POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

RASD Document

Author: **Mattia Piccinato, Gabriele Puglisi, Jacopo Piazzalunga**

Academic Year: 2023-24

Contents

1 | Introduction

1.1. Purpose

The purpose of the project CodeKataBattle (CKB) is to develop a platform where Students can practice coding together. Educators set up challenges (called Battles) within Tournaments, and Students work in teams to solve them. The platform checks their code automatically, giving scores based on how well it works, how quickly they finish, and how good their code quality is. Educators may optionally give extra scores by checking the work themselves, in a so-called Consolidation Stage which starts right after the end of every Battle. Students get ranked in Tournaments based on their scores obtained in teams, and can earn Badges for some achievements, in order to make learning programming more fun and competitive.

1.1.1. Goals

- G1 Allows registered Students who enrolled according to the right modalities to participate in a Tournament of Code Kata Battles and take part in its Battles.
- G2 Allows registered Educators to manage Tournaments for which they have been granted permission.
- G3 Allows registered Students who participate in a Tournament of Code Kata Battles to be rewarded of different achievements.
- G4 Allows registered Users to visualize information for which they have granted permission.
- G5 Automates code evaluation process using GitHub Actions and some static analysis tools.

1.2. Scope

The main features which should be provided in order to achieve the aim of the project are:

- Creating Challenges: Educators can make coding challenges (Battles) that Students can join, alone or in teams (groups), within the context of a Tournament.
- Using GitHub Actions for Code Submsissions: Students are supposed to submit their code for a Battle in their GitHub repository, and the system must be informed of a new commit by a participating Student making use of GitHub Actions.
- Checking Code both Automatically and Manually: The platform assigns a score to Students' code automatically, without the intervention of any Educator. Optionally, Educators may also decide to evaluate the code themselves.
- Rankings: During each Battle, a live ranking of the involved teams is available, enabling participating Students to track their performance. Additionally, live Tournament rankings show how well each Student is performing in the Battles within a given Tournament.
- Badges for Achievements: At the end of every Tournament, Students who achieved good results may be awarded with a special Badges, which are ruled by the Educator who created the Tournament.

The main goal is to help Students practice coding, giving them feedback and comparing their results to others.

1.2.1. World phenomena

WP1 An Educator wants to create a Tournament competition.

WP2 A Student wants to participate in a Tournament competition.

WP3 A Student forks a Battle GitHub repo.

WP4 A Student sets up the update workflow using GitHub Actions.

WP5 An Educator wants to end a Tournament competition.

WP6 A User wants to visualize data about Tournaments, Battles, and Student profiles.

WP7 An Educator is given permission by another Educator to manage a Tournament.

1.2.2. Shared phenomena

Controlled by Machine

SP1 The system notifies the Students about the creation of a Tournament.

SP2 The system notifies the groups about the creation of a Battle.

SP3 The system notifies the groups about the evaluation of a Battle.

SP4 The system notifies the Students about the end of a Tournament.

SP5 The system shows some information about a Battle.

SP6 The system shows some information about a Tournament.

SP7 The system shows some information about a Student.

SP8 The system creates a GitHub repository containing the code kata.

SP9 The system assigns the score to the code of a group.

SP10 The system assigns a Badge to a Student.

Controlled by World

SP11 An Educator creates a Tournament of code kata Battles.

SP12 An Educator defines Badges (see section 1.3) for the Tournament.

SP13 An Educator gives permission to another Educator to manage a Tournament.

SP14 An Educator creates a code kata Battle.

SP15 A Student creates a group for a Battle.

SP16 A Student invites another Student to join the group they created.

SP17 A Student joins a group for a Battle.

SP18 An API call is received by the system when a commit is pushed.

SP19 An Educator closes a Battle.

SP20 An Educator closes a Tournament.

SP21 An Educator evaluates the code of a Student.

1.3. Definitions, Acronyms, Abbreviations

In this section some information about terminology is provided, in order to clarify terms, acronyms, and abbreviations used in the document, ensuring easy understanding and reference for readers.

1.3.1. Definitions

- Battle: A Code Kata, that is, a challenge in which teams of players need to solve a problem in a specific coding language and submit their code to get a score according to the rules of the Battle.
- Tournament: A competition composed of many Battles in which participants' overall score is the sum of all the scores obtained in every Battle they participated in, individually or in team with other players.
- Student: The User which takes part into the Tournaments of Battles.
- Educator: The User which organizes Tournaments of Battles to which the Students can participate and who manages every aspect about them.
- Consolidation stage: The Consolidation Stage is the phase of a Battle which starts as soon as the submission deadline expires, during which the Educators who manage the Tournament can eventually assign an additional score to every team, which will be summed to the score previously assigned by the platform.
- Badge: A Badge is an achievement marker related to a certain Tournament, obtained by Students if they satisfied the specific conditions defined by the Educator during the creation process of the Tournament.

1.3.2. Acronyms

- CK: Code Kata, that is, a Battle.
- CKB: Code Kata Battle, that is, the name of the platform.

1.3.3. Abbreviations

- WPn: n-th World Phenomena
- SPn: n-th Shared Phenomena
- Gn: n-th Goal
- Dn: n-th Domain Assumption
- Rn: n-th Requirement

1.4. Revision History

Revised on	Version	Description
22-Dec-2023	1.0	Initial Release of the document

1.5. Reference Documents

- Assignment document A.Y. 2023/2024
  ("Requirement Engineering and Design Project: goal, schedule and rules")
- Software Engineering 2 A.Y. 2023/2024 Slides

(Lecture slides provided during the course)

1.6. Document Structure

This document is composed of six sections:

- 1st Chapter: We begin by presenting the problem statement and outlining the system's objectives. In the scope subsection, we offer insights into the various real-world and shared phenomena that the system addresses. Finally, we provide essential resources for readers, including definitions and abbreviations, to facilitate a comprehensive understanding of this document.
- 2nd Chapter: We offer a comprehensive overview of the system, including insights into User profiles and their primary functions. Additionally, we present Domain Diagrams to illustrate system components and describe various scenarios. Finally, we establish the key domain assumptions underpinning the system's operation
- 3rd Chapter: We delineate the system's requirements, encompassing both functional and non-functional aspects. In addition, we present Use Case Diagrams that illustrate system interactions, accompanied by detailed descriptions of each use case and related Sequence Diagrams. Lastly, we establish a clear mapping of these requirements to both system goals and use cases for comprehensive understanding.
- 4th Chapter: We provide a formal analysis of the system to be with Alloy.
- 5th Chapter: We provide an estimate of the effort spent by each group member.
- 6th Chapter: We provide a list of the references used in this document.

---

2 | Overall Description

2.1. Product Perspective

CKB is a platform meant to help Students practice coding through CKs. Educators set up these Battles for teams of Students to compete in coding exercises using different programming languages like Java or Python. Each CK has a description and tasks to complete, but Students must write the actual code themselves and make sure it passes certain tests before a deadline.

Educators decide the Battle rules, like team sizes and deadlines. Once the Battles are set, Students form teams and get their own GitHub pages. They need to use GitHub Actions to automate checking their code, which the CKB evaluates.

Scores are given automatically based on how well the code works, how quickly it's finished after signing up, and its quality. Educators might also add scores based on other things. CKB updates the team standings as the Battle goes on and gives final scores when it's done.

Students' scores from different Battles add up to give a total for the whole Tournament. Also, the platform gives out Badges for achievements and participation that everyone can see on a Student's profile.

2.1.1. Scenarios

- 1st Scenario: Signing up and logging in User signs up, logs out and signs in: The User Mark opens the platform and starts the signing up procedure, fills the required data and completes the signign up procedure. Then he logs out to be sure that everything went fine, and a few seconds later he logs in again.
- 2nd Scenario: Creating a Tournament Kevin the Educator creates a Tournament and

sets the subscription deadline. He creates the Badges for the Tournament, specifying their title and rules by using the specific language, referring to existing variables in the system.

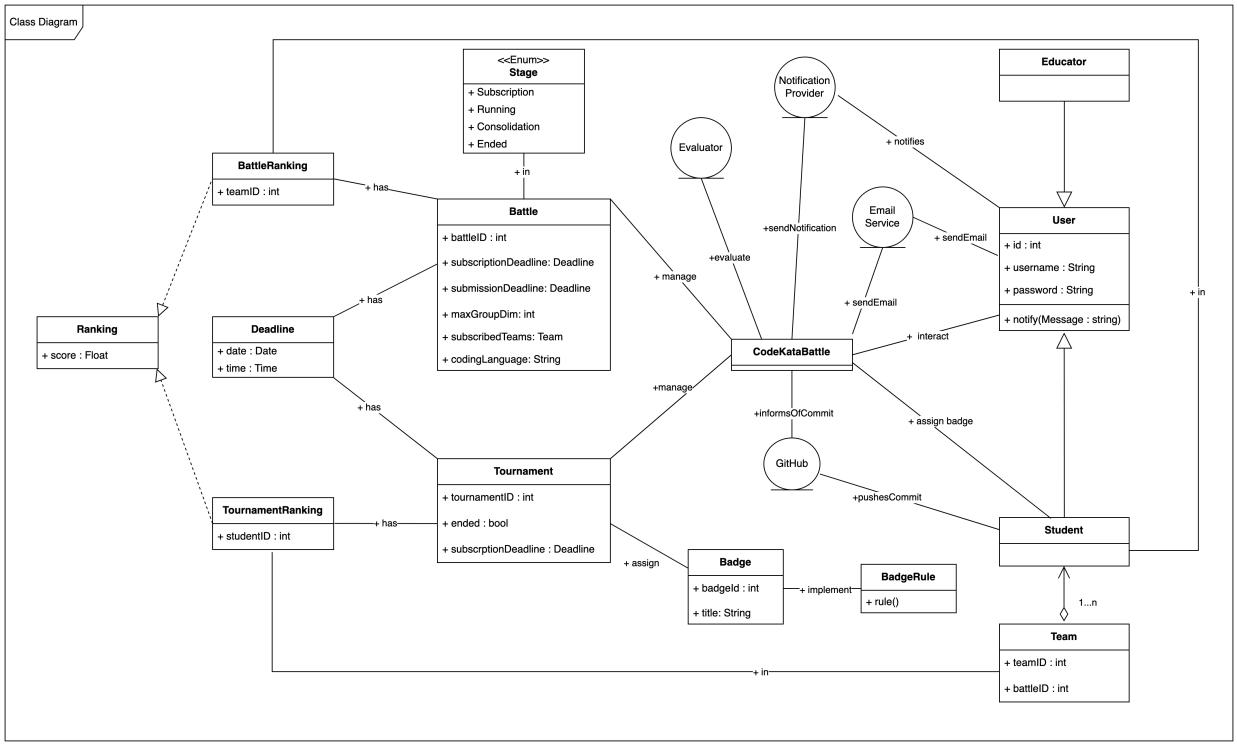
- 3rd Scenario: Giving permission to manage a Tournament Kevin the Educator gives the permission to his colleague Lara to create further Battles within the context of the Tournament.
- 4th Scenario: Creating a Battle in a Tournament Kevin the Educator creates a first Battle for the Tournament and sets registration and submission deadlines, providing a brief textual description, the software project and the set of test cases; then, he sets minimum and maximum number of members per group and specifies the additional mandatory configurations for scoring and whether the manual evaluation is needed. All the Students on the platform are notified.
- 5th Scenario: Joining a Tournament Jim the Student subscribes to a certain Tournament Battle, forming a new group for an existing Battle. Right after, Jim invites Kyle, another Student, which accepts Jim's invite to join his existing group.
- 6th Scenario: Starting Battle The registration deadline of a Battle expires, then the system creates a GitHub repository and sends the link to all the registered Students.
- 7th Scenario: Forking the Battle repository: Jim the Student forks the repository related to the Battle. He then sets up the automated workflow.
- 8th Scenario: Pushing a commit for a Battle Jim the Student pushes a commit (before the expiration of the submission deadline). The system pulls the commit, executes the runnable to test the outcome, computes the mandatory aspects' (functional aspects, timeliness and sources quality) score between 0 and 100 and updates both the Battle score of the group and the Battle ranking if the new commit score is better than the previous best one for the group.
- 9th Scenario: Ending Battle The submission deadline of a Battle expires. The Battle enters the Consolidation Stage. Kevin the Educator assigns the personal optional scores. Then the system assigns the score for every group in the Battle ranking, and the personal score for every Student. The system notifies all the participating Students as soon as the Battle ranking has been updated. The system updates the Tournament ranking adding the single Battle score, for each Student.
- 10th Scenario: Ending Tournament Kevin the Educator ends a Tournament. Any running

Battle is stopped. The system computes the scores for the last commits which were pushed and waiting to be computed and notifies all the Students. The personal scores are updated by the system. Badges are assigned to the Students by the system.

- 11th Scenario: Visualizing content The User Michael visualizes the current Battle ranking before the submission deadline expires. He also visualizes the current list of Tournaments, picks one of them and visualizes its ranking. Finally, he picks the first Student in the Tournament ranking and visualizes their profile, in order to see their Badges.

2.1.2. Class Diagram

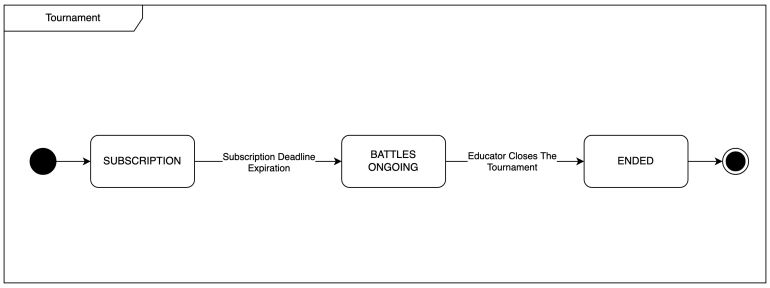
The UML Class Diagram below represents a conceptual, high-level model of the software to be. Given its nature, it may model objects that will not be represented in the actual system that will be developed. Moreover, at this level, it should not include any references to methods and other low-level details that will be detailed during the design phase.



2.1.3. State Charts

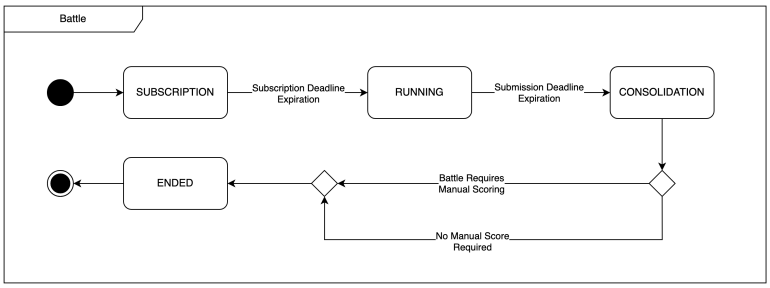
The subsequent section details the principal elements of the CodeKataBattle (CKB) system and their progression through different phases. For this purpose, some UML State Charts are proposed.

Tournament



- The diagram in question outlines the potential phases of a Tournament. Initiated by an Educator, the Tournament promptly enters the 'Subscription' phase, allowing Students to register their participation. Following the closure of the subscription window, the system transitions the Tournament to the 'Ongoing' phase, wherein Battles are created and unfold. Upon the resolution of these contests, the Educator finalizes the Tournament, propelling it into the 'Ended' phase.

Tournament



- The second state diagram offers an overview of the stages a Battle undergoes on the CKB platform. Once the Educator creates a Battle, it is set in "SUBSCRIPTION"

state, where Students can form teams and register for the challenge. Following the end of the subscription deadline, the Battle transitions to the "RUNNING" state. This phase is characterized by active participation as Students develop solutions and submit their code before the submission deadline. After the submission deadline has passed, the platform moves the Battle into the "CONSOLIDATION" phase, which is a critical juncture where the platform assesses the submissions. Depending on the Battle's configuration, there may be a need for manual scoring by Educators. If so, the Educators review the submissions and assign additional scores. Once the manual evaluation is complete, or if no manual scoring is necessary, the Battle proceeds to the "ENDED" state, signaling its conclusion and the finalization of results and rankings.

2.2. Product Functions

The platform CKB offers several key functions:

- Battles and Tournaments Creation Educators can create coding challenges (Battles) within Tournaments, specifying details like descriptions, team sizes, deadlines, scoring configurations and, eventually, Badges.
- Student Participation On the other hand, Students can join such Tournaments and take part in Battles, individually or in teams.
- GitHub Integration The system allows the Students to perform code submissions just by pushing their code on GitHub, thanks to automated workflows triggered by GitHub Actions service.
- Automated Evaluation The platform automatically evaluates Student code based on test cases, timeliness, and code quality using external static analysis tools.
- Scoring and Ranking It continuously updates team rankings during Battles and provides overall Tournament rankings at the end of each Battle.
- Badges and Recognition Educators define Badges which can be obtained by Students, serving as recognition for accomplishments and participation to a certain Tournament.
- Manual Optional Evaluation Educators can manually evaluate Students' work and assign additional scores at the end of every Battle.

2.2.1. Requirements

R1 The system must allow an unregistered Educator to sign up.

R2 The system must allow an unregistered Student to sign up.

R3 The system must allow a registered User to log in.

R4 The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.

R5 The system must provide registered Educators of a list of Tournament-related statistics for the Badges definition, during the Tournament creation process.

R6 The system must provide registered Educators of a specific language which lets them define the Badges, during the Tournament creation process.

R7 The system must allow registered Educators to grant other registered Educators the permission to manage the Tournament, during the Tournament creation process.

R8 The system must allow registered Educators to end the creation process of a Tournament that they started themselves.

R9 The system must be able to send notifications to every registered User.

R10 The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.

R11 The system must allow a registered Student to create a group for a Battle in a Tournament.

R12 The system must allow a registered Student to accept an invitation to a group for a Battle in a Tournament.

R13 The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline is not expired yet.

R14 The system must allow registered Students who are enrolled in a Battle to perform code submissions.

R15 The system must be provided of proper APIs to let registered Students perform code submissions through GitHub Actions.

R16 The system must update the Battle ranking when a valid code submission is performed.

R17 The system must set the Consolidation Stage of a Battle when its submission deadline expires.

R18 The system must let Educators to end the Consolidation Stage of a Battle if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.

R19 The system must update Tournament ranking when a Battle exits the Consolidation Stage.

R20 The system must let Educators who are either the creator of the Tournament or who have been granted the permission to by the latter to close a Tournament if and only if there is no Battle such that either their subscription or submission deadline is not expired yet or such that they are still in the Consolidation Stage.

R21 The system must assign an achievements' Badge for a given Tournament to any Student who satisfied the conditions defined by the creator of the Tournament.

R22 The system must allow every User to see the Badges which were ever obtained by a given Student.

R23 The system must notify every registered Student about the creation of a new Tournament.

R24 The system must notify every registered Student about the creation of a new Battle within a Tournament they are enrolled in.

R25 The system must notify every registered Student about the end of a Battle they are participating in.

R26 The system must notify every registered Student about the end of a Tournament they are enrolled in.

2.3. User Characteristics

In the CKB platform, a User can be either a Student or an Educator, each with distinct roles and motivations.

- Students They join Battles to practice coding in a collaborative environment, aiming to enhance their programming skills through practical challenges, seeking to measure their performance against peers and track their progress. Students can achieve recognition by obtaining Badges within the context of a Tournament. Students are often notified about new activities that they may want to join, such as a new Tournament, or a new Battle within the context of a Tournament that they are enrolled in.
- Educators They set up and manage the coding challenges, individually or cooperating with other Educators. They act as facilitators, creating Battles within Tournaments to engage Students in practical exercises. Educators may want to personally evaluate Students' code submissions, in order to reward particularly brilliant solutions or to penalize major mistakes. Additionally, they are responsible for defining Badge criteria, which act as a motivating factor.

2.4. Assumptions, Dependencies, Constraints

This section serves as a comprehensive overview of critical factors which must be considered during the implementation of the platform. It consolidates the foundational assumptions made during project planning and highlights eventual dependencies.

2.4.1. Domain Assumptions

t

D1 The User must have a working Internet connection.

D2 The Students who participate to a Battle properly set up the GitHub Action for code submissions.

D3 The Users always receive every notification which is sent by the system.

D4 The availability of the static analysis tools utilized for the code evaluation process is consistent.

D5 The availability of GitHub Actions service is consistent.

2.4.2. Dependencies

The static analysis tools which provide a score based on the quality level of each code submission can either be implemented or integrated in the system as Evaluator. Similarly, the NotificationProvider can either be implemented or integrated in the system. For the registration process, a verification email must be sent by the system through to let Users successfully sign up, thus requiring the integration of an EmailService.

3 Specific requirements

3.1. External Interface Requirements

The system provides all the main functions described previously (see section 2.2) and lets Users access all the information they are granted permission for. Given such purposes, any personal computer is a suitable device to make use of all the CKB functionalities, allowing convenient acces through any web browser.

3.1.1. User Interfaces

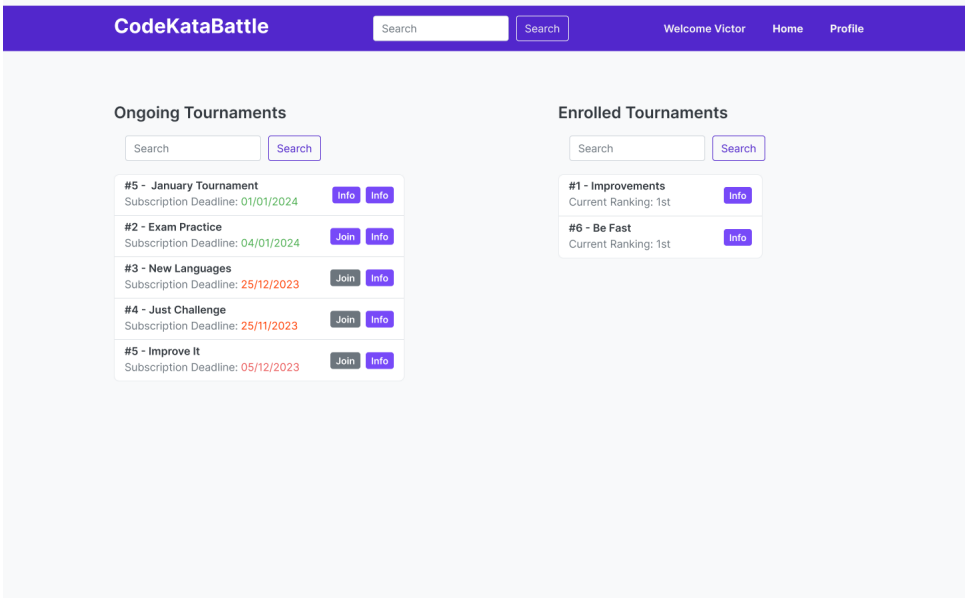


Figure 3.1: Home Page (Student)

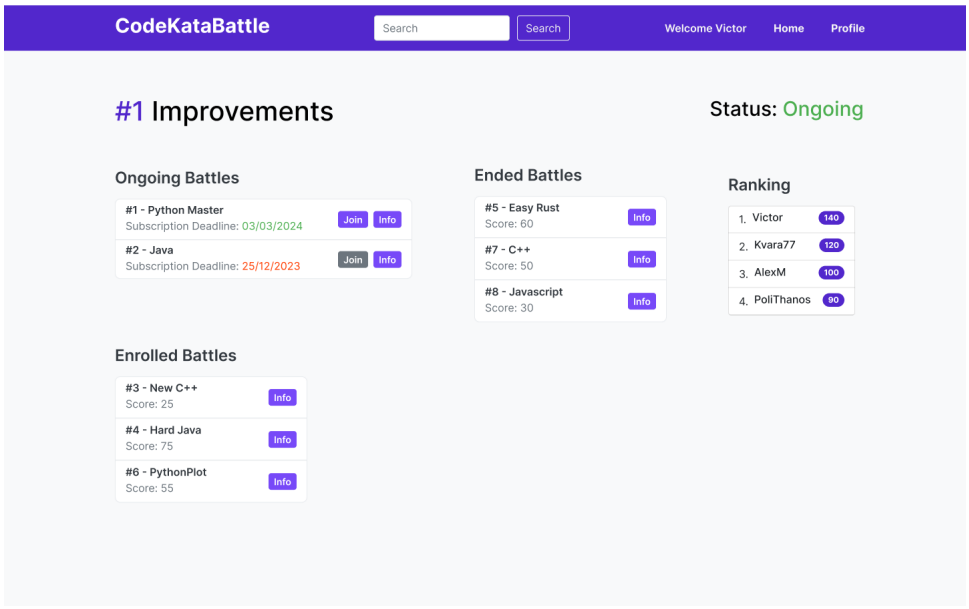


Figure 3.2: Tournament Page (Student)

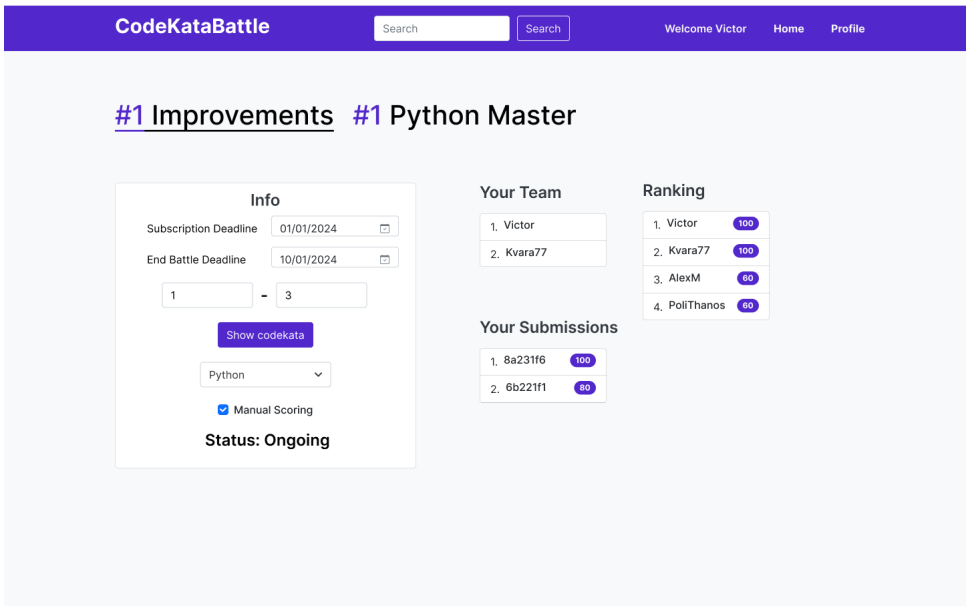


Figure 3.3: Battle Page (Student)

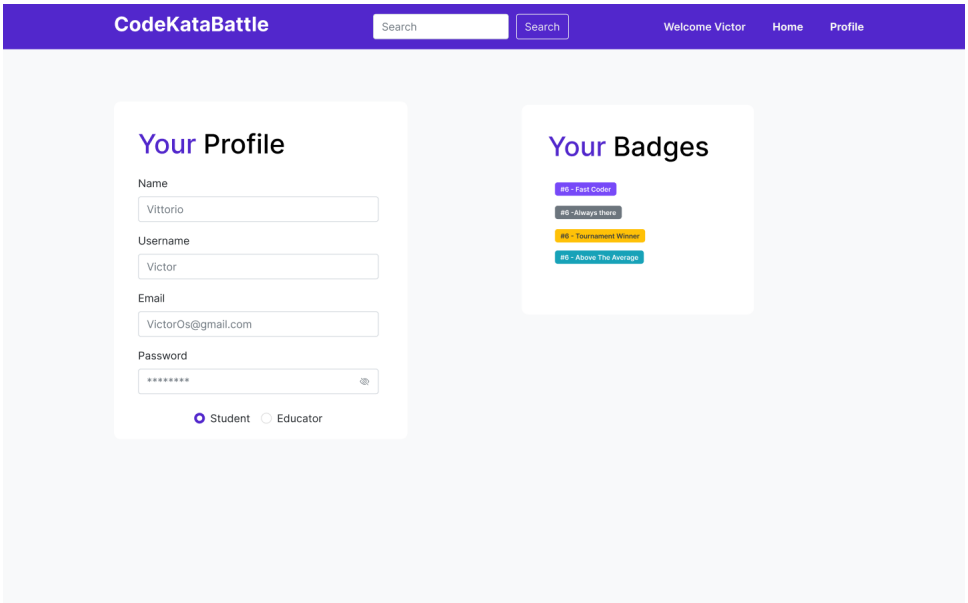


Figure 3.4: Profile Page (Student)

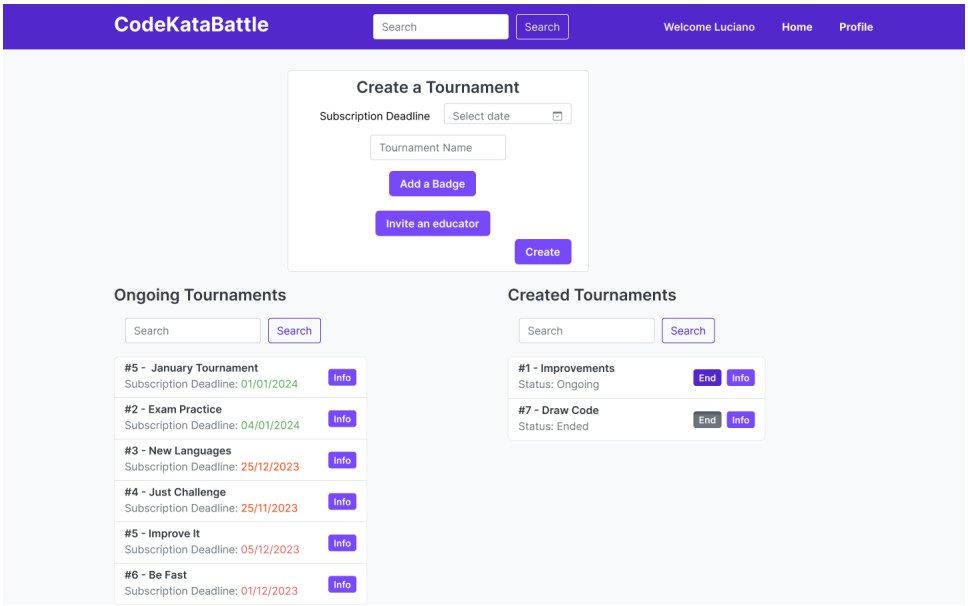


Figure 3.5: Home Page (Educator)

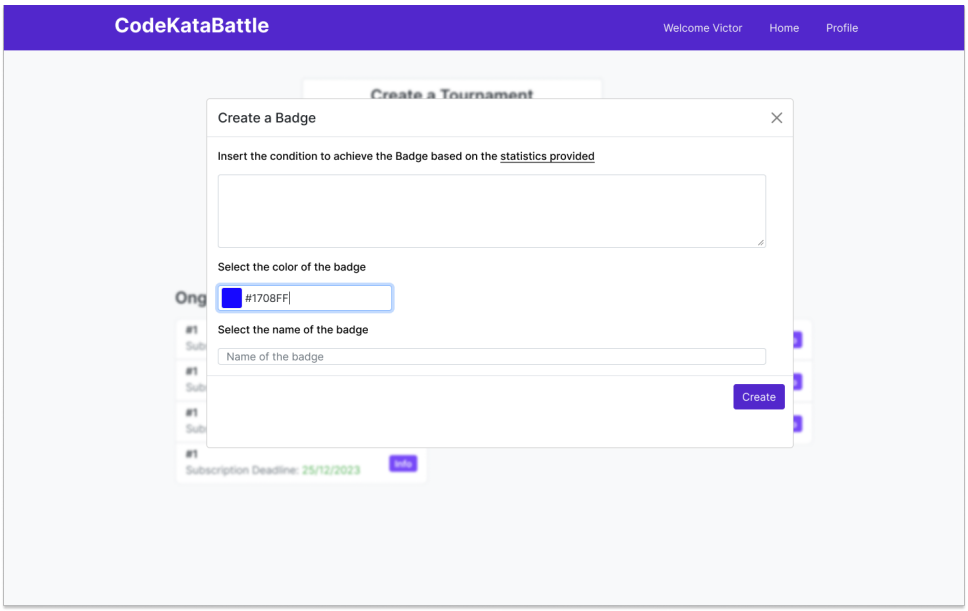


Figure 3.6: Badge definition for a tournament (Educator)

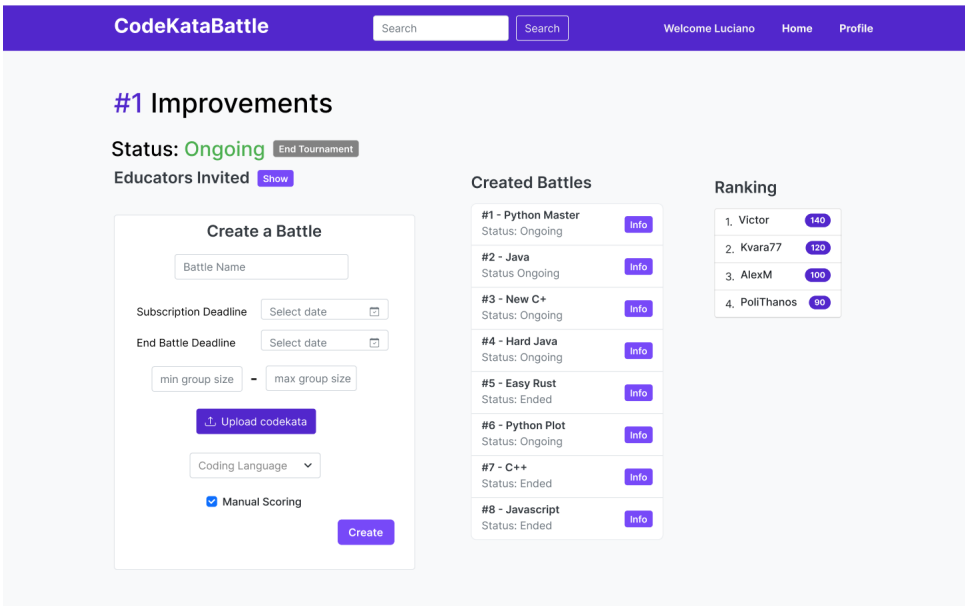


Figure 3.7: Tournament Page where the educator is the creator of the tournament (Educator)

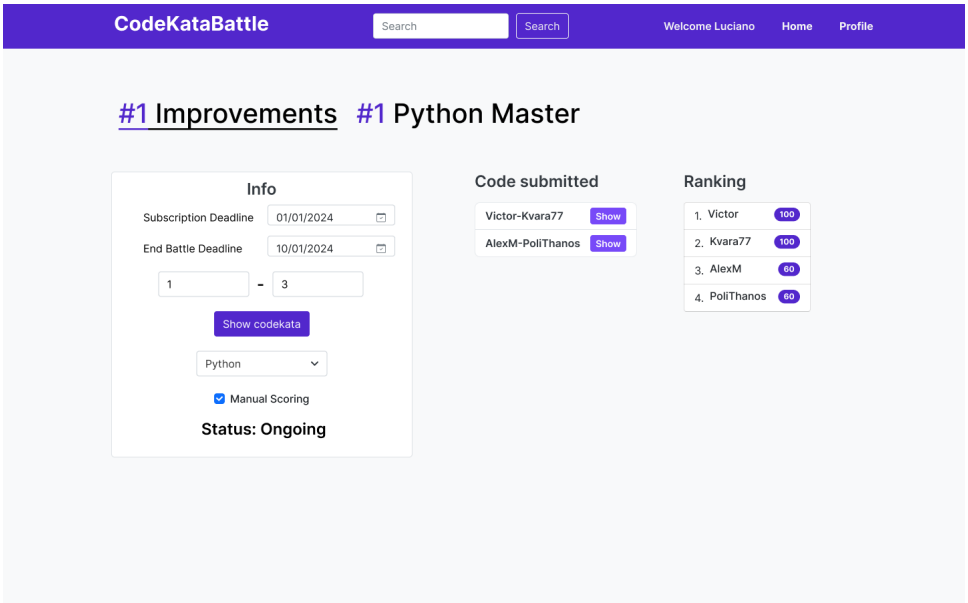


Figure 3.8: Battle Page where the educator is the creator of the tournament (Educator)

3.1.2. Hardware Interfaces

To use the system, both Educators and Students must use a suitable device. Due to the communication capabilities needed, any personal computer results once again being a suitable device for the User purpose.

3.1.3. Software Interfaces

The system should integrate a NotificationService to keep up Users of any interesting event on the platform, an EmailService for the registration process, and proper static analysis tools as an Evaluator of the Students' code submissions.

3.1.4. Communication Interfaces

The system requires a stable internet connection to work properly. This connection is used to exchange data between the Users and the central database which contains the information regarding ongoing Tournaments and Battles.

3.2. Functional Requirements

In this section, all the Use Cases are listed attached to their corresponding Use Case Diagram. Then, the mapping between Goals, Domain Assumptions and Requirements is provided.

3.2.1. Use Case Diagrams

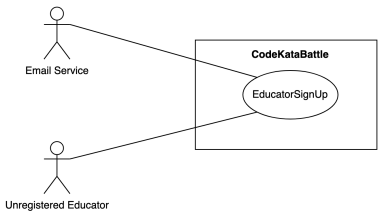


Figure 3.9: EducatorSignUp

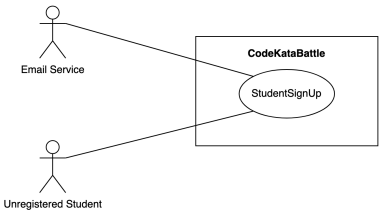


Figure 3.10: StudentSignUp

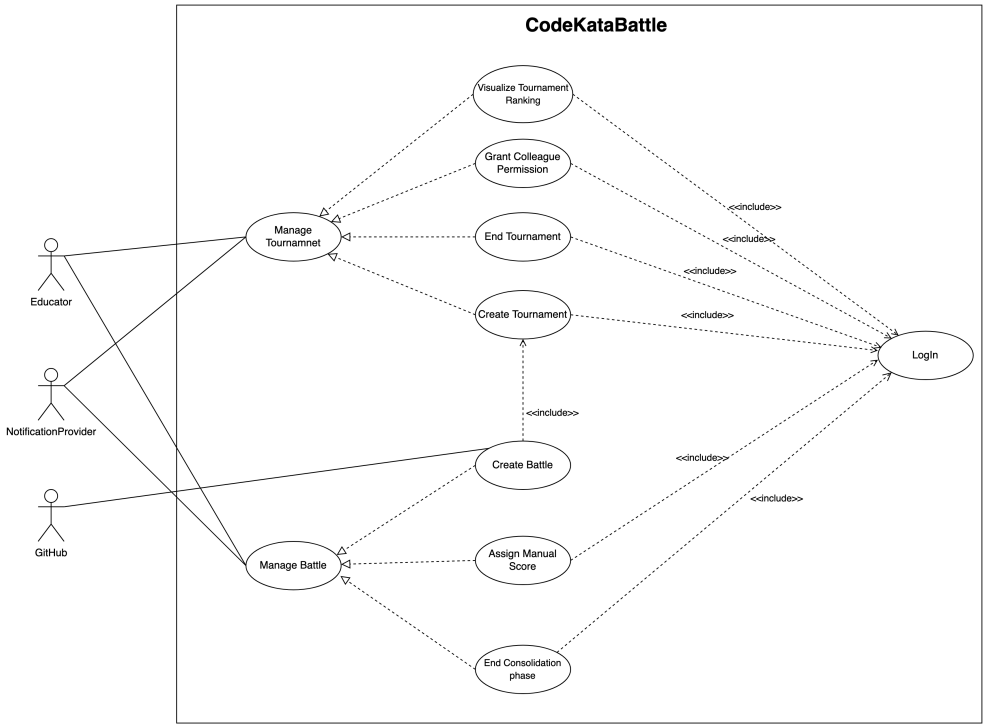


Figure 3.11: Educator

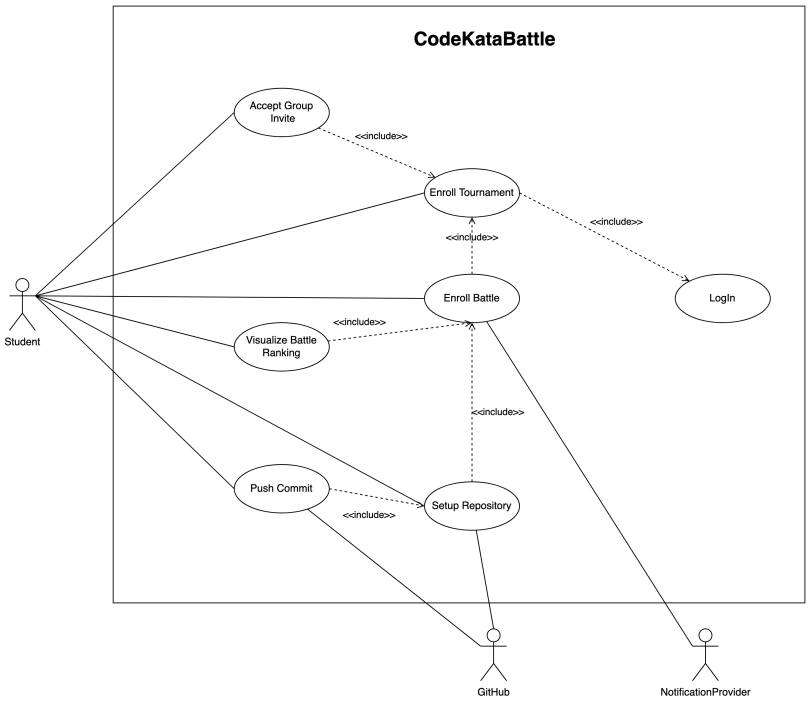


Figure 3.12: Student

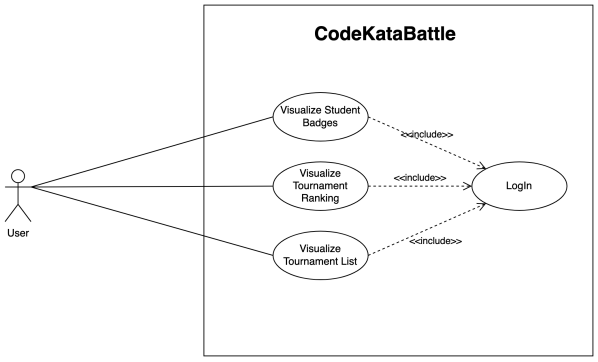


Figure 3.13: User

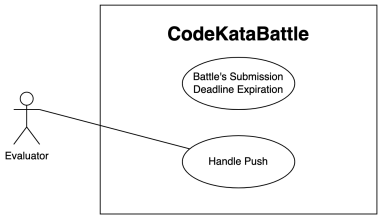
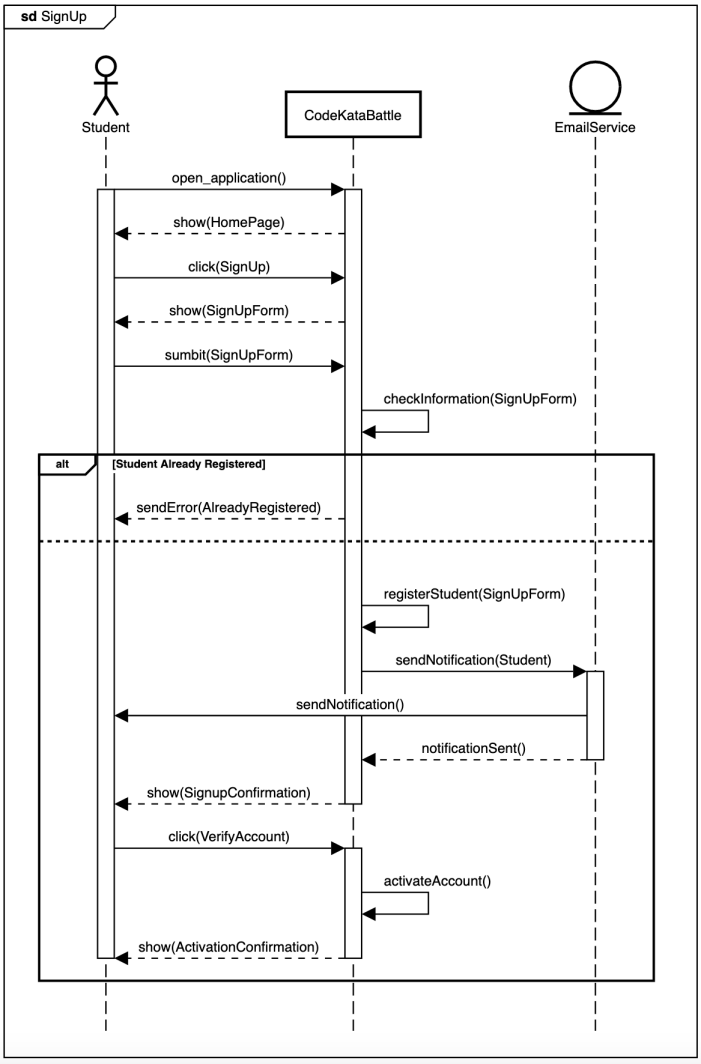


Figure 3.14: Evaluator

3.2.2. Use Cases

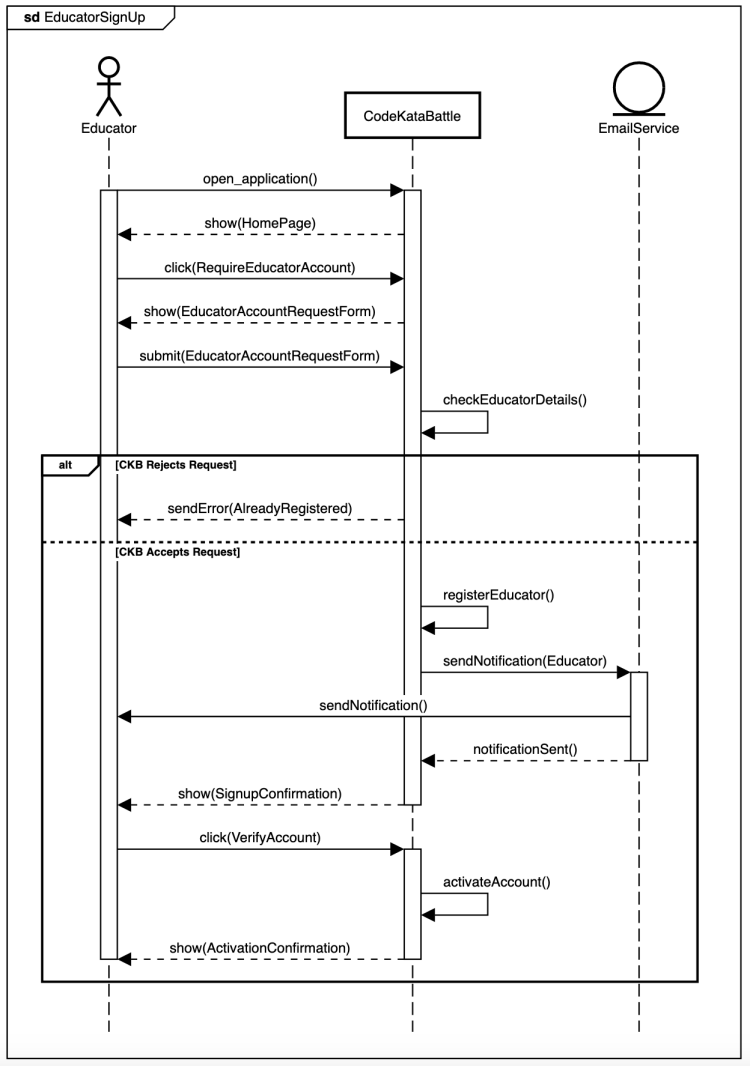
StudentsSignUp	
Participating Actors	Student, EmailService, CodeKataBattle
Entry Condition	True

Flow of Events	• 1. The Student opens the “Sign up” page • 2. CodeKataBattle shows the page to sign up • 3. The Student fills the required informations and clicks “Sign Up” button • 4. CodeKataBattle checks the information provided by the Student • 5. CodeKataBattle registers the Student • 6. CodeKataBattle sends confirmation of signup and inform the Student to verify the account • 7. CodeKataBattle sends a notification to verify the account through the EmailService • 8. The Student verifies the account • 9. CodeKataBattle activates the Student’s account • 10. CodeKataBattle sends to the home page
Exit Condition	The Student successfully signed up and activated his account
Exceptions	The Student was already registered



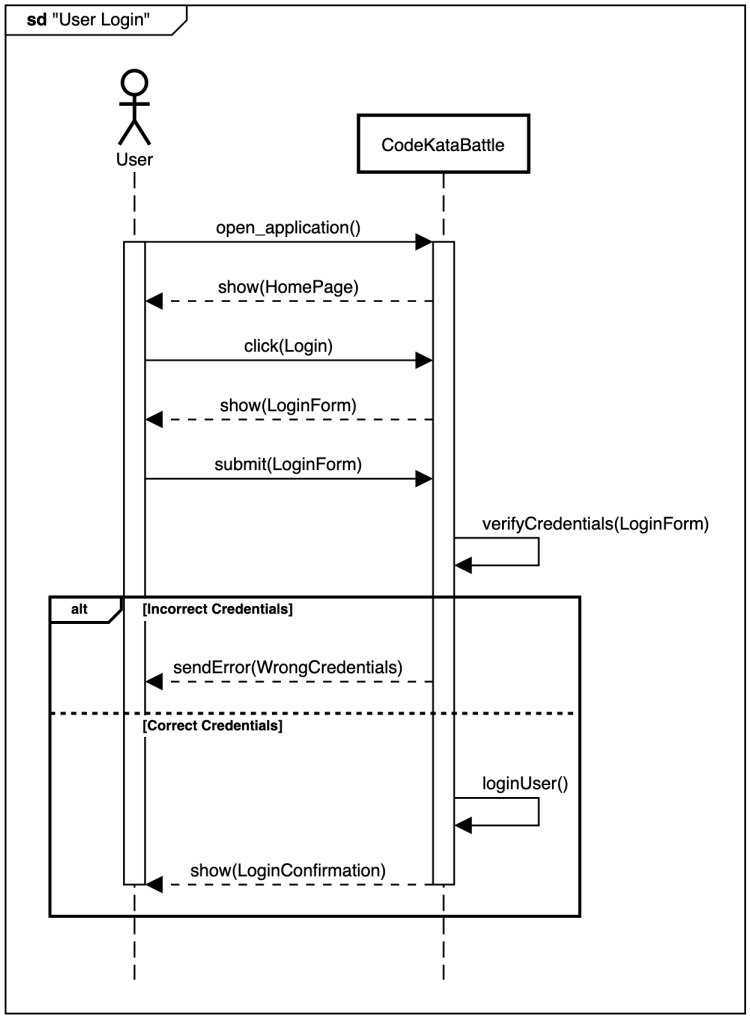
EducatorsSignUp	
Participating Actors	Educator, EmailService, CodeKataBattle
Entry Condition	True

Flow of Events	• 1. The Educator opens the “Sign up” page • 2. CodeKataBattle shows the page to sign up • 3. The Educator fills the required informations and clicks “Require An Educator Account” button • 4. CodeKataBattle shows the form to require an Educator account • 5. CodeKataBattle registers the Student • 6. CodeKataBattle sends confirmation of signup and inform the Educator to verify the account • 7. CodeKataBattle sends a notification to verify the account through the EmailService • 8. The Educator verifies the account • 9. CodeKataBattle activates the Educator’s account • 10. CodeKataBattle sends to the home page
Exit Condition	The Educator successfully signed up and obtained an Educator account
Exceptions	The Educator was already registered



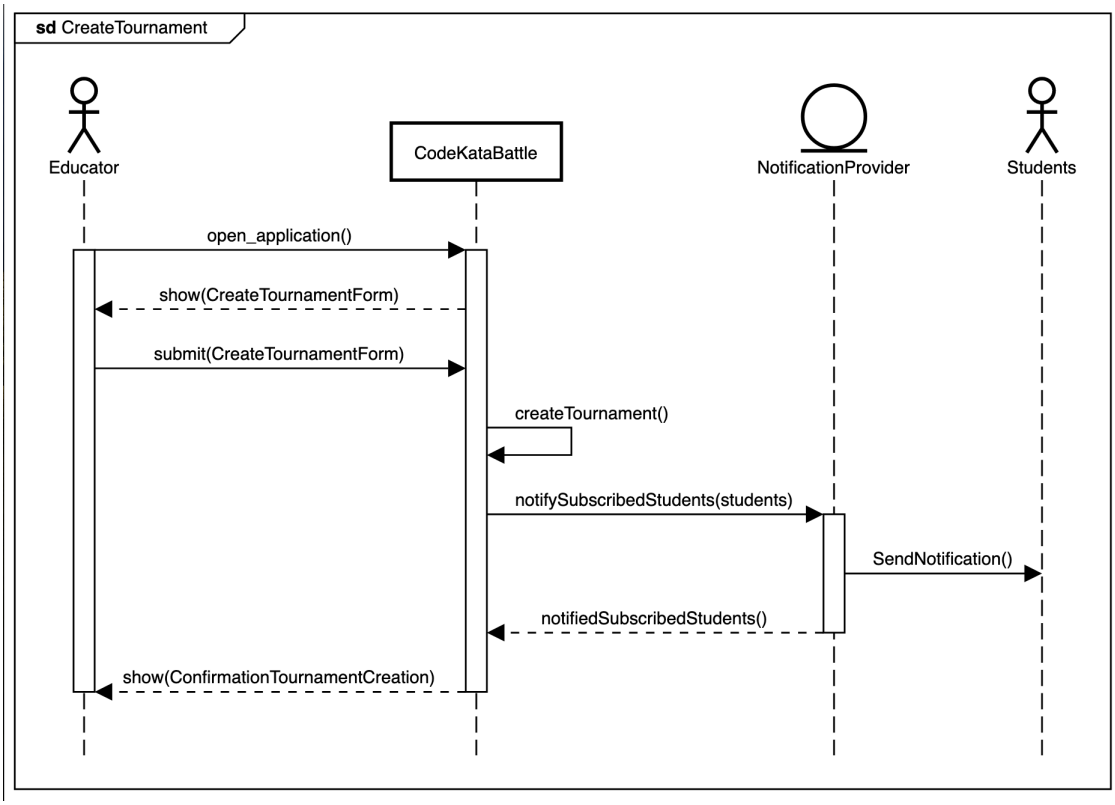
UserLogsIn	
Participating Actors	User, CodeKataBattle
Entry Condition	User has an account

Flow of Events	• 1. The User opens CodeKataBattle • 2. The User clicks the “Log in” button • 3. CodeKataBattle shows the form to log in • 4. The User fills the required informations and clicks “Log in ” button • 5. CodeKataBattle verifies the credentials of the User • 6. CodeKataBattle logs in the User
Exit Condition	The User successfully logged in
Exceptions	The User filled wrong credentials



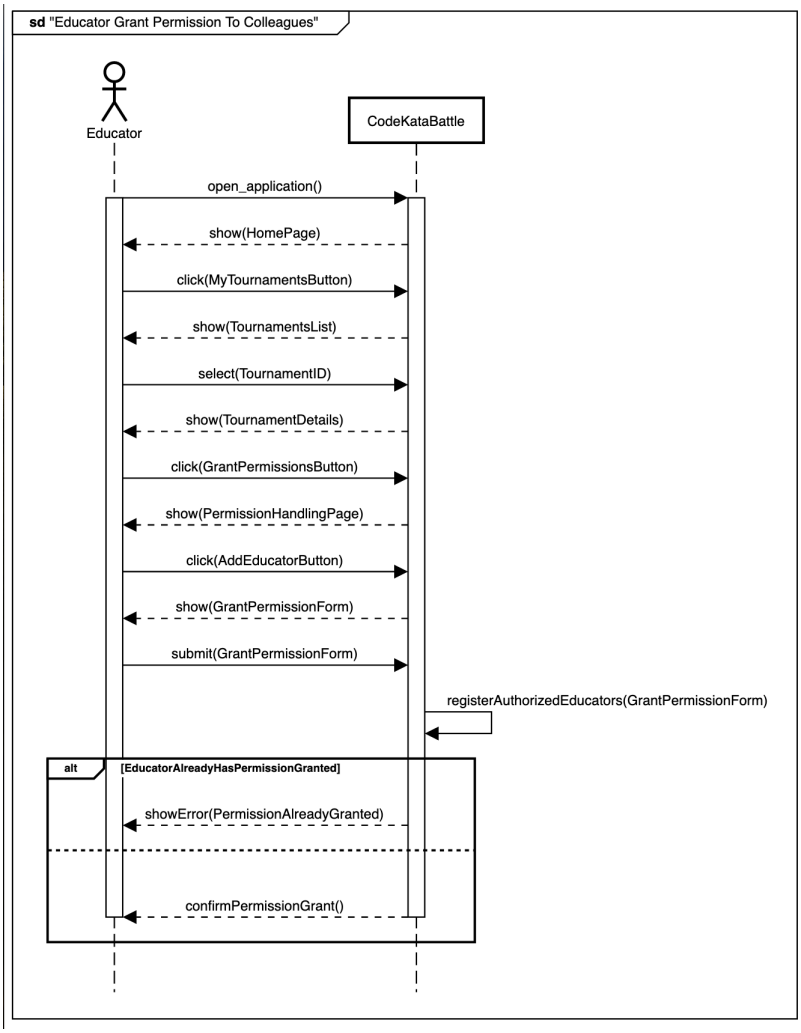
CreateTournament	
Participating Actors	Student, Educator, NotificationProvider, CodeKataBattle
Entry Condition	Educator is logged in

Flow of Events	• 1. The Educator clicks on the “Create a new Tournament” button • 2. CodeKataBattle shows the form to create a new Tournament • 3. The Educator chooses a subscription deadline and fills the correct field • 4. The Educator chooses whether to use Badges and eventually fills the relative fields • 5. The Educator chooses to which other colleagues grant the permission to manage the Tournament • 6. The Educator clicks the “Create” button • 7. CodeKataBattle creates the Tournament • 8. CodeKataBattle notifies through the NotificationProvider all the Students subscribed to the platform of the creation of the Tournament
Exit Condition	The Educator successfully created a new Tournament and CodeKataBattle successfully notifies all the subscribed Students
Exceptions	None



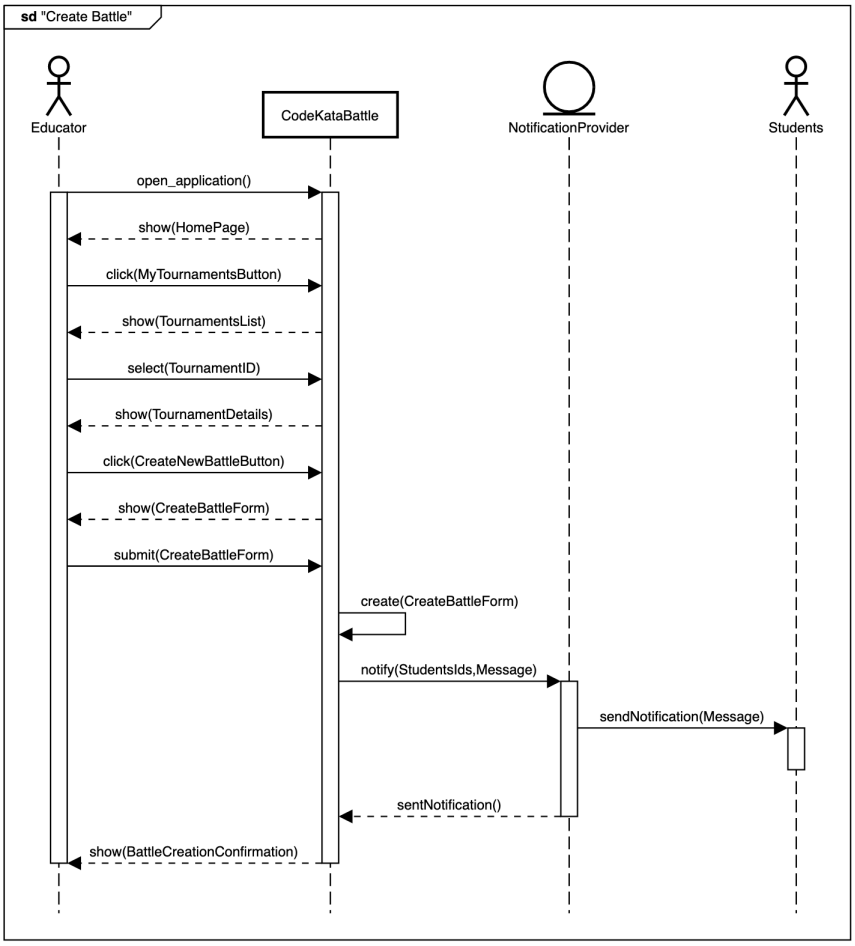
EducatorGrantsPermissionToColleagues	
Participating Actors	Educator, CodeKataBattle
Entry Condition	The Educator is creating a Tournament
Flow of Events	• 1. CodeKataBattle shows the Tournament's details • 2. The Educator clicks on the "Grant Permissions" button • 3. CodeKataBattle shows the page to handle permissions for the Tournament • 4. The Educator clicks on the "Add Educator" button • 5. CodeKataBattle shows a form to grant other Educators permissions on the Tournament • 6. The Educator fills the requested information and clicks "Confirm" button • 7. CodeKataBattle recognizes the newly authorized Students

Exit Condition	The Educator successfully granted permission to manage the Tournament to another Educator
Exceptions	None



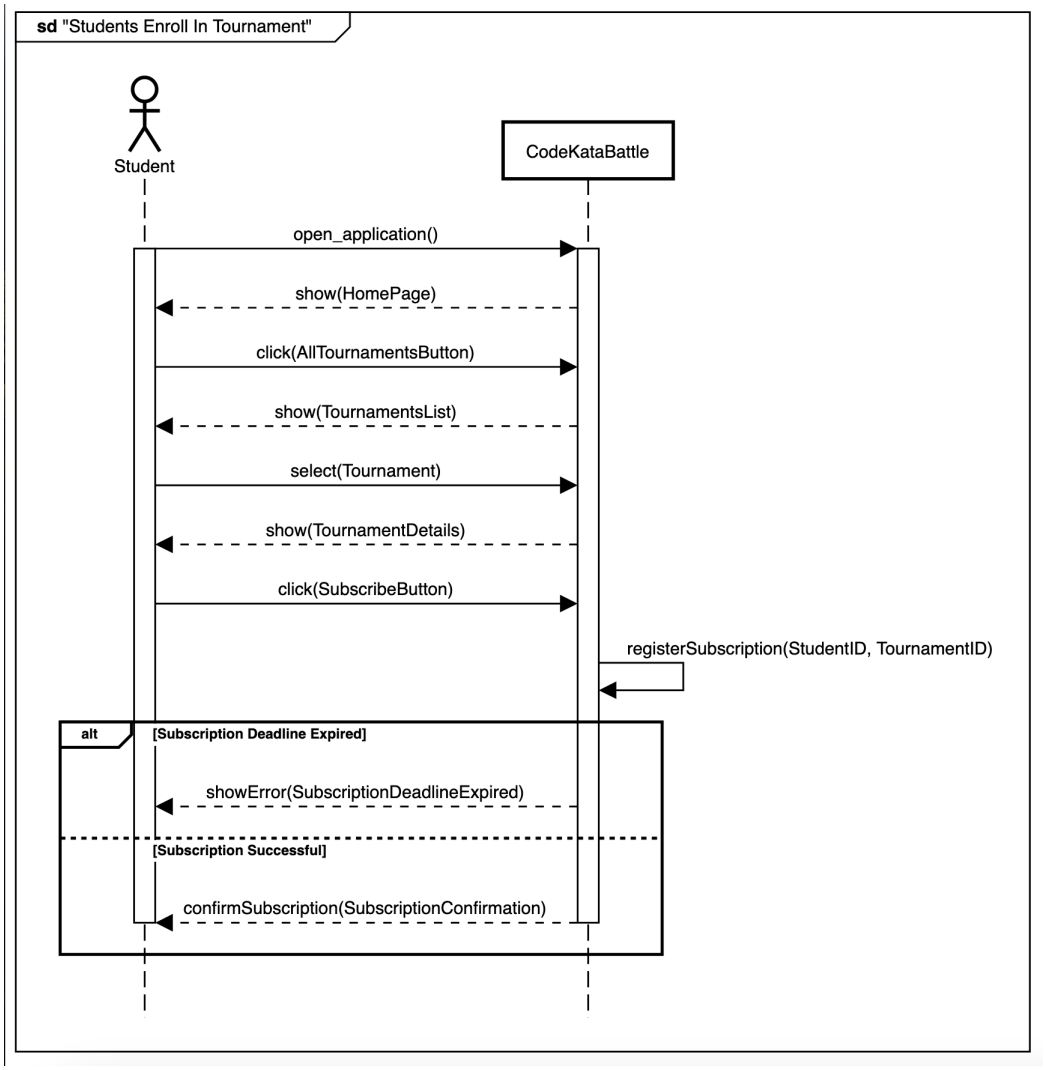
CreateBattle	
Participating Actors	Student, Educator, NotificationProvider, CodeKataBattle
Entry Condition	The Educator is logged in and is granted the permission to manage a Tournament

Flow of Events	• 1. The Educator clicks on the "My Tournaments" button • 2. CodeKataBattle shows the list of the Tournaments that the Educator is allowed to manage • 3. The Educator clicks on a Tournament • 4. CodeKataBattle shows the Tournament's details • 5. The Educator clicks on "Create a new Battle" button • 6. CodeKataBattle shows the form to create a new Battle • 7. The Educator uploads the problem • 8. The Educator uploads the CodeKata • 9. The Educator fills the minimum and maximum number of Students per team • 10. The Educator fills the minimum and maximum number of Students per team • 11. The Educator chooses a registration deadline • 12. The Educator chooses a submission deadline • 13. The Educator chooses the scoring configuration • 14. The Educator clicks the "Create" button • 15. CodeKataBattle creates the Battle • 16. CodeKataBattle notifies through the NotificationProvider all the Students subscribed to the Tournament of the creation of the Battle
Exit Condition	The Educator successfully created a new Battle in a Tournament and CodeKataBattle successfully notifies all the Students subscribed to the Tournament
Exceptions	None



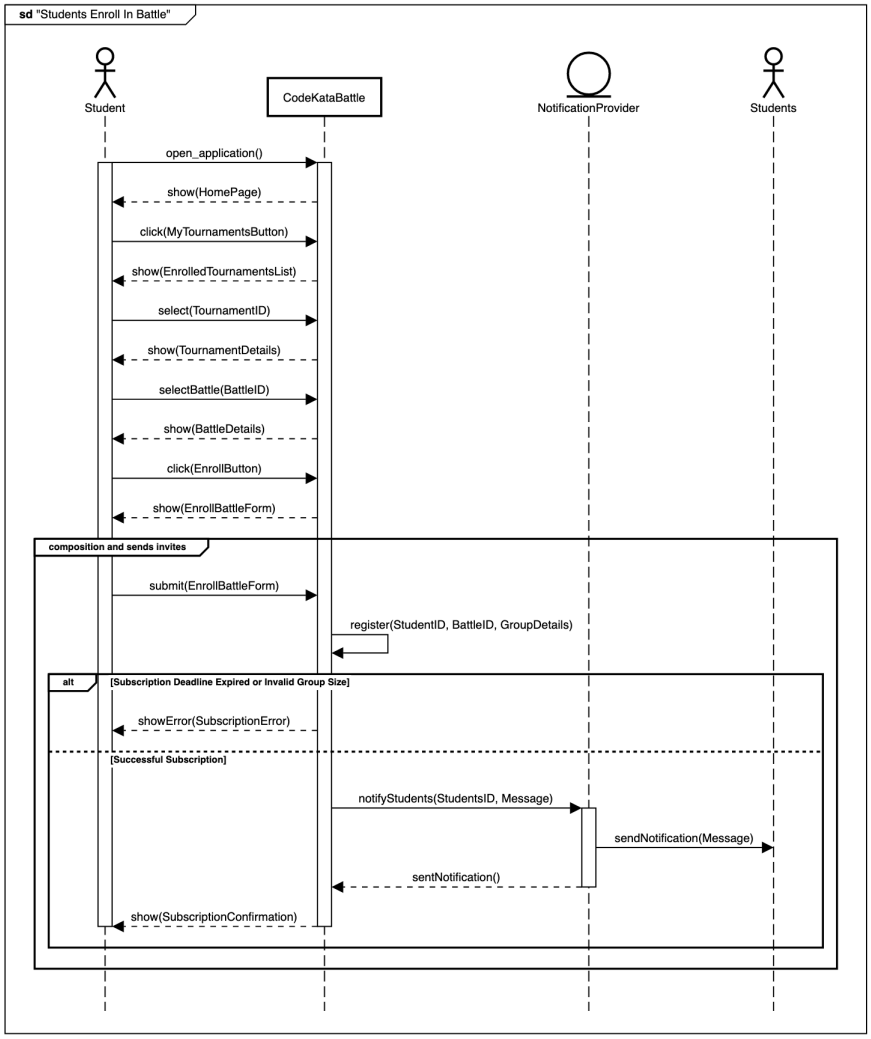
StudentEnrollsInTournament	
Participating Actors	Student, CodeKataBattle
Entry Condition	The Student is logged in and at least a Tournament has been created
Flow of Events	• 1. The Student clicks on the "All Tournaments" button • 2. CodeKataBattle shows the list of all the Tournaments • 3. The Student clicks on a Tournament • 4. CodeKataBattle shows the Tournament's details • 5. The Student clicks on "Subscribe" button • 6. CodeKataBattle performs the Student's subscription

Exit Condition	The Student successfully subscribed to the Tournament
Exceptions	The subscription deadline for the chosen Tournament is expired



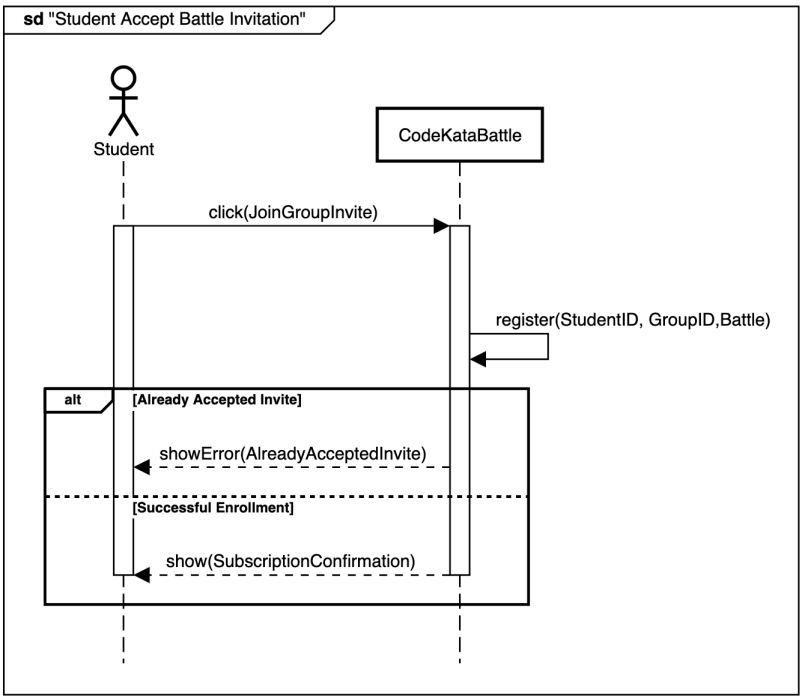
StudentEnrollsInBattle	
Participating Actors	Student, CodeKataBattle
Entry Condition	Student is logged in and subscribed to the Battle's Tournament

Flow of Events	• 1. The Student clicks on the "My Tournaments" button • 2. CodeKataBattle shows the list of the Tournaments the Student is enrolled in • 3. The Student clicks on a Tournament • 4. CodeKataBattle shows the Tournament's details • 5. The Student clicks on the Battle he wants to enroll in • 6. CodeKataBattle shows the Battle's details • 7. The Student clicks on the "Subscribe" button • 8. CodeKataBattle shows the form to specify the group composition • 9. The Student makes his choice and invites other Students • 10. CodeKataBattle performs the Student's subscription
Exit Condition	The Student successfully subscribed to the Battle and eventually invites other Students
Exceptions	• The subscription deadline for the chosen Battle is expired • The number of Students in the group is invalid • Some Students who are being invited are not enrolled in the Tournament



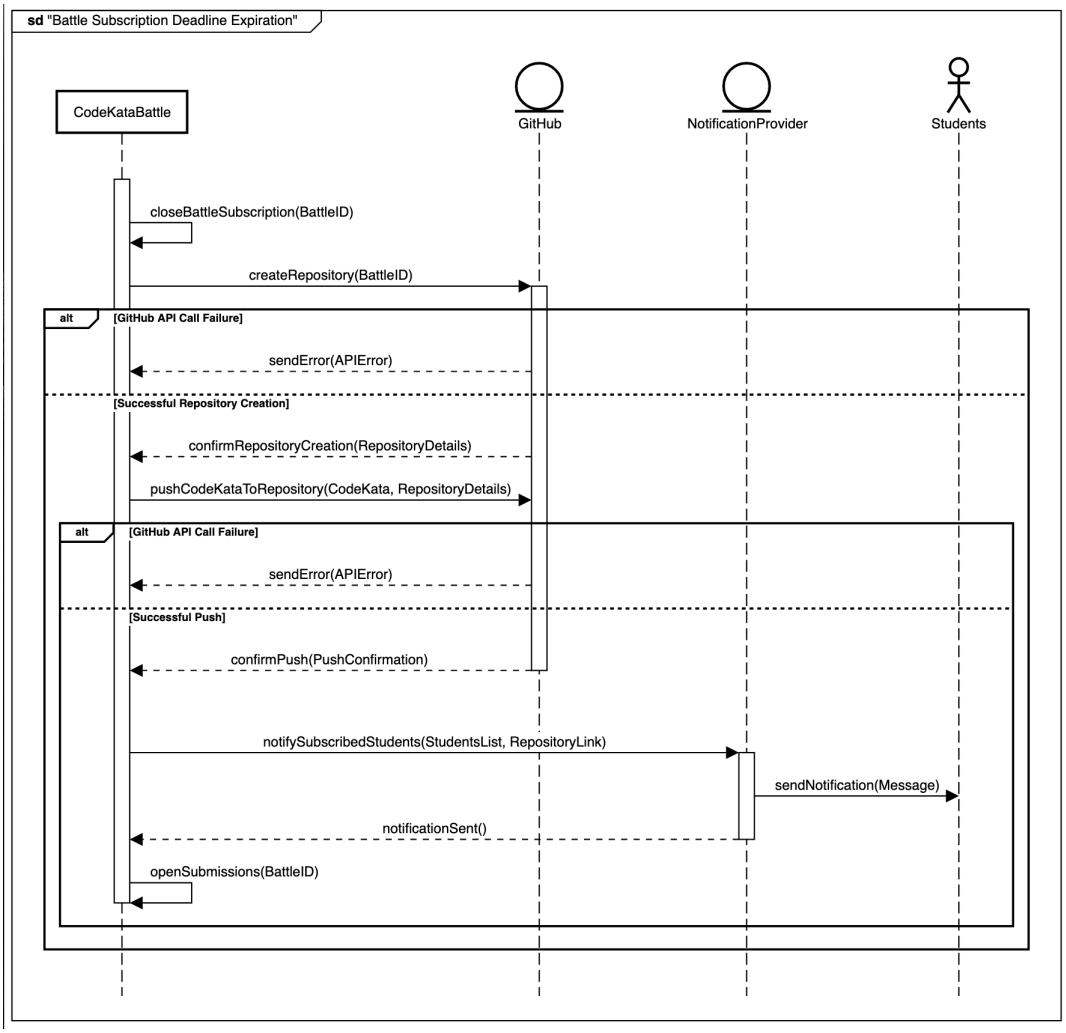
StudentAcceptsBattleInvitation	
Participating Actors	Student, CodeKataBattle
Entry Condition	Student received an invite to join a group for a Battle
Flow of Events	• 1. The Student clicks on the "Join Group" button of the received invite • 2. CodeKataBattle registers the Student's subscription to the Battle and the enrollment in the group

Exit Condition	The Student successfully subscribed to the Battle and joined the team of the Student who invited him
Exceptions	The Students had already accepted the invite



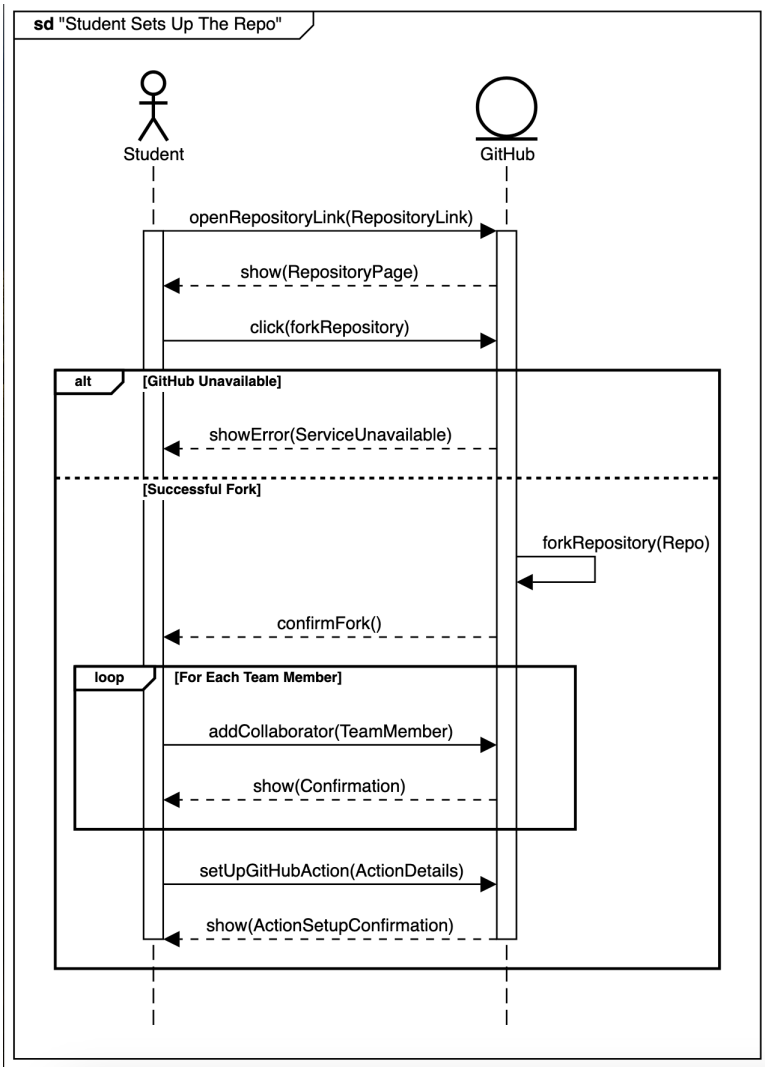
BattlesSubscriptionDeadlineExpiration	
Participating Actors	GitHub, CodeKataBattle
Entry Condition	A Battle's subscription deadline has reached its expiration point

Flow of Events	• 1. CodeKataBattle closes Battle's subscription stage • 2. CodeKataBattle creates the Battle's GitHub repository • 3. CodeKataBattle pushes the CodeKata to the newly created repository • 4. CodeKataBattle shares with them the link to the GitHub repository • 5. CodeKataBattle starts registering submissions for the subscribed members
Exit Condition	The Battle's subscription is closed, preventing further participation after the deadline and the GitHub repository associated with the Battle has been created
Exceptions	The GitHub API call fails



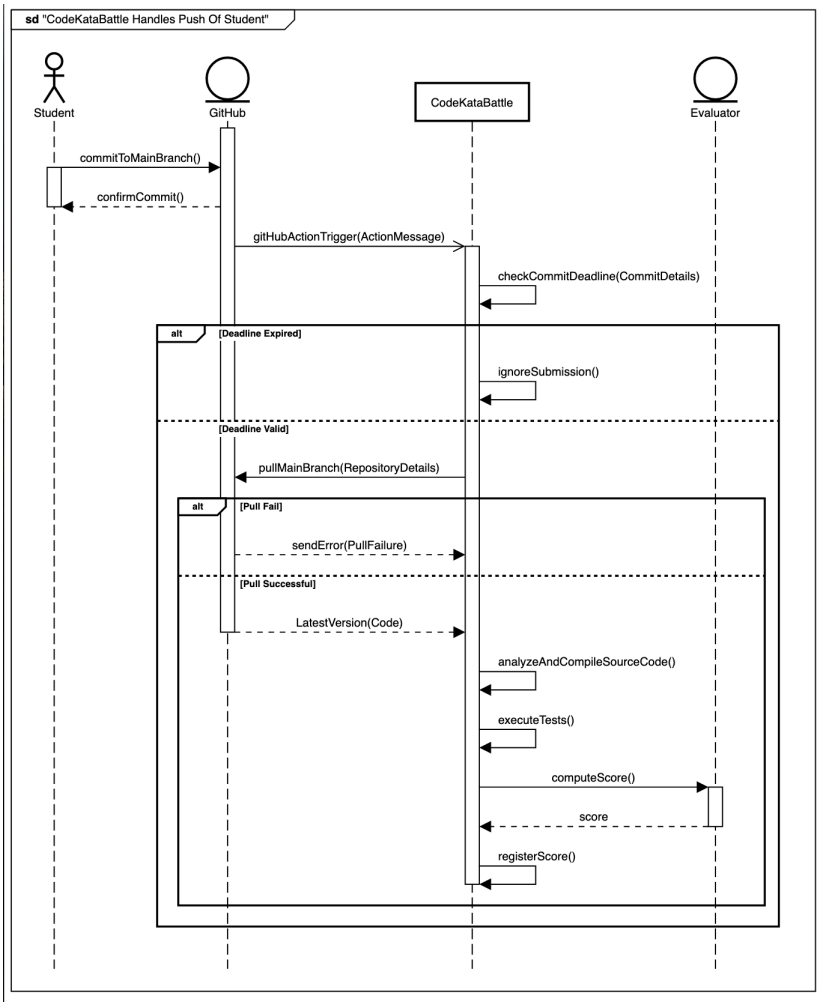
StudentSetsUpTheRepository	
Participating Actors	GitHub, Student
Entry Condition	The Student is enrolled in the Battle and has a GitHub account
Flow of Events	• 1. The Student opens the link to the repository • 2. The Student forks the repository • 3. The Student adds his team members as collaborators in the repository • 4. The Student sets up the “GitHub Action”

Exit Condition	The Student successfully forked the Battle's GitHub repository and set up the GitHub Action
Exceptions	The GitHub website is temporarly unavailable



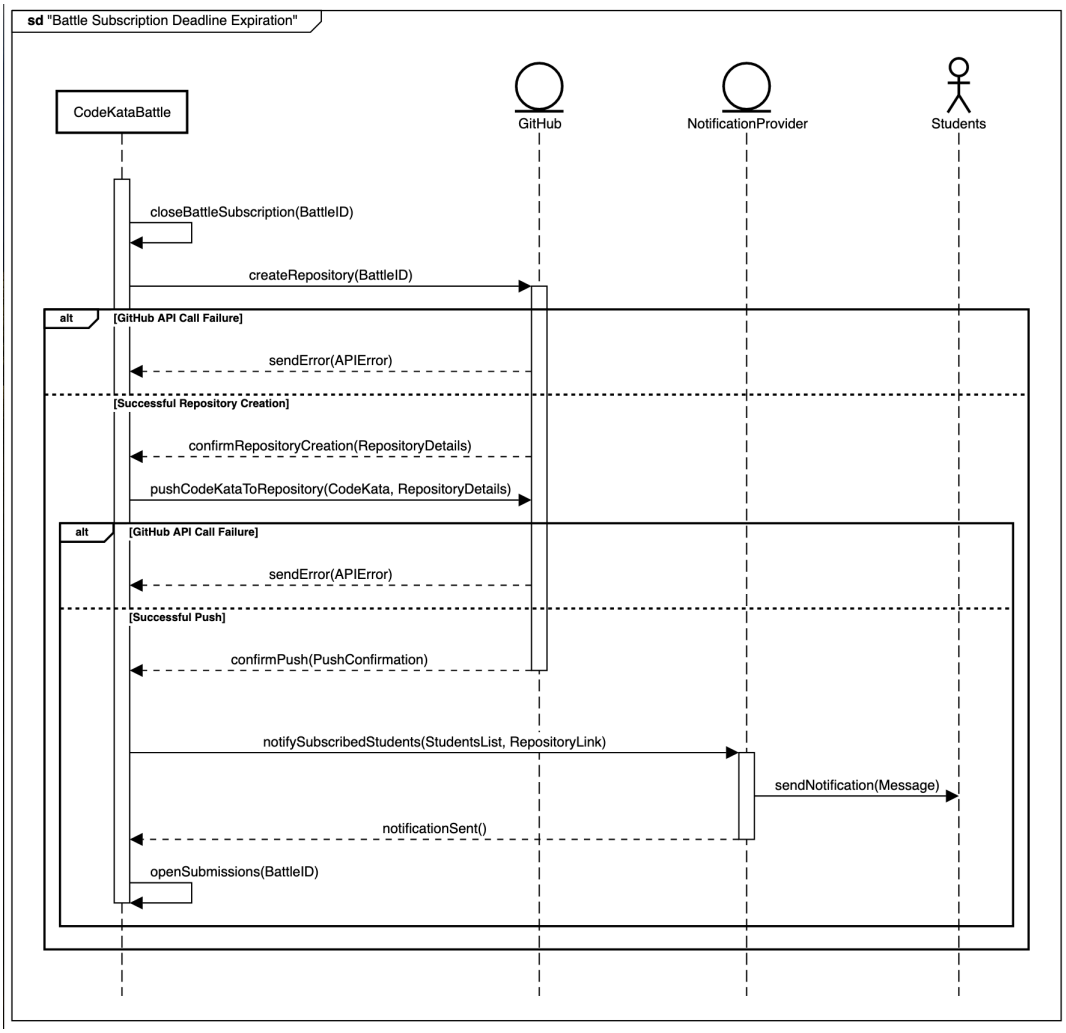
CodeKataHandlesStudentsPush	
Participating Actors	GitHub, Student, CodeKataBattle
Entry Condition	The Student is subscribed to the Battle and CodeKataBattle APIs got notified of a new commit for the Battle

Flow of Events	• 1. The Student commits a new version of his code to the repository's main branch • 2. The GitHub Action set up in the initial phase notifies CodeKataBattle APIs of the new available source code • 3. CodeKataBattle checks the commit happened before the Battle's submission deadline • 4. CodeKataBattle pulls the latest version of the source code from the Student's repository's main branch • 5. CodeKataBattle analyzes and compiles the source code • 6. CodeKataBattle executes the Battle's tests • 7. CodeKataBattle computes and registers the score of the latest commit • 8. CodeKataBattle compares the new scores with the previous best one of the group, in the case the former is better • 9. CodeKataBattle updates score of the group in the Battle and their ranking
Exit Condition	CodeKataBattle handled the latest commit, it has evaluated it, and updated score and ranking
Exceptions	• The push fails • The deadline was already expired when the push occurred • The pull fails • The automated evaluation encounters an issue



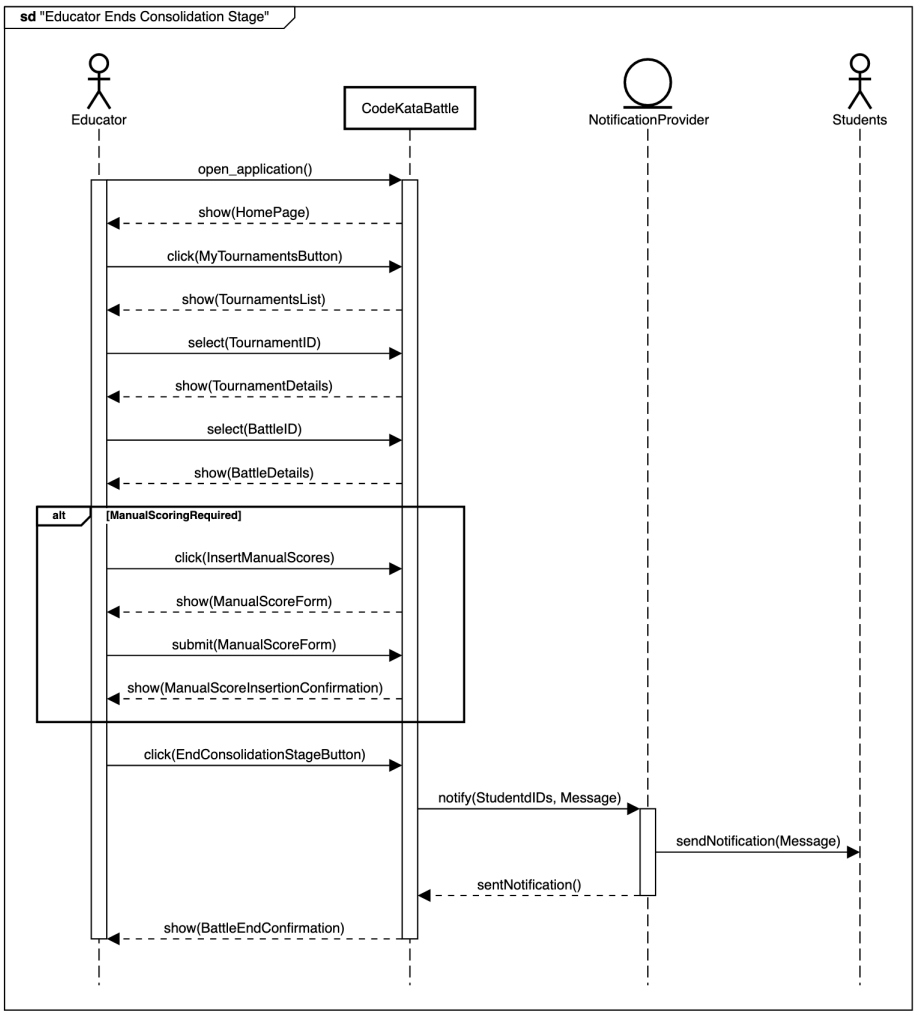
BattleSubmissionDeadlineExpiration	
Participating Actors	CodeKataBattle
Entry Condition	A Battle's submission deadline has reached its expiration point
Flow of Events	• 1. CodeKataBattle stops handling Students' commits • 2. CodeKataBattle processes all enqueued commits • 3. CodeKataBattle moves Battle to Consolidation Stage
Exit Condition	The Battle scores and rankings are updated and it is in Consolidation Stage

Exceptions	None



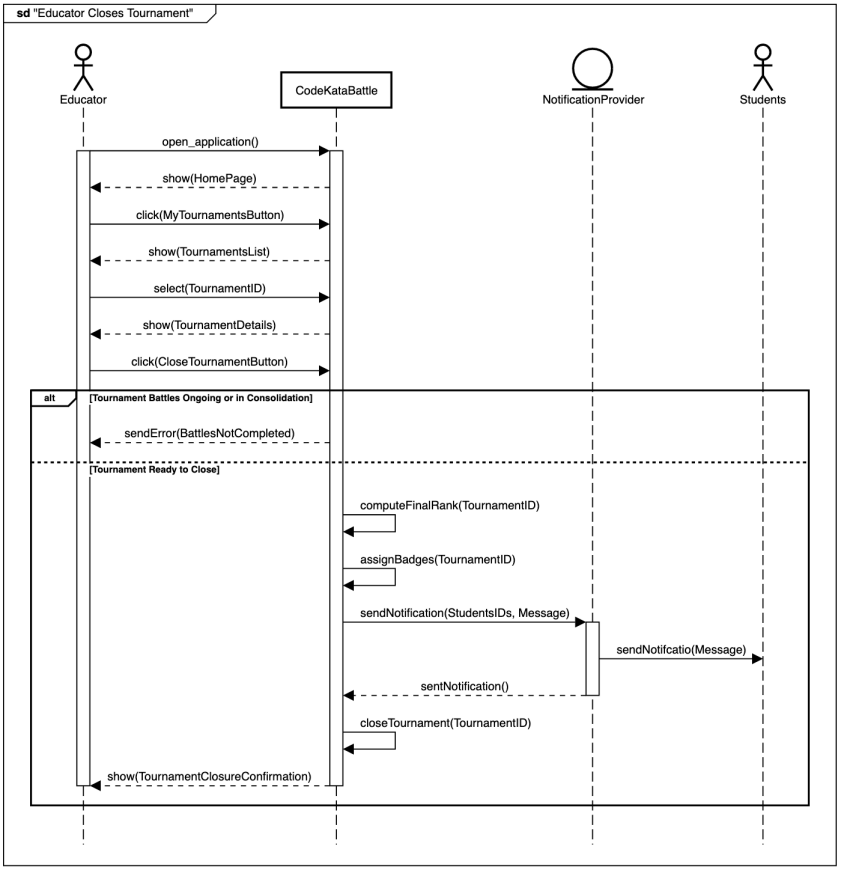
EducatorEndsBattle	
Participating Actors	Student, Educator, NotificationProvider, CodeKataBattle
Entry Condition	A Battle is in Consolidation Stage

Flow of Events	• 1. The Educator clicks on the “My Tournaments” button • 2. CodeKataBattle shows the list of the Tournaments that the Educator is allowed to manage • 3. The Educator clicks on a Tournament • 4. CodeKataBattle shows the Tournament’s details • 5. The Educator picks a Battle • 6. The Educator eventually assigns manual further scores • 7. The Educator clicks on "End Battle" button • 8. CodeKataBattle ends the Battle • 9. CodeKataBattle notifies through the NotificationProvider all the Students who participated in the Battle
Exit Condition	The Battle has finished, the Tournament’s personal scores of the Students are updated and the Students have been notified
Exceptions	None



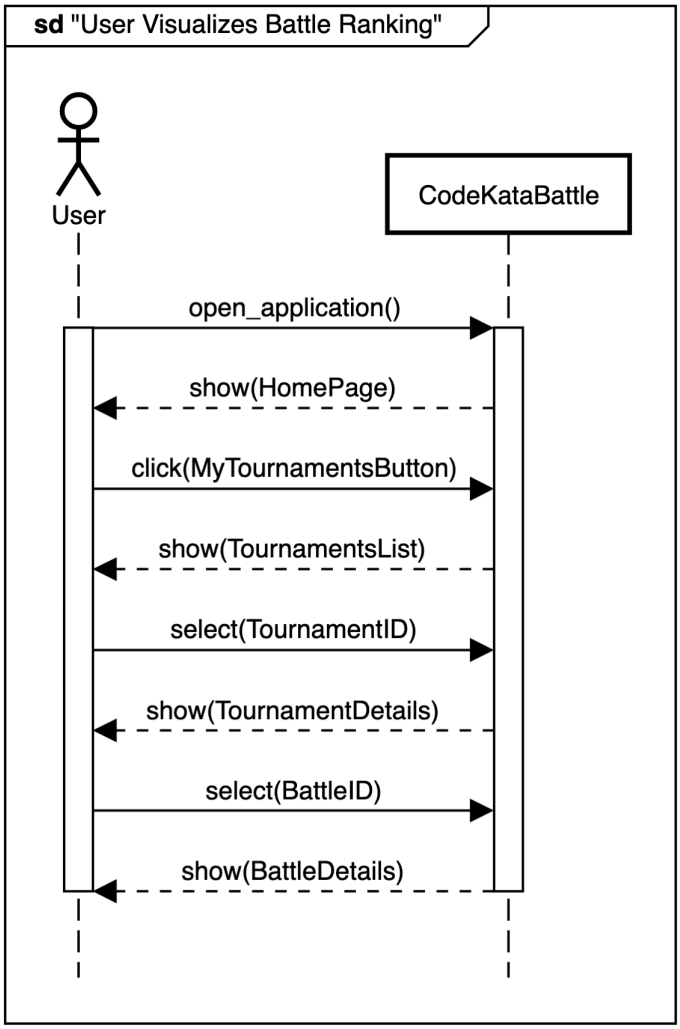
EducatorClosesTournament	
Participating Actors	Student, Educator, NotificationProvider, CodeKataBattle
Entry Condition	The Educator has the permission to manage the Tournament

Flow of Events	• 1. The Educator clicks on the "My Tournaments" button • 2. CodeKataBattle shows the list of the Tournaments that the Educator is allowed to manage • 3. The Educator clicks on a Tournament • 4. CodeKataBattle shows the Tournament's details • 5. The Educator clicks on the "Close Tournament" button • 6. CodeKataBattle closes the Tournament • 7. CodeKataBattle computes the final Tournament ranking • 8. CodeKataBattle eventually assigns Badges to the Students • 9. CodeKataBattle notifies through the NotificationProvider all the Students who were subscribed to the Tournament
Exit Condition	The Tournament is closed, all involved Students have been notified and Badges have been assigned
Exceptions	There is at least one Battle in the Tournament which is not finished



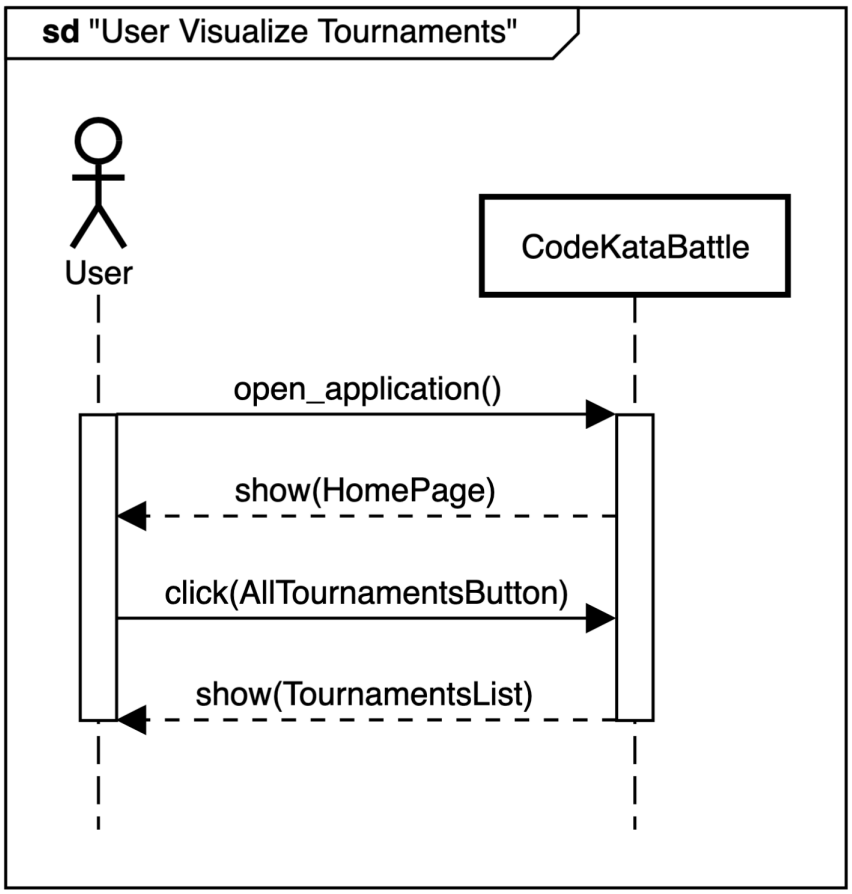
UserVisualizesBattleRanking	
Participating Actors	User, CodeKataBattle
Entry Condition	The Battle is started and the User is either an Educator who manages the Tournament of the Battle or a Students who is participating to the Battle
Flow of Events	• 1. The User clicks on the "My Tournaments" button • 2. CodeKataBattle shows the list of the Tournaments • 3. The Educator clicks on a Tournament • 4. CodeKataBattle shows the Tournament's details • 5. The User clicks on a Battle • 6. CodeKataBattle shows the Battle's details

Exit Condition	The User visualizes the current ranking for the Battle
Exceptions	None



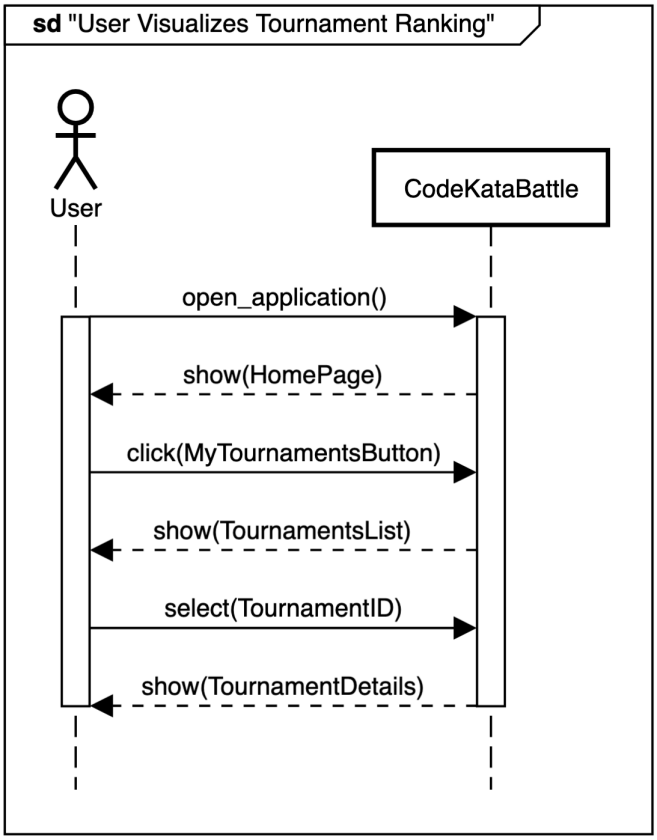
UserVisualizesListTournaments	
Participating Actors	User, CodeKataBattle
Entry Condition	User is logged in

Flow of Events	• 1. The User click on the “All Tournaments” button • 2. CodeKataBattle shows the list of the Tournaments
Exit Condition	The User visualizes the list of Tournaments
Exceptions	None



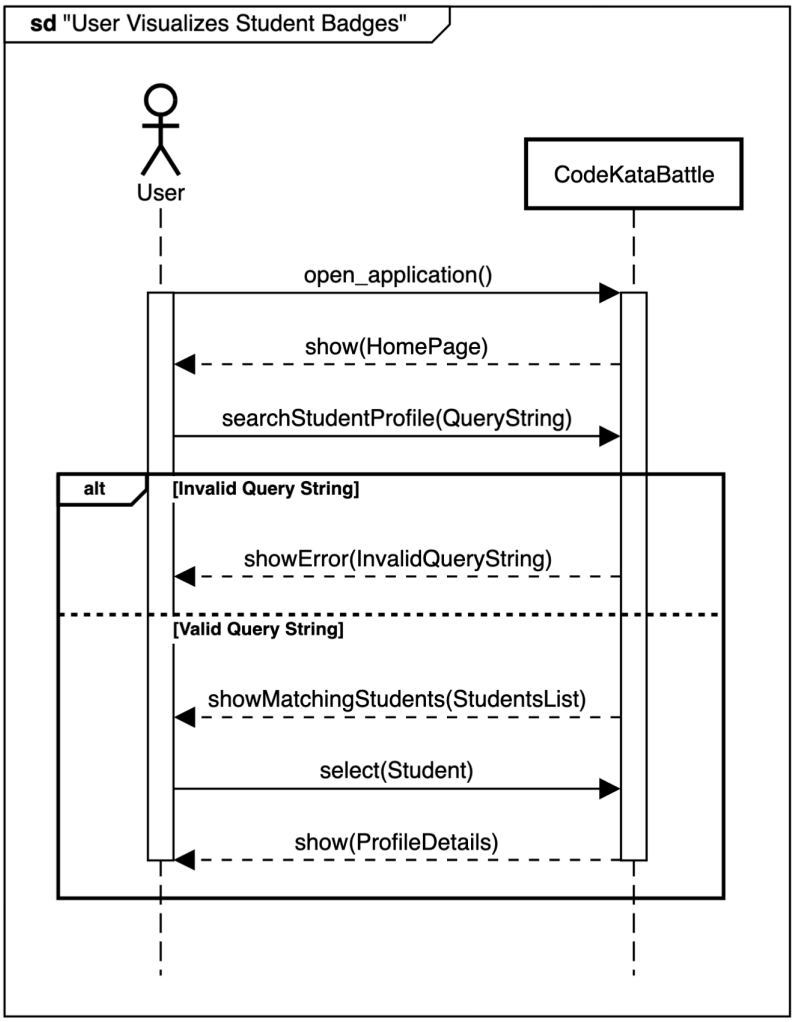
UserVisualizesTournamentRanking	
Participating Actors	User, CodeKataBattle

Entry Condition	The Tournament has started and the User either an Educator who manages the Tournament or a Students who is subscribed to the Tournament
Flow of Events	• 1. The User click on the “My Tournaments” button • 2. CodeKataBattle shows the list of Tournaments • 3. The User clicks on a Tournament • 4. CodeKataBattle shows the Tournament’s details
Exit Condition	The User visualizes the current ranking
Exceptions	None



UserVisualizesStudentBadges

Participating Actors	User, CodeKataBattle
Entry Condition	True
Flow of Events	• 1. The User looks for the Student's profile through the "Search" field • 2. CodeKataBattle shows the Students matching the query string • 3. The User clicks on a Student • 4. CodeKataBattle shows the Student's profile details
Exit Condition	The User visualizes the Student's Badges
Exceptions	The given query string is invalid



3.2.3. Mapping

G1

G1	**Allows registered Students who enrolled according to the right modalities to participate in a Tournament of Code Kata Battles and take part in its Battles.**
D1	The User must have a working Internet connection.
D2	The Students who participate to a Battle properly set up the GitHub Action for code submissions.
D3	The Users always receive every notification which is sent by the system.
D4	The availability of the external tools utilized for the code evaluation process is consistent.
D5	The availability of GitHub Actions service is consistent.
R1	The system must allow an unregistered Educator to sign up.
R2	The system must allow an unregistered Student to sign up.
R3	The system must allow a registered User to log in.
R4	The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
R8	The system must allow registered Educators to end the creation process of a Tournament that they started themselves.
R10	The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.
R11	The system must allow a registered Student to create a group for a Battle in a Tournament.
R12	The system must allow a registered Student to accept an invitation to a group for a Battle in a Tournament.
R13	The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline is not expired yet.
R14	The system must allow registered Students who are enrolled in a Battle to perform code submissions.
R17	The system must set the Consolidation Stage of a Battle when its submission deadline expires.
R18	The system must let Educators to end the Consolidation Stage of a Battle if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.

R20	The system must let Educators who are either the creator of the Tournament or who have been granted the permission to by the latter to close a Tournament if and only if there is no Battle such that either their subscription or submission deadline is not expired yet or such that they are still in the Consolidation Stage.
R23	The system must notify every registered Student about the creation of a new Tournament.
R24	The system must notify every registered Student about the creation of a new Battle within a Tournament they are enrolled in.
R25	The system must notify every registered Student about the end of a Battle they are participating in.
R26	The system must notify every registered Student about the end of a Tournament they are enrolled in.

G2	
G2	**Allows registered Educators to manage Tournaments for which they have been granted permission.**
D1	The User must have a working Internet connection.
R1	The system must allow an unregistered Educator to sign up.
R3	The system must allow a registered User to log in.
R4	The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
R7	The system must allow registered Educators to grant other registered Educators the permission to manage the Tournament, during the Tournament creation process.
R8	The system must allow registered Educators to end the creation process of a Tournament that they started themselves.
R10	The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.
R18	The system must let Educators to end the Consolidation Stage of a Battle if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.

R20	The system must let Educators who are either the creator of the Tournament or who have been granted the permission to by the latter to close a Tournament if and only if there is no Battle such that either their subscription or submission deadline is not expired yet or such that they are still in the Consolidation Stage.

G3	
G3	**Allows registered Students who participate in a Tournament of Code Kata Battles to be rewarded of different achievements.**
D1	The User must have a working Internet connection.
D2	The Students who participate to a Battle properly set up the GitHub Action for code submissions.
D3	The Users always receive every notification which is sent by the system.
D4	The availability of the external tools utilized for the code evaluation process is consistent.
D5	The availability of GitHub Actions service is consistent.
R1	The system must allow an unregistered Educator to sign up.
R2	The system must allow an unregistered Student to sign up.
R3	The system must allow a registered User to log in.
R4	The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
R5	The system must provide registered Educators of a list of Tournament-related statistics for the Badges definition, during the Tournament creation process.
R6	The system must provide registered Educators of a specific language which lets them define the Badges, during the Tournament creation process.
R8	The system must allow registered Educators to end the creation process of a Tournament that they started themselves.
R10	The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.
R11	The system must allow a registered Student to create a group for a Battle in a Tournament.

R12	The system must allow a registered Student to accept an invitation to a group for a Battle in a Tournament.
R13	The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline is not expired yet.
R14	The system must allow registered Students who are enrolled in a Battle to perform code submissions.
R17	The system must set the Consolidation Stage of a Battle when its submission deadline expires.
R18	The system must let Educators to end the Consolidation Stage of a Battle if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.
R20	The system must let Educators who are either the creator of the Tournament or who have been granted the permission to by the latter to close a Tournament if and only if there is no Battle such that either their subscription or submission deadline is not expired yet or such that they are still in the Consolidation Stage.
R21	The system must assign an achievements' Badge for a given Tournament to any Student who satisfied the conditions defined by the creator of the Tournament.

G4	
G4	**Allows registered Users to visualize information for which they have granted permission.**
D1	The User must have a working Internet connection.
R1	The system must allow an unregistered Educator to sign up.
R2	The system must allow an unregistered Student to sign up.
R3	The system must allow a registered User to log in.
R4	The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
R8	The system must allow registered Educators to end the creation process of a Tournament that they started themselves.
R10	The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.

R11	The system must allow a registered Student to create a group for a Battle in a Tournament.
R12	The system must allow a registered Student to accept an invitation to a group for a Battle in a Tournament.
R13	The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline is not expired yet.
R16	The system must update the Battle ranking when a valid code submission is performed.
R17	The system must set the Consolidation Stage of a Battle when its submission deadline expires.
R18	The system must let Educators to end the Consolidation Stage of a Battle if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.
R19	The system must update Tournament ranking when a Battle exits the Consolidation Stage.
R20	The system must let Educators who are either the creator of the Tournament or who have been granted the permission to by the latter to close a Tournament if and only if there is no Battle such that either their subscription or submission deadline is not expired yet or such that they are still in the Consolidation Stage.
R21	The system must assign an achievements' Badge for a given Tournament to any Student who satisfied the conditions defined by the creator of the Tournament.
R22	The system must allow every User to see the Badges which were ever obtained by a given Student.

G5	
G5	**Automates code evaluation process using GitHub Actions and other external tools.**
D2	The Students who participate to a Battle properly set up the GitHub Action for code submissions.
D4	The availability of the external tools utilized for the code evaluation process is consistent.
D5	The availability of GitHub Actions service is consistent.
R1	The system must allow an unregistered Educator to sign up.

R2	The system must allow an unregistered Student to sign up.
R3	The system must allow a registered User to log in.
R4	The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
R8	The system must allow registered Educators to end the creation process of a Tournament that they started themselves.
R10	The system must allow a registered Educator to create a Battle in a Tournament if and only if he is the creator of the Tournament or if he was granted the permission to by the latter.
R11	The system must allow a registered Student to create a group for a Battle in a Tournament.
R12	The system must allow a registered Student to accept an invitation to a group for a Battle in a Tournament.
R13	The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline is not expired yet.
R14	The system must allow registered Students who are enrolled in a Battle to perform code submissions.
R15	The system must be provided of proper APIs to let registered Students perform code submissions through GitHub Actions.

3.3. Performance Requirements

Due to the non-critical nature of the system, too strict performance requirements are not needed. However, in order to offer the best possible User experience, the system should provide:

- The system should let Educators create a Tournament within 2 seconds.
- The system should let Educators create a Battle within 2 seconds.
- The system should let Students join a Tournament within 2 seconds
- The system should let Students join a Battle within 2 seconds
- The system should receive a code submission by a Student within 5 seconds

- The system should evaluate code submissions within 180 seconds
- The system should update the Battle ranking within 2 seconds
- The system should update the Tournament ranking within 2 seconds
- The system should let Educators end the Consolidation Stage of a Battle within 5 seconds.
- The system should let Educators end a Tournament within 5 seconds.
- The system should let new Badges be available in the Students profiles within 2 seconds

---

3.4. Design Constraints

In this section some information about the design constraint are provided.

3.4.1. Standards compliance

Specifications described in this document must be respected by the system. The source code of the application must be commented on and documented adequately.
The system should respect the guidelines described by the GDPR.

3.4.2. Hardware limitations

The system requires any device and a stable internet connection.

3.5. Software System Attributes

In this section typically, the non-functional requirements and quality attributes of the system which should be provided are discussed.

3.5.1. Reliability and Availability

The system should offer its functionalities with an availability equal to 99.5%, or more. In other words, the system must be inaccessible for less than two days every year. To achieve this goal, the system should provide a high redundancy for the most critical components.
Furthermore, in order to guarantee better reliability performances, all the scheduled maintenance actions on the system should be done during the night.

3.5.2. Security

The connection between the application and the server must be safe. To keep a good level of security, the system should use the Transport Layer Security protocol. For this purpose, it is needed an SSL/TSL certificate.

3.5.3. Maintainability

The source code must be commented on as well as possible and the correlated documentation must be kept updated during the whole life cycle of the system.
Modularity and low coupling between components must be a focus during the designing and developing phases.

4 | Formal analysis using alloy

4.1. Objectives of the analysis

In this section, a presentation of the formal modelling activity that has been done using the Alloy formal notation. The main goal of this activity is to formally describe the domain and properties of the system to be.
In particular, the main objective of this activity is to model and formally represent Educators, Students, Tournaments, Battles and Badges, as well as the main constraints which regard such entities.
In order to ensure straightforward comprehension of the following, each constraint and entity will be annotated in natural language, along with any predicates and assertions which have been verified.

4.2. Alloy Code

```
// ———— Definition of auxiliar entities ———— //


// Defined in order to model the deadlines

sig Time {
timestamp: Int
}{
timestamp >= 0
}


// Defined in order to model any flag, such as whether a Battle is in the Consolidation Stage
```

```
abstract sig Boolean {} one sig True extends Boolean {} one sig False extends Boolean {}

// ———— Definition of game components ———— //

// Defined in order to model the score of a team in a Battle and the score of a Student in a Tournament sig Score {
value: Int
}{
value >= 0
}

// Defined in order to model the rank of a team in a Battle and the rank of a Student in a Tournament sig Rank {
value: Int
}{
value >= 1
}

// Defined in order to model the Tournament sig Tournament {
managers: some Educator,
subscription_deadline: one Time,
battles: set Battle,
scores: Student -> one Score,
ranks: Student -> one Rank, badges: set Badge,
achievements: Student -> set BadgeRule
}

// Defined in order to model the Battle (unnecessary to model the programming language) sig Battle {
tournament: one Tournament,
teams: Student -> some Student,
scores: Student -> one Score,
ranks: Student -> one Rank,
```

subscription_deadline: one Time,
submission_deadline: one Time,
consolidation_stage: one Boolean
}

// Defined in order to model the Badge (unnecessary to model the title) sig BadgeRule {
} sig Badge {
rule: one BadgeRule
}

// ———— Definition of actors ———— //

// Defined in order to model the nature of Student and Educator as Users abstract sig User {}

// Defined in order to model the Student sig Student extends User {
badges: set Badge
}

// Defined in order to model the Educator sig Educator extends User{
}

// ———— Facts ———— //

// Tournaments: A Student must both have a score and a rank, if a Student has higher score than another Student than it has a lower rank (e.g. 1st is lower than 2nd), only achievements of Badges which relate to existing Badges for the Tournament are stored, a Student has one only score, a Student has one only rank fact TournamentFact {

all t1: Tournament | all s: Student | some sc: Score | some r: Rank | (sc in s.(t1.scores)) => (r in s.(t1.ranks)) and (r in s.(t1.ranks) => sc in s.(t1.scores))
all t2: Tournament | all s1, s2: Student | ((s1.(t2.ranks).value < s2.(t2.ranks).value) <=> (s1.(t2.scores).value > s2.(t2.scores).value))

```
all t3: Tournament | all s3: Student | all b: Badge | ( b.rule in s3.(t3.achievements) ) => ( b
in t3.badges )
all t4: Tournament | all disj sc1, sc2: Score | some s: Student | ( sc1 in s.(t4.scores) ) => not
( sc2 in s.(t4.scores) )
all t5: Tournament | all disj r1, r2: Rank | some s: Student | ( r1 in s.(t5.ranks) ) => not ( r2
in s.(t5.ranks) )
}
```

// Battles: The subscription deadline must expire before the submission deadline, teams relationships are symmetrical, teams relationships are reflexive, if a Student s1 is in team with a Student s2 and not with a Student s3 then s2 is not in team with s3, Students in the same team have same score and rank, a Student has one only score, a Student has one only rank

```
fact BattleFact {
all b: Battle | b.subscription_deadline.timestamp < b.submission_deadline.timestamp
all b1: Battle | all s1, s2: Student | s2 in s1.(b1.teams) <=> s1 in s2.(b1.teams)
all b2: Battle | all s: Student | s in s.(b2.teams)
all b3: Battle | all s3, s4, s5: Student | ( ( s4 in s3.(b3.teams) ) and ( not ( s5 in s3.(b3.teams)
) ) ) => ( not (s4 in s5.(b3.teams) ) )
all b4: Battle | all s6, s7: Student | s7 in s6.(b4.teams) => ( ( s6.(b4.scores).value = s7.(b4.scores).value
) and ( s6.(b4.ranks).value = s7.(b4.scores).value ) )
all b5: Battle | all disj sc1, sc2: Score | some s8: Student | ( sc1 in s8.(b5.scores) ) => ( not (
sc2 in s8.(b5.scores) ) )
all b6: Battle | all disj r1, r2: Rank | some s9: Student | ( r1 in s9.(b6.ranks) ) => ( not ( r2
in s9.(b6.ranks) ) )
}
```

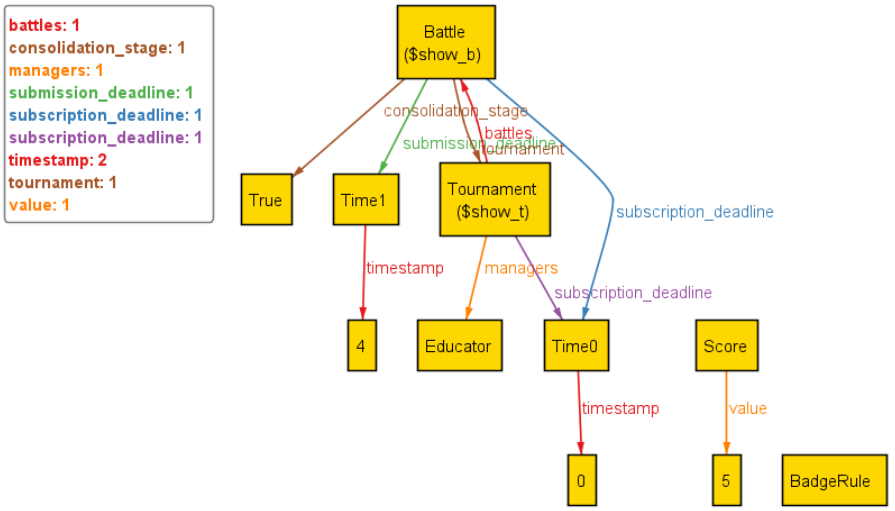
// A Student must have obtained the Badges in some Tournament

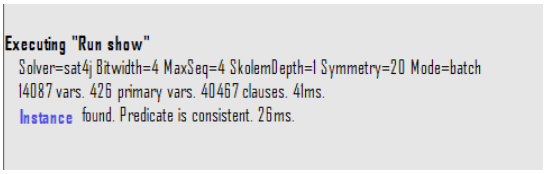
```
fact StudentObtainedBadges {
one t: Tournament | all s: Student, b: Badge | b in s.badges => b.rule in s.(t.achievements)
}
```

// ———— Predicates and assertions ———— //

// 1. A basic example

```
pred show {
some b: Battle | some t: Tournament | b in t.battles
}
run show
```





// 2. An example to show that a higher position in the ranking corresponds to a lower score

```
pred RankingExample {
some disj b1, b2: Battle | some t: Tournament | b1 in t.battles and b2 in t.battles
some disj s3, s4: Student | some b3: Battle | ( not ( s3 in s4.(b3.teams) ) ) and ( s3.(b3.ranks).value
```

```
!= s4.(b3.ranks).value )
some r1, r2: Rank | r1.value = 1 and r2.value = 2
some sc1, sc2: Score | ( sc1. value = 5 ) and ( sc2.value = 6 )
}
run RankingExample
```

this/Battle	tournament	teams		scores		ranks	
Battle$0	Tournament$0	Student$0	Student$0	Student$0	Score$0	Student$0	Rank$1
		Student$1	Student$1	Student$1	Score$2	Student$1	Rank$0
Battle$1	Tournament$0	Student$0	Student$0	Student$0	Score$0	Student$0	Rank$1
		Student$1	Student$1	Student$1	Score$2	Student$1	Rank$0
Battle$2	Tournament$0	Student$0	Student$0	Student$0	Score$2	Student$0	Rank$0
		Student$1	Student$1	Student$1	Score$0	Student$1	Rank$1

this/Rank	value
Rank$0	5
Rank$1	2
Rank$2	1

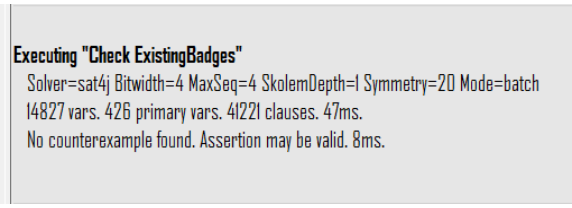
this/Score	value
Score$0	2
Score$1	6
Score$2	5

```
Executing "Run RankingExample"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  15533 vars. 450 primary vars. 42690 clauses. 39ms.
  Instance found. Predicate is consistent. 73ms.
```

// 3. Ensuring that Badges obtained by Students must be existed in some Tournaments (in this model, Tournament can both represent both a running one or a finished one)

```
assert ExistingBadges {
all b: Badge | all s: Student | some t: Tournament | ( b in s.badges ) => ( b in t.badges)
```

}
check ExistingBadges



5 | Effort spent

5.1. Effort spent per unit

This section shows the amount of time that each member has spent to produce the document. Please notice that each unit is the result of coordinated work among all the members.

UNIT	MEMBERS	HOURS
SetUp	Puglisi	1h
Scenarios	Piccinato	3h
Use Cases	Piazzalunga, Piccinato	8h
Phenomena	Puglisi, Piccinato	5h
Goals	Puglisi, Piccinato	2h
Domain Assumptions	Piccinato, Puglisi	1h
Requirements	Piccinato, Puglisi	7h
Mapping	Piccinato	1h
Sequence Diagrams	Piazzalunga	7h
Class Diagram	Piazzalunga	3h
Use Case Diagrams	Piazzalunga	2h
State Charts	Piazzalunga	1h
UI Mockups	Puglisi	15h
Alloy	Piccinato	8h
Chapter 1 Redaction	Puglisi, Piccinato	4h
Chapter 2 Redaction	Piccinato	5h
Chapter 3 Redaction	Piccinato	4h
Chapter 4 Redaction	Piccinato	2h
Chapters 5 and 6 Redaction and Final Review	Piccinato, Puglisi, Piazzalunga	5h

6 References

6.1. References and Tools

1. GitHub: https://www.github.com
2. GitHub Actions: https://github.com/features/actions
3. The UI Mockups have been made with: https://www.figma.com
4. Alloy Language Reference: https://alloytools.org/download/alloy-language-reference.pdf
5. Alloy Tools: https://alloytools.org
6. Sequence Diagrams have been made with: https://sequencediagram.org
7. Use Case Diagrams have been made with: http://draw.io
8. Class Diagrams have been made with: http://draw.io
9. State Charts have been made with: http://draw.io