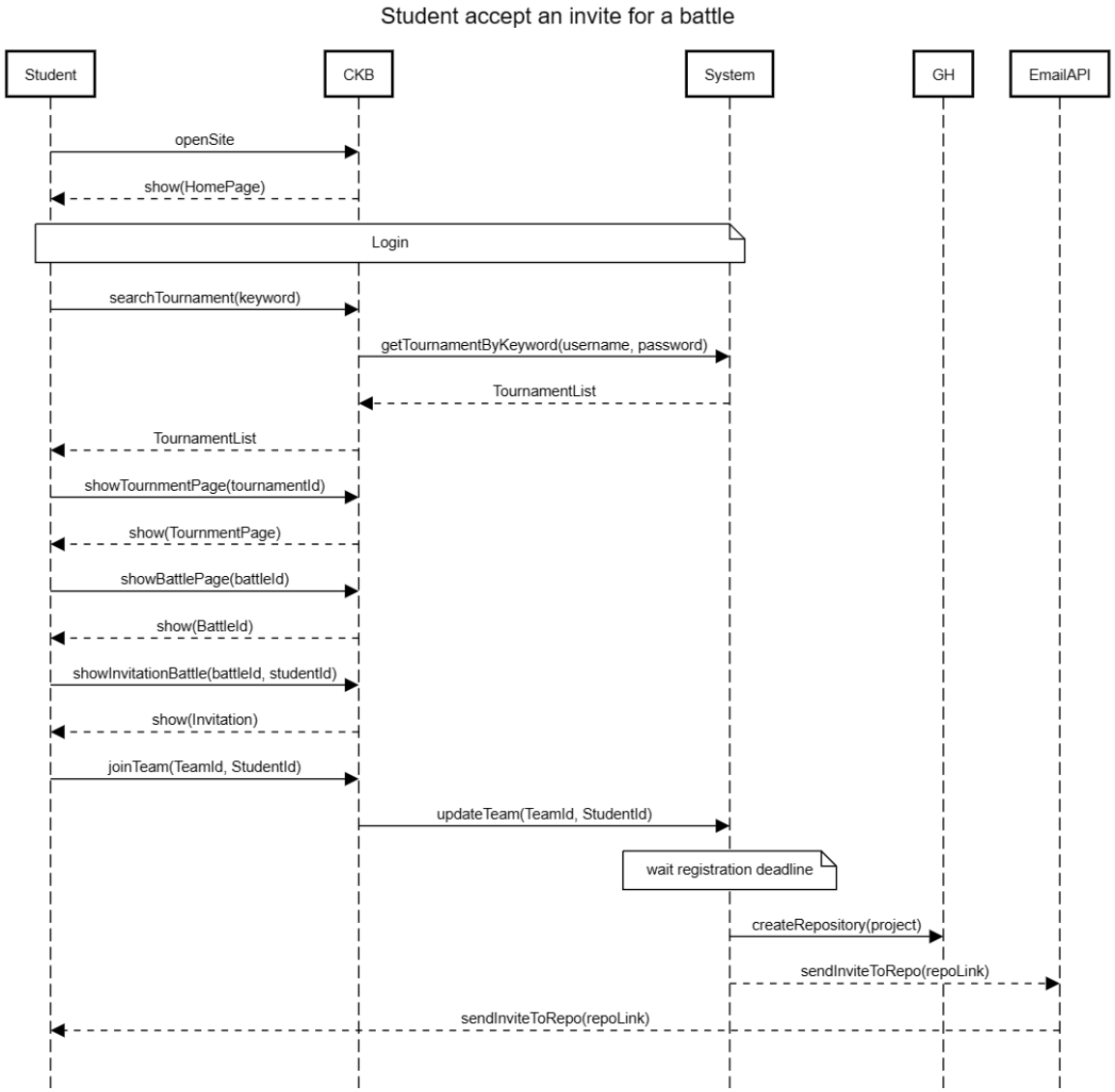
[SD5]. Student joins a tournament



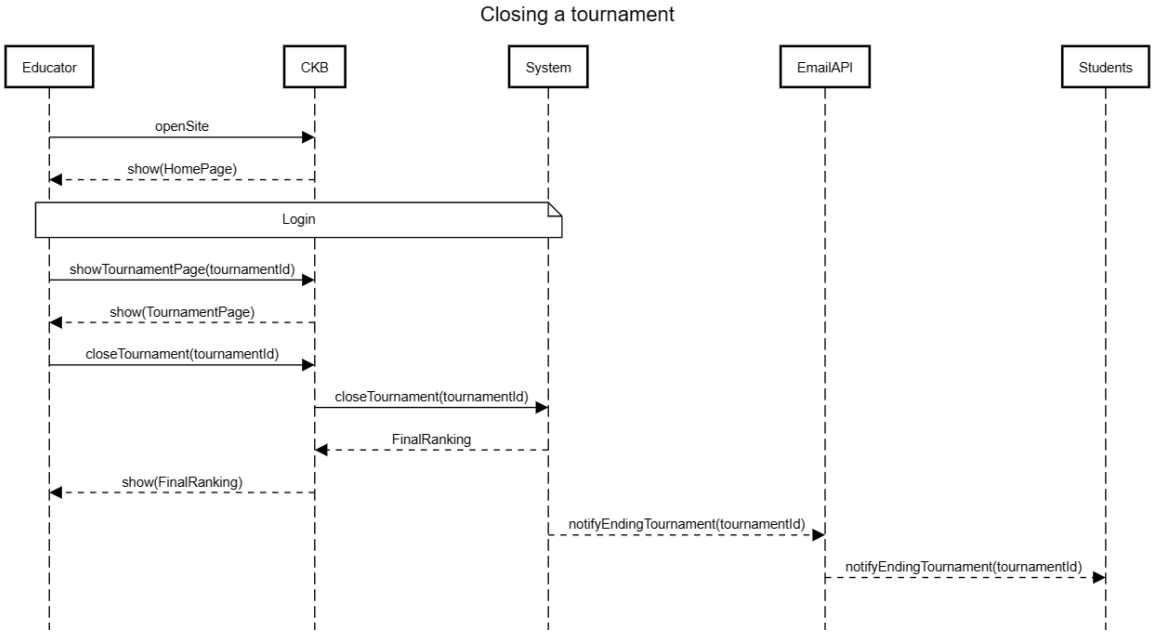
[SD6]. Student joins a battle



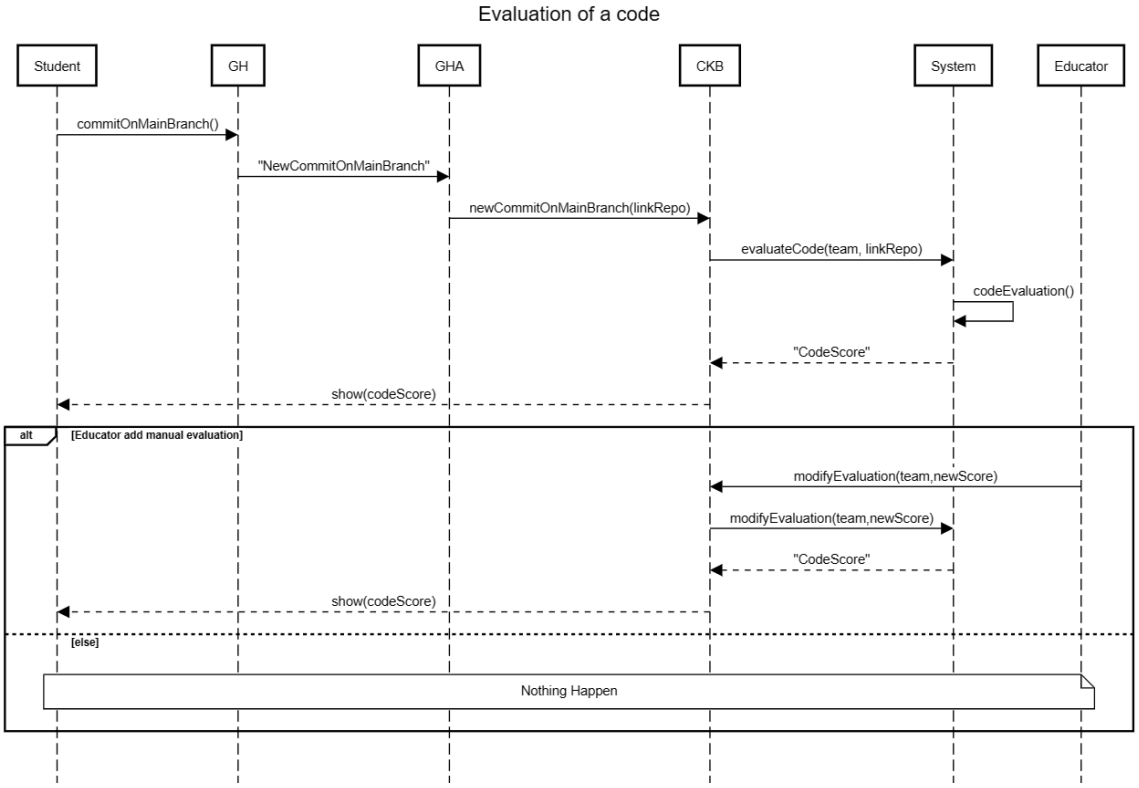
[SD7]. Student accepts an invite for a battle



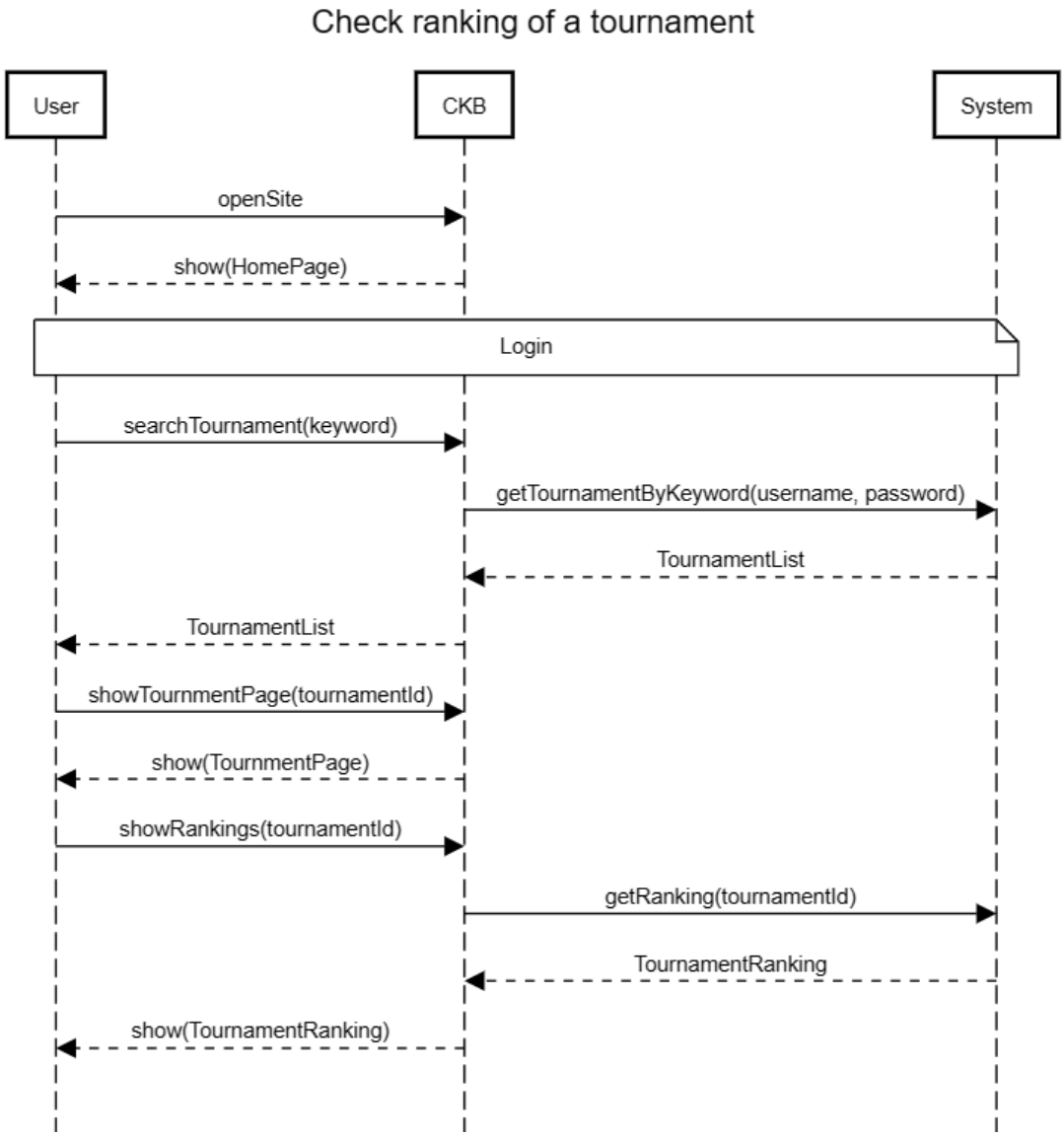
[SD8]. Closing a tournament



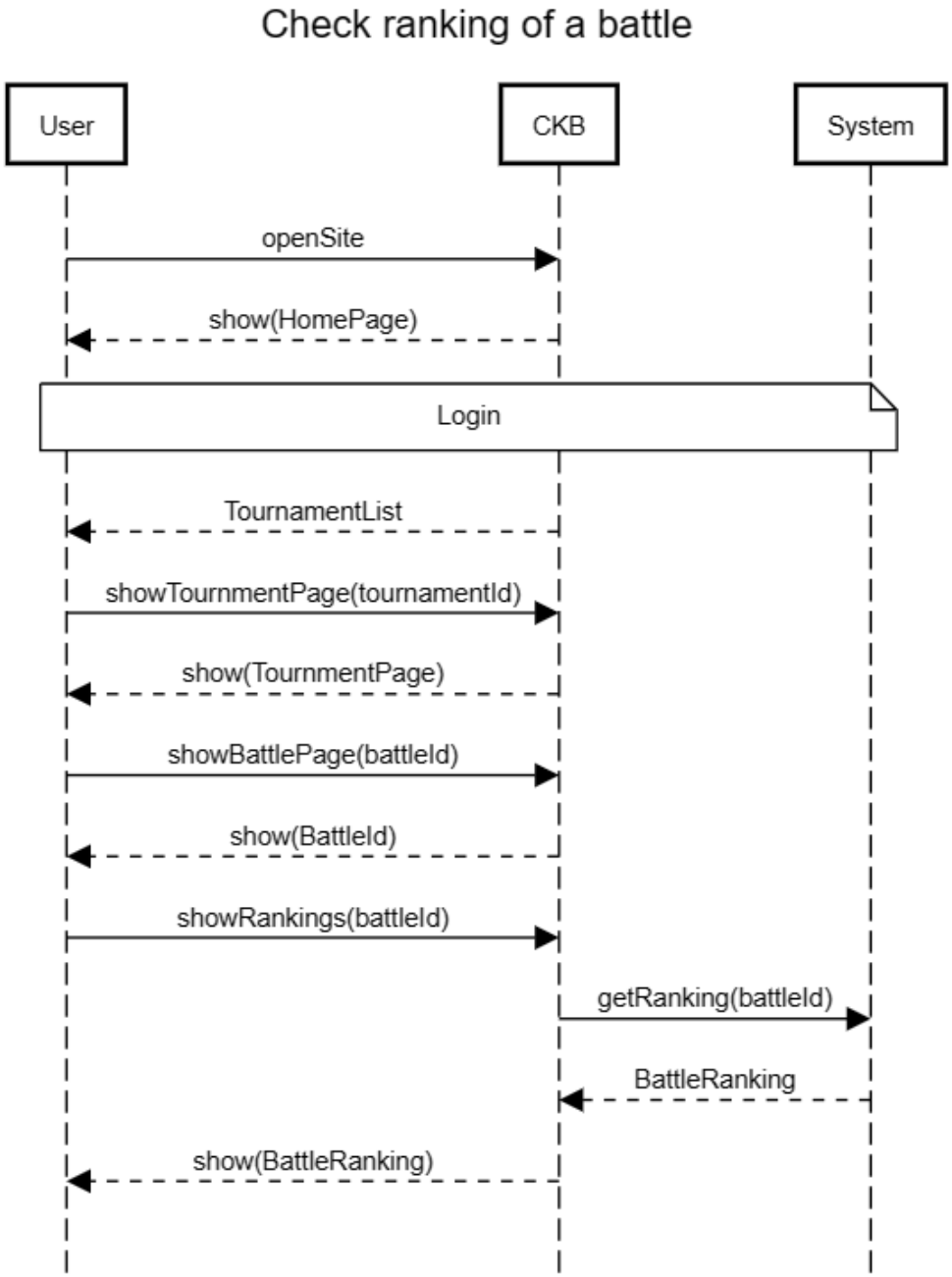
[SD9]. Evaluation of a code



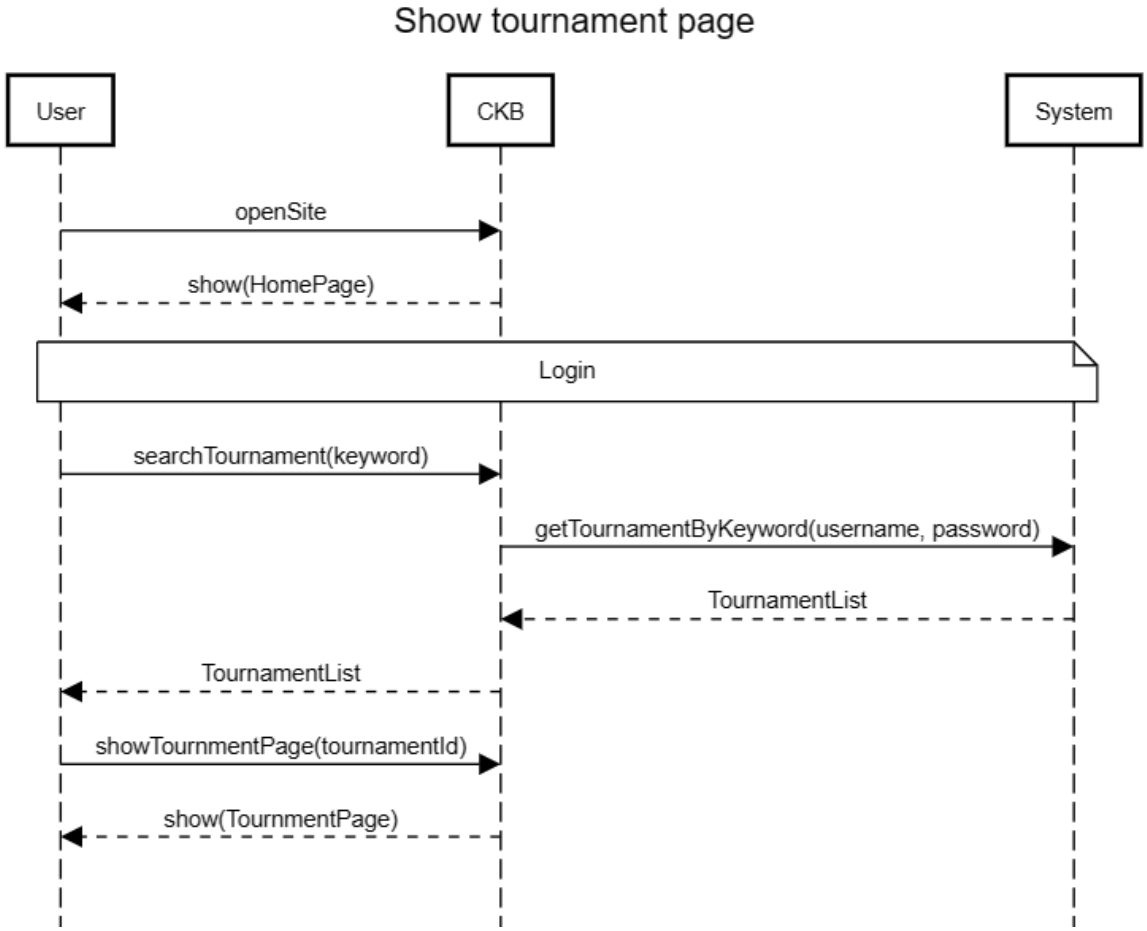
[SD10]. Check ranking of a tournament



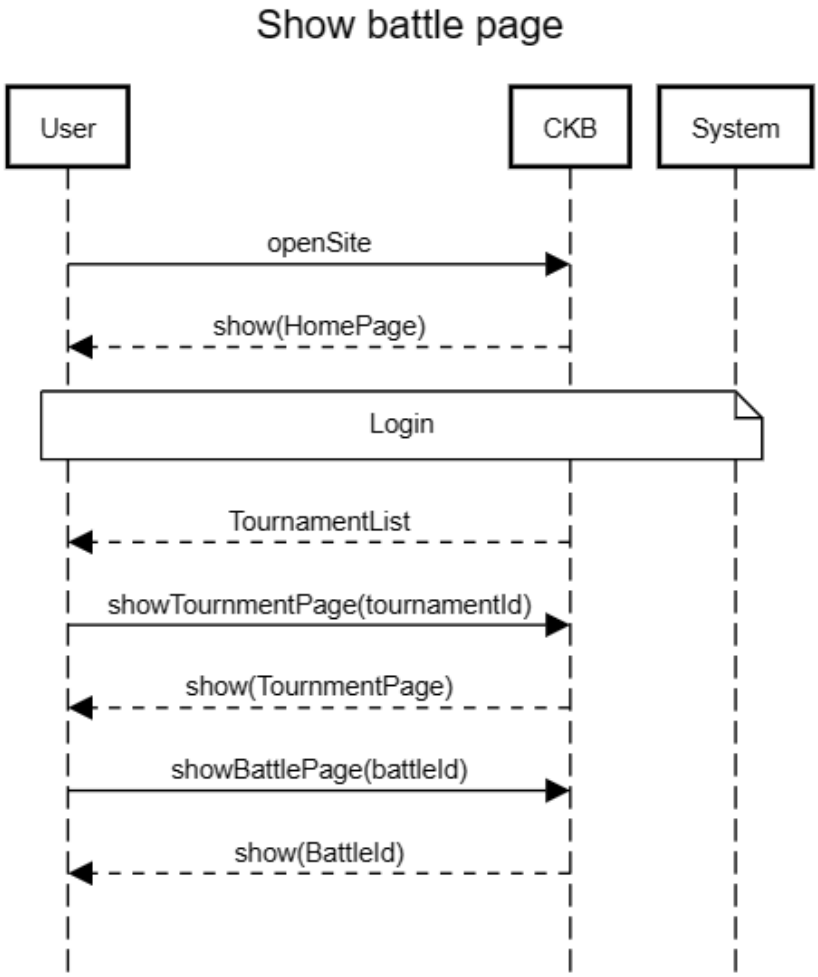
[SD11]. Check ranking of a battle



[SD12]. Show tournament page



[SD13]. Show battle page



3.2.5. Requirement mapping

[G1] Allow registered users as educators to create and manage tournaments and battles to offer enrolled students the opportunity to improve their skills in the field of software development.

Requirements	*Domain Assumption*
[R2]. System allows educators to sign up. [R4]. System allows educators to login. [R5]. System allows educator to create a new tournament and to add all the needed information. [R6]. System allows educator to grant the permission to other colleagues to create battle in a specific tournament. [R7]. System allows educator to create a battle inside a particular tournament by adding CK and the deadlines. [R8]. System allows educator to see rankings of each tournament, which he created or in which he has been nominated collaborator. [R9]. System allows educator to see ranking of a specific battle. [R11]. System allows educator to close a tournament.	[DA1]. Users must have a valid email address. [DA4]. Each user will create an account. [DA5]. Personal data given by user during the registration process are assumed truthful. [DA6]. User must have an internet connection. [DA7]. Educator must upload the correct CK.

[G2] Allow users registered as students to participate in tournaments and battles of their interest to enhance their skills in software development. Student joins a battle in a tournament they are register for and invite some friends to participate with them, thus creates a team for the battle.

Requirements	*Domain Assumption*
[R1]. System allows students to sign up. [R3]. System allows students to login. [R12]. System notifies students about the creation of a new tournament. [R13]. System notifies students about the creation of a new battle in a specific tournament in which they have subscribed. [R14]. System notifies students about publication of final ranking of a specific tournament. [R15]. System notifies students about publication of final ranking of a specific battle. [R16]. System allows students to join a tournament. [R17]. System allows students to join a battle. [R18]. System allows students to see rankings of each tournament, which they are sub scripted. [R19]. System allows students to see ranking of a specific battle. [R20]. System, at end of the registration period of a battle, create a GH repo and send invitation to all member of the group. [R22]. System allows students to search tournament by key words. [R23]. System allows interaction with GH platform.	[DA1]. Users must have a valid email address. [DA2]. Students must have a GH profile. [DA3]. Students need to know how to access GitHub, how to fork a repository and how to enable GHA. [DA4]. Each user will create an account. [DA5]. Personal data given by user during the registration process are assumed truthful. [DA6]. User must have an internet connection. [DA8]. It is assumed that interaction with GH is always possible.

[G3] Provide an opportunity for students to receive an evaluation for their implementation concerning the required problem, considering different parameters.

Requirements	*Domain Assumption*
[R10]. System allows educator to modify the evaluation manually of a specific group in a battle. [R21]. System automatically evaluates the code in the main branch of the repo after each commit in it. [R23]. System allows interaction with GH platform.	[DA1]. Users must have a valid email address. [DA2]. Students must have a GH profile. [DA3]. Students need to know how to access GitHub, how to fork a repository and how to enable GHA. [DA4]. Each user will create an account. [DA5]. Personal data given by user during the registration process are assumed truthful. [DA6]. User must have an internet connection. [DA8]. It is assumed that interaction with GH is always possible.

3.3. Performance requirements

Our application, designed to enhance software programming skills, commits to ensuring a timely and reliable response at every stage of the process. Notifications regarding the creation of new tournaments will be sent in real-time to all subscribers, while notifications related to battles will be limited to participants in the corresponding tournament. This selection will occur swiftly, allowing all users to have an ample time frame to register.

User interactions with rankings, tournament registrations, and battles will proceed seamlessly, even in high-load situations, ensuring an optimal user experience.

A crucial aspect is the efficient management of the creation of repositories on GitHub, occurring at the end of the registration period for a battle. The system must create these repositories swiftly and efficiently, adapting to simultaneously manage various battles without negative impacts on overall performance.

Commit monitoring and automatic code testing are critical phases in our system. It is essential to ensure that notifications are timely whenever a commit is made to the battle repositories and that code testing occurs swiftly and efficiently, providing immediate feedback to students.

During tournaments, the system will maintain high performance, simultaneously managing communications, rankings, and repository creation. The display of results and rankings for ongoing tournaments will occur with rapid response times.

Furthermore, our platform will guarantee high reliability and availability. During critical phases, such as repository creation and the execution of automatic tests, the system will minimize downtime. In case of spikes in participation or high loads, it will dynamically adapt to maintain optimal performance levels, preserving the quality of service offered to users.

3.4. Design constraints

3.4.1. Standards compliace

Regarding data privacy, the CKB adheres to the GDPR, a European Union regulation establishing rules for the protection of personal data and privacy for individuals within the European Union and the European Economic Area. Therefore, users' personal data and information will not be used for commercial purposes.

The system will be fully compliant with the D0-178C standard. This implies that all software tests will be conducted in accordance with specified requirements, enabling the association of each requirement with a dedicated test case used to verify whether it has been satisfied.

Additionally, the system must adopt international standards concerning the use and representation of date and time. This will ensure consistency and international compatibility in interpreting temporal information, contributing to a coherent and accurate management of temporal data within the system.

3.4.2. Hardware limitations

The system requires a robust hardware infrastructure to ensure reliable and timely performance during daily activities and potential usage spikes. The server's processing capacity must be sufficiently scaled to handle large volumes of data, ensuring smooth operations even during high-traffic situations. Resilience to peak requests is crucial to ensure the application can concurrently manage a significant number of users and activities without compromising performance.

Simultaneously, the server's storage capacity must be adequate to efficiently handle user data, including details related to tournaments and battles. Efficient data management necessitates a solid storage capacity to ensure the availability and accessibility of information at all times. This aspect is critical to support daily operations and provide a strong foundation for future expansion.

On the user's side, it is essential to use a browser enabled for JavaScript management. This technological dependency is necessary to fully leverage the interactive and dynamic features of the application, offering an optimal user experience. Ensuring that the user has a JavaScript-compatible browser is therefore a fundamental prerequisite for the proper functioning of the system and accessing all its advanced features.

3.5. Software system attributes

3.5.1. Reliability

To ensure high reliability, the system will implement advanced mechanisms for real-time error detection and management. Clear and contextualized alerts will be adopted, providing users with precise information in case of any issues. This approach ensures immediate and targeted intervention to resolve problems, preserving the integrity and reliability of operations.

Furthermore, the system's architecture will be designed with a distribution across multiple servers and the implementation of data replication strategies. This design aims to prevent single points of failure that could compromise the system's integrity as a whole. Distribution across multiple servers and data replication will contribute to ensuring operational continuity even in the presence of hardware failures or malfunctions.

To reduce the risk of errors during data-saving operations in the database, the system will use dedicated triggers for error management. These triggers will provide active control during critical phases, intervening promptly to correct any inconsistencies and ensure the coherence of stored data.

Additionally, if a comprehensive traceability of system activities and in-depth analysis of errors are desired, it will be possible to implement a logging system. This tool will allow for detailed recording of all operations, facilitating the understanding and resolution of any issues. Overall, this integrated strategy of active error management, distribution across

multiple servers, data replication, and detailed logging of activities will contribute to ensuring high reliability of the application over time.

3.5.2. Availability

System availability is a fundamental requirement to ensure uninterrupted operation over time. To optimize availability and ensure operational continuity even in the face of failures, the server implementation will be designed with a distributed approach. This means that the system will be deployed across multiple machines that constitute the server, allowing load balancing and mitigating potential impacts from hardware failures or malfunctions in a single machine.

Data replication will be implemented to ensure redundancy of critical information. In the event of machine malfunctions or data loss, data replicas will be readily available, ensuring service continuity without significant interruptions. This approach will help minimize the risk of data loss and maintain high availability standards, even in scenarios of unforeseen failures.

Additionally, the system will be designed to automatically detect faults and respond accordingly, activating failover mechanisms to ensure that user requests can be handled without interruptions. Efficient management of distribution, replication, and data redundancy is essential to ensure high system availability, minimizing downtime, and overall enhancing user experience.

3.5.3. Security

In accordance with security practices, the system will take several measures to safeguard sensitive information and maintain data integrity.

All communication between the client and the server, as well as with the database management system (DBMS), will be via HTTPS, so that there will be a secure, encrypted transfer of data.

Passwords will be hashed before being stored in the database, thus reducing the risk associated with storing passwords in plain text. In addition, data transmission from client to server will be encrypted to protect against potential eavesdropping and unauthorized access.

To further enhance data security, sensitive information will be encrypted before storage, minimizing the impact of potential data leaks. In addition, strict access controls will be implemented to prevent third parties from accessing sensitive data through group requests, strengthening the overall resilience of the system against unauthorized access and potential security breaches.

3.5.4. Maintainability

Code maintainability is a fundamental cornerstone in our approach to software development.

To ensure a code structure that is easily manageable and adaptable over time, we will adopt well-known and established design patterns. The use of these patterns provides a coded architecture that reflects best practices and promotes an intuitive understanding of the code by developers.

Simultaneously, our focus on maintainability is reflected in extensive documentation detailing the implemented functions. This documentation not only provides clarity on the behaviour of functions but also on the broader context in which they are integrated into the system's architecture.

Comprehensive documentation facilitates the work of developers making changes, aiding in understanding the relationships between different parts of the code and thus contributing to the long-term maintainability and improvement of our software.

3.5.5. Portability

The developed CKB application is designed to run on any operating system, ensuring significant flexibility in access across various platforms.

The only required condition is the presence of a browser capable of running JavaScript. This design choice aims to maximize the accessibility of the application, enabling users to enjoy its features regardless of the operating system used. Thanks to this approach, users will experience a seamless and comprehensive interface, irrespective of their chosen device or operating system, ensuring greater adaptability and accessibility of the application.

4. FORMAL ANALYSIS USING ALLOY

4.1. Signatures

- sig DateTime {} //it represents a couple <data, time>
- sig CodeKata {} //it represents a CK
- abstract sig User {} //it represents a user
- sig Student extends User {} //it represents a student
- sig Educator extends User {} //it represents an educator
- sig Tournament { //it represents a tournament
  registrationDeadline: one DateTime, //the tournament's registration deadline
  students: some Student, //the students registered for the tournament
  educators: some Educator //the educators that are responsible for managing
  }
- sig Battle { //it represents a battle
  registrationDeadline: one DateTime, //the battle's registration deadline
  submissionDeadline: one DateTime, //the deadline by which possible solutions to the battle must be administered
  tournament: one Tournament, //the tournament of which the battle is a part
  code: one CodeKata //it represents the battle's CK
  } {
  registrationDeadline != submissionDeadline and
  registrationDeadline != tournament.registrationDeadline and
  submissionDeadline != tournament.registrationDeadline
  }
- sig Team { //it represents a team
  students: some Student, //students make up the team
  battleT: one Battle //the battle in chich the team is enrolled
  }
- sig Evaluation { //it represents an evaluation
  time: one DateTime, //the time at which the assessment was made
  team: one Team, //the team that received the evaluation
  battleE: one Battle, //the battle in which the team that received the evaluation is enrolled
  grade: one Int, //the team grade
  code: one CodeKata //it represents the evaluation's CK
  } {
  time != battleE.registrationDeadline and
  time != battleE.submissionDeadline and
  time != battleE.tournament.registrationDeadline and
  grade > 0
  }

4.2. Facts

- fact EachBattleHasAnEvaluation { //each battle has an evaluation
  all e1, e2: Evaluation |
  (e1 != e2) implies ((e1.battleE !=e2.battleE) or (e1.team!= e2.team))
  }

- fact EachTeamInABattleHasDifferentStudent { //each team in a battle has different student
  all t1, t2: Team,
  b: Battle |
  (((t1 != t2) and (b in t1.battleT) and (b in t2.battleT)) implies
  (all s: Student |
  (s in t1.students and s not in t2.students) or (s not in t1.students and s in t2.students)
  or (s not in t1.students and s not in t2.students)))
  }

- fact EachStudentInABattleIsInTheTournament { //each student in a battle is in the tournament
  all t:Team,
  s:Student |
  (s in t.students) implies (s in t.battleT.tournament.student)
  }

- fact BattleAndEvaluationHasSomeCodeKata { //each evaluation with its corresponding battle have the same CodeKata
  all e: Evaluation |
  e.battleE.code = e.code
  }

- fact TeamAndEvaluationHasSomeBattle { //each evaluation with its relative team have the same battle
  all e: Evaluation |
  e.team.battleT = e.battleE
  }

- fact EachTeamHasAnEvaluation{ //each team has an evaluation
  all t:Team |
  #team.t = 1
  }

- fact EachBattleHasDifferentCodeKata{ //each battle has different CodeKata
  all b1, b2: Battle |
  (b1 != b2) implies (b1.code != b2.code)
  }

4.3. Predicate

- pred show {
  #Battle = 2
  #Team = 4
  #Student = 6
  #Tournament = 2
  }

- run show for 8

Below we find two representations of the system as specified above.

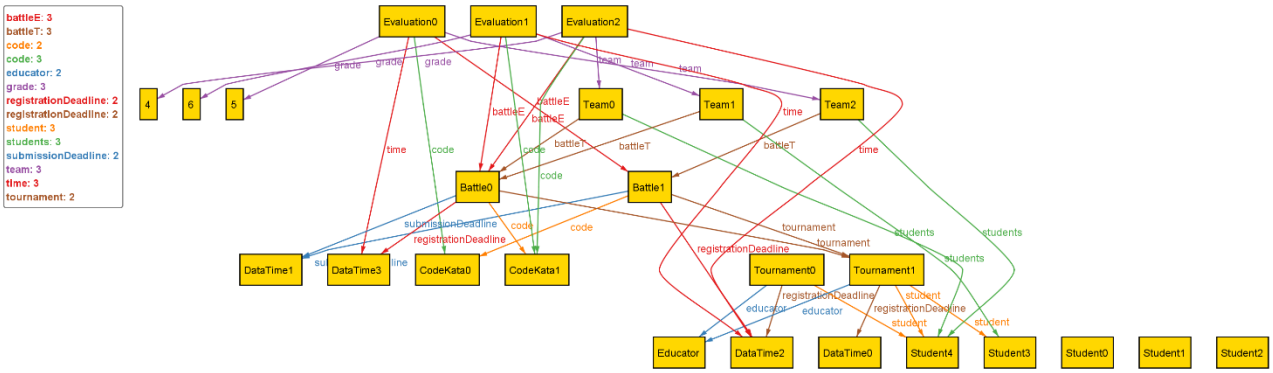


Figure 12 Alloy Static Model

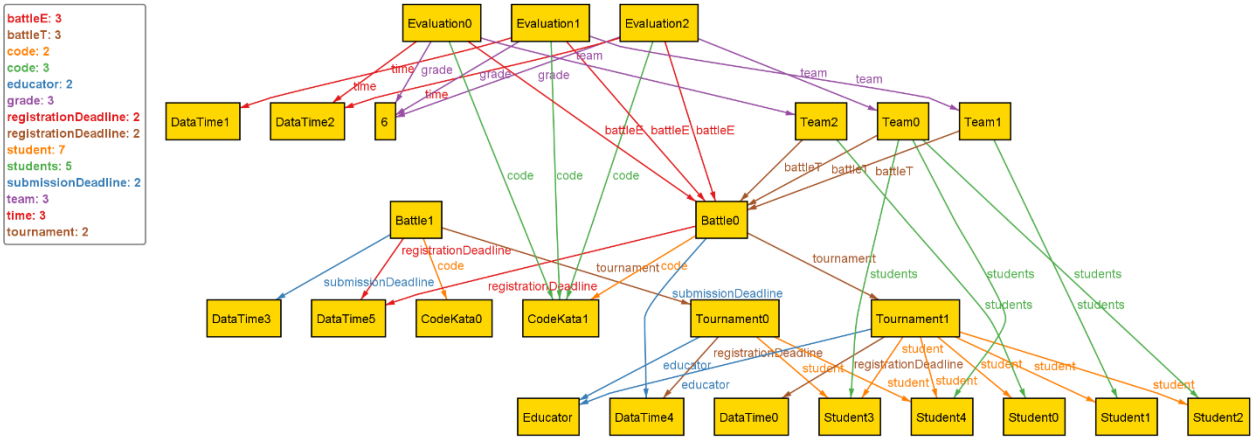


Figure 13 Alloy Static Model

5. EFFORT SPENT

In the following tables we will summarize the effort spent by each member of the team on the RASD Document

- Conti Alessandro

Chapter	*Effort (in hours)*
1	5
2	10
3	11
4	2

- Di Paola Antonio

Chapter	*Effort (in hours)*
1	3
2	4
3	15
4	7

6. REFERENCES

- State Diagrams made with: *draw.io*.
- Class Diagrams made with: *StarUML*.
- Sequence Diagrams made with: *SequenceDiagram.org*.
- Alloy models made, ran and checked with: *Alloy 6*.
- The User Interface overview was written in HTML with: *Visual Studio Code*.