Requirements Analysis & Specification Document

Karl Monrad Kieler {karlmonrad.kieler@mail.polimi.it}
Aske Schytt Meineche {askeschytt.meineche@mail.polimi.it}
Leonie Dragun {leonie.dragun@mail.polimi.it}

15 November 2023

Contents

1 Introduction

1.1 Purpose

The purpose of this Requirement Analysis and Specification Document (RASD) is to document and reason about fundamental choices relating to the CodeKataBattles software. CKB is a platform designed to enhance and hone skills within software development through gamified coding challenges and team work. This document serves three distinct purposes: First and foremost as an approximate guide to the developers responsible for building the software, as the document will present a proposed class structure including the logical cardinalites of each through as class diagram. The logic of the class structure is tested through static analysis using Alloy, with both facts and environments presented along with an example instance. Detailed sequence diagrams are also provided in order for developers to completely understand the interaction between core system components.

Second, the document is also a support for management who needs to understand the implications of the software, through the stated goals, atomic functional requirements and domain assumptions. We also supply management with use-cases and use-case diagrams to understand the intended interaction between the system and the potential user-base.

Lastly, the document also serves as a type of contract, as the stated outcomes and constraints are specifically stated, both aiding potential users and management.

1.1.1 Goal

The objectives we aim to fulfill through the implementation of the software are as follows.

1. Enable students to improve their software development skills through practice and competition.
2. Enable Educators to set up test-driven coding challenges including automated feedback online.
3. Simulate a Real-world software development scenario through use of GitHub and GitHub Actions.
4. Allow students to compare performances on specific challenges and coding tournaments.

1.2 Scope

In recent years, online availability of scalable educational offers have increased within languages (DuoLingo) and math and science (Brilliant) . The aim of this project is to build a platform that supports educators around the world in hosting small to large scale coding challenges, honing the skills of inquisitive students.

CodeKataBattles presents an environment where students form teams to engage in code kata battles, challenging them to develop solutions that meet specific coding requirements and pass predefined tests. The platform's intuitive user interface enables students to participate in these battles, receive immediate feedback, and learn from both successes and mistakes.

CKB supports coding challenges that promote hand-on learning, as well as collaboration through team development and knowledge sharing. Users are also able to track their progress in both tournaments and battles, being awarded with badges when achieving predefined goals.

CKB provides two primary interfaces: a student interface for participating in battles, reviewing codes, collaborating, and tracking progress, and an educator interface for setting up battles, monitoring student progress, and accessing analytics to improve educational content.

Following the World and Machine paradigm by M. Jackson and P. Zave, we identify the Machine as the CKB system to be developed and the educational environment as the World. This distinction allows categorization of phenomena into those within the World (educational needs, team interactions), those controlled by the Machine (coding challenges, feedback mechanisms), and shared phenomena (student engagement, learning outcomes).

1.2.1 World Phenomena

1. Student/Team forks CoteKataBattle Github Repository
2. Student/Team Sets Up Relevant Automated WorkFlow through Github Actions

1.2.2 Shared Phenomena

1. Educator creates a CodeKataBattle
2. Educator creates a tournament
3. Student registers as part of a team
4. Student/Team registers for a CodeKataBattle
5. Student/Team registers for a tournament
6. User Submits Solution to GitHub
7. User Subscribes to CodeKataBattle-Platform
8. Publishing of tournament rankings
9. Educator Creates New Badge
10. Student Receives New Badge

1.3 Definitions, Acronyms

1.3.1 Definitions

1. **Educator**: A type of user that is unable to participate in battles, but can create battles and tournaments.
2. **Student**: A type of user that can participate in battles and subscribe to tournaments
3. **GitHub**: One of the most widely used version control platforms for code.

1.3.2 Acronyms

1. **Educator**: A type of user that is unable to participate in battles, but can create battles and tournaments.
2. **Student**: A type of user that can participate in battles and subscribe to tournaments
3. **GitHub**: One of the most widely used version control platforms for code.

1.4 Revision History

1. Version 1.0 (16th December 2023)

1.5 Reference Documents

This Document is strictly based on

1. Specification of RASD project of the Software Engineering II course, held by professor Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli at the Politecnio di Milano, A.Y 2023/2024
2. Slides of Software Engineering 2 course on WeBeep

1.6 Document Structure

The document is divided into three overall parts:

1. **Overall Description** which describes the intended use-cases and user descriptions. We also describe a proposed class structure with brief explanations for each class along with notes on certain cardinalities, where we deem relevant.

2. **Specific Requirements** aims to supply concrete, atomic functional requirements, to aid developers in building the software, as well as outlining the proposed hardware and software interfaces.

3. The final part **Formal Analysis** uses Alloy to test the coherence of the proposed class structure and assumptions about the system.

2 Overall Description

2.1 Scenarios

1. **Create Tournament**
   Håkon teaches a class in introduction to programming at Middelfart Gymnasium and is teaching his class about basic programming logic. The students have a hard time grasping the concepts, so he decides to use CodeKataBattles to allow them to apply the theory in practice. He creates a tournament named "Introduction2Programming" for the class and asks all participants to create a user and subscribe to the tournament.

2. **Create a Battle**
   Håkon is preparing a battle for the Introduction2Programming tournament on recursive programming. He creates the required build automation scripts, test cases and description, uploads it to the battle and specifies the number of students and relevant deadlines. Due budget cuts he does not have time to give personal feedback so he sets the scoring configuration to automatic.

3. **Register a Team**
   Torben receives a notification from a tournament about a new Battle in a tournament he is participating in. He logs into CKB and reads the specifications of the new battle and sees that the registration deadline expires tomorrow. Groups may contain up to 4 participants so he registers a team 'TorbensTæskehold' and invites his 3 classmates.

4. **Participate in a Battle**
   Bertram is participating solo in a battle due to his lack of social skills. The registration deadline has just ended so he receives a link for a GitHub repository titled "binary_search". He forks the repository and sets up the required automated workflow through GitHub Actions, allowing the platform to grade and test his solution.

5. **Receive an evaluation**
   Bertram pushes a new commit to his forked repository "binary_search" with a new solution to the battle. After a few minutes, he receives a notification that he has received a new score in this battle. To his dismay, he realizes that the assigned score is 0/100 and places him dead last in the current rankings.

6. **Assign Manual Score**
   The submission deadline for the battle "binary_tree_inversion" has now been transgressed. Since Håkan has specified that scores include manual feedback, the battle now enters a consolidation phase. He assigns additional scores based on the code's similarity to his own solution and updates the score for each participant.

7. **Receive a Badge**
   Jørgen has just finished participating in the battle "birds_on_a_line" where I placed 2nd. He receives a notification that he has placed top 3 in 5 battles in a row in the current tournament, earning him the badge "CodeNinja". He is very pleased and calls his mother.

2.2 Product perspective

The Domain Class Diagram depicted below illustrates the structure and relationships within the CodeKataBattle system's class domain. This diagram is instrumental in understanding the interactions among the various elements of the system. Below are highlighted the most salient relationships and their constraints to facilitate comprehension of the system's domain.

- **Educator and User:** The system distinguishes between two types of users: students and educators, each being a specialized type of User. Every Educator can create multiple Tournaments and Battles, representing a one-to-many relationship (`1..*`). This design choice reflects the need for Educators to manage various coding challenges and competitive events within the platform.

- **Tournament:** Tournaments are each uniquely created and owned by an Educator, but an Educator can create multiple Tournaments implying a one-to-many relationship (`1..*`) regarding creation. Tournaments can encompass multiple Battles (`0..*`) and can linked to many Users who participate in them, indicating a 0-to-many relationship (`0..*`).
- **Battle:** Battles, the individual challenges within Tournaments, are designed to accommodate participation by multiple Groups, with a multiplicity of zero-to-many (`0..*`). Each Battle is capable of receiving multiple Submissions from each Group.
- **Group:** The Group entity denotes teams of Users (students) who participate in Battles. A Group can consist of one or more Users, thus forming a many-to-many relationship (`*..*`) with the User entity.
- **Submission:** Submissions are created by Groups and are associated with specific Battles. Each Submission is linked to one Battle, signifying a one-to-one relationship (`1..1`) in the context of evaluation. However, a Battle can have numerous Submissions linked to it, showcasing a one-to-many relationship (`1..*`).
- **Notification:** Notifications are triggered by system events and are sent to Users. A Notification has a `(*..1)` relationship to Tournament, as a Notification will always be sent due to events in the tournament, battles within the tournament, submission to the battle in the tournament or badges achieved in the tournament. This implies a (`0..*`) relationship from certain classes to Notifications, as a Notification *can* be triggered by a Battle, Badge or submission, but not necessarily by any of them. A many-to-many relationship (`*..*`) exists from a Notification to a Student, indicating that each Notification can be sent to a variety of Student (e.g. Registration Deadline Has Ended) and that a Student can receive several Notifications.
- **Badge:** The Badge entity is connected to Users based on their achievements within the system. Multiple Badges can be awarded to a single User, and many users can achieve a badge, highlighting a many-to-many relationship (`*..*`).

2.3 State charts

These two state diagrams gives an overview over the two most essential process within CodeKata-Battles, namely the Battle and Tournament.

Figure 2 shows the progression of a tournament in CodeKataBattles. Initiated by an educator, the "Drafting" phase involves filling in the tournament's details such as the title and registration deadline. Once established, the system transitions the tournament to "Open registration," allowing students to join until the deadline, after which it moves to "Closed registration." When the start date is surpassed, the tournament enters the "In progress" stage, which is where the battle can be had. This continues until the educator decides to end the event, bringing the tournament to its end and finalizing the standings.

Figure 3 displays the lifecycle of a battle within CodeKataBattles. A battle initial state the "Drafting" starts when an educator decides to create a battle within a tournament. Here the educator details the deadlines, the title, the description, and the automation scripts containing the test cases. Once that is done, and the battle's details are validated by the system, the battle is created and precedes to be open for registration. Here students within the tournament can in teams of 1 or more join the battle, however, once the registration deadline is passed the battle is closed for registrations. This waiting state is over when the start date of the battle is surpassed, commencing the "in progress" stage of the battle. Here students can submit as many solutions for the problem and get feedback. The stage ends when the submission deadline is finished.

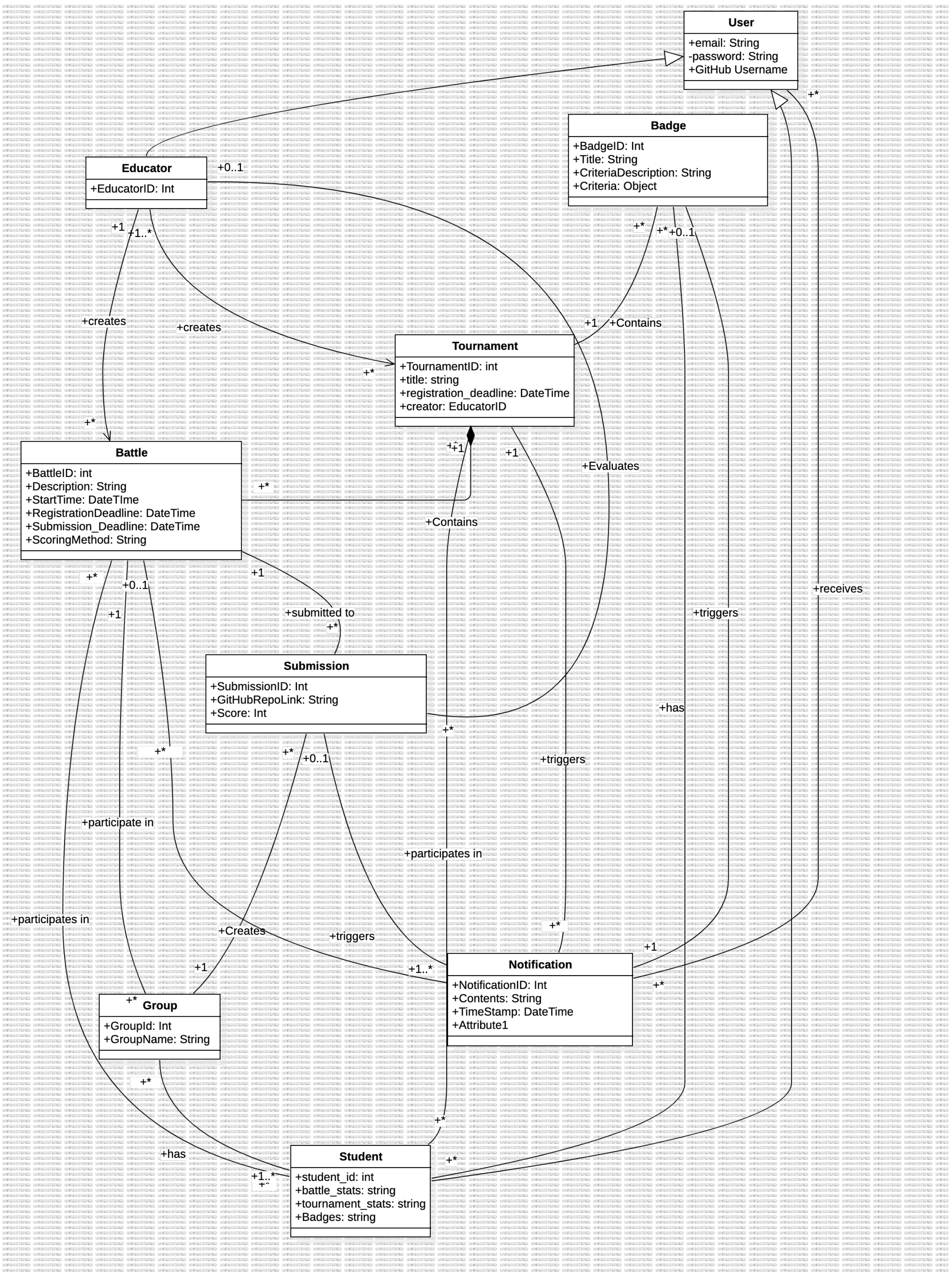


Figure 1: UML Domain Class Diagram

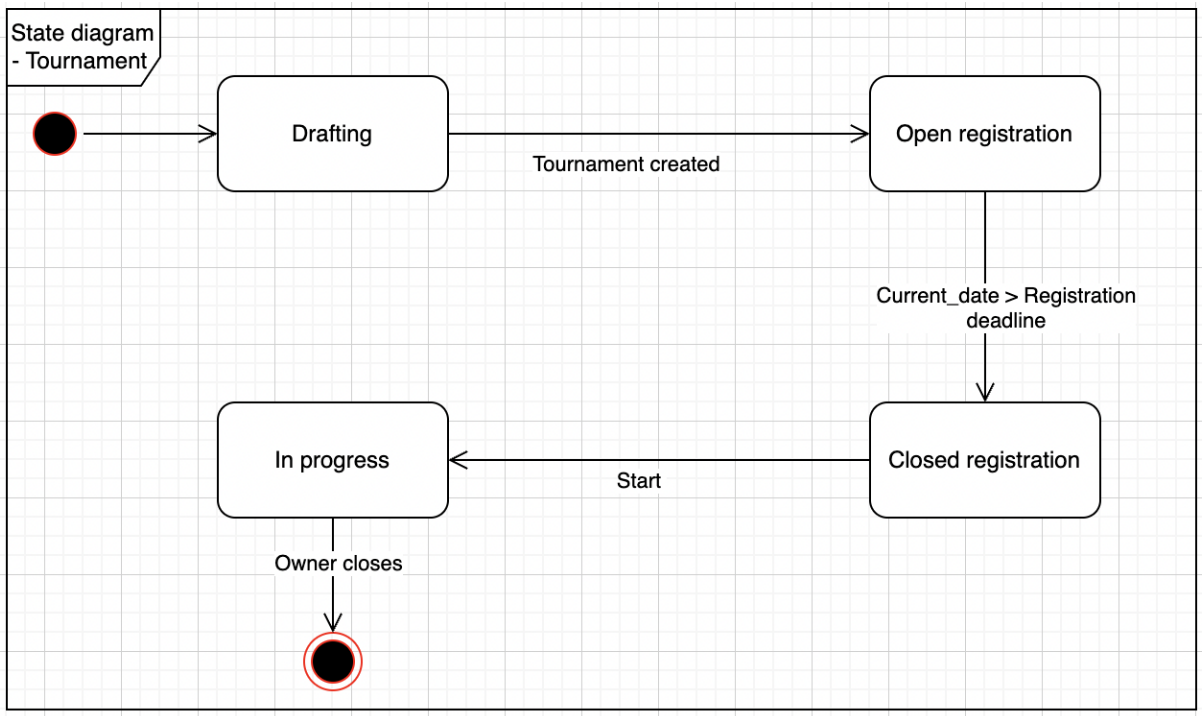


Figure 2: State Diagram - Tournament

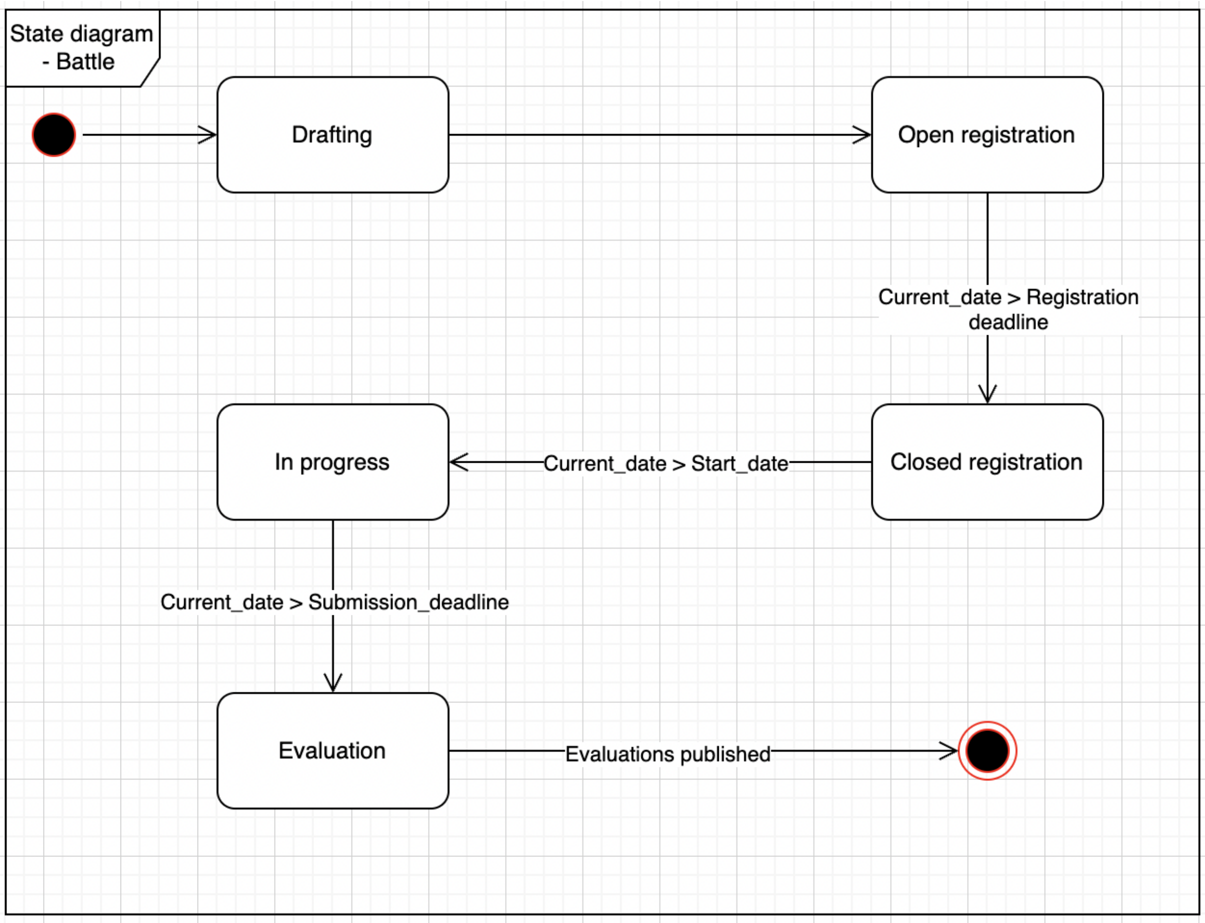


Figure 3: State Diagram - Battle

Now all the team's best solutions are put into the final ranking, and every participant in the battle is notified. Hereafter the Battle has ended.

2.4 Product functions

The following section will have a detailed description of the CodeKataBattles functions.

1. **Sign-Up & Log In**
   This functionality should enable the user to register as a user on the platform. Each user has to specify their email and their GitHub username. We will use the email as their username, and let them specify their passwords under standard security requirements. For the user to finalize their registration, and verification email is sent to the given email with a link for the user to confirm their access to the email. After the initial sign-up process is finished, the user can log on to the platform via their specified email and password.

2. **Create battle**
   Available to every educator is the ability to create a battle under a given tournament, where the educator has the required permissions. The educator then needs to specify a number of elements to finalize the creation of the battle. One element concerns the technical documents making up the code kata. Here the educator is required to specify a description of the battle, test cases specifying how a satisfactory solution should behave, and lastly include the automation scripts. Thereafter the educator has to make the formal configuration of the code kata, which entails setting the minimum and maximum size of the groups and setting the registration and final submission deadline. The last element to specify is the configuration of the scoring methodology for the battle. Here the educator can enable manual scoring or automate the process in a weighted manner to their liking.

3. **Join battle**
   Students can see the tournaments open for registration, and the open battles, in tournaments they already have joined. When selecting to join a battle, the user is prompted to create a group. The group can consist of the sole creator of the group, or the student can invite as many additional students as the battle allows. The other students need to already be registered for the tournament to be able to be invited. Each group needs a name unique to the battle participants.

4. **Create tournament**
   Only an educator can create a tournament. A tournament requires a title, registration deadline, and specification of badges for the tournament. Each badge needs a title, and a set of rules enabling the automation of giving out the badge to participants that fulfill the requirements for the badge. Beyond the registration deadline putting a threshold of when students can join the battle, the deadline also constrains battle deadlines in the tournaments to be after the battle. To enable other educators to create battles within the tournament, the original creator of the tournament has to add the educator by username to a list of educators with permission.

5. **Receive Notifications from battles**
   When a student or group has joined a battle they are placed on a notification list that enables them to receive notifications in several scenarios:

   (a) When the registration deadline is transgressed users receive a link to the GitHub repository containing code descriptions, build automation scripts, and other relevant files.

   (b) When a submission has been evaluated the user receives a notification containing indexes of the tests that the program has passed as well as the score for the solution.

   (c) When the submission deadline is transgressed the user receives a notification with the final score and rank in the battle.

6. **Receive Notifications from Tournaments**
   If a user is subscribed to a tournament the user is placed on a notification list so they can be notified in certain situations. The user is notified when a battle is created under a tournament. The user is notified of their tournament rank & score when the tournament is ended by a permitted educator. The user is notified if they qualify for any badges specified in the tournament.

7. **See tournament information**
   In order to keep up with their performance and upcoming battles, users can inspect Tournament information. This includes seeing previous, future, and ongoing battles of the tournaments along with relevant summary statistics, such as average score and number of participants. Additionally, the user can see their current rank and score on the tournament leaderboard. Users can also inspect if they have received any badges in the tournament.

8. **See battle information**
   This functionality allows users to keep up with the battles they are involved in. This includes seeing the attempts made by the user or group along with the number of tests passed by each attempt. The battle leaderboard reveals the rank and score of the user or group in the current battle. The information on remaining time until the submission deadline and the number of participants is also visible.

9. **Display Student Profile**
   This functionality allows students and educators to view student's battle and tournament involvement. This includes rankings, badges and achievements.

2.5 User characteristics

The CodeKataBattles system is primarily interacted with by two categories of users: Students and Educators.

2.5.1 Student

In the context of CodeKataBattles, a student is a type of user registered on the CodeKataBattle platform solely for participating in code kata battles and tournaments. They cannot create battles or tournaments. They have a device with an internet browser and internet connection so they can access the web application.

2.5.2 Educator

In the context of CodeKataBattles, an educator is a type of user registered on the CodeKata-Battle platform for creating and hosting tournaments in code kata battles and tournaments. They can also score submissions in battles where manual evaluation is enabled and she/he is owner of the tournament. They can also close down tournaments at will. They have a device with an internet browser and internet connection so they can access the web application.

2.6 Assumptions, dependencies and constraints

2.6.1 Regulatory Policies

The CodeKataBattle platform will ask for user personal information like email address linked with a GitHub user. Email addresses and Github user information won't be used for commercial purposes. Personal information will be processed in compliance with the GDPR.

2.6.2 Domain Assumptions

This section outlines the assumptions that are considered as given within the operational domain of the CodeKataBattle system. These are conditions that the system presumes to be true and are necessary for the correct operation and use of the platform.

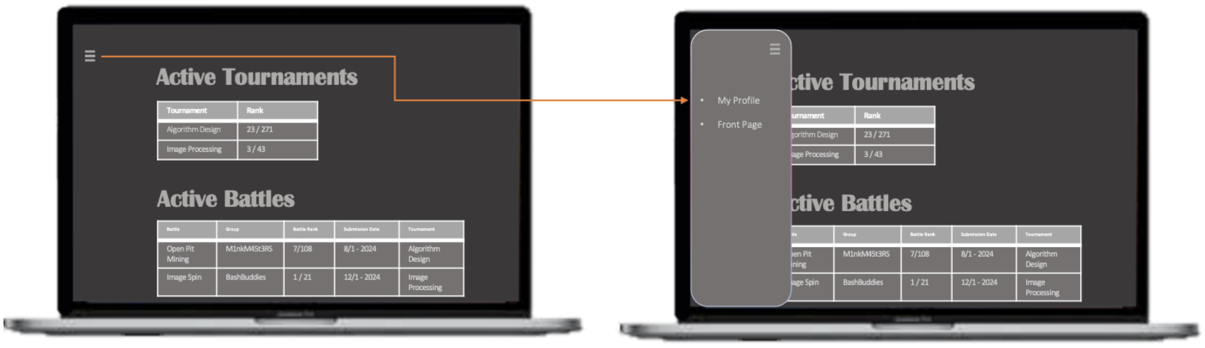
1. [**D1**] Users must have a stable internet connection to access and interact with the CodeKataBattle platform.
2. [**D2**] Users' computing devices must be capable of running the necessary software development tools for coding battles.
3. [**D3**] GitHub services are available and reliable as the platform depends on GitHub for repository hosting and Actions for automated workflows.
4. [**D4**] Educators and students must have a basic understanding of how to use version control systems, particularly GitHub.
5. [**D5**] Automated testing scripts provided in code katas are correct and capable of accurately evaluating the students' code submissions.
6. [**D6**] Users must allow notification services to receive timely updates from the CodeKataBattle platform.
7. [**D7**] The static analysis tools integrated with the platform for code quality assessment are functioning and accessible.
8. [**D8**] Students' submissions through GitHub are timestamped accurately to ensure fair evaluation of timeliness.
9. [**D9**] Educators inputting the battle information, including test cases and scoring configurations, must ensure their accuracy and clarity.
10. [**D10**] The computing environment used by students for development is compatible with the code kata requirements and specifications.
11. [**D11**] The system's timekeeping is synchronized with an accurate time source to ensure deadline enforcement.
12. [**D12**] Educators are responsible for defining clear and measurable objectives for battles and tournaments.
13. [**D13**] Students are expected to follow the test-first approach as specified in the battle requirements, creating tests before implementation.
14. [**D14**] Scores and feedback are assumed to be constructive and used for educational purposes, not as punitive measures.
15. [**D15**] The platform assumes all users act in good faith, not attempting to game the system or manipulate scores through unfair means.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

This section presents the user interfaces for the CodeKataBattles platform. The first section is dedicated to the user interfaces encountered by the student, and the second section to those encountered by the educator. While similar on many accounts as both students and educators need the ability to monitor and receive notifications about relevant tournaments and battles. However, students' interactions with battles and tournaments are limited to monitoring and subscribing, while educators can create new battles and tournaments. While the service needs to be accessed by a heterogeneous set of users, because a lot of the intricacies of the difference in use-cases are handled by GitHub, a lot of the functionalities available to students and educators are very closely related.



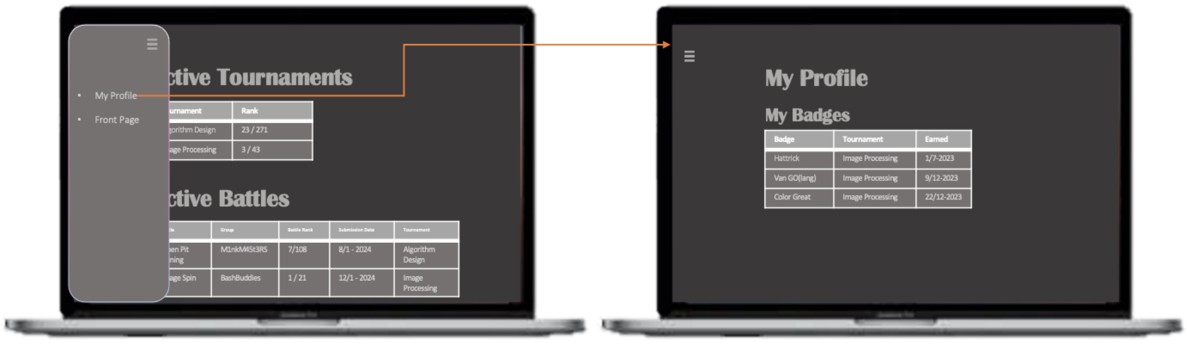
Active Tournaments

Tournament	Rank
Algorithm Design	23 / 271
Image Processing	3 / 43

Active Battles

Battle	Group	Battle Rank	Submission Date	Tournament
Open Pit Mining	M1nkM45t3RS	7/108	8/1 - 2024	Algorithm Design
Image Spin	BashBuddies	1 / 21	12/1 - 2024	Image Processing

Figure 4: User Interface: Front Page & Sidebar

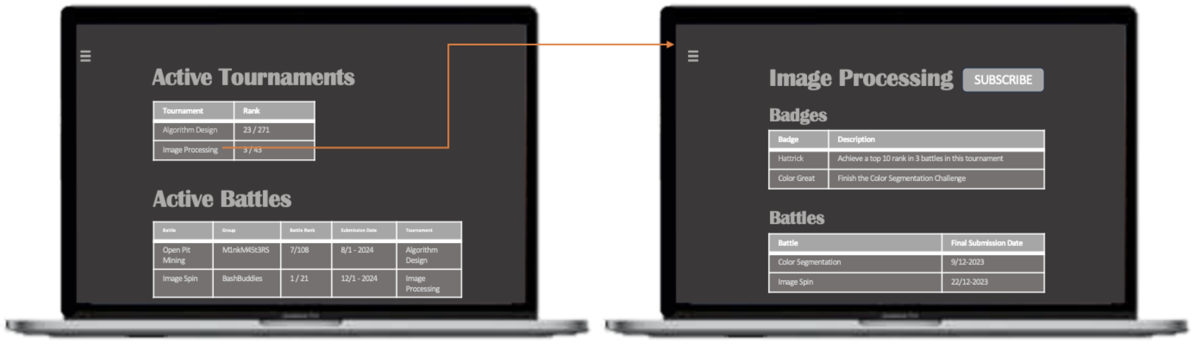


My Profile

My Badges

Badge	Tournament	Earned
Hattrick	Image Processing	1/7-2023
Van GO(lang)	Image Processing	9/12-2023
Color Great	Image Processing	22/12-2023

Figure 5: User Interface: Profile



Active Tournaments

Tournament	Rank
Algorithm Design	23 / 271
Image Processing	3 / 43

Active Battles

Battle	Group	Battle Rank	Submission Date	Tournament
Open Pit Mining	M1nkM45t3RS	7/108	8/1 - 2024	Algorithm Design
Image Spin	BashBuddies	1 / 21	12/1 - 2024	Image Processing

Image Processing

Badges

Badge	Description
Hattrick	Achieve a top 10 rank in 3 battles in this tournament
Color Great	Finish the Color Segmentation Challenge

Battles

Battle	Final Submission Date
Color Segmentation	9/12-2023
Image Spin	22/12-2023

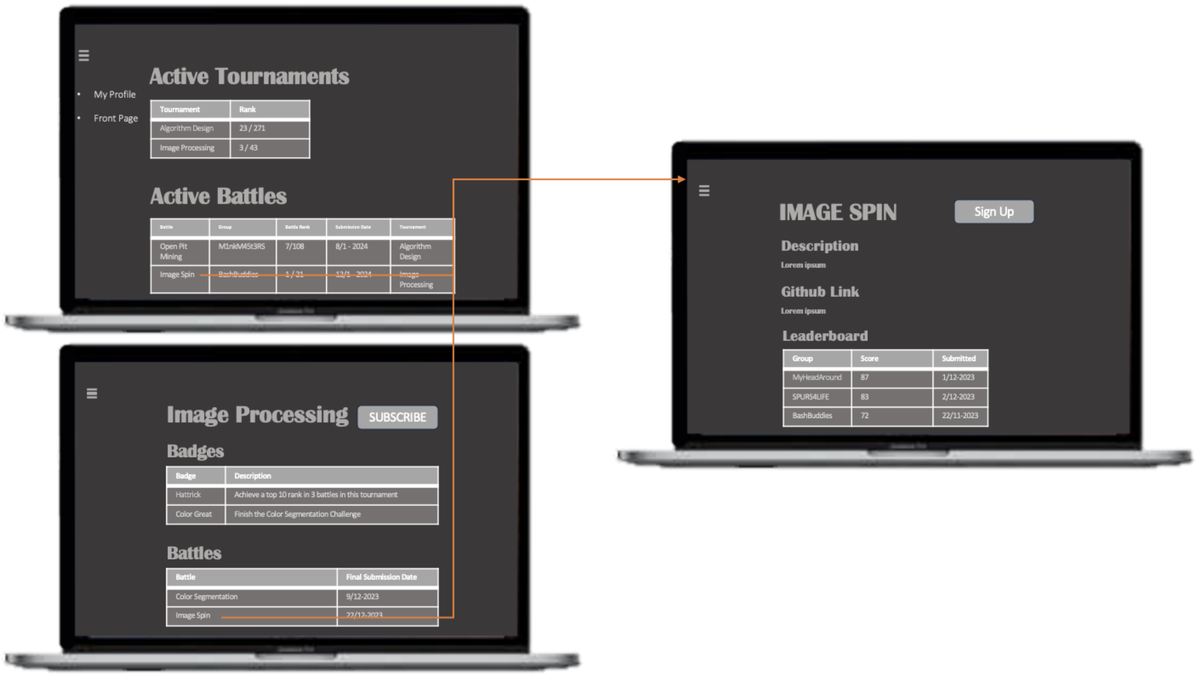
Figure 6: User Interface: Tournament Page

3.2 Students

3.3 Educators

3.3.1 Hardware Interfaces

All users, both Students and Educators can access the platform through a web browser. Therefore, all users need access to a device with an internet connection and browser installed. While note strictly required, a text editor is also recommended.



Active Tournaments

- My Profile
- Front Page

Tournament	Rank
Algorithm Design	23 / 271
Image Processing	3 / 43

Active Battles

Battle	Group	Battle Rank	Submission Date	Tournament
Open Pit Mining	M1nkM45t3RS	7/108	8/1 - 2024	Algorithm Design
Image Spin	BashBuddies	1 / 21	12/1 - 2024	Image Processing

Image Processing SUBSCRIBE

Badges

Badge	Description
Hattrick	Achieve a top 10 rank in 3 battles in this tournament
Color Great	Finish the Color Segmentation Challenge

Battles

Battle	Final Submission Date
Color Segmentation	9/12-2023
Image Spin	22/12-2023

IMAGE SPIN Sign Up

Description

Lorem ipsum

Github Link

Lorem ipsum

Leaderboard

Group	Score	Submitted
MyHeadAround	87	1/12-2023
SPURS4LIFE	83	2/12-2023
BashBuddies	72	22/11-2023

Figure 7: User Interface: Battle Page

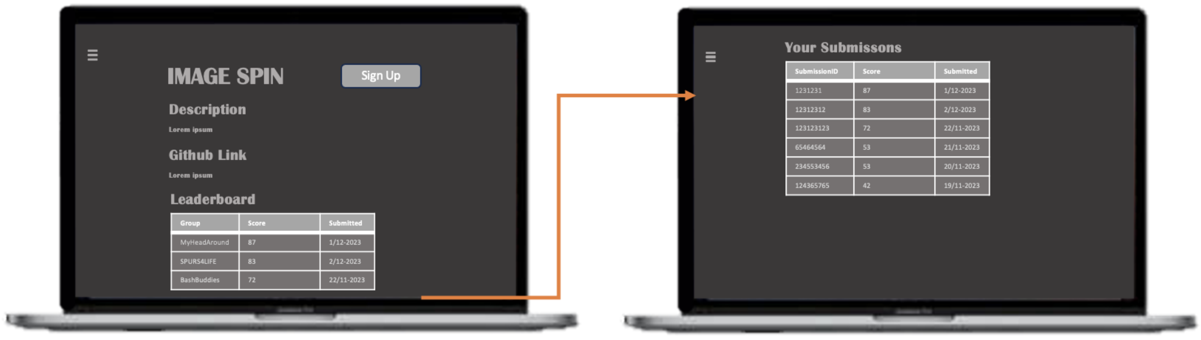


IMAGE SPIN Sign Up

Description

Lorem ipsum

Github Link

Lorem ipsum

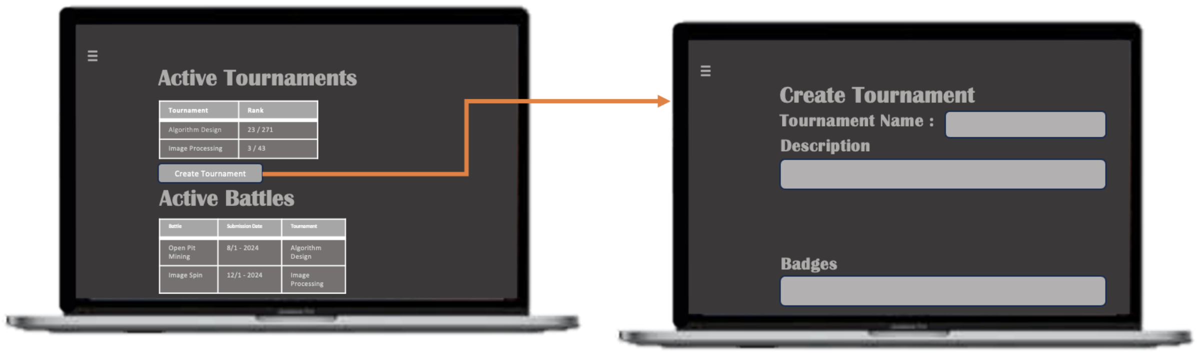
Leaderboard

Group	Score	Submitted
MyHeadAround	87	1/12-2023
SPURS4LIFE	83	2/12-2023
BashBuddies	72	22/11-2023

Your Submissons

SubmissionID	Score	Submitted
1231231	87	1/12-2023
12312312	83	2/12-2023
123123123	72	22/11-2023
65464564	53	21/11-2023
234553456	53	20/11-2023
124365765	42	19/11-2023

Figure 8: User Interface: Battle Page (Submission Log)



Active Tournaments

Tournament	Rank
Algorithm Design	23 / 271
Image Processing	3 / 43

Create Tournament

Active Battles

Battle	Submission Date	Tournament
Open Pit Mining	8/1 - 2024	Algorithm Design
Image Spin	12/1 - 2024	Image Processing

Create Tournament

Tournament Name :

Description

Badges

Figure 9: User Interface: Create Tournament Page

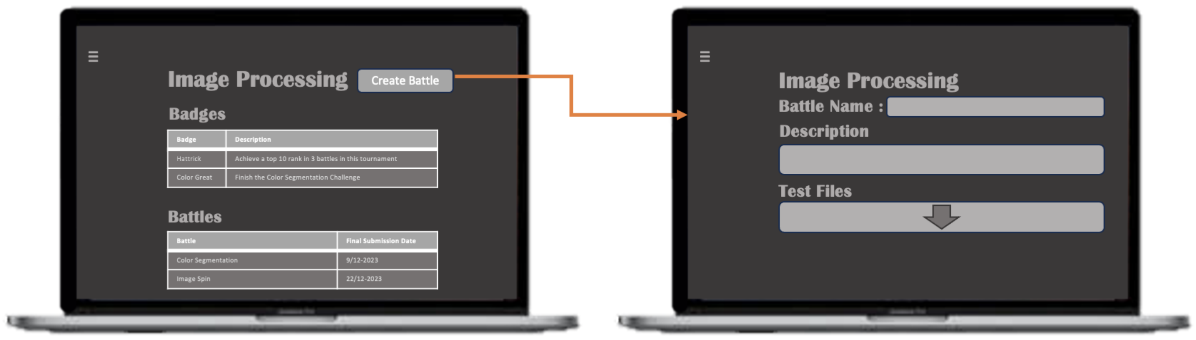


Image Processing

Badges

Badge	Description
Hattrick	Achieve a top 10 rank in 3 battles in this tournament
Color Great	Finish the Color Segmentation Challenge

Battles

Battle	Final Submission Date
Color Segmentation	9/12-2023
Image Spin	22/12-2023

Figure 10: User Interface: Create Battle Page

3.3.2 Software Interfaces

The system utilizes one primary external service, namely Github. A Github action is set up, such that any pushes to a forked repository from a given battle is automatically pulled by the system. Then a predefined test script evaluates the implementation on a variety of tests.

3.4 Functional Requirements

The CodeKataBattle system is designed to facilitate and manage competitive coding exercises, known as code katas. The system shall support the following functional requirements to ensure a comprehensive and engaging learning experience for students:

1. [**R1**] The system shall allow users to sign up using their GitHub credentials (i.e. their email and GitHub Username).
2. [**R2**] The system shall send a verification email to confirm a user's account upon sign-up.
3. [**R3**] Educators shall be able to create tournaments with a specified title.
4. [**R4**] Educators shall be able to set registration deadlines for tournaments.
5. [**R5**] Educators shall be able to define tournament badge criteria.
6. [**R6**] The system shall notify users of new tournaments.
7. [**R7**] Users shall be able to subscribe to tournaments by a given deadline.
8. [**R8**] Educators shall be able to create battles within tournaments.
9. [**R9**] Educators shall be able to upload technical documents for battles.
10. [**R10**] Educators shall be able to set the minimum and maximum number of students per group for battles.
11. [**R11**] Educators shall be able to set a registration deadline for battles.
12. [**R12**] Educators shall be able to set a final submission deadline for battles.
13. [**R13**] Educators shall be able to configure scoring methodologies for battles.
14. [**R14**] Students shall be able to form teams to join battles.
15. [**R15**] The system shall provide a GitHub repository link for the code kata to registered teams after the registration deadline.

16. [**R16**] The system shall automatically score submissions based on the results of test cases.
17. [**R17**] The system shall evaluate the timeliness of submissions.
18. [**R18**] The system shall assess the quality level of source code through static analysis.
19. [**R19**] Educators shall have the option to manually score submissions.
20. [**R20**] The system shall update battle scores in real-time upon new commits on GitHub.
21. [**R21**] The system shall notify participants of final battle ranks after the consolidation phase.
22. [**R22**] The system shall update personal tournament scores with the sum of battle scores.
23. [**R23**] The system shall maintain a visible tournament rank for each student.
24. [**R24**] Educators shall be able to define gamification badges with specific rules.
25. [**R25**] The system shall display badges on student profiles.
26. [**R26**] Educators shall be able to close tournaments and trigger final rank notifications.
27. [**R27**] The system shall enable boolean expressions to define rules of badges and achievements.
28. [**R28**] Students shall be able to visualize their performance.
29. [**R29**] Students shall be able to visualize their tournament information.
30. [**R30**] Students shall be able to track their battle involvement.
31. [**R31**] Students shall be able to see detailed logs of their attempt outcomes.

Definition of use case diagrams, use cases and associated sequence/activity diagrams, and mapping on requirements

3.5 Performance Requirements

The CodeKataBattles platform, recognizing the unique nature of code kata activities, has tailored performance requirements, especially concerning system downtime. Since a substantial part of the participation occurs in personal text editors or IDEs, the platform can accommodate longer downtimes without significantly impacting user productivity. An accepted downtime of up to half an hour is considered reasonable, allowing users to continue their work offline during such periods.

To minimize disruption, scheduled maintenance is planned during off-peak hours, and users are notified in advance through various channels. In case of unexpected downtimes, prompt communication is ensured, providing users with information and an estimated resolution time. Despite this tolerance for downtime, ensuring data integrity remains paramount. The platform is equipped with robust backup mechanisms to safeguard user data and maintain accessibility after downtimes.

These performance requirements reflect a balance between user experience and the practicalities of system maintenance for CodeKataBattles, catering to the platform's specific workflow.

3.6 Design Constraints

3.6.1 Standards compliance

Regarding data privacy, the CKB platform adheres to the General Data Protection Regulation (GDPR), which governs data protection and privacy for individuals within the European Union (EU) and the European Economic Area (EEA). Additionally, the system must comply with international standards for using and representing dates and times.

3.6.2 Hardware limitations

Here we report relevant hardware requirements. The only real hardware requirement is a machine with an Internet connection (2G/3G/4G/Wi-Fi) and a modern web browser

3.7 Software System Attributes

3.7.1 Reliability

As mentioned in section 3.5 this service does not warrant complex infrastructure to aggressively reduce downtime, due to the amount of work being done in text editors or IDEs not directly connected to the platform. This is also the case for data, as all code is stored in Github repositories, however, all data related to the gamification aspect of the platform is stored only in the system.

3.7.2 Security

As for any system handling personal information, such as CodeKataBattles does with information regarding educators and students, security is vital. To ensure compliance with security standards, encryption of passwords within the database must be applied. The same encryption need is present for data in transmission over the internet, to avoid interception-tactics. User's access to data must be a constraint to information deemed relevant for them through the use of role-based access control.

3.7.3 Availability

CodeKataBattles is not a critical service, so the availability role is mainly to create a good user experience. Furthermore as mentioned in 3.5 user productivity can exist in downtime. Therefore the system should aim to provide at least 99% availability, meaning that the average time between occurrences of a failure and service recovery (MTTR) must be less or equal to 3,65 days per year.

3.7.4 Maintainability

The system is required to ensure a high level of maintainability. This entails the utilization of appropriate design patterns and adherence to recognized coding standards. The code must be well-documented, and practices such as hard-coding are strictly avoided. Moreover, the system should include a comprehensive testing routine, which is expected to cover a minimum of 75% of the entire codebase, with the exception of interface code.

3.7.5 Portability

The web application should be compatible with all operating systems that support a web browser, including Windows, macOS, Linux, and others.

Name	Login User
Actors	Student & educator
Entry Condition	The student/educator has opened the CodeKata web application
Event flow	1 - The user inserts his username and password in the form 2 - The user clicks on the "Login" button 3 - The system checks the credentials 4 - The application shows the proper dashboard
Exit condition	The user has access to the services for the right interface provided by the CodeKata web application
Exception	3 - The data inserted are not valid. The system returns to the entry condition.

Table 1: Use Case Specification

Name	Join tournament
Actors	Student
Entry Condition	1 - The student is logged into the CodeKata platform 2 - The student is on the home page of the CodeKata web application 3 - The registration deadline has not yet passed
Event flow	1 - The student selects one of the available tournaments on the homepage 2 - The student clicks on the "Join" button 3 - The system adds the specific user to the participant list for the given tournament 4 - The tournament is added to the overview of tournaments the student is currently participating in
Exit condition	The student now has access to the battles within the tournament and is returned to the tournament home page
Exception	

Table 2: Use Case Specification

Name	Join battle
Actors	Student
Entry Condition	1 - The student is logged in 2 - The student is subscribed to the tournament, to which the battle belongs 3 - The battle is open for registration 4 - The student resides on the battles page
Event flow	1 - The student clicks on the "Join" button 2 - The student is prompted to specify participant(/s) by username in its group 3 - The student enters the name of the group 4 - The system adds the specific user/team to the participant list for the given battle 5 - The battle is added to the overview of battles the student is currently participating in
Exit condition	The student will now receive notifications relevant to the given battle and is returned to the battle's home page

Table 3: Use Case Specification

Name	Create Battle
Actors	Educator
Entry Condition	1 - The educator is logged into the CodeKata platform. 2 - The educator has the details of a specific tournament displayed.
Event flow	1 - The educator clicks the “Create Battle” button. 2 - The educator uploads a brief textual description of the battle. 3 - The educator uploads a software project with build automation scripts. 4 - The educator sets the registration and submission deadlines. 5 - The educator selects the scoring methodology. 6 - The system confirms the creation of the battle.
Exit condition	The battle is created, enabling educator management within the tournament.
Exception	If required details are missing or incorrect, the educator is prompted for correction.

Table 4: Use Case Specification for Creating a Battle

Name	Manual Battle Evaluation
Actors	Educator
Entry Condition	1 - The educator is logged in. 2 - The educator is the owner of the battle. 3 - The submission deadline for the battle has passed.
Event flow	1 - The educator navigates to the battle’s submission list. 2 - The educator selects a submission for review. 3 - The educator assesses the submission, assigning a manual score. 4 - The system updates the battle’s rankings with the new score. 4 - The system notifies the creator/s of the submissions of evaluation.
Exit condition	The evaluated submission has been scored and displayed in the battle’s rankings.
Exception	If an error occurs during scoring, the educator is notified to retry.

Table 5: Use Case Specification for Evaluating a Submission

Name	Create a Badge
Actors	Educator
Entry Condition	1 - The educator is logged into the CodeKata platform. 2 - The educator is the creator of the tournament.
Event flow	1 - The educator selects the option to create a new badge. 2 - The educator inputs the badge name, description, and criteria. 3 - The system validates the input and creates the badge.
Exit condition	A new badge is created and available for awarding to students in the tournament.
Exception	If the criteria are not valid, the educator is prompted to correct the information.

Table 6: Use Case Specification for Creating a Badge

Name	Check Tournament Ranking
Actors	Student
Entry Condition	1 - The student is logged into the CodeKata platform. 2 - The student is participating in the tournament.
Event flow	1 - The student navigates to the tournament overview. 2 - The student selects the tournament to view. 3 - The system displays the current ranking and score of all participants.
Exit condition	The student views their current standing within the tournament rankings.
Exception	If the rankings are not available, the battle’s start time is yet to be surpassed.

Table 7: Use Case Specification for Checking Tournament Ranking

Name	Automatic Scoring of a Submission
Actors	System
Entry Condition	A group has submitted their solution.
Event flow	1 - The system retrieves the latest commit from the group's GitHub repository. 2 - The system runs automated tests against the submission. 3 - The system calculates the score based on the test results. 4 - The system updates the battle's leaderboard with the new score.
Exit condition	The group's submission is scored and the leaderboard reflects the updated score.
Exception	Errors during test execution or score calculation prompt system retry.

Table 8: Use Case Specification for Automatic Scoring of a Submission

Name	Announcing Tournament Results
Actors	System
Entry Condition	1 - All battles' submission deadlines within the tournament have passed. 2 - The tournament creator clicks the button on the tournament page to end it.
Event flow	1 - The system compiles final scores from all tournament battles. 2 - The system determines tournament winners and badge recipients. 3 - The system sends notifications to participants with final standings. 4 - The system updates the tournament page with the final results.
Exit condition	Participants get final results; standings are shown on the tournament page.
Exception	Failure in result compilation or notification delivery triggers a retry.

Table 9: Use Case Specification for Announcing Tournament Results

Name	View Tournament Ranking
Actors	Student, Educator
Entry Condition	The user is logged into the CodeKata platform and participating in a tournament.
Event flow	1 - The user navigates to the specific tournament section. 2 - The user selects the option to view rankings. 3 - The system displays the leaderboard with student names and scores. 4 - User reviews the ranking positions and individual scores.
Exit condition	The user views their tournament ranking and scores.
Exception	If the leaderboard isn't updated, the system suggests a page refresh.

Table 10: Use Case Specification for Viewing Tournament Ranking

Name	Push a new commit to a battle solution
Actors	Student
Entry Condition	1 - The student is logged into the CodeKata platform. 2 - The student is part of a group in an ongoing battle.
Event flow	1 - The student updates their submission, by pushing a new commit in the forked GitH 2 - The system validates the submission and confirms receipt. 3 - The system queues the submission for scoring.
Exit condition	Submission is successfully uploaded and awaiting evaluation. Automatic submission ev

Table 11: Use Case Specification for Submitting a Solution

Name	Give out badges
Actors	System
Entry Condition	1 - The educator in charge has closed the tournament. 2 - The final tournament rank became available.
Event flow	1 - The badges associated to the turnament and the badges‘s rules, which must be fulfi
Exit condition	Students involved in the tournament are notified of the final tournament rank and the

Table 12: Use Case Specification for Submitting a Solution

Name	Create GitHub battle repository
Actors	System
Entry Condition	1 - The registration deadline of a battle has expired.
Event flow	1 - The system creates a GitHub repository for the battle, containing the kata code.
Exit condition	Students of teams subscribed to the battle are being sent the link to the repository.

Table 13: Use Case Specification for Submitting a Solution

Name	Finalise team battle score
Actors	System
Entry Condition	1 - The submission deadline of the battle has expired 2 - Both mandatory automatic as well as optional manual evaluation have been comple
Event flow	1 - The teams final score of the battle is determined
Exit condition	1 - The teams members‘ tournament score is updated based on their achieved final bat

Table 14: Use Case Specification for Submitting a Solution

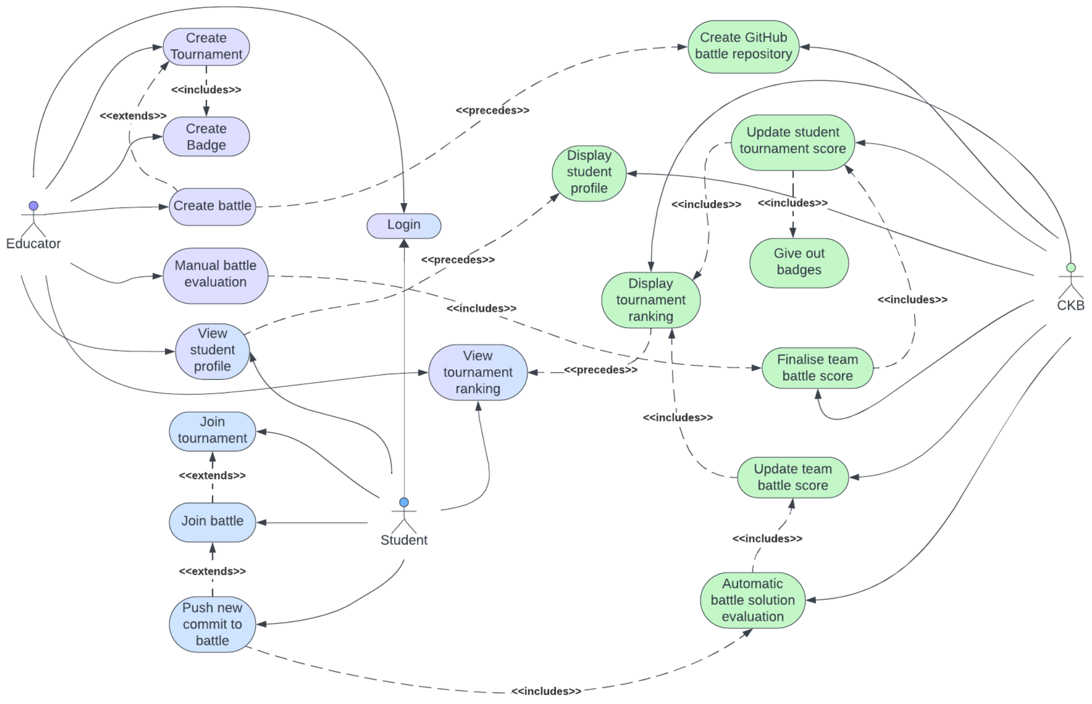


Figure 11: Use case diagram.

3.8 Requirements Mapping

This section maps the functional requirements and domain assumptions to the specific goals of the CodeKataBattle project. This mapping ensures that each goal is achievable through the fulfillment of certain requirements under the given assumptions.

3.9 Goals to Requirements and Domain Assumptions

[G2] Enable Educators to set up test-driven coding challenges including automated feedback online	
Requirements	**Domain Assumptions**
R3-R5 Tournament creation with badges	D4 Educators' understanding of version control
R8-R13 Battle setup with technical documents and deadlines	D5 Correct automated testing scripts
R19 Manual scoring option for educators	R27 Boolean expressions for badge rules

[G2] Enable Educators to set up test-driven coding challenges including automated feedback online	
Requirements	**Domain Assumptions**
R3-R5 Tournament creation with badges	D4 Educators' understanding of version control
R8-R13 Battle setup with technical documents and deadlines	D5 Correct automated testing scripts
R19 Manual scoring option for educators	R27 Boolean expressions for badge rules

[G3] Simulate a Real-world software development scenario through the use of GitHub and GitHub Actions.	
Requirements	**Domain Assumptions**
R15 Use of GitHub repositories for code katas	D10 Compatibility with code kata requirements
R17 Timeliness evaluation of submissions	D13 Adherence to test-first development approach

[G4] Enable Educators to set up test-driven coding challenges including automated feedback online	
Requirements	**Domain Assumptions**
R22 Update of personal tournament scores	D14 Use of scores for constructive feedback
R23 Maintenance of visible tournament rank	D15 Fair use of the platform by users
R28-R30 Performance visualization tools	

3.10 Sequence Diagrams

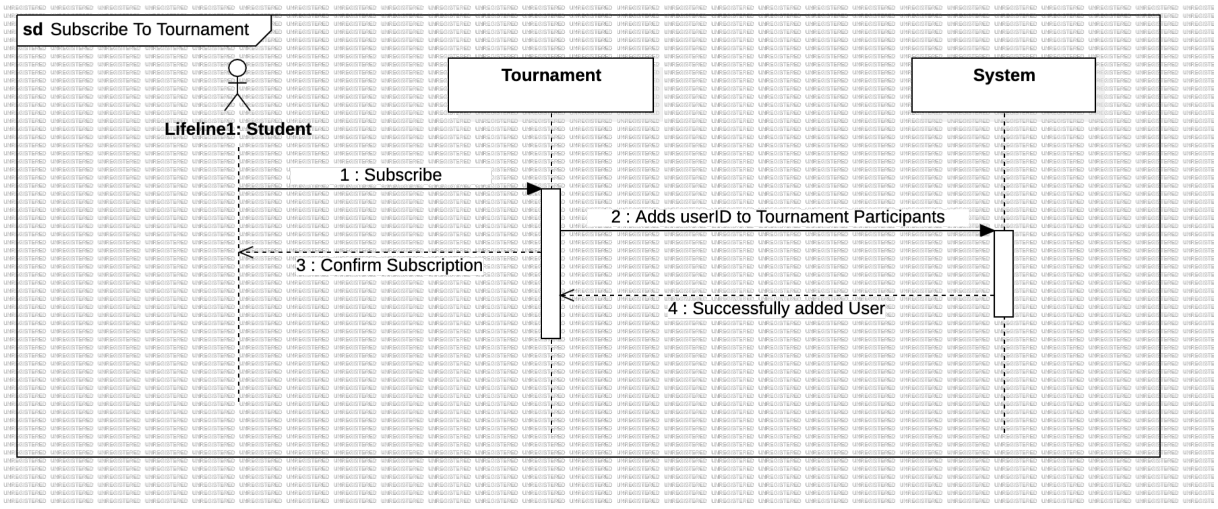

sd Subscribe To Tournament
Lifeline1: Student
Tournament
System
1 : Subscribe
2 : Adds userID to Tournament Participants
3 : Confirm Subscription
4 : Successfully added User

Figure 12: Sequence Diagram: Subscribe to Tournament

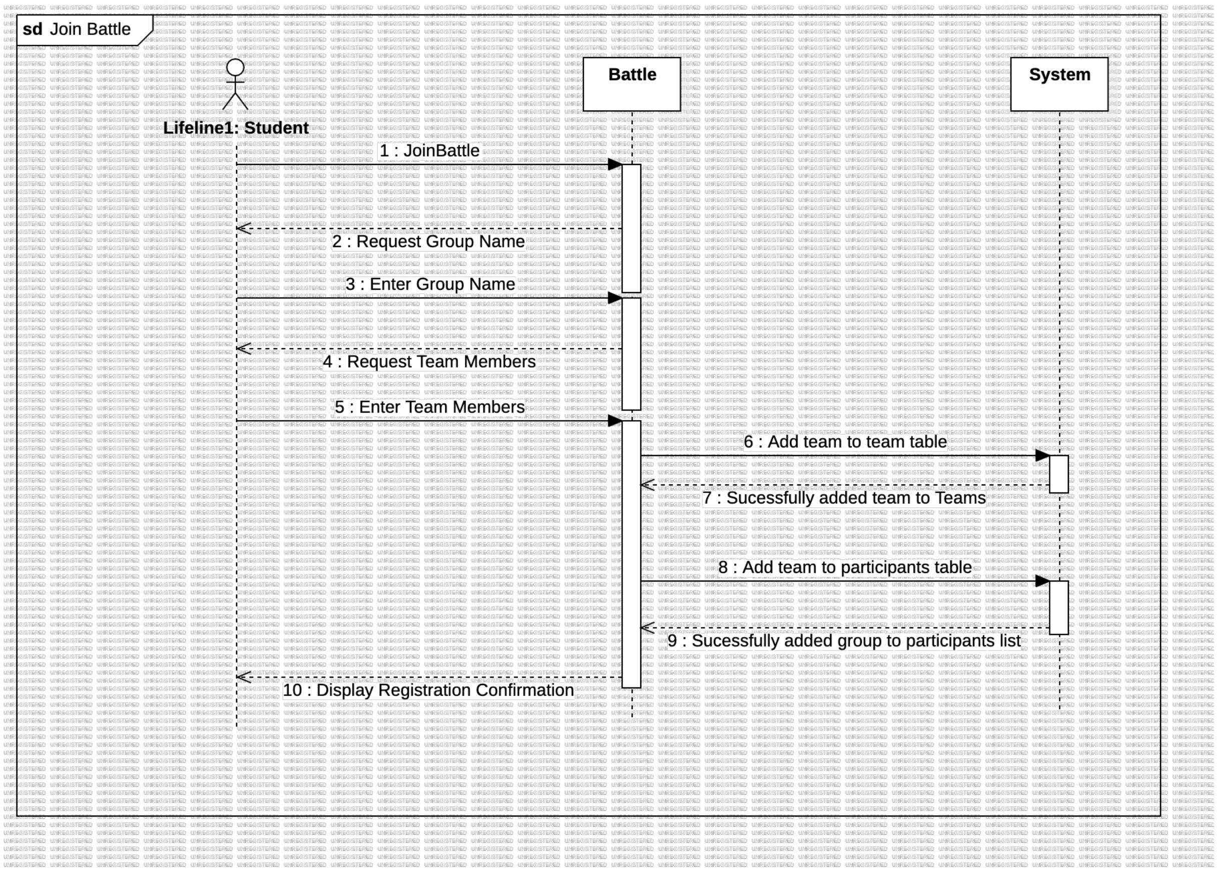

sd Join Battle
Lifeline1: Student
Battle
System
1 : JoinBattle
2 : Request Group Name
3 : Enter Group Name
4 : Request Team Members
5 : Enter Team Members
6 : Add team to team table
7 : Sucessfully added team to Teams
8 : Add team to participants table
9 : Sucessfully added group to participants list
10 : Display Registration Confirmation

Figure 13: Sequence Diagram: Join Battle

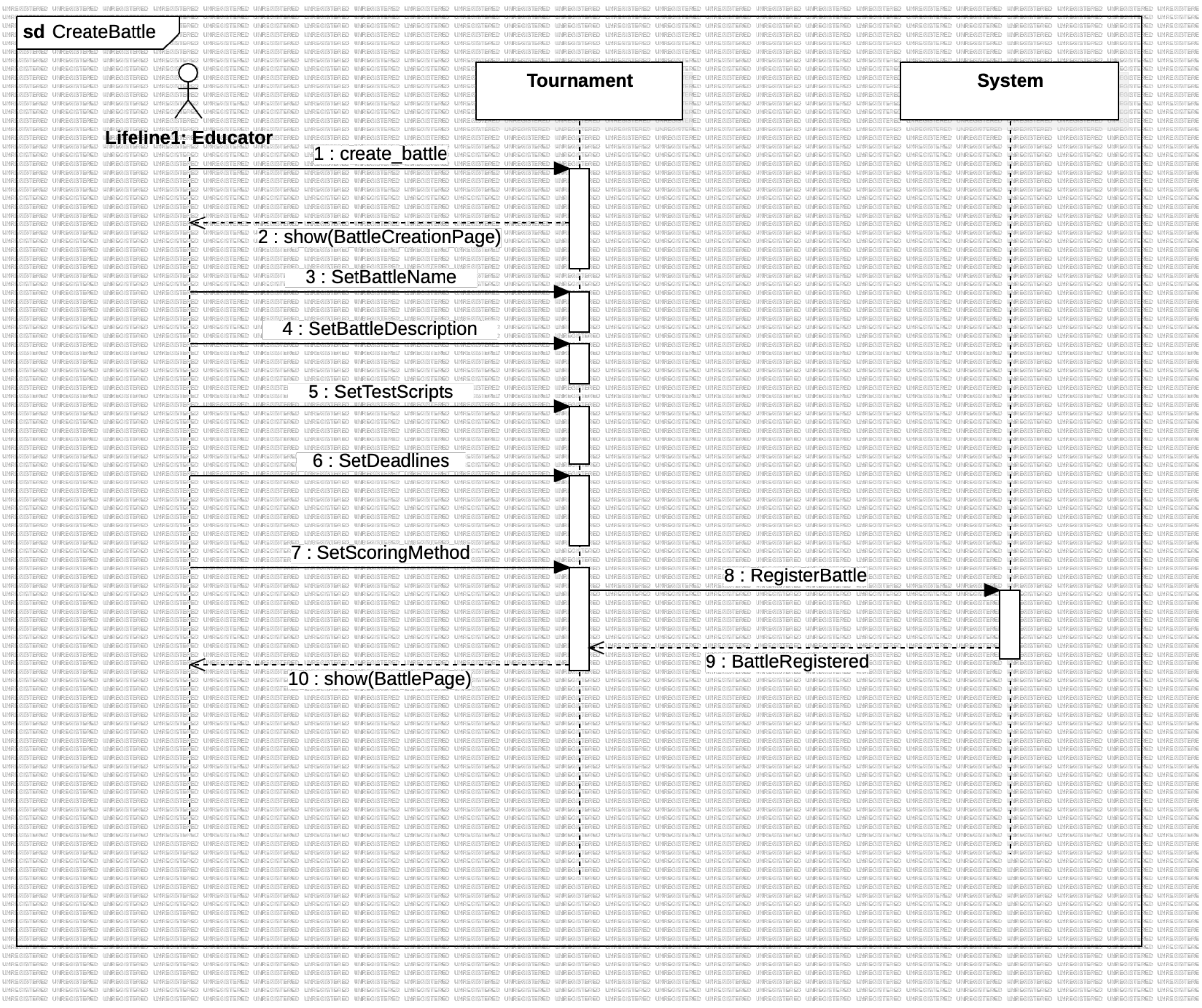


Figure 14: Sequence Diagram: Create Battle

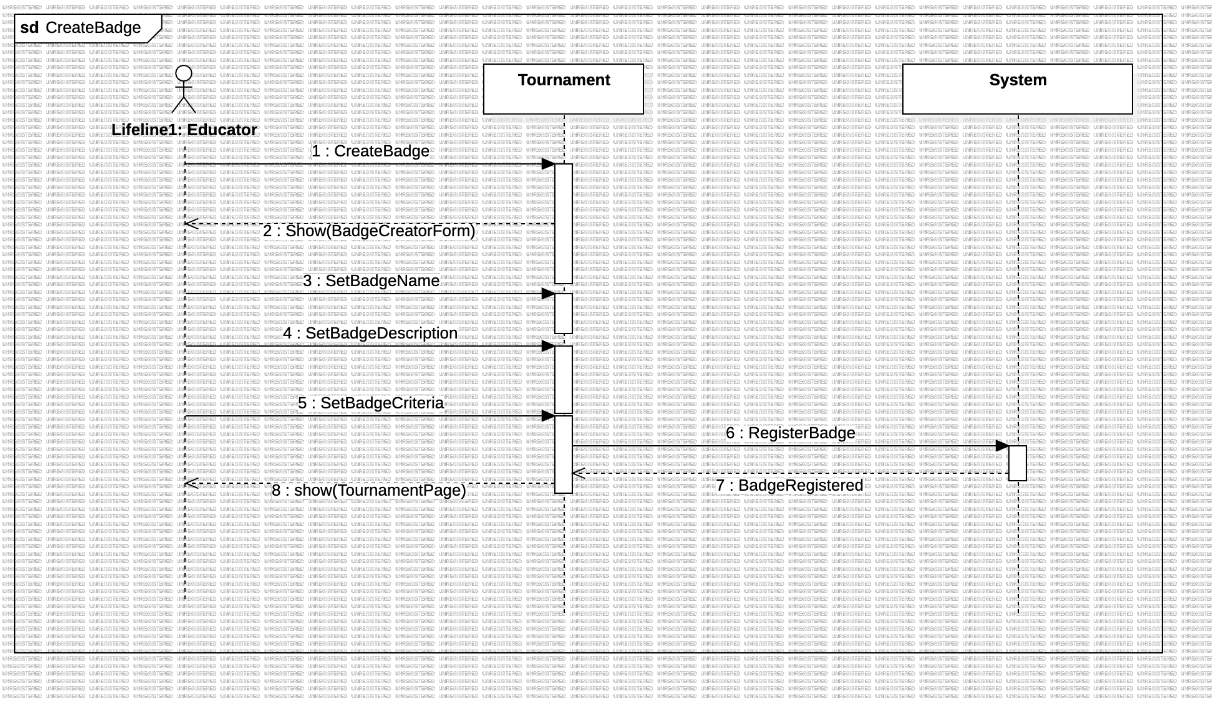


Figure 15: Sequence Diagram: Create Badge

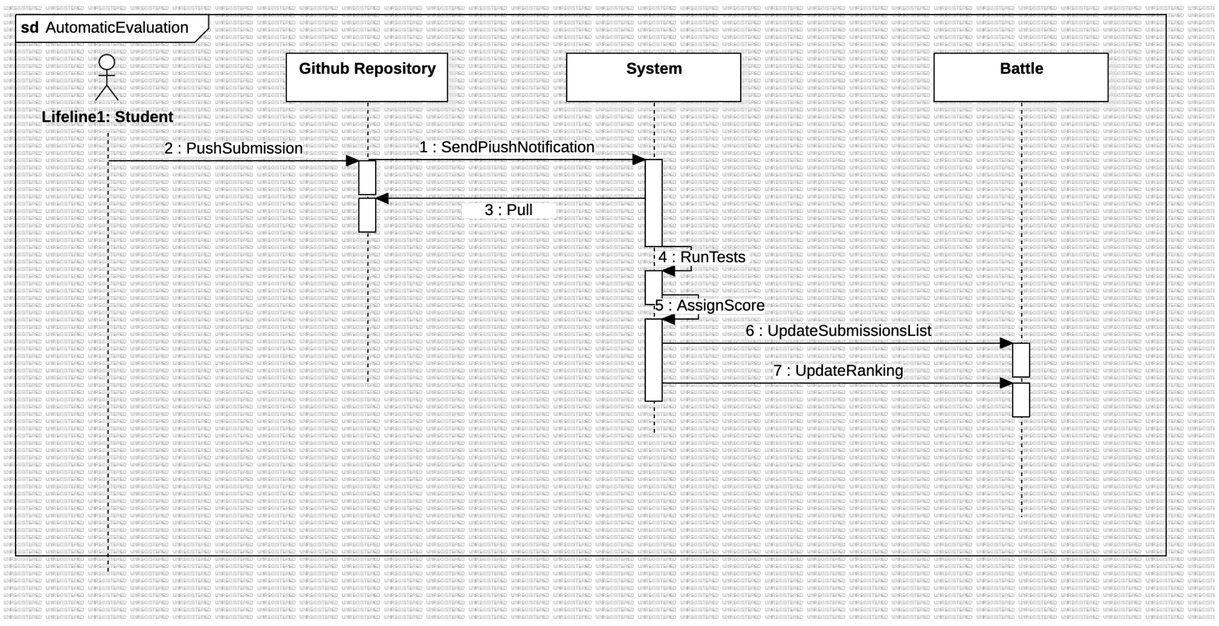


Figure 16: Sequence Diagram: Automatic Evaluation

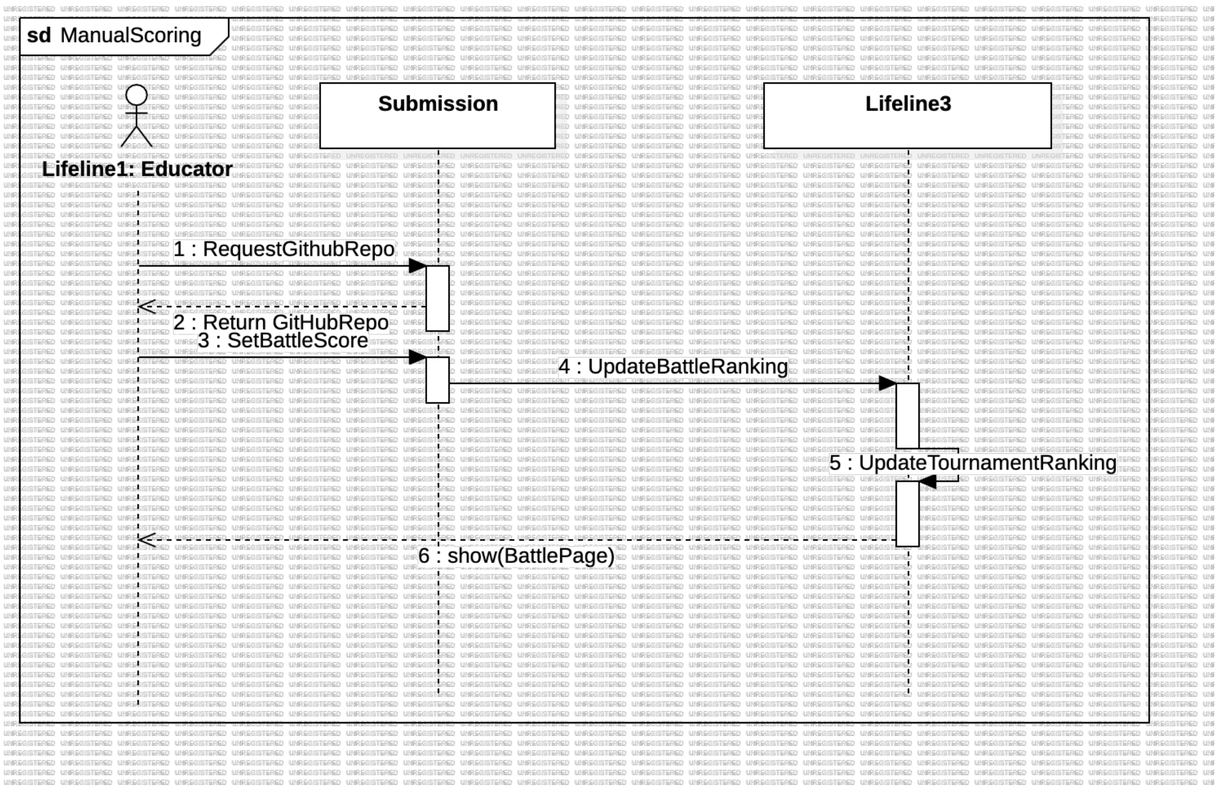


Figure 17: Sequence Diagram: Manual Evaluation

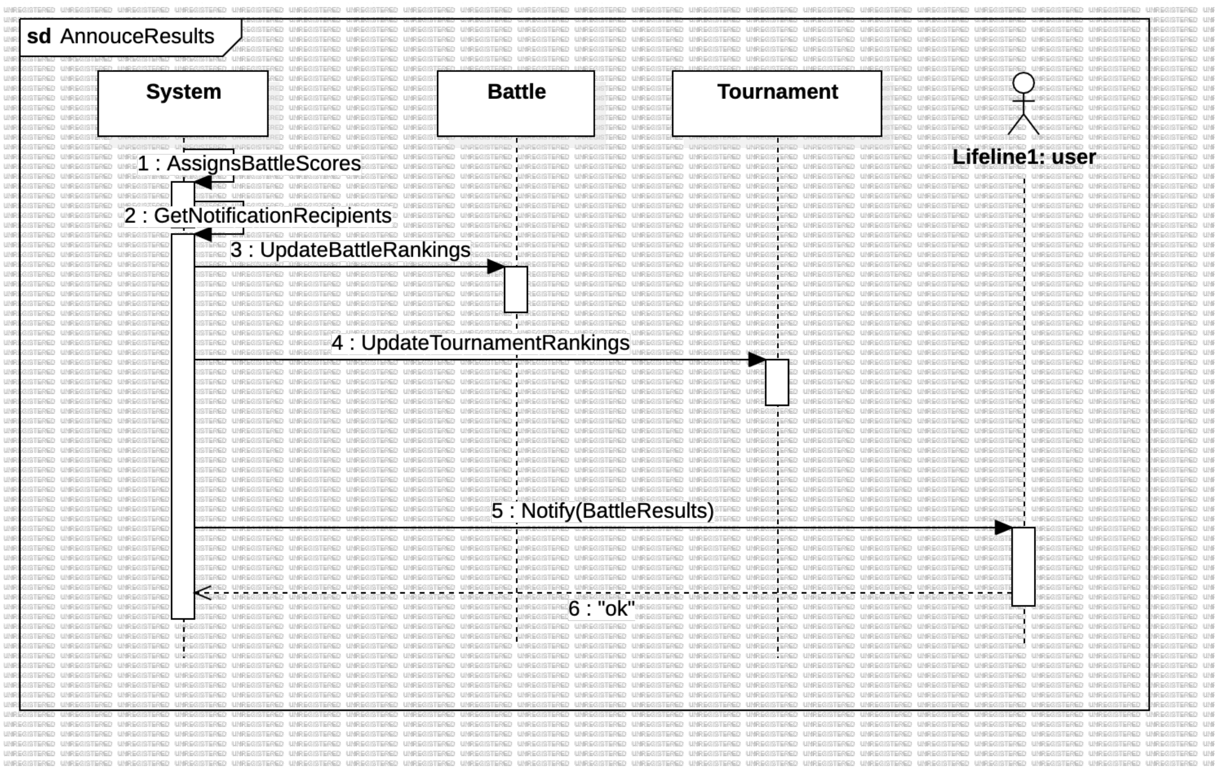


Figure 18: Sequence Diagram: Announce Battle Results

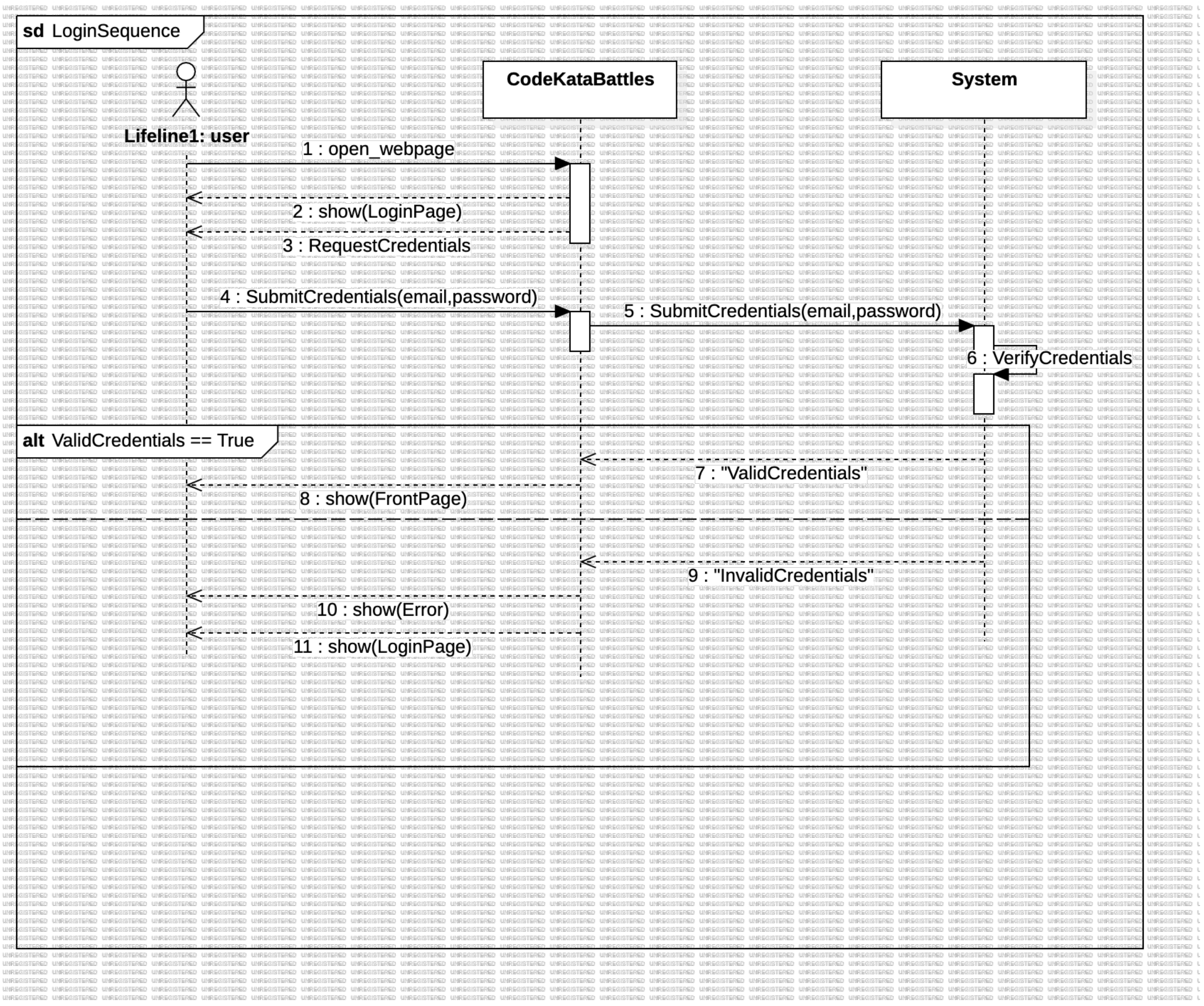
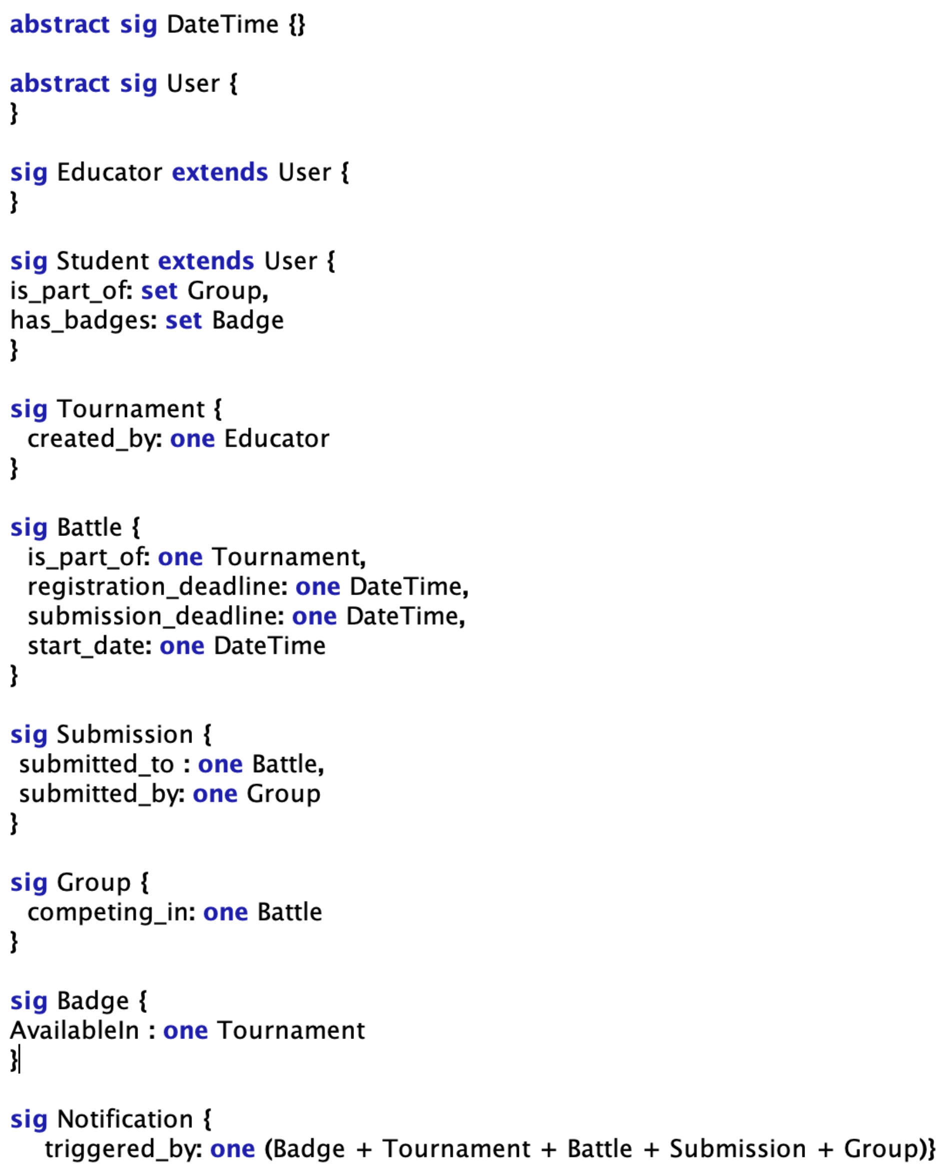


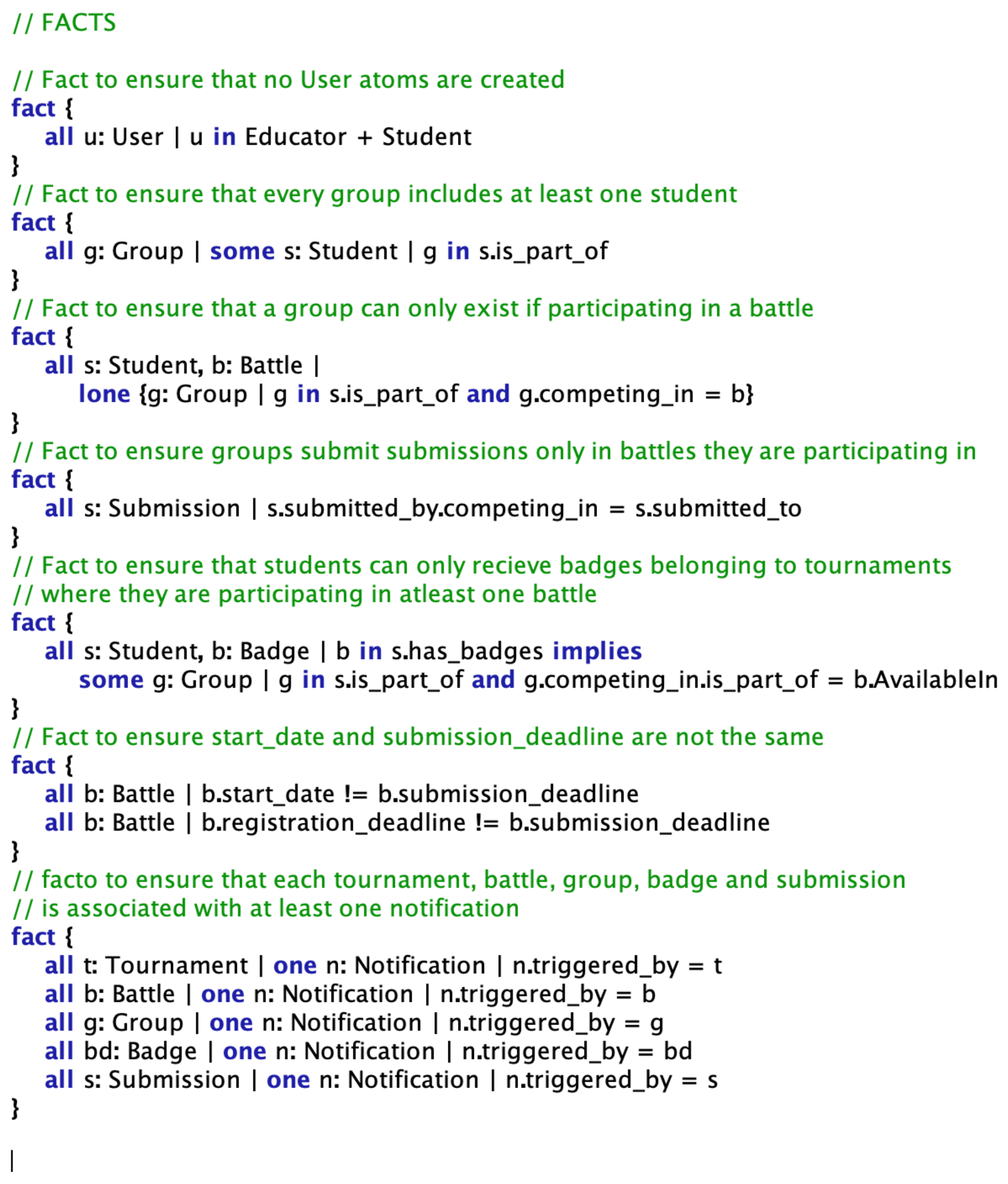
Figure 19: Sequence Diagram: Log-in Sequence

4 Formal Analysis Using Alloy



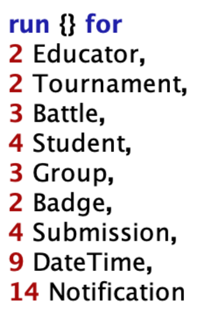
```
abstract sig DateTime {}

abstract sig User {
}

sig Educator extends User {
}

sig Student extends User {
is_part_of: set Group,
has_badges: set Badge
}

sig Tournament {
  created_by: one Educator
}

sig Battle {
  is_part_of: one Tournament,
  registration_deadline: one DateTime,
  submission_deadline: one DateTime,
  start_date: one DateTime
}

sig Submission {
 submitted_to : one Battle,
 submitted_by: one Group
}

sig Group {
  competing_in: one Battle
}

sig Badge {
AvailableIn : one Tournament
}

sig Notification {
    triggered_by: one (Badge + Tournament + Battle + Submission + Group)}
```

Figure 20: Sigs



```
// FACTS

// Fact to ensure that no User atoms are created
fact {
    all u: User | u in Educator + Student
}
// Fact to ensure that every group includes at least one student
fact {
    all g: Group | some s: Student | g in s.is_part_of
}
// Fact to ensure that a group can only exist if participating in a battle
fact {
    all s: Student, b: Battle |
        lone {g: Group | g in s.is_part_of and g.competing_in = b}
}
// Fact to ensure groups submit submissions only in battles they are participating in
fact {
    all s: Submission | s.submitted_by.competing_in = s.submitted_to
}
// Fact to ensure that students can only recieve badges belonging to tournaments
// where they are participating in atleast one battle
fact {
    all s: Student, b: Badge | b in s.has_badges implies
        some g: Group | g in s.is_part_of and g.competing_in.is_part_of = b.AvailableIn
}
// Fact to ensure start_date and submission_deadline are not the same
fact {
    all b: Battle | b.start_date != b.submission_deadline
    all b: Battle | b.registration_deadline != b.submission_deadline
}
// facto to ensure that each tournament, battle, group, badge and submission
// is associated with at least one notification
fact {
    all t: Tournament | one n: Notification | n.triggered_by = t
    all b: Battle | one n: Notification | n.triggered_by = b
    all g: Group | one n: Notification | n.triggered_by = g
    all bd: Badge | one n: Notification | n.triggered_by = bd
    all s: Submission | one n: Notification | n.triggered_by = s
}
```

Figure 21: Facts



```
run {} for
2 Educator,
2 Tournament,
3 Battle,
4 Student,
3 Group,
2 Badge,
4 Submission,
9 DateTime,
14 Notification
```

Figure 22: Run Command

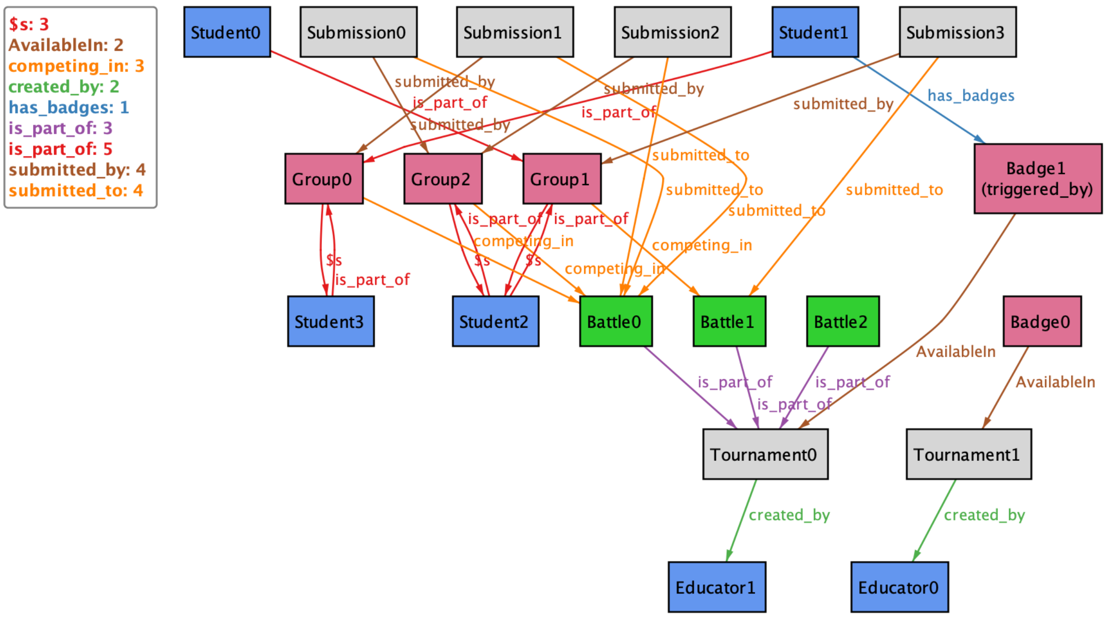


Figure 23: Example

5 Efforts Spent

The tables below indicate how much time each participant has spent on each of the sections in the report.

5.1 Karl Kieler

Section	Time Spent (Hours)
Introduction	2
Overall Description	11
Specific Requirements	16
Formal Analysis	12

5.2 Leonie Dragun

Section	Time Spent (Hours)
Introduction	4
Overall Description	12
Specific Requirements	16
Formal Analysis	12

5.3 Aske Schytt Meineche

Section	Time Spent (Hours)
Introduction	4
Overall Description	9
Specific Requirements	19
Formal Analysis	12

6 References

1. Diagrams made with: StarUML
2. Mockups made with Microsoft PowerPoint
3. Alloy Models made with AlloyTools