POLITECNICO DI MILANO

Computer Science and Engineering

Requirements Analysis and Specification Document

CodeKataBattle

Authors:

Manuela Marenghi
Luca Cattani
Tommaso Fellegara

Professor:

Matteo Giovanni Rossi

Table of Contents

1. Introduction

1.1 Purpose

CodeKataBattle is a platform where students can join code competitions and experiment team working. CKB lets educators create tournaments where to publish coding battles, that are software development exercises whose focus is not only on solving a task or problem, but on learning new skills and developing successful routines. Using CKB students join these tournaments and collaborate using GitHub to experiment different programming languages and development platforms. This document has the purpose of giving a complete description of the requirements and specifications of the CKB platform.

1.1.1 Goals

- [G1] Educators create tournaments
- [G2] Educators create battles for students to solve
- [G3] Educators evaluate student's solutions
- [G4] Students take part in tournaments together with other students
- [G5] Students solve coding problems and see educator's evaluations

1.2 Scope

The platform allows students to take part in coding tournaments, where they will have to solve coding problems in the form of battles.

A code kata consists of a project containing:

- a textual problem description
- a set of test cases the implementation must pass
- any necessary build automation tool

Each tournament is created by an educator, who can choose to allow other educators to create battles for the tournament. To create a new battle within a tournament on the platform, an educator must have been given permission to create battles for that tournament and has to provide the following data:

- a code kata
- the minimum and maximum number of students per group
- a registration deadline
- a final submission deadline
- configurations for scoring

Educators can also create gamification badges for their tournaments, these are elements in the form of individual rewards with a title and a rule about how to obtain them. Each badge can be assigned to one or more students, depending on the rules. When a student obtains a badge, it will show up on their profile, and everyone else (educators and other students) will be able to see it.

All students subscribed to the CKB platform are notified whenever a new tournament is created, and they can subscribe to the tournament by a given deadline (chosen by the tournament creator). If they subscribe, they are notified of all upcoming battles created within that tournament.

After the creation of a battle, students use the platform to form teams for that battle. In particular, each student can join a battle on their own or by inviting other students to their team (respecting the minimum and maximum number of students per group set for that battle by the creator).

When a battle's registration deadline expires, CKB creates a GitHub repository containing the code kata and sends the link to all students who are members of a valid subscribed team. In particular, students are asked to fork the GitHub repository of the code kata and set up an automated workflow through GitHub actions that informs the CKB platform (through proper API calls) as soon as a commit is pushed into the main branch of their repository. Each commit pushed to the main branch of a group's repository must trigger the CKB platform (through GitHub actions) to pull the repository's source, analyze it by running tests on the corresponding executables, and calculate and update the battle score for that team. The score is a number between 0 and 100 and is calculated considering the following:

- number of test cases passed
- time passed between the start of the contest and the time of the submission
- quality of the code (in the matter of security, maintainability and reliability)
- a personal score assigned by the educator (optional)

At the end of each battle, the platform updates the personal tournament score of each student, that is, the sum of all battle scores received in that tournament. Thus, for each tournament, there is a rank that measures how a student's performance compares to other students in the context of that tournament. All users can see the list of ongoing tournaments as well as the corresponding tournament rank.

When an educator closes a tournament, as soon as the final tournament rank becomes available, the CKB platform notifies all students involved in the tournament.

Each user (student or educator) may also browse the list of present and past tournaments, look at tournament and battle rankings, and check out any student or educator profile.

1.2.1 World phenomena

- [W1] Students collaborate on a project
- [W2] Students choose a language in which to solve a battle problem
- [W3] Educators create code battles
- [W4] Students fork a GitHub repository
- [W5] Students create a GitHub action to automatically use CKB's API

1.2.2 Shared phenomena

controlled by the world and observer by the system
- [SP1] Educators create a new tournament
- [SP2] Educators create a new battle
- [SP3] Educators assign a score to student's battle solutions
- [SP4] Students join a tournament
- [SP5] Students create or join battle groups
- [SP6] Students and Educators login to CKB
- [SP7] Students and Educators sign up to CKB
- [SP8] Students trigger a Github action to call CKB's API
- [SP9] Educators create badges
- [SP10] Deadlines for battles registrations expire
- [SP11] Deadlines for battles submission expire
- [SP12] Deadlines for tournament registrations expire
- [SP13] Educators allow other educators to create battles on their tournament

controlled by the system and observed by the world
- [SP14] The system sends a battle's repository link to all registered users
- [SP15] The system generates rankings for each battle
- [SP16] The system assigns score to students
- [SP17] The system warns users about expired deadlines
- [SP18] The system warns users new tournaments being created
- [SP19] The system warns users when battles are created in a tournament they are subscribed in
- [SP20] The system assigns a badge to a user

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Code kata** - the set of: textual description, test cases and build automation scripts that form a coding problem users on the platform have to solve.
- **Code battle** - the grouping of code kata and battle settings, described by an educator, that constitute a coding challenge on the platform. Note that code battles are also simply referred to as "battles" in this document.
- **Tournament** - a collection of code battles created by one or more educators.
- **Users** - everyone using the platform, that is, students, educators and everyone who is browsing the platform and is not logged in yet.
- **Consolidation phase** - interval of time defined after the submission deadline of a battle expires and the educator could evaluate manually the solution of the students. After that there is a phase in which the standings are updated.

1.3.2 Acronyms

- **CKB** - CodeKataBattle
- **API** - Application Programming Interface
- **UML** - Unified Modeling Language
- **SMD** - State Machine Diagram

- **SD** - Sequence Diagram

1.3.3. Abbreviations
- **[Gi]** - system's i-th goal
- **[SPi]** - i-th shared phenomena
- **[Wi]** - i-th world phenomena
- **[Ri]** - i-th requirement
- **[Di]** - i-th domain assumption
- **[UCi]** - i-th use case

1.4 Document Structure

Mainly the current document is divided in 4 chapters, which are:
- **Introduction**: it aims to describe the environment of the project, highlighting shared and world phenomena. In particular it's focused on the reasons and the goals that are going to be achieved with its development.
- **Overall Description**: it's a high-level description of the system focusing on the description of possible scenarios, a domain model and state machine diagrams, also general product functions are highlighted together with domain assumptions.
- **Specific Requirements**: it describes in detail the requirements and the system's use cases. General design constraints are also included in this part.
- **Formal Analysis**: this section contains a formal description of the environment where the system operates and an analysis of constraints performed using Alloy.

2. Overall Description

2.1 Product perspective

2.1.1 Scenarios

The following scenarios provide a general description about how the main functions of they system will be executed by the users:

- **Tournament Creation**: an educator wants to create a tournament, they sign up as and navigate to the tournament creation page. The tournament creation page asks the educator to provide a registration deadline for the tournament, they can also choose to define as many gamification badges as they want, these are elements in the form of rewards that will be awarded to students at the end of the tournament, defining such elements requires to define both a title and a set of rules to define who will be awarded the badge. When they are done with setting up the tournament, the educator clicks the "create tournament" button to create the tournament, as a consequence the system creates the tournament and notifies all students registered on the platform.

- **Battle creation**: an educator wants to create a new battle for a tournament, to do so they have to either have been granted permission from the creator of the tournament or be the creator of the tournament theirself. After going to the tournament page and clicking the "create new battle" button, they will be shown a form to provide all that's necessary to create a battle, that is: a code kata, scoring settings, registration and submission deadlines and boundaries regarding the allowed number of students for each team. After filling the form, the educator clicks the "create battle" button to finally prompt the system to create a new battle. The system also notifies all students registered to the tournament about the creation of a new battle.

- **Battle joining**: a student who wants to join a battle can do so by clicking the "join battle" button on a battle page, by doing so the system creates a team for them. Once in a team a student can invite friends to collaborate or leave their current team to join another one by either accepting an invite or selecting another team from the battle's "teams" tab, they can also just leave the battle and the system will remove them from their team.

- **Solution submission**: CodeKataBattle provides an API for students to integrate in a GitHub actions workflow on their repository. When a battle starts, students will first be provided a GitHub repository to fork, then they'll have to create a new GitHub action to make an API call each time a commit is pushed to the main branch of their repository. By doing so the team can try to build many solutions to the problem and test them locally using the test provided in the forked repository, once they are ready to submit their work they just need to push a commit to their main repository, or just merge their current branch (ahead of the main branch) to the main branch. When the API is called, the system will run automated tests using static analysis and give the team a score between 0 and 100.

2.1.2 Domain model

The following picture represents all elements of the domain in which the system operates, the relations between them and what attributes identify the entities. In particular we highlight the following relations:

- There are two types of users: students and educators
- A single educator can create multiple tournaments
- A single educator can create multiple coding battles
- Each battle is related to a single tournament
- Each battle may contain a number of gamification badges
- A tournament is composed of multiple battles
- Each student can join one or more tournaments
- Each student can form teams to take part in different battles
- Each team can submit solutions to a single battle
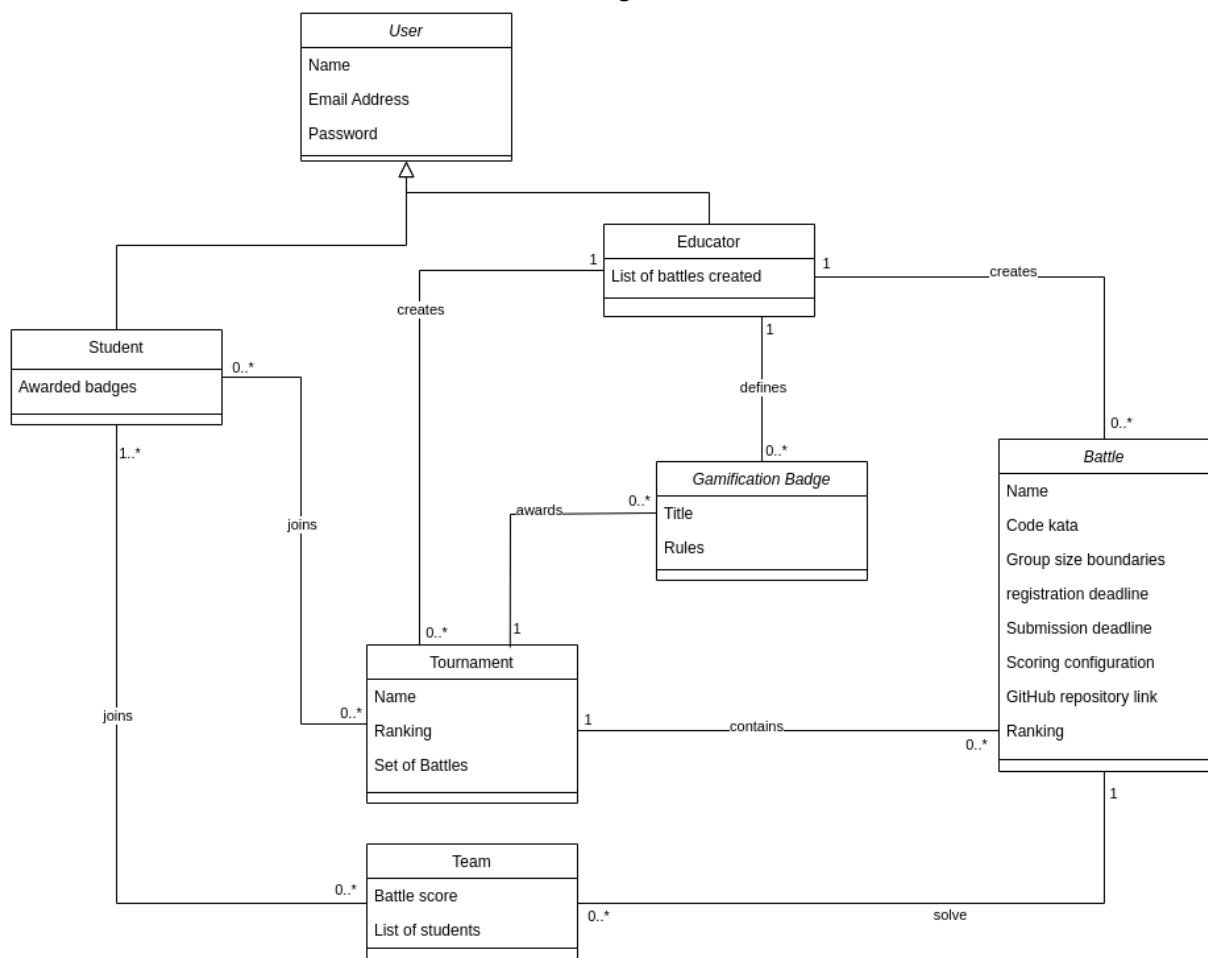
2.1.3 State machine diagrams

The following state diagrams provide a general understanding about the lifecycle of different elements of the system:

- **Battle**: a battle is first created by an educator, students can form teams until the registration deadline expires, then they can submit their solutions to the problem. After the submission deadline expires, educators could evaluate solutions.
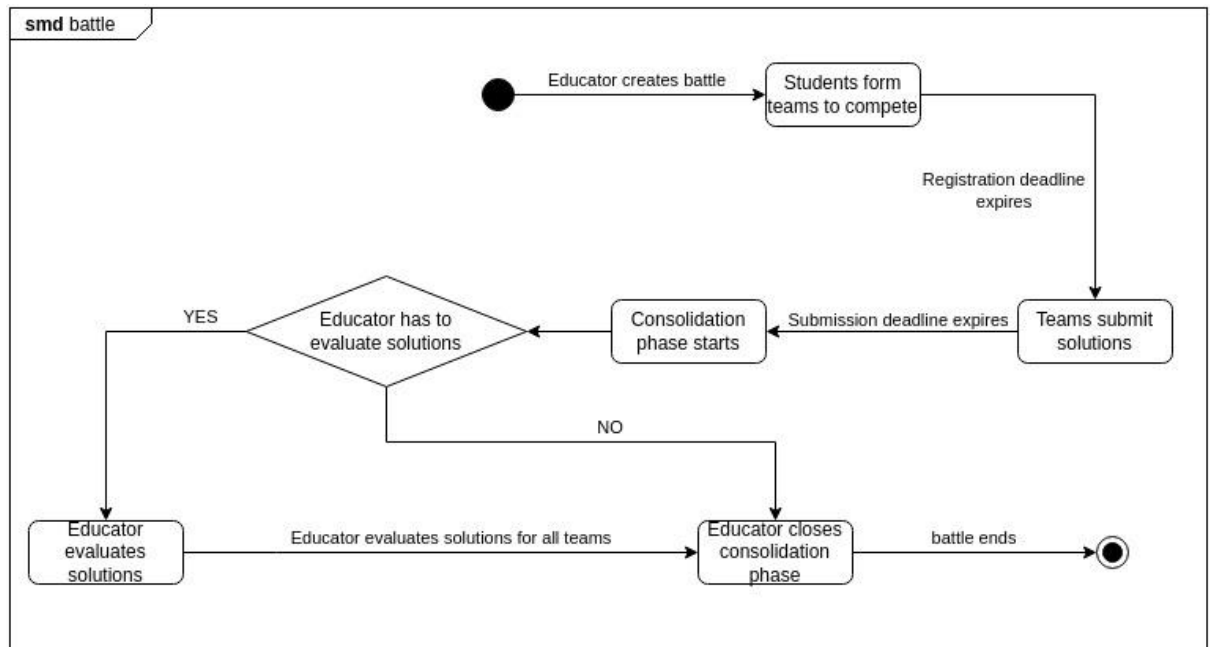
- **Team**: a student can create a team, after that students could join or leave the team up until the expiration of the registration deadline, then the team can start submitting solutions to the coding battle for which it's been created, after the submission deadline, the team cannot essentially do anything.
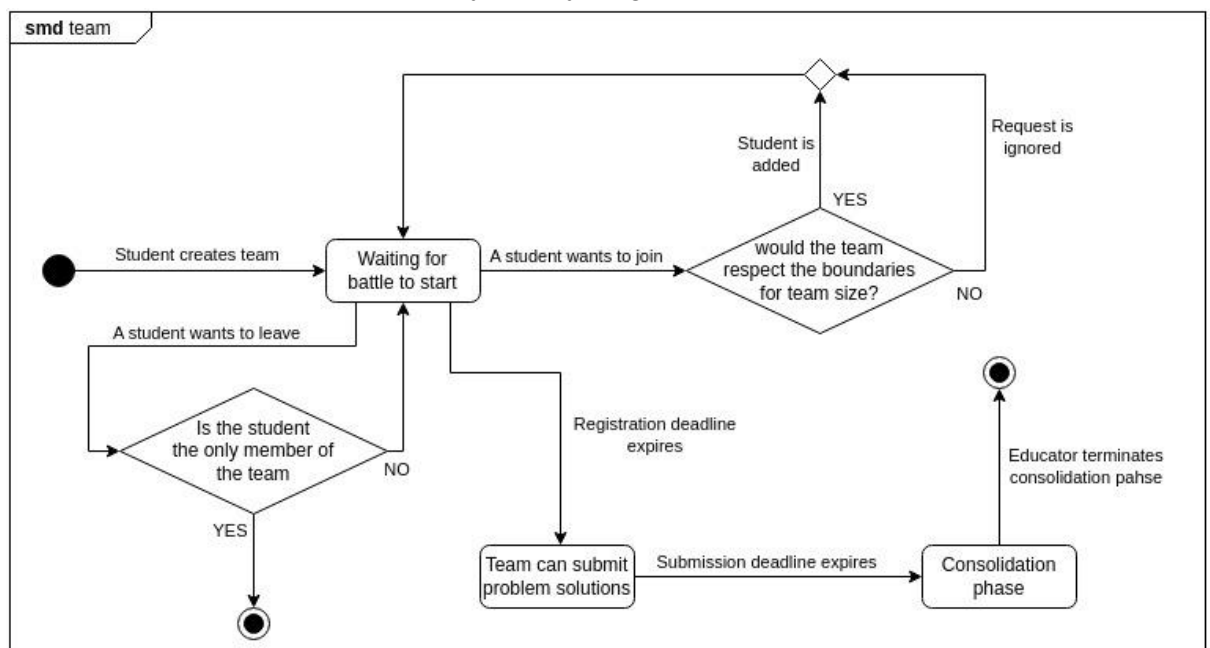
- **Tournament**: educators can create a tournament, build battles for it and allow other educators to do so throughout the life of the tournament, the rankings are updated each time a battle ends (ranking updates occur sequentially with respect to the end of battles) and become final once the creator of the tournament closes it
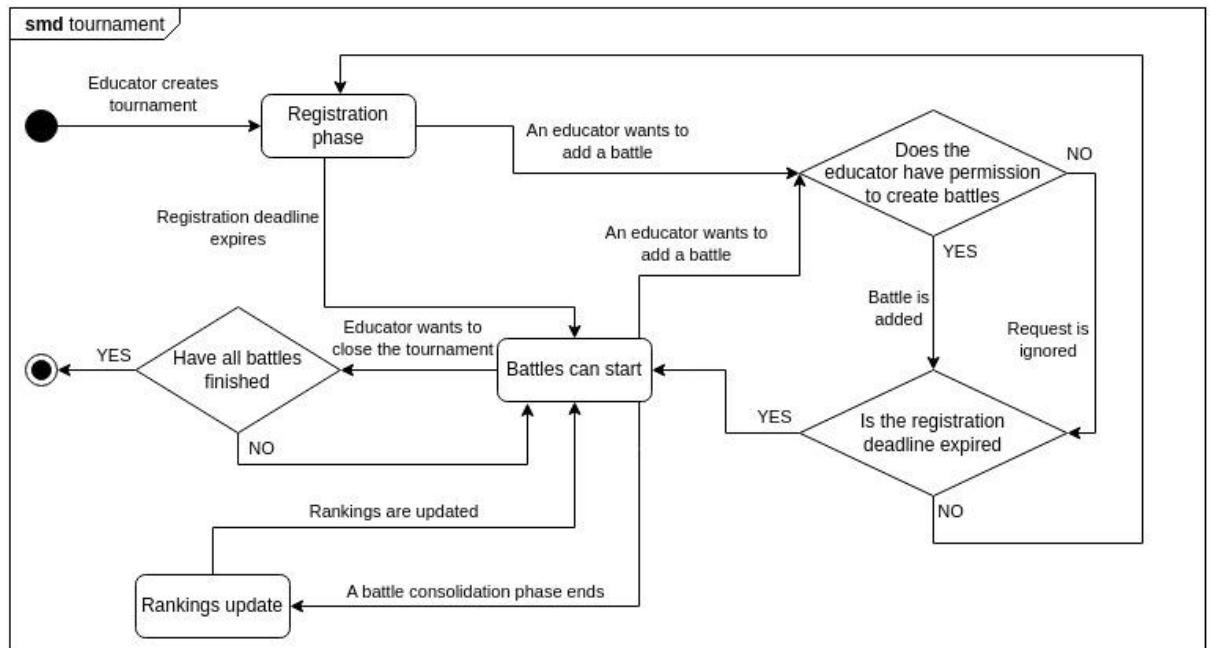
2.2 Product functions

2.2.1 User sign up and login

To access all the functionalities of the platform users must create an account by inserting their personal information. When creating their account they can choose between two types :
- educator, that gives you the permission to create tournaments,battles and to give evaluations to students' work.
- student, that gives you the possibility to join tournaments and battles and to create groups with other students.

After choosing one type of account, users will be asked to provide an email and a password that will become its credentials.

Each time a user needs to use the platform it has to login with its credentials defined in the sign up phase.

2.2.2 Tournament creation

An educator can create a tournament after signing in, to do so he/she has to specify the subscription deadline and any badges that the students can be awarded, if any.

After creating a tournament, educators can decide at any time to give permission to other educators to create new coding battles for that tournament.

Once the tournament is created, a notification is sent to all the students subscribed to the platform and they can choose to subscribe.

If they subscribe, they will be notified of all the upcoming battles created in the tournament.

2.2.3 Battle creation

An educator, with the permission to create a battle, can create a battle using the CKB platform by going through the following steps:
- upload the code kata, containing a textual description of the problem, the test cases and the build automation scripts
- set the minimum and the maximum members of the group
- set a registration deadline
- set a final submission deadline
- set additional configuration for scoring

At the end of the configuration, the platform notifies all the students subscribed to that tournament about the creation of a new battle.

2.2.4 Battle registration

After the creation of a battle, the students use the platform to form teams for the battle. Each student can join the battle on his/her own or by inviting other students, respecting the limit of minimum and maximum members per group.
When the registration deadline expires, the system creates a GitHub repository containing the code kata and then sends the link to all the students subscribed to the battle.

2.2.5 Battle solution evaluation

When a battle deadline expires the system performs a final evaluation about each group's solution. The first part of the evaluation is automatically done by CKB using static analysis tools. If the educator decided to personally evaluate solutions during the creation of the battle, it does so during the consolidation phase, by looking at the repository of each group for that specific battle and assigning a score.
The score is a natural number between 0 and 100 and is based on: time passed between the registration deadline and the last commit, number of test cases passed out of all test cases and the results of the measurement of the quality level of the sources, performed with the aforementioned tools.
When the educator confirms evaluations for all participants then CKB automatically generates ranking for that battle.
At the end of each battle, the platform updates the personal tournament score of each student, that is, the sum of all battle scores received in that tournament.
At each tournament end the platform takes the sum of each battle score to calculate the final score for every student and assign gamification badges.

2.2.6 Platform browsing

The system allows each type of user to browse the platform and view different information. Users can check other's profiles and see their personal information by searching their nicknames. Another action allowed is to view tournaments or battles ranking by looking at the dedicated section and gamification badges that will compare on their profiles.

2.3 User characteristics

2.3.1 Non-registered users

Non-registered users are all users who are neither logged in as a student or an educator, their actions are restricted to only signing up, registering and browsing the platform. All non-registered users can log in or sign up to become either students or educators.

2.3.2 Student

Student is a type of user that utilizes the platform to participate in tournaments and battles. In order to do that they need to have a GitHub account and be registered to CKB.

2.3.3 Educator

An educator is a user that creates tournaments and coding battles for students to solve. In order to use the functionalities the educator needs to subscribe to the platform and be logged in.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumption
- [D1] Students have a GitHub profile
- [D2] Students use CKB's API to share their code with the evaluation system
- [D3] Tests provided by the educators are correct and consistent with the problem description
- [D4] Students know how to use GitHub in particular, GitHub Actions
- [D5] Repositories who use CKB's API to submit their code actually belong to students subscribed to that specific battle
- [D6] Educators will close the "consolidation phase"

3. Specific Requirements

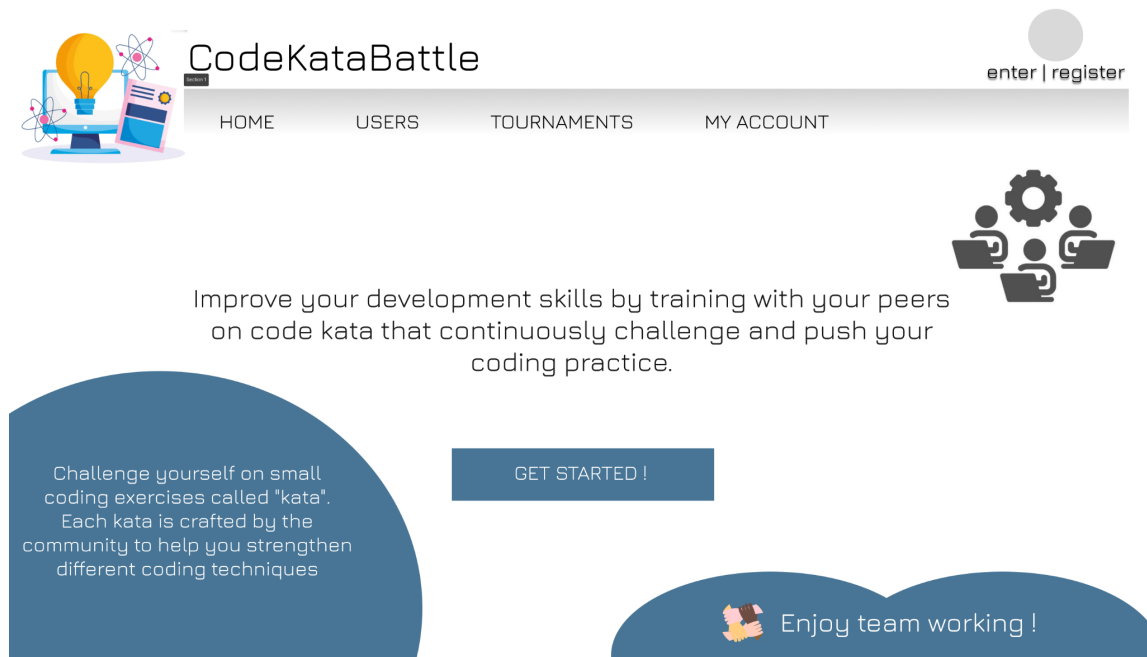## 3.1 External interface requirements

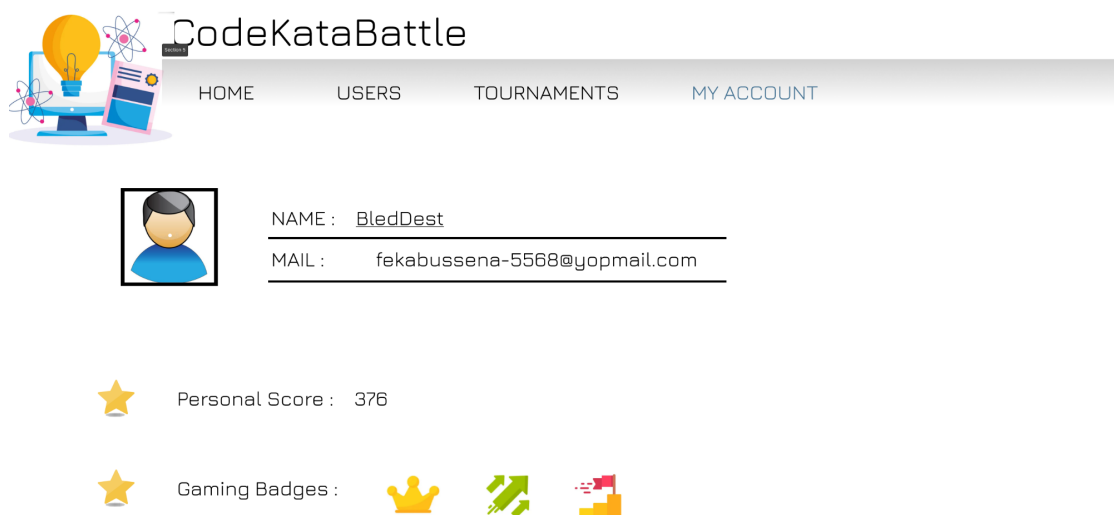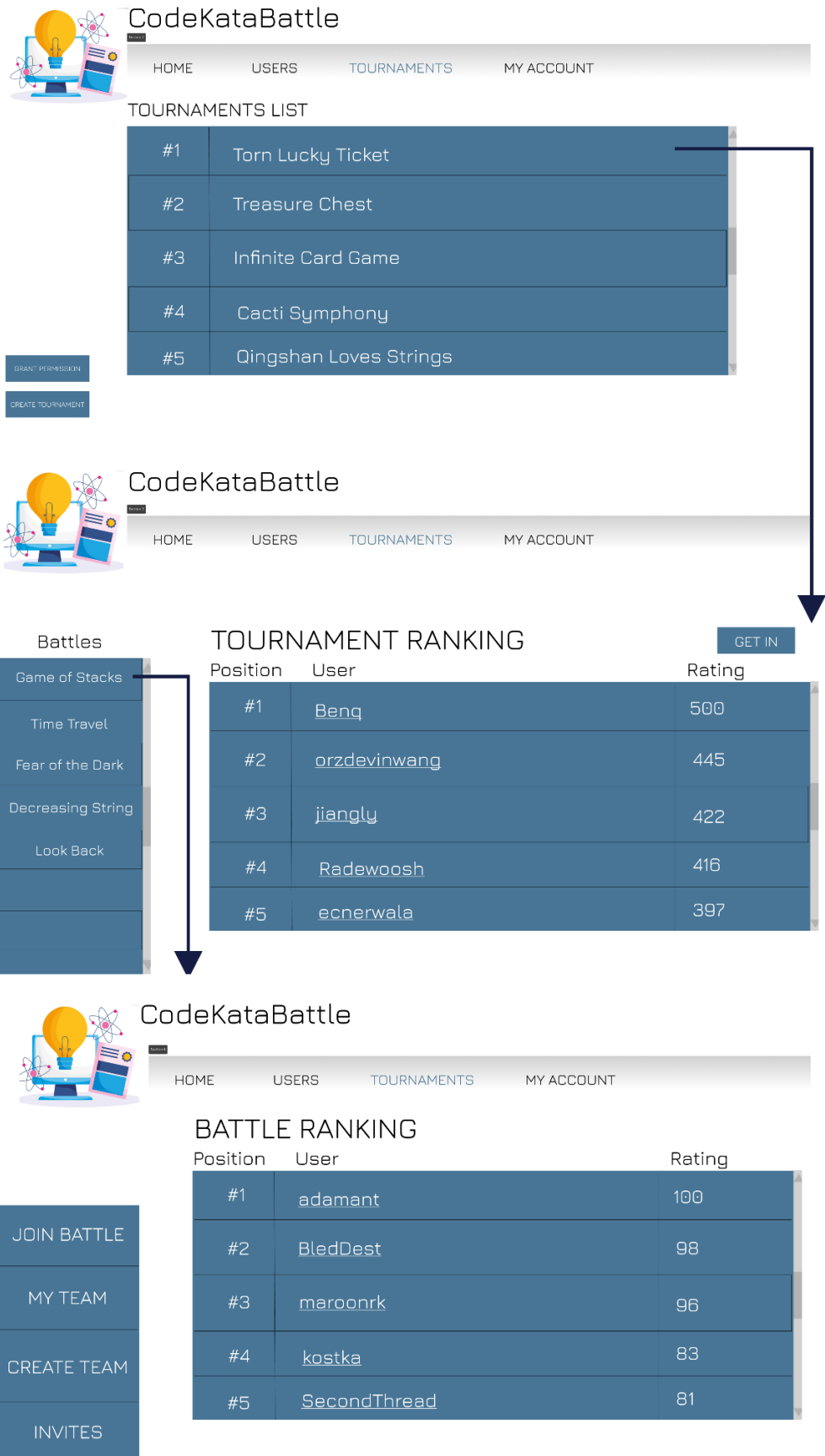### 3.1.1 User interface

Fig.1 Home page

Fig.2 Personal page

CodeKataBattle

HOME   USERS   TOURNAMENTS   MY ACCOUNT

TOURNAMENTS LIST

#1	Torn Lucky Ticket
#2	Treasure Chest
#3	Infinite Card Game
#4	Cacti Symphony
#5	Qingshan Loves Strings

GRANT PERMISSION

CREATE TOURNAMENT

CodeKataBattle

HOME   USERS   TOURNAMENTS   MY ACCOUNT

Battles

Game of Stacks
Time Travel
Fear of the Dark
Decreasing String
Look Back

TOURNAMENT RANKING

GET IN

Position	User	Rating
#1	Benq	500
#2	orzdevinwang	445
#3	jiangly	422
#4	Radewoosh	416
#5	ecnerwala	397

CodeKataBattle

HOME   USERS   TOURNAMENTS   MY ACCOUNT

BATTLE RANKING

Position	User	Rating
#1	adamant	100
#2	BledDest	98
#3	maroonrk	96
#4	kostka	83
#5	SecondThread	81

JOIN BATTLE

MY TEAM

CREATE TEAM

INVITES

Fig.3

Tournament and Battle pages

Fig.4 Battle creation page

3.1.2 Hardware interface

In order to use the CKB platform, users can use any device with access to an internet connection, but the system is better suited for desktop use.

3.1.3 Software interface

CKB provides an API that students can use to submit their solutions. In particular, students are expected to set up GitHub actions workflows to automatically execute API calls.

3.1.4 Communication interface

All communications from and to CKB are performed via REST APIs, using HTTP or HTTPS. The platform also provides an API for students to use in the github action workflows in their forks of battle repositories to make the system automatically evaluate their solutions.

3.2 Functional requirements

- [R1] The system allows users to sign up
- [R2] The system allows users to sign in
- [R3] The system allows users to browse other users profiles
- [R4] The system allows users to browse the list of tournaments
- [R5] The system allows users to browse tournament rankings
- [R6] The system allows users to browse battle rankings
- [R7] The system allows educators to create tournaments
- [R8] The system allows educators to specify a tournament registration deadline

- [R9] The system allows students to join tournaments
- [R10] The system allows educators to define gamification badges, consisting in a title and one or more rules that must be fulfilled for a student to obtain the badge
- [R11] The system requires educators to define a way to assign scores to students submission in an automated way
- [R12] The system allows educators to decide whether they have to assign personal scores to students solutions during the consolidation phase
- [R13] The system allows educators to assign a personal score during the consolidation phase if they decided to allow it when creating the battle
- [R14] The system allows the creator of a battle to terminate the consolidation phase after having evaluated all of the groups sources (if they decided to do so when creating the battle), effectively terminating the battle
- [R15] The system notifies all students subscribed to the platform whenever a new tournament is created
- [R16] The system notifies students when a battle is created if they are registered to that battle's tournament
- [R17] The system notifies students when the battle's final rankings become available
- [R18] The system notifies students when the final tournament ranking become available
- [R19] The system provides all students subscribed to a battle with that battles's code kata by notifying them with a link to that code kata's GitHub repository when a battle's registration deadline expires if they are subscribed
- [R20] The system allows educators to create coding battles for a specific tournament if they either have been given permission from the tournament creator to do so or they created that tournament
- [R21] The system allows educators to grant permission to create new coding battles for a tournament they have created to other educators
- [R22] The system allows students to create a group for each battle
- [R23] The system allows students to invite other students to their group for a battle
- [R24] The system allows students to join a group for a battle
- [R25] The system allows students to compete in a coding battle if, when the registration deadline expires, they are part of a team composed by a number of students within the boundaries defined by that battle's creator
- [R26] The system allows educators to specify battle deadlines when creating a new battle
- [R27] The system allows educators to specify boundaries for the number of students in each group when creating a new battle
- [R28] The system allows students and educators to see evolving rankings before a code battle has reached its submission deadline
- [R29] The system allows educators to upload code katas when creating a new battle by providing a textual description, a set of test cases and build automation scripts
- [R30] The system provides an API to allow users to submit their solution to a code battle, triggering the system to run automated tests to analyze the students code
- [R31] The system allow educators to create new variables to use during the definition of the rules for gamification badges, using a pseudo-language

3.2.1 Use case diagrams

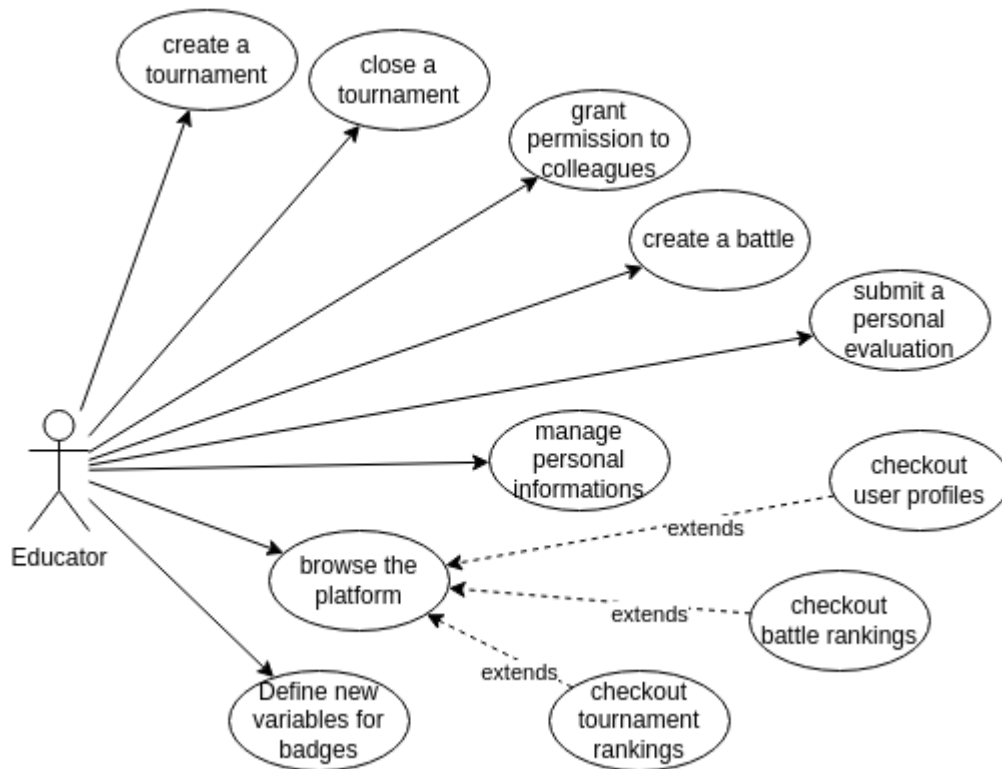The system identifies 3 types of users, which can perform a different set of operations:
- **Non-registered user**: a user that is not logged in can perform few operations on the platform, they can become either student or educator by signing in
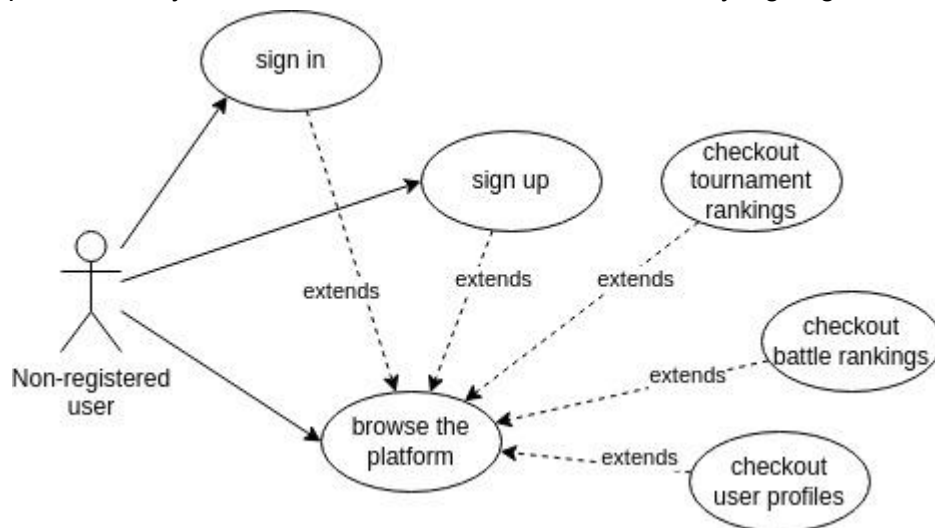
- **Student**: this type of user uses the platform to solve coding challenges with peers and is assumed to be logged in
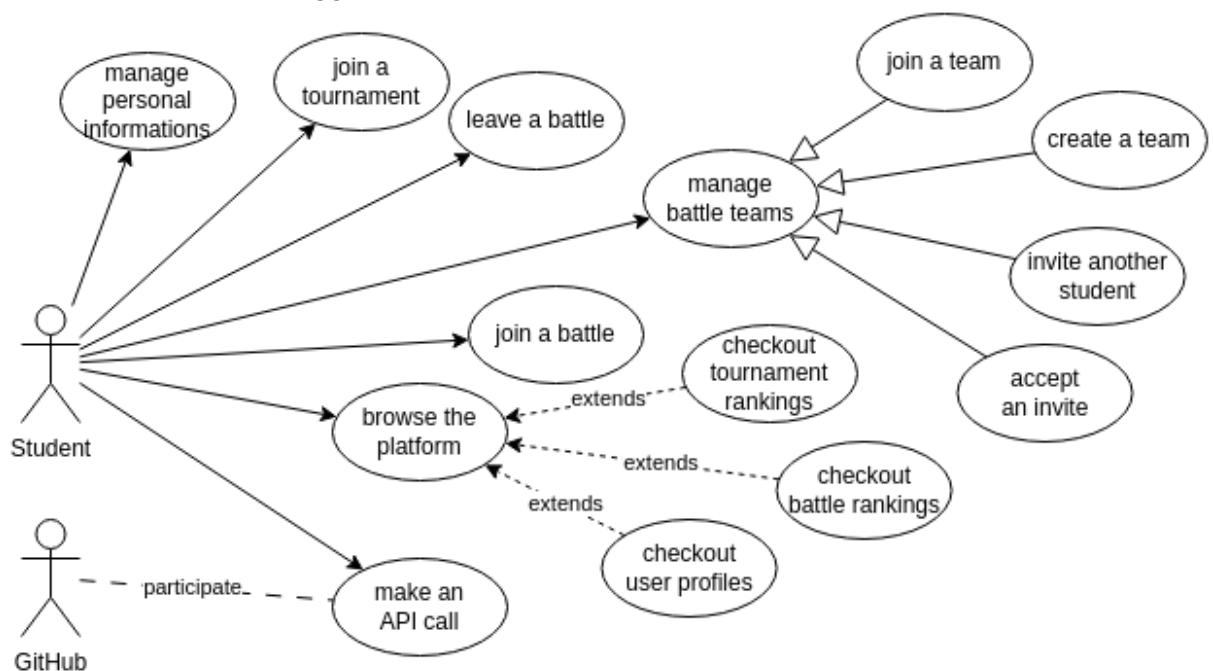
● **Educator**: this type of user uses the platform to create tournaments and challenges for other students to solve and is assumed to be logged in

3.2.2 Use cases

Name	Sign in
ID	UC1
Actors	Non-registered users
Entry condition	The actor is connected to the platform and wants to log in
Event flow	1. The actor navigates to the signin page 2. The actor fills the signin form with email and password 3. The actor submits the form 4. The system singes the actor in
Exit condition	The actor is signed in
Exceptions	● Credentials are not correct ○ The system displays an error message informing the actor that the provided information is not correct

Name	Sign up
ID	UC2
Actors	Non-registered users
Entry condition	The actor is connected to the platform and wants to create an account

Event flow	1. The actor navigates to the signup page
	2. The actor fills the signup form providing the following:
	   ○ Full name
	   ○ Password
	   ○ Email
	   ○ Selection between educator/student profile
	3. The actor submits the form
	4. The system creates a new account with the informations the actor provided

Exit condition	The actor has created a new account

Exceptions	● The provided email is already linked to a registered account
	   ○ The system displays an error message informing the actor that there already is an account registered using that email
	● The actor tries to send the form without providing all the requested information
	   ○ The system displays an error message informing the actor that the provided information is not complete

Name	Manage personal informations
ID	UC3
Actors	Students, Educators
Entry condition	The actor is logged in and wants to change some of his/her personal informations
Event flow	1. The actor navigates to the manage personal informations page 2. The actor can change any personal information using a form 3. The actor submits the form 4. The system updates the actor's personal information 5. The system redirects the actor to the landing page
Exit condition	The actor's personal informations are updated
Exceptions	● The actor updates his email with one already associated with another registered account ○ The system displays an error message informing the actor that the provided email is taken

Name	Browse the platform (checkout tournament rankings)
ID	UC4.1
Actors	Non-registered users, Students, Educators
Entry condition	The actor is using the platform and wants to have a look at the rankings for a certain tournament
Event flow	1. The actor navigates to the rankings of a tournament 2. The system shows the actor the latest available rankings for that tournament
Exit condition	The actor has been shown the desired tournament rankings
Exceptions	● The client tries to query the rankings of a tournament which is unavailable or nonexistent ○ The system displays an error message informing the actor that the rankings for that tournament cannot be found

Name	Browse the platform (checkout battle rankings)
ID	UC4.2
Actors	Non-registered users, Students, Educators
Entry condition	The actor is using the platform and wants to have a look at the rankings for a certain battle
Event flow	1. The actor navigates to the rankings of a battle 2. The system shows the actor the latest available rankings for that battle
Exit condition	The actor has been shown the desired battle rankings
Exceptions	● The client tries to query the rankings of a battle which is unavailable or nonexistent ○ The system displays an error message informing the actor that the rankings for that tournament cannot be found

Name	Browse the platform (checkout user profiles)
ID	UC4.3
Actors	Non-registered users, Students, Educators
Entry condition	The actor wants to have a look at some other user's information

Event flow	1. The actor navigates to another user's profile page
	2. The system shows the actor the user's informations, which depend on the type of account the user has:
	    a. If the profile belongs to a student, the actor will be able to see the user's name, email and any awarded gamification badges
	    b. If the profile belongs to an educator, the actor will be able to see the user's name and email

Exit condition	The actor has been shown the desired profile
Exceptions	● The actor tries to query a non-existing user profile ○ The system displays an error message informing the actor that requested user profile does not exist

Name	Join a tournament
ID	UC5
Actors	Student
Entry condition	The actor is logged in as a student and wants to join a tournament to solve coding battles
Event flow	1. The actor navigates to a tournament page 2. The actor registers for the tournament by clicking the "register" button 3. The system registers the actor to the tournament
Exit condition	The user is registered to the tournament

Exceptions	● The user was already registered to the tournament
	    ○ The system displays an error message informing the actor that he is already registered for that tournament
	● The tournament registration deadline is expired
	    ○ The system displays an error message informing the actor that the registration deadline has expired, so there is no way for them to join that tournament

Name	Manage battle teams (join a team)
ID	UC6.1
Actors	Student
Entry condition	The actor is logged in as a student and wants to join an existing battle team
Event flow	1. The actor navigates to a coding battle 2. The actor navigates to the team list by clicking the "teams" tab in the battle page 3. The actor selects a team that he/she wants to join 4. The system registers the actor to the desired team
Exit condition	The actor is part of the desired team for a coding battle

Exceptions	● The registration deadline for the battle is expired
	  ○ The system displays an error message informing the actor that the registration deadline for that battle has expired and he can no longer join a team
	● The chosen team already has the maximum number of students
	  ○ The system displays an error message informing the actor that the selected team already has reached the maximum number of students for that battle
	● The actor is already part of another team
	  ○ The system informs the actor that they are already registered to the coding battle in with a different team and have to leave their current team if they want to join the new one
	  ○ The actor can choose to leave their team and, by doing so they request the system to join the new team
	  ○ If the new team is full their action will be voided, otherwise they will be added to the new team and removed from the old one
	  ○ If the old team is now empty, the system deletes it
	● The actor is already part of the selected team
	  ○ The system displays an error message informing the actor that they are already part of that team

Name	Manage battle teams (invite another student)
ID	UC6.2
Actors	Student
Entry condition	The actor wants to invite another student to their battle team
Event flow	1. The actor navigates to a coding battle page 2. The actor navigates to it's own team page by clicking the team tab 3. The actor clicks the "invite friend" button 4. The actor inserts the email of a registered student 5. The actor confirms his action by clicking the "invite" button 6. The system sends the student specified by the actor an invite to join the actor's team for that coding battle
Exit condition	The student specified by the actor has received an invite to join the actor's team for that coding battle

Exceptions	● The actor requested to invite a non-existing student
	  ○ The system displays an error message informing the actor that the provided email does not belong to that of a registered student
	● The actor is not registered to the coding battle
	  ○ The system displays an error message informing the actor that they have to be registered to a coding battle to invite someone to a team

Name	Manage battle teams (create a team)
ID	UC6.3
Actors	Student
Entry condition	The actor wants to create a new team for a coding battle to which he is already registered
Event flow	1. The actor navigates to a coding battle page 2. The actor creates a new team for that battle by clicking on the "create new team" button 3. The system creates a new team for the actor for that coding battle 4. The system registers the actor to the new team
Exit condition	The actor is part of a new team for the battle

Exceptions	● The battle registration deadline has expired
	  ○ The system displays an error message informing the actor that the registration deadline for that battle has expired, therefore no new teams can be created
	● The actor is already part of another team
	  ○ The system displays an error message informing the actor that they are already registered to the coding battle in with a different team and have to leave their current team if they want to join the new one
	  ○ The actor can choose to leave their team and, by doing so they request the system to create and join the new team
	  ○ If the old team is now empty, the system deletes it

Name	Manage battle teams (accept an invite)
ID	UC6.4
Actors	Student
Entry condition	The actor wants to accept an invite to a coding battle
Event flow	1. The actor navigates to a coding battle page 2. The actor navigates to its invites page for that battle by clicking on the "teams" tab 3. The actor is shown a list of invites from other students requesting for them to join their team 4. The actor join a team by clicking on the "accept" button of an invite 5. The system adds the actor to the team for which they accepted an invite
Exit condition	The actor is part of the team for which they accepted an invite
Exceptions	● The actor is already part of another team ○ The system informs the actor that they are already registered to the coding battle in with a different team and have to leave their current team if they want to join the new one ○ The actor can choose to leave their team and, by doing so they request the system to join the new team ○ If the new team is full their action will be voided, otherwise they will be added to the new team and removed from the old one ○ If the old team is now empty, the system deletes it

Name	Join a battle
ID	UC7
Actors	Student
Entry condition	The actor wants to join a coding battle
Event flow	1. The actor navigates to a coding battle page 2. The actor join the battle by clicking the "join" button 3. The system creates a new team for the actor 4. The system adds the actor the new team
Exit condition	The actor is part of a new team for the selected battle
Exceptions	● The actor is already registered to the battle ○ The system displays an error message informing the actor they are already registered to that coding battle

Name	Leave a battle
ID	UC8
Actors	Student
Entry condition	The actor wants to leave a coding battle
Event flow	1. The actor navigates to a coding battle page 2. The actor leaves a battle by clicking on the "leave battle" button 3. The system removes the actor from its current team 4. The system deletes the team the actor was in if they were the only member
Exit condition	The actor is not registered for the battle
Exceptions	● The actor is not registered to that battle ○ The system displays an error message informing the actor that they are not registered for that battle

Name	Make an API call
ID	UC9
Actors	Student (may be more than one, acting as a team)
Entry condition	A team of students wants to submit a solution to a battle
Event flow	1. The team pushes a commit to the main branch of its repository 2. The GitHub action they set up is triggered and calls CKB's API 3. The system fetches the sources of the team repository 4. The system runs automatic code evaluations using static analysis 5. The system analyzes the team's solution using static analysis 6. The system assigns a score to the team's solution
Exit condition	The team's score is updated considering the evaluation of their solution

Exceptions	● The team is not registered to the battle for which they called the API
	   ○ The system aborts the analysis and notifies all the members of the team who called the API that they tried to submit a solution for a battle for which they are not registered
	● The submission deadline is expired
	   ○ The system aborts the analysis and notifies all the members of the team who called the API that they tried to submit a solution for a battle for which the submission deadline has expired
	● The submission phase has not started yet
	   ○ The system aborts the analysis and notifies all the members of the team who called the API that they tried to submit a solution for a battle for which the submission phase has not started yet

Name	Create a battle
ID	UC10
Actors	Educator
Entry condition	The educator has created the tournament in which he wants to create a battle or has been given permission to create a battle by the tournament's creator
Event flow	1. The actor navigates to the tournament page 2. The actor starts the process of creating a new battle by clicking on the "create new battle" button 3. The actor is shown a battle creation form 4. The actor uploads a code kata, containing a textual description of the problem, a set of test cases and build automation scripts 5. The actor fills the form providing: boundaries for the number of students in groups, a registration deadline and a submission deadline 6. The actor fills the form configuring the scoring and deciding whether they will have to assign a personal score to students solutions 7. After providing all the requested information, the educator clicks "create battle" to submit the form 8. The system adds the battle to the set of battles for that tournament
Exit condition	The new battle has been added to the set of battles in the tournament
Exceptions	• The submission deadline is before the registration deadline ○ The system displays an error message informing the actor that they provided a wrong set of deadlines • The submission deadline or the registration deadline are before the current time (in the past) ○ The system displays an error message informing the actor that they must have made a mistake while providing the deadlines as time travel is not possible • The boundaries for the number of team members do not form a valid interval ○ The system displays an error message informing the actor that the boundaries they provided for the number of allowed members for each team are not coherent • The actor has not been given permission by the tournament creator to create battles (or is not the tournament creator) ○ The system displays an error message informing the actor that they cannot create a battle for that tournament because they have not been granted permission from the creator of the tournament to do so

Name	Submit a personal evaluation
ID	UC11
Actors	Educator
Entry condition	The educator decided that scoring would also be determined by its own evaluation when creating a battle and the battle is now in the consolidation phase
Event flow	1. The educator navigates to the page of a coding battle 2. The educator navigates to the team list by clicking the "teams" tab in the battle page 3. The educator clicks on the "assign evaluation" button 4. The educator is presented an input box where to insert the score and a "submit" button to click to submit the score 5. The educator inserts a score in the form of a natural number between 0 and 100 6. The educator clicks the "submit" button 7. The system updates the team score accounting for the educator's evaluation
Exit condition	The score is updates with the educator's personal evaluation

Exceptions	● The educator did not include the possibility to assign personal scores to students solutions when creating the battle
	  ○ The system displays an error message informing the actor that the educator did not include the possibility to assign personal scores to students solutions when creating the battle
	● The battle is not in the consolidation phase
	  ○ The system displays an error message informing the actor that the battle is not in its consolidation phase
	● The actor did not provide a valid score
	  ○ The system displays an error message informing the actor that they provided an invalid score and the that the assigned score has to be a natural number between 0 and 100
	● The actor did not create that battle
	  ○ The system displays an error message informing the actor that only who created the battle can submit a personal evaluation

| Name	Create a tournament
ID	UC12
Actors	Educator
Entry condition	The educator wants to create a tournament for students to solve coding problems in the form of battles
Event flow	1. The educator navigates to the tournaments page 2. The educator navigates to the tournament creation page by clicking the "create new tournament" button 3. The educator is shown a form to set up the tournament 4. The educator provides a registration deadline for the tournament 5. The educator defines any gamification badges (they can also decide to not define any) 6. The educator submits the form by clicking the "create tournament" button 7. The system creates the tournament 8. The system notifies all students registered to the platform about the creation of a new tournament
Exit condition	A new tournament is created and the students registered to the platform have been notified
Exceptions	● The educator tries to define a rule for a gamification badge using the wrong syntax ○ The system displays an error message informing the actor that they made a syntax error while defining the rule

Name	Close a tournament
ID	UC13
Actors	Educator
Entry condition	The educator wants to terminate a tournament
Event flow	1. The educator navigates to the tournament page 2. The educator clicks on the "close tournament" button 3. The system computes the tournament rankings 4. The system awards the gamification badges 5. The system notifies all the students about the publication of the final ranking 6. The system closes the tournament
Exit condition	The gamification badges are awarded, tournament rankings are computed and students have been notified about them. The tournament is closed.

Exceptions	● The user is not the educator who created the tournament
	  ○ The system displays an error message informing the actor that they have to the tournament creator to close it
	● Not all battles have finished, i.e., they are still ongoing or have not gone through the consolidation phase
	  ○ The system displays an error message informing the actor that there are still ongoing battles

| Name	Grant permission to colleagues
ID	UC14
Actors	Educator
Entry condition	The educator wants to allow another educator to create battle for a tournament
Event flow	1. The educator navigates to the tournament page 2. The educator clicks on the "add collaborators" button 3. The educator is presented an input box where to insert the email of the educator they want to add as a collaborator and an "invite" button 4. The educator inserts the email of the educator they want to grant access to create battles for that tournament to 5. The educator clicks on the "invite" button 6. The system grants the invited educator permission to create battles for that tournament 7. The system notifies the invited educator that they have been granted permission to create battles for the tournament
Exit condition	The invited educator has been granted permission to create battles for that tournament and has been notified about it

Exceptions	● The actor is not the creator of the tournament
	    ○ The system displays an error message informing the actor that they have to be the creator of the tournament if they want to allow an educator to create battles for it
	● The provided email does not belong a registered educator account
	    ○ The system displays an error message informing the actor that the email they provided does not belong to a registered educator account

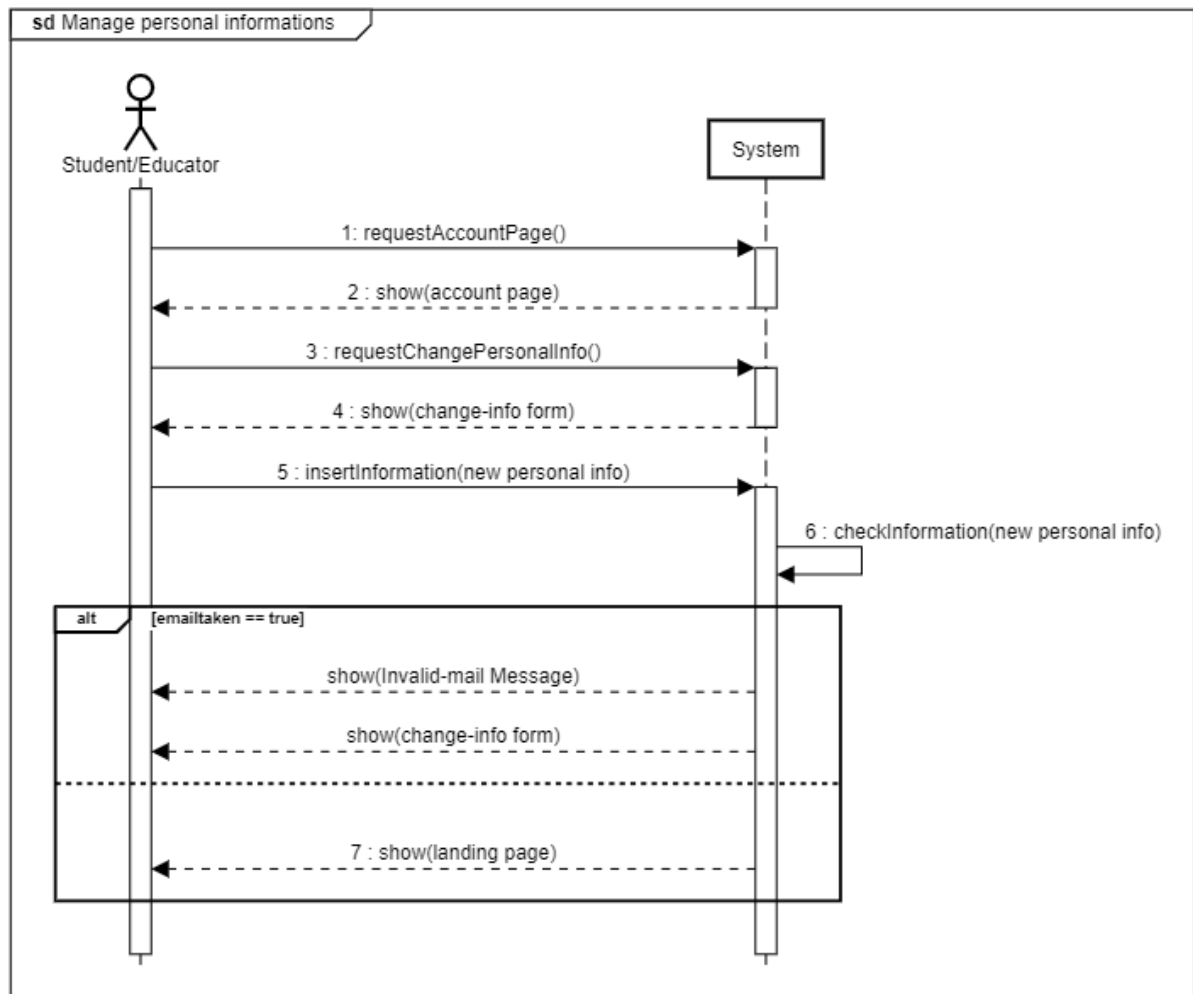| Name	Define new variables for badges
ID	UC15
Actors	Educator
Entry condition	The educator is creating a tournament and wants to define a new variable to be used when defining badges
Event flow	1. During the process of creating a tournament, the educator clicks on the button to create new badges 2. The educator clicks on the button "create new variable" 3. The educator inputs the name of the variable 4. The educator uses the pseudo-language requested by the system to define what data the new variable will provide 5. The system creates the new variable
Exit condition	The defined variable is created and can be used to create new badges
Exceptions	● The actor did not use the language correctly 　○ The system displays an error message informing the actor that they have made a syntax error when defining the new variable

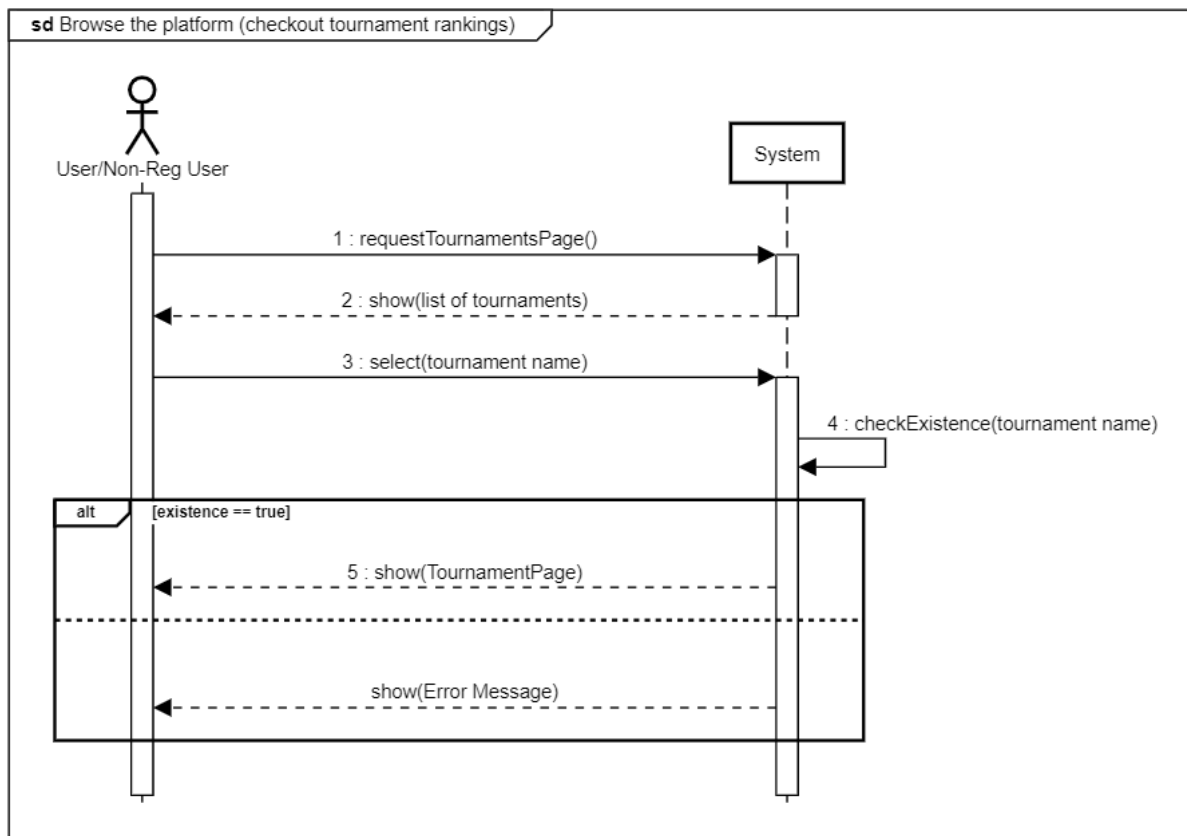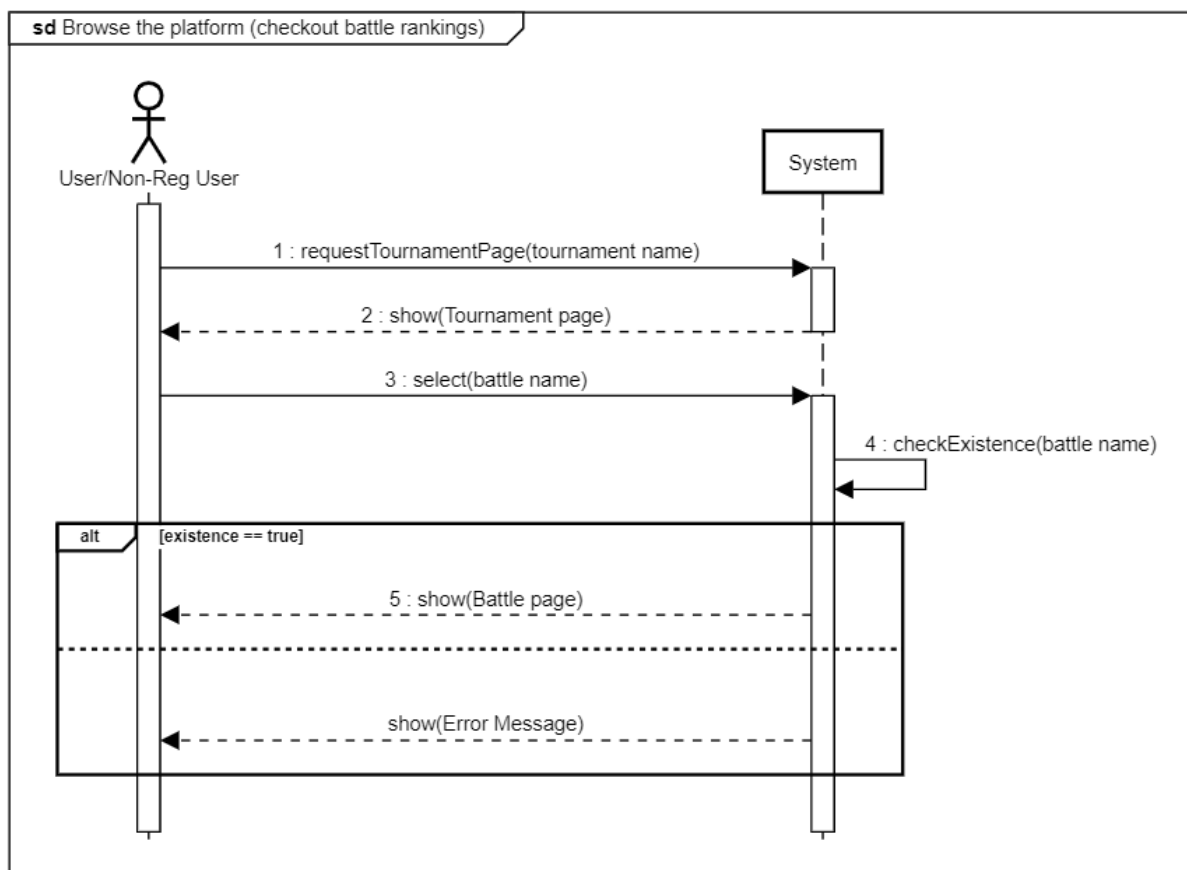## 3.2.3 Sequence diagrams

[UC1] Sign in
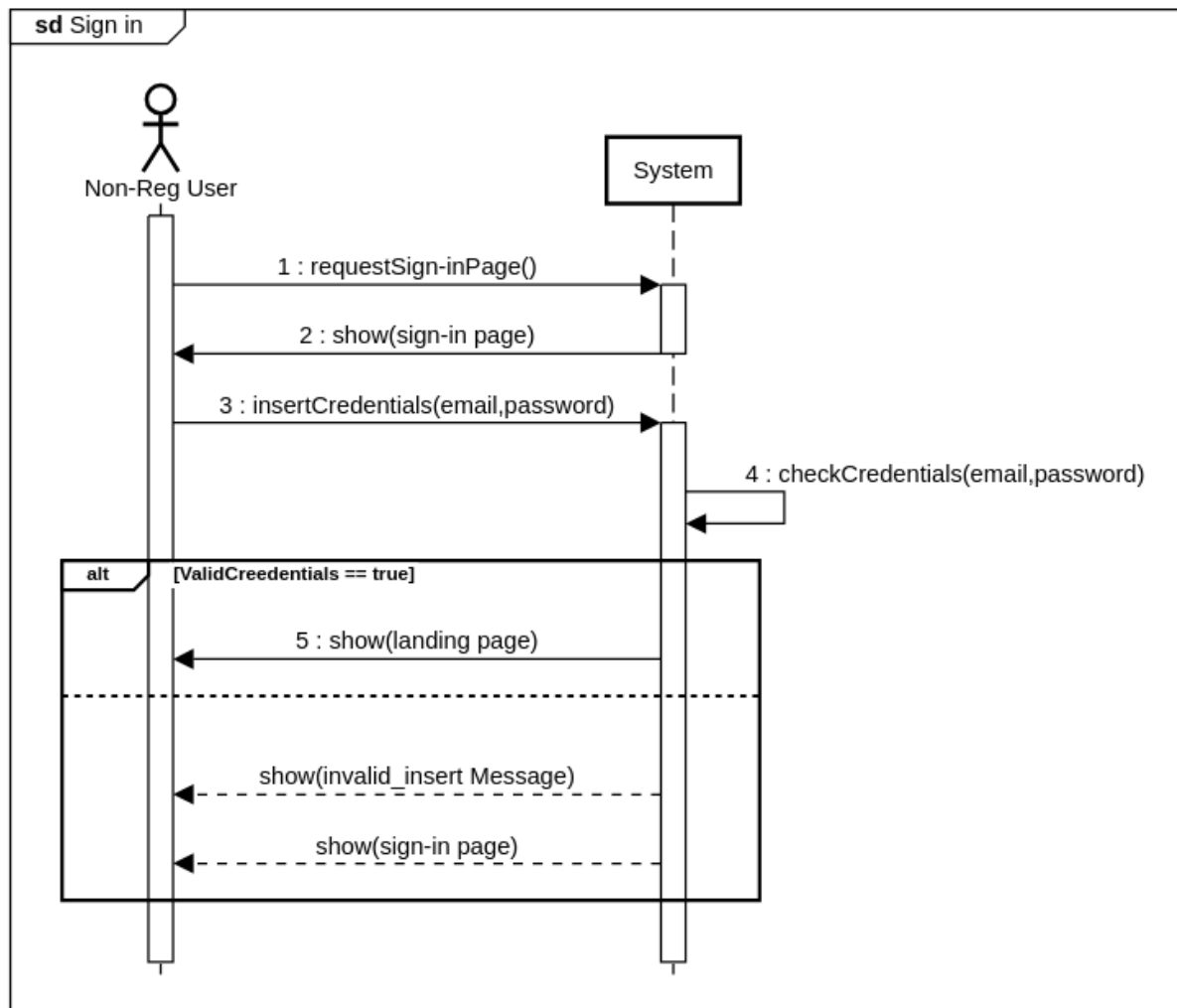
[UC2] Sign up
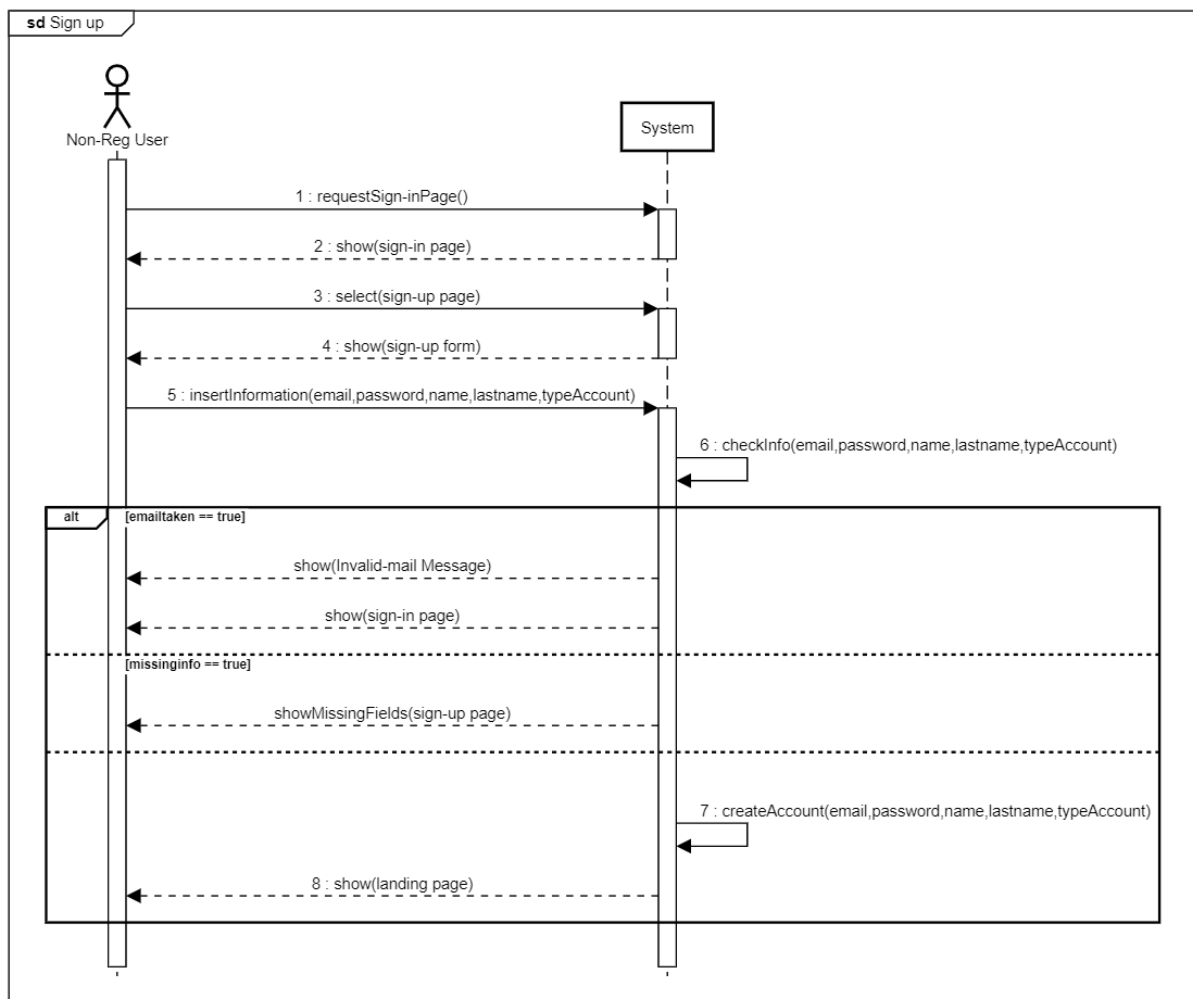
[UC3] Manage personal informations

[UC4.1] Browse the platform (checkout tournament rankings)

sd Browse the platform (checkout tournament rankings)

User/Non-Reg User → System

1 : requestTournamentsPage()

2 : show(list of tournaments)

3 : select(tournament name)

4 : checkExistence(tournament name)

alt [existence == true]

5 : show(TournamentPage)

------

show(Error Message)

[UC4.2] Browse the platform (checkout battle rankings)

sd Browse the platform (checkout battle rankings)

User/Non-Reg User → System

1 : requestTournamentPage(tournament name)

2 : show(Tournament page)

3 : select(battle name)

4 : checkExistence(battle name)

alt [existence == true]

5 : show(Battle page)

------

show(Error Message)

[UC4.3] Browse the platform (checkout user profiles)

sd Browse the platform (checkout user profiles)

User/Non-Reg User

System

1 : requestUserPage(username)

2 : checkExistence(username)

alt [existence == true]

3 : show(User page)
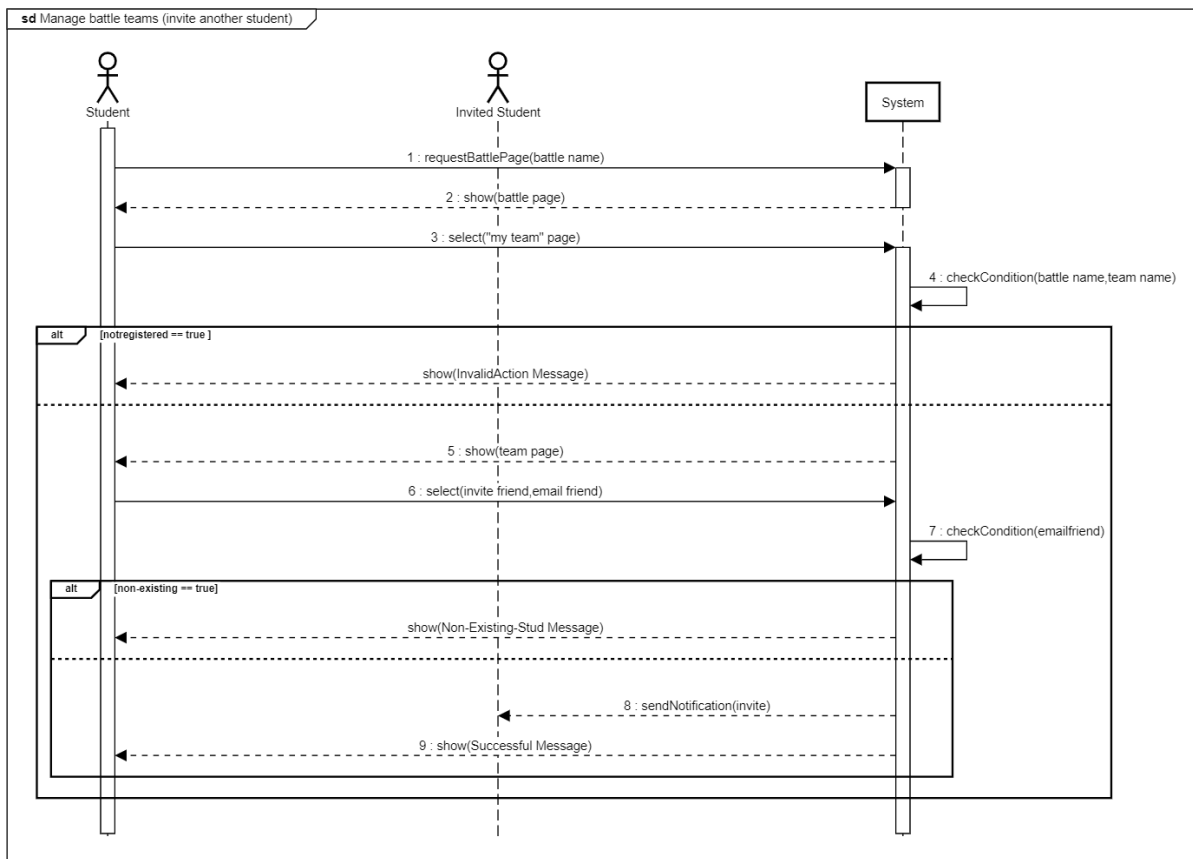
show(Inesistent-User Message)
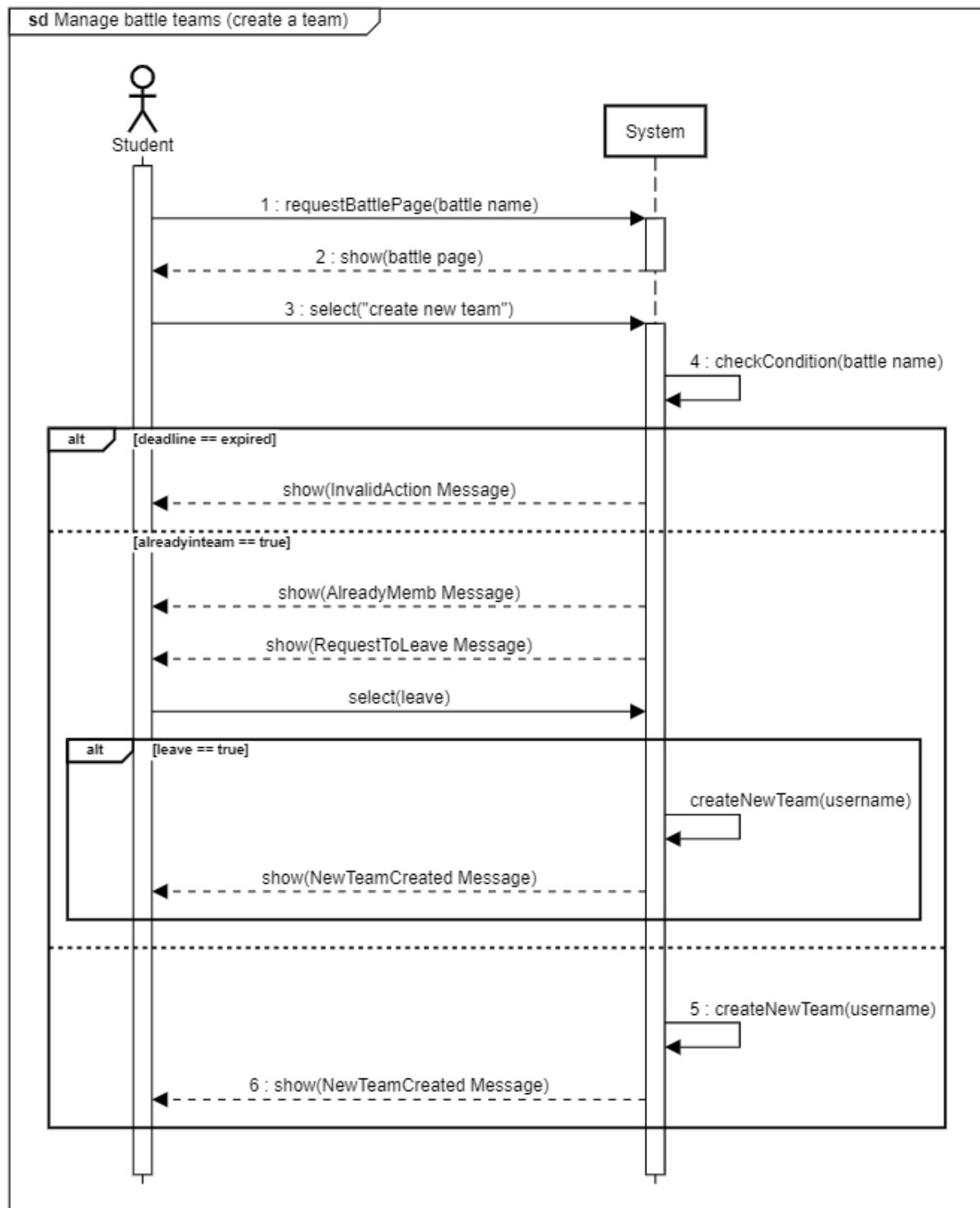
[UC5] Join a tournament

[UC6.1] Manage battle teams (join a team)
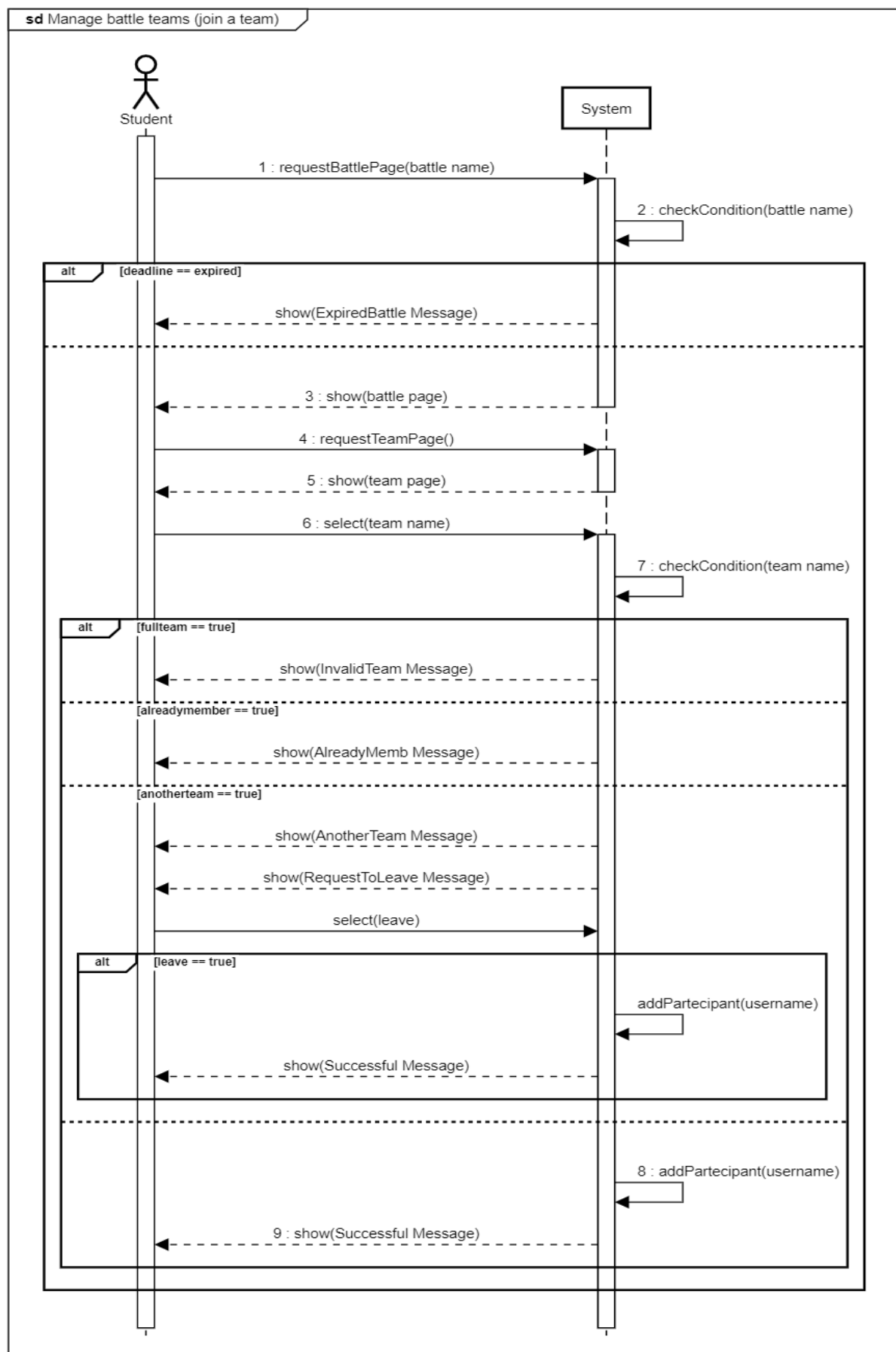
[UC6.2] Manage battle teams (invite another student)

[UC6.3] Manage battle teams (create a team)

[UC6.4] Manage battle teams (accept invite)
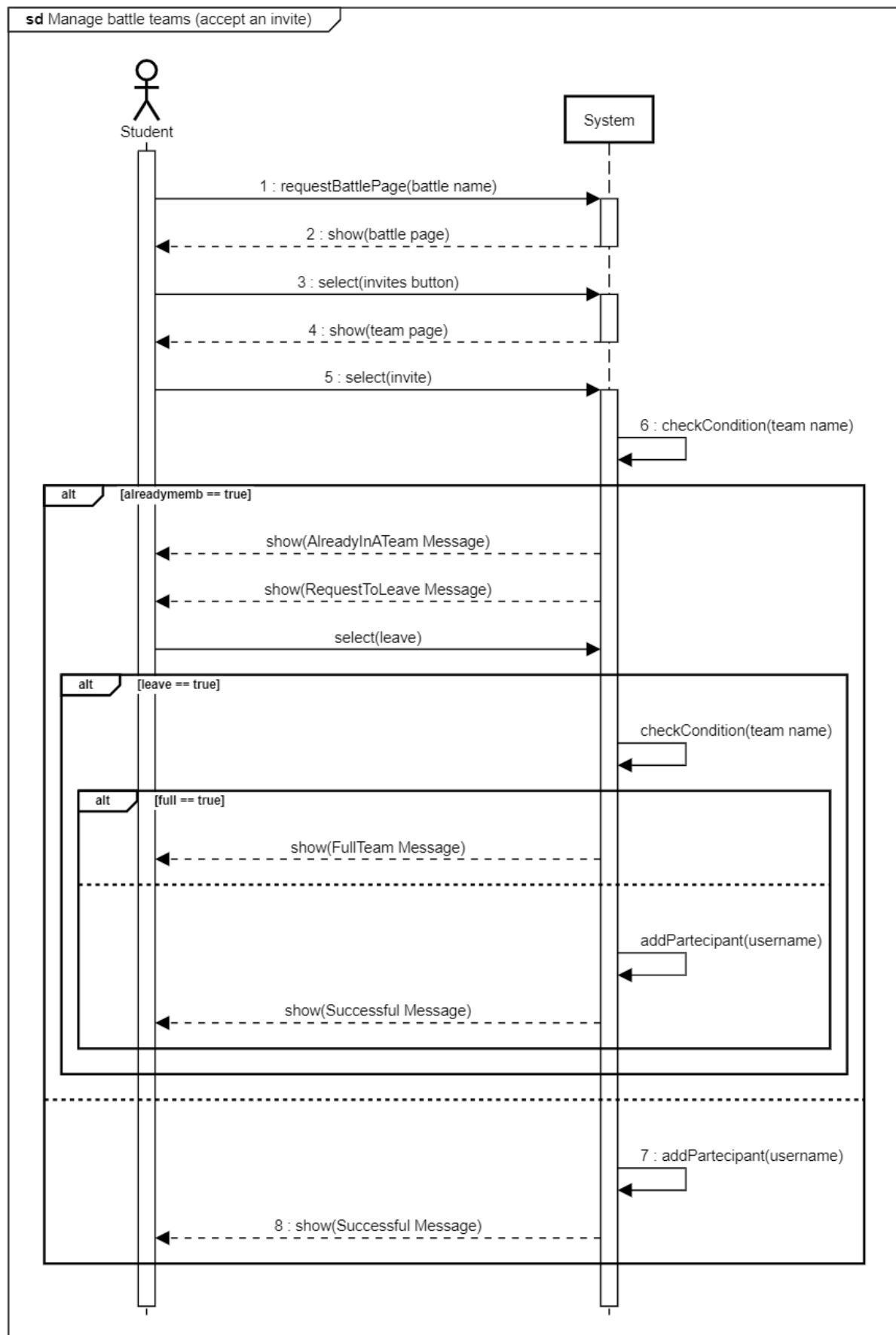
[UC7] Join a battle
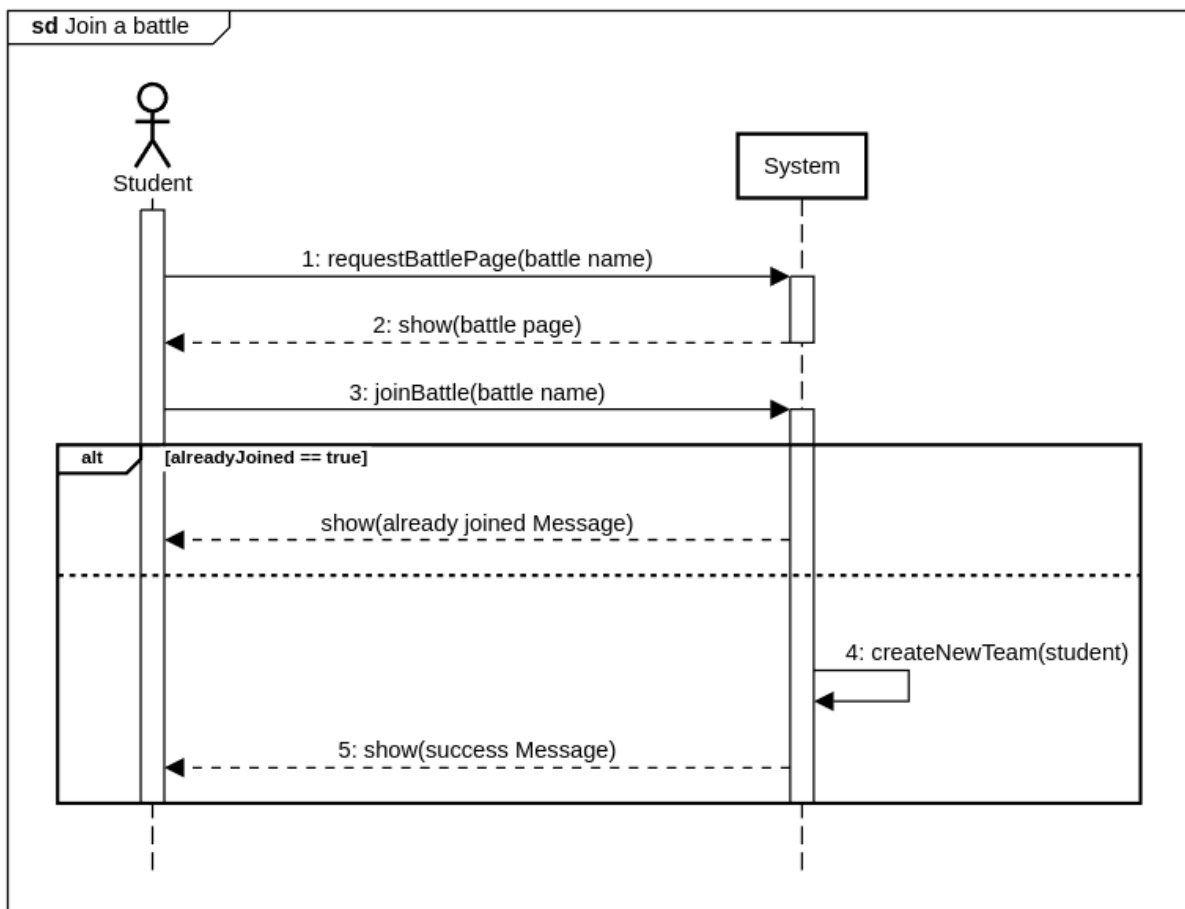
[UC8] Leave a battle
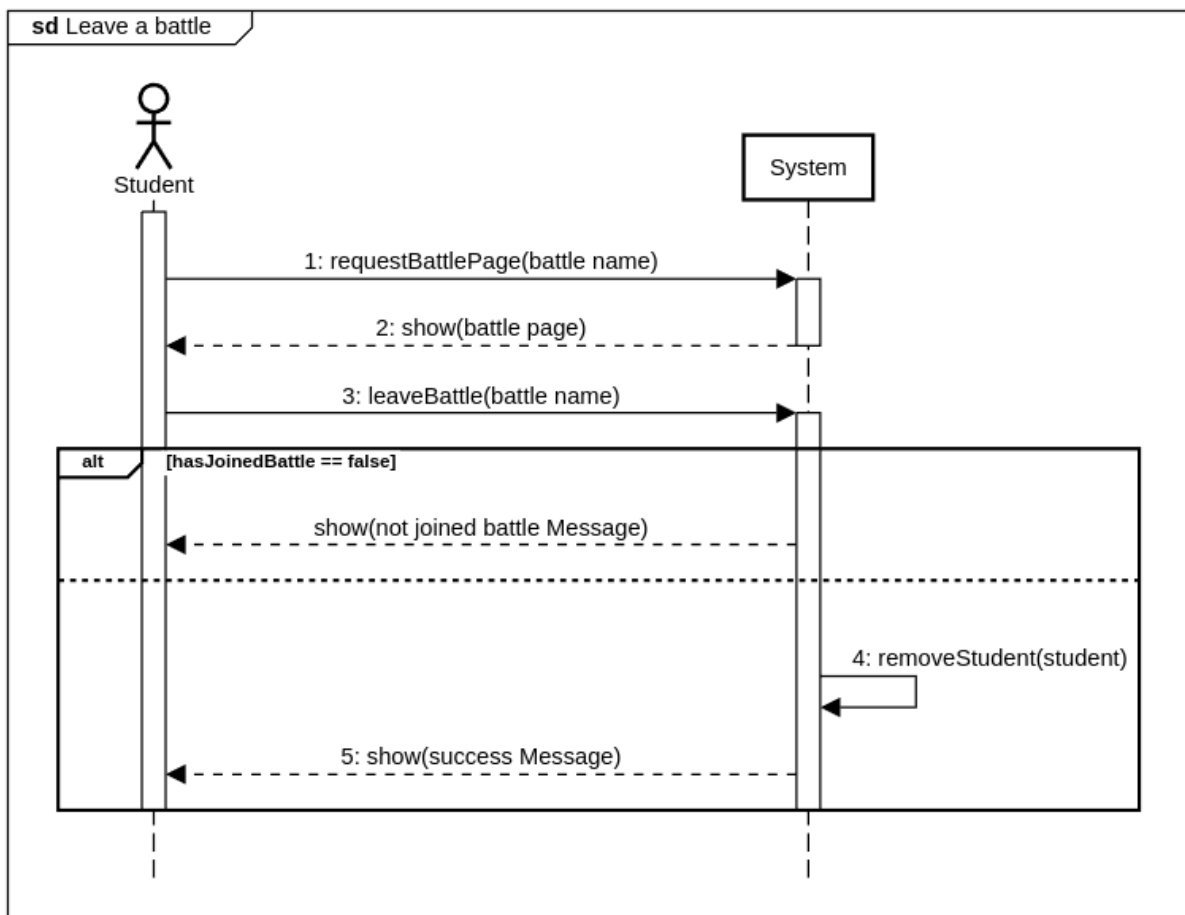
[UC9] Make an API call
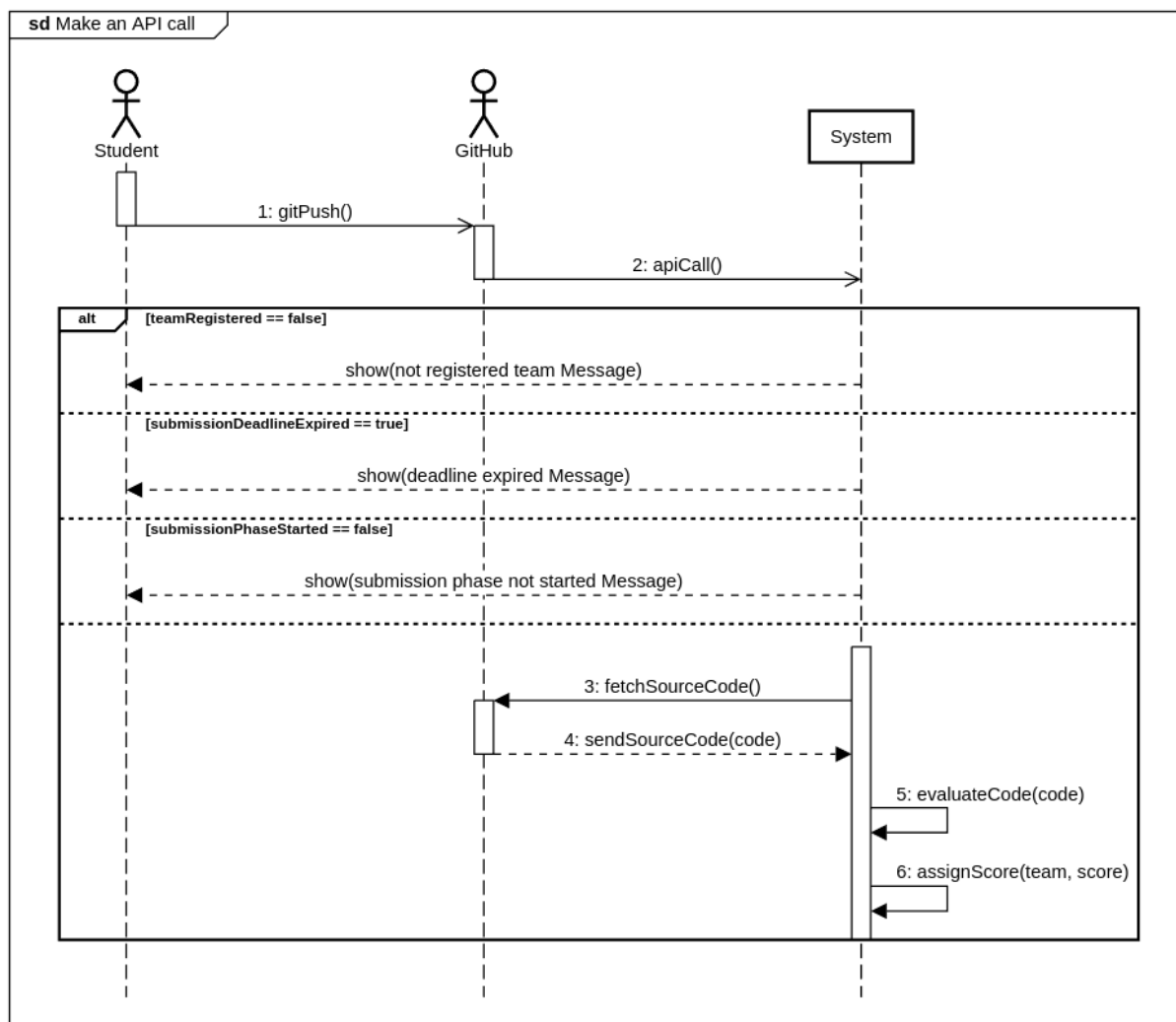
[UC10] Create a battle

[UC11] Submit a personal evaluation

sd Submit a personal evalutation

Educator

System

1: requestBattlePage(battle name)

2: show(battle page)

3: requestTeamsPage()

4: show(teams page)

5: assignEvaluation(score, team)

alt [battleInConsolidationPhase == false]

show(not consolidation phase Message)

[personalEvaluationIncluded == false]

show(invalid option Message)

[score < 0 or score > 100]

show(invalid score Message)

6: show(success Message)
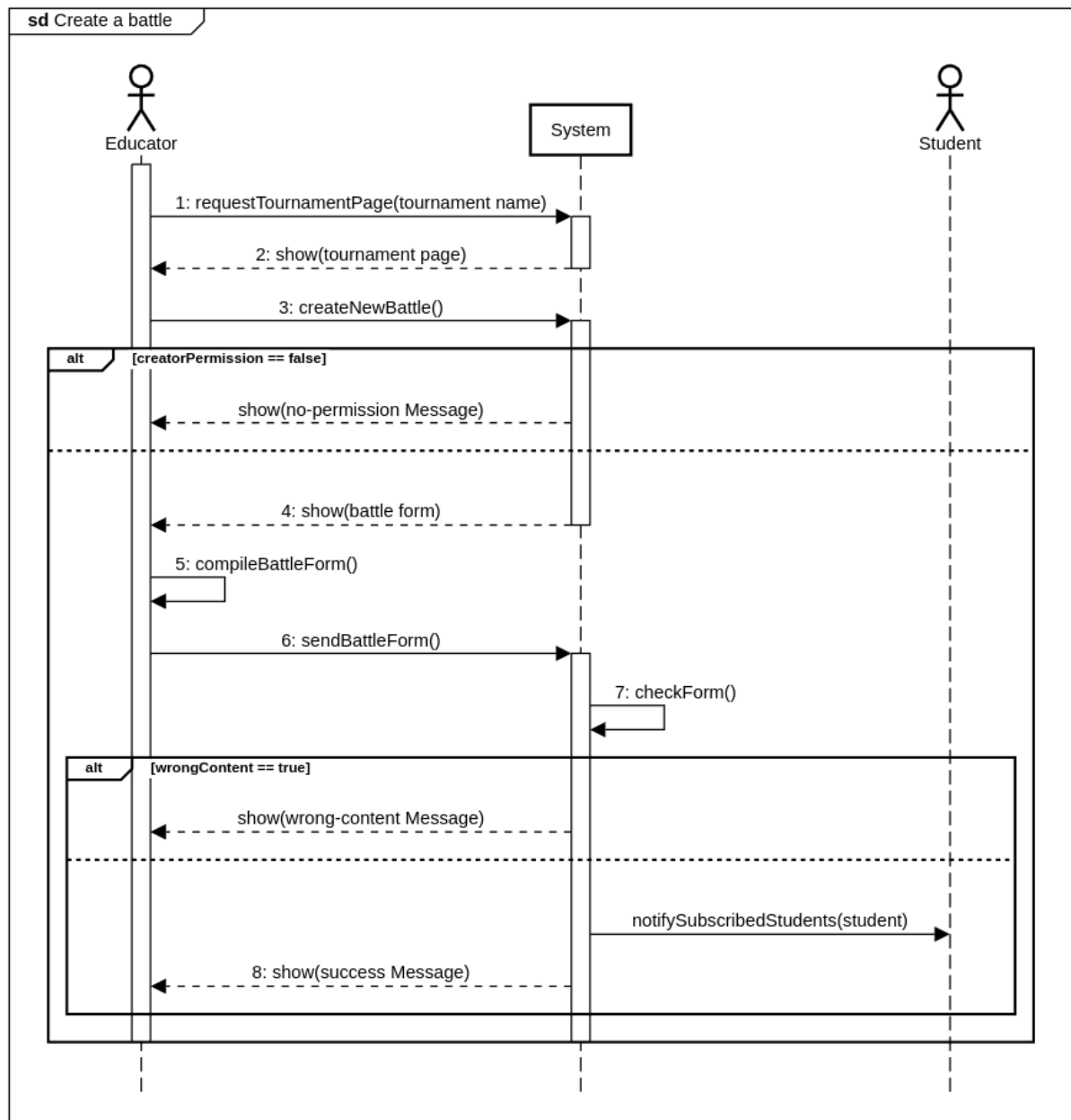
[UC12] Create a tournament

[UC13] Close a tournament

sd Close a Tournament

Educator → System: 1: requestTournamentPage(tournament name)
System --> Educator: 2: show(tournament page)
Educator → System: 3: closeTournament()

alt [creator == false]
System --> Educator: show(not-creator Message)

[allBattlesTerminated == false]
System --> Educator: show(battles not finished Message)

4: calculateRankAwards(rank, awards)
System → Student: 5: awardStudent(student)
System → Student: 6: notifyStudentEndTournament(student)
System --> Educator: 7: show(success Message)

[UC14] Grant permission to colleagues

[UC15] Define new variables for badges

sd Define new variable for badges

Educator

System

1: createBadge()

2: show(badge page)

3: createVariable()

4: show(variable page)

5: defineVariable()

6: sendVariableDefinition()

7: checkDefinition()

alt [syntaxError == true]

show(syntax error Message)

8: show(success Message)
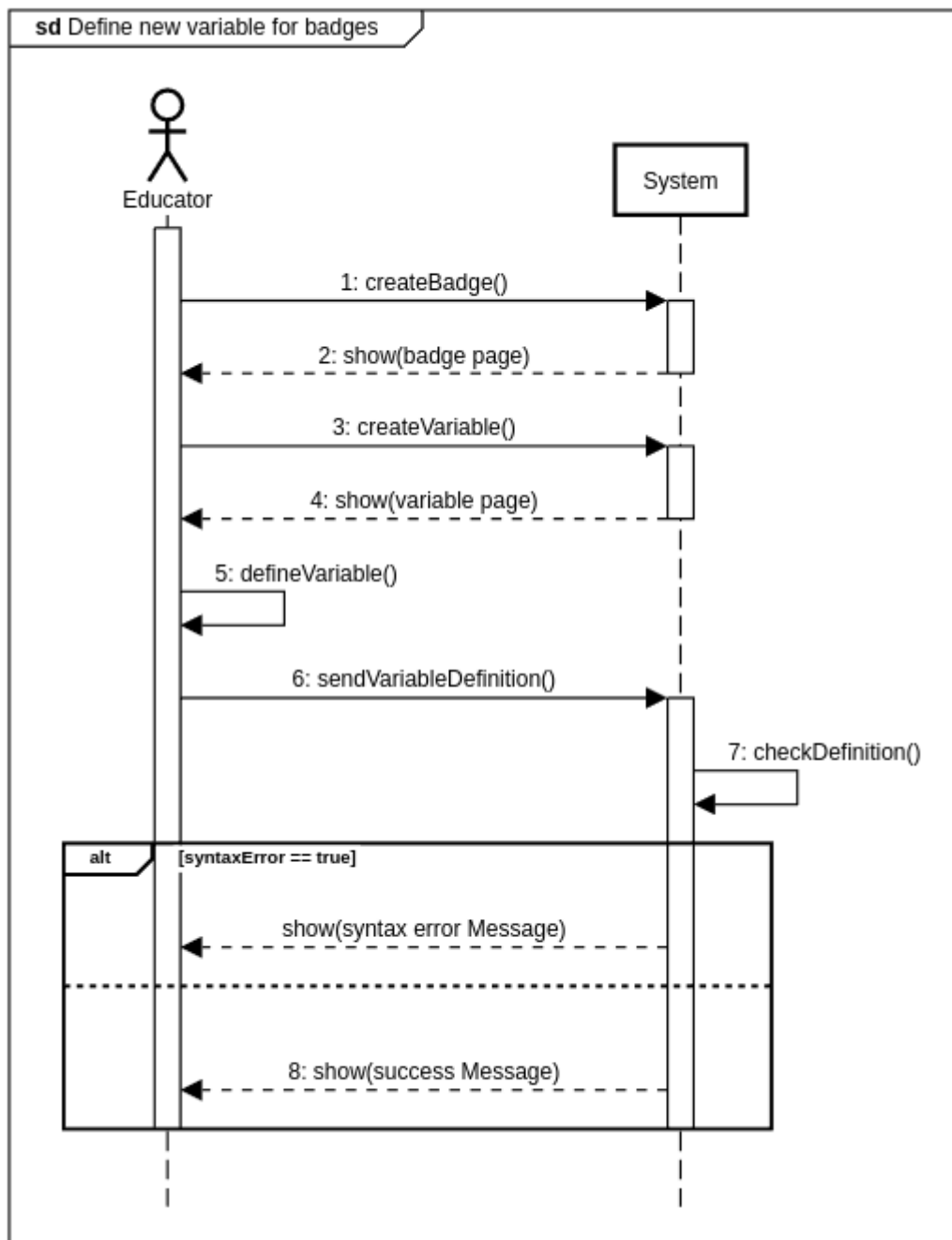
3.2.4 Requirements mapping

In this section are illustrated correlations with the defined requirements and domain assumptions for each goal.

[G1] Educators create tournaments	
[R1] The system allows users to sign up [R2] The system allows users to sign in [R7] The system allows educators to create tournaments [R8] The system allows educators to specify a tournament registration deadline [R10] The system allows educators to define gamification badges, consisting in a title and a one or more rules that must be fulfilled for a student to obtain the badge [R15] The system notifies all students subscribed to the platform whenever a new tournament is created [R31] The system allow educators to create new variables to use during the definition of the rules for gamification badges, using a pseudo-language	[D3] Tests provided by the educators are correct and consistent with the problem description.

[G2] Educators create battles for students to solve	
[R1] The system allows users to sign up [R2] The system allows users to sign in [R16] The system notifies students when a battle is created if they are registered to that battle's tournament [R20] The system allows educators to create coding battles for a specific tournament if they either have been given permission from the tournament creator to do so or they created that tournament [R21] The system allows educators to grant permission to create new coding battles for a tournament they have created to other educators [R26] The system allows educators to specify battle deadlines when creating a new battle [R27] The system allows educators to specify boundaries for the number of students in each group when creating a new battle [R29] The system allows educators to upload code katas when creating a new battle by providing a textual description, a set of test cases and build automation scripts	[D3] Tests provided by the educators are correct and consistent with the problem description

[G3] Educators evaluate student's solutions

[R1] The system allows users to sign up
[R2] The system allows users to sign in
[R11] The system requires educators to define a way to assign scores to students submission in an automated way
[R12] The system allows educators to decide whether they have to assign personal scores to students solutions during the consolidation phase
[R13] The system allows educators to assign a personal score during the consolidation phase if they decided to allow it when creating the battle
[R30] The system provides an API to allow users to submit their solution to a code battle, triggering the system to run automated tests to analyze the students code

[D2] Students use CKB's API to share their code with the evaluation system
[D5] Repositories who use CKB's API to submit their code actually belong to students subscribed to that specific battle

[G4] Students take part in tournaments together with other students

[R1] The system allows users to sign up
[R2] The system allows users to sign in
[R3] The system allows users to browse other users profiles
[R4] The system allows users to browse the list of tournaments
[R9] The system allows students to join tournaments
[R15] The system notifies all students subscribed to the platform whenever a new tournament is created
[R19] The system provides all students subscribed to a battle with that battles's code kata by notifying them with a link to that code kata's GitHub repository when a battle's registration deadline expires if they are subscribed
[R22] The system allows students to create a group for each battle
[R23] The system allows students to invite other students to their group for a battle
[R24] The system allows students to join a group for a battle

[D1] Students have a GitHub profile
[D4] Students know how to use GitHub in particular, GitHub Actions

[G5] Students solve coding problems and see educator's evaluations	
[R1] The system allows users to sign up [R2] The system allows users to sign in [R5] The system allows users to browse tournament rankings [R6] The system allows users to browse battle rankings [R14] The system allows the creator of a battle to terminate the consolidation phase after having evaluated all of the groups sources (if they decided to do so when creating the battle), effectively terminating the battle [R17] The system notifies students when the battle's final rankings become available [R18] The system notifies students when the final tournament ranking become available [R19] The system provides all students subscribed to a battle with that battles's code kata by notifying them with a link to that code kata's GitHub repository when a battle's registration deadline expires if they are subscribed [R25] The system allows students to compete in a coding battle if, when the registration deadline expires, they are part of a team composed by a number of students within the boundaries defined by that battle's creator [R28] The system allows students and educators to see evolving rankings before a code battle has reached its submission deadline	[D1] Students have a GitHub profile [D2] Students use CKB's API to share their code with the evaluation system [D4] Students know how to use GitHub in particular, GitHub Actions [D5] Repositories who use CKB's API to submit their code actually belong to students subscribed to that specific battle [D6] Educators will close the "consolidation phase"

3.3 Performance requirements

The CodeKata web application must ensure a seamless user experience through swift response times and efficient handling of user interactions. The system should respond to user actions within 2 seconds under standard conditions and maintain an uptime of at least 99% over any given month. Server resources should be optimized to prevent CPU utilization from exceeding 70%, and memory usage should not surpass 80% to mask network variance. These performance benchmarks collectively aim to ensure a reliable, responsive, and scalable CodeKata platform.

3.4 Design constraints

3.4.1 Standards compliance

The CKB platform will use the email address only to send notifications about events already written and explained.
The system will adopt the date and time used in the timezone UTC+1 and the format will be the following dd/MM/yyyy hh:mm:ss where:
- dd = days
- MM = month in the format 1-12
- yyyy = year
- hh = hours in the format 0-23
- mm = minutes
- ss = seconds

3.4.2 Hardware limitations

The users should have the necessary hardware to connect to the CKB platform and GItHub, that is, the ability to use an internet connection.

3.5 Software system attributes

3.5.1 Reliability

The system will be designed with robust error handling mechanisms to manage unexpected events, preventing catastrophic failures. Regular automated testing and continuous monitoring will be implemented to identify and rectify any potential issues proactively. Additionally, the system will have regular backup procedures in place to safeguard against data loss, providing users with a dependable platform for publishing and participating in code kata battles.

3.5.2 Availability

The system doesn't provide emergency services so its availability can be around 99% that means having 3.65 days of downtime per year allowed. This will be achieved through the implementation of redundant servers, load balancing techniques, and scalable infrastructure.

3.5.3 Security

Security is a paramount concern for the CodeKata software. Robust measures will be implemented to safeguard user data and protect against potential threats. This includes employing secure authentication and authorization protocols, encryption of sensitive information, and regular security audits to identify and address vulnerabilities. Additionally, the system will adhere to industry best practices and compliance standards to ensure a safe environment for users to engage in code kata battles.

3.5.4 Maintainability

The system shall be characterized by scalable and reusable modules which will be easier to maintain and replace in case of failure. Ordinary maintenance, for bug fixes and improvements, will be scheduled during night time, when the user traffic is minimal. The core aspects of maintainability and modularity will be addressed in the design document.

3.5.5 Portability

The CKB is inherently portable as it is designed to be accessed through a web browser. This means that users can utilize the platform on a wide range of devices, including desktops, laptops, tablets, and smartphones, irrespective of their operating systems. The website will be developed using responsive design techniques, ensuring optimal viewing and interaction experiences across various screen sizes and resolutions. Additionally, compatibility with popular web browsers will be a priority, enabling users to access and participate in code kata battles seamlessly from any location with an internet connection. The CKB portability ensures that users have the flexibility to engage in coding challenges on their preferred devices, enhancing accessibility and user convenience.

4. Formal Analysis Using Alloy

The model is created to show how the system can be at any point in time, but most importantly how the system cannot be!

4.1 Signatures

```
sig Student {

}

sig Submission {

}

sig Team {
    team_students: some Student,
    team_submissions: set Submission
}

enum BattleStatus { RegistrationPhase, SubmissionPhase, Closed }

sig Battle {
    teams_registered: set Team,
    min_members: one Int,
    max_members: one Int,
    battle_status: one BattleStatus,
    battle_submissions: set Submission
} {
    min_members > 0
    max_members >= min_members
}

sig Tournament {
    tournament_battles: set Battle
}
```

4.2 Facts

```
// Tournaments should not have battles in common
fact {
     all disj t1, t2 : Tournament | #(t1.tournament_battles &
t2.tournament_battles) = 0
}

// In all the battles that are started the number of member of the team
// should respect the battle's rule
fact {
     all b : Battle, t : b.teams_registered |
     b.battle_status != RegistrationPhase implies
     #(t.team_students) >= b.min_members and
     #(t.team_students) <= b.max_members
}

// A student cannot be in two different teams in the same battle
fact {
     all b : Battle | all disj t1, t2 : Team |
     (t1 in b.teams_registered and t2 in b.teams_registered) implies
     #(t1.team_students & t2.team_students) = 0
}

// A submission can only be performed by a single team
fact {
     all s : Submission | one t : Team | s in t.team_submissions
}

// All the battles should belong to a tournament
fact {
     all b : Battle | one t : Tournament | b in t.tournament_battles
}

// All the teams should belong to a battle
fact {
     all t : Team | one b : Battle | t in b.teams_registered
}

// A battle can have submissions only if it is in a SubmissionPhase or
// it is Closed (because it had submissions before)
fact {
     all b : Battle | #(b.battle_submissions) > 0 implies
     (b.battle_status = SubmissionPhase or b.battle_status =
     Closed)
}
```

```
// For all the battles a submission belongs to only one team
fact {
      all b : Battle, s : b.battle_submissions | one t : Team | s in
      t.team_submissions and t in b.teams_registered
}


// For all the team registered in a battle, if a team has a solution then
// the solution must be in the battle submissions
fact {
      all b : Battle, t : b.teams_registered |
      #(t.team_submissions) > 0 implies
      (all sub : t.team_submissions | sub in b.battle_submissions)
}
```

4.3 Predicates

```
 pred show {
       #Tournament <= 1
       #Battle <= 1
       #Team <= 3
       #Student <= 4
       #Submission <= 5
 }
```

4.4 Assertions

```
// There are no submissions during the registration phase
assert no_submission_in_registration_phase {
      all b: Battle | b.battle_status = RegistrationPhase implies
      #(b.battle_submissions) = 0
}

// If a team is subscribed in a battle that is not in the registration
//phase the number of participants must follow the rule
assert number_students_follows_the_conditions {
      all t : Team, b : Battle | t in b.teams_registered and
b.battle_status != RegistrationPhase implies
      #(t.team_students) >= b.min_members and #(t.team_students) <=
b.max_members
}
```

60

```
// All the students can be part of a team or stay alone
assert student_could_not_be_in_team {
    all s : Student | s not in Team.team_students or s in
Team.team_students
}


// A team can submit 0, 1 or more solutions to a single battle
assert submission_assert {
    all b : Battle, t : b.teams_registered | #(t.team_submissions &
b.battle_submissions) >= 0
}
```
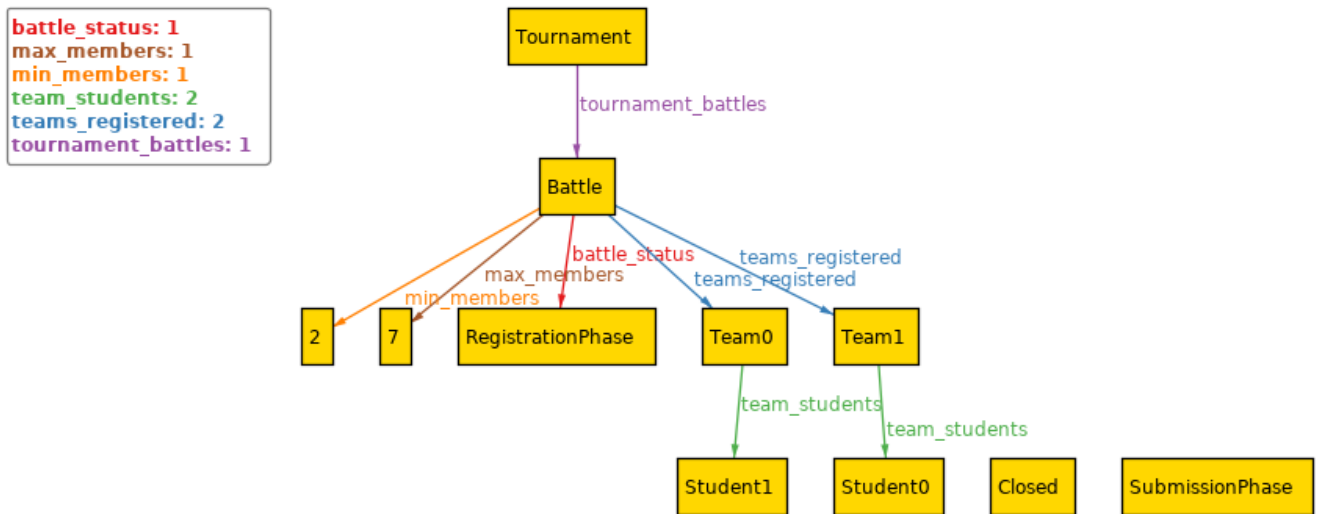
4.5 Run show

In this paragraph are shown a few simulations run using Alloy Analyzer.
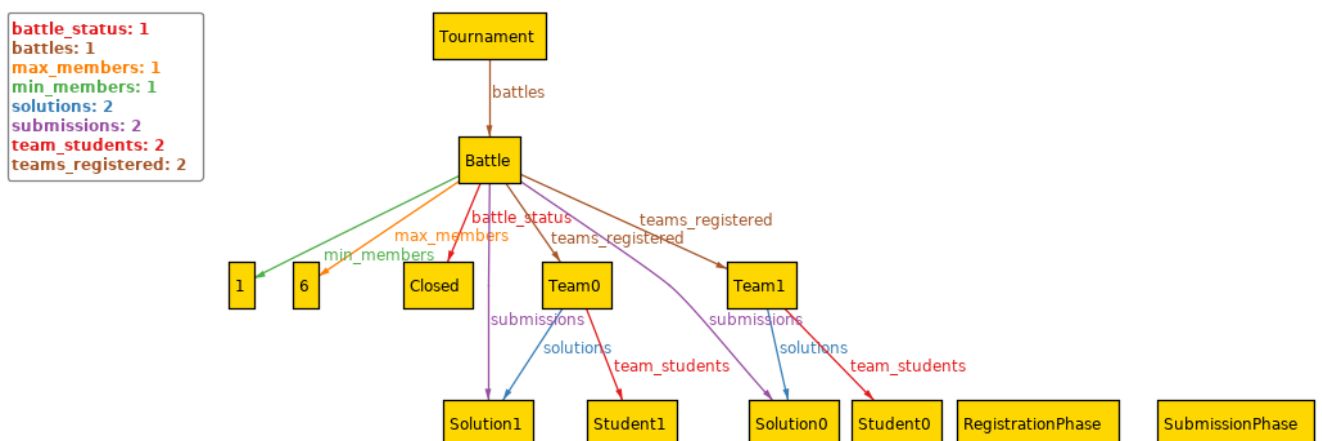- Submission during the submission phase
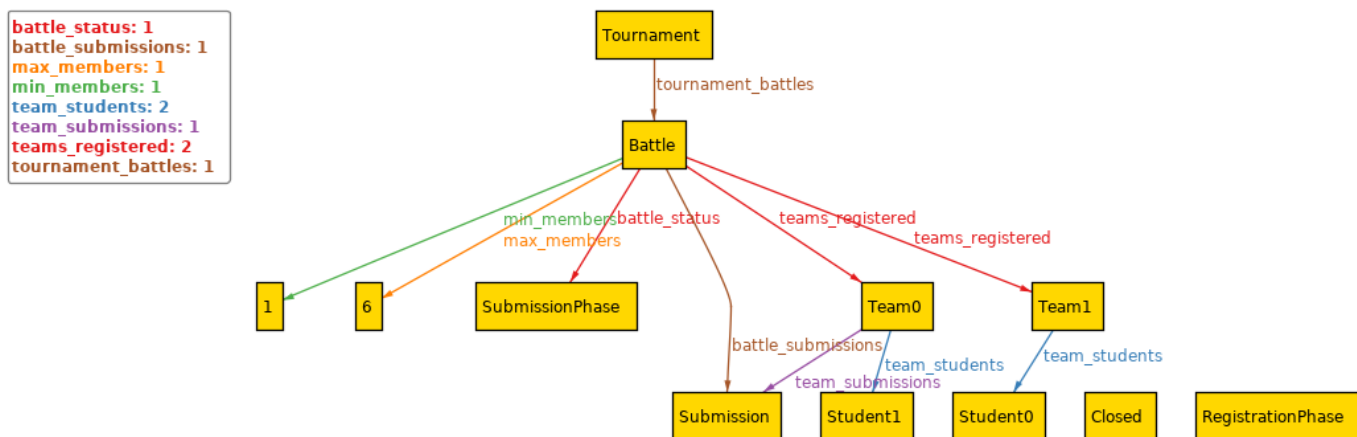
- Highlight the fact that a team can be registered also with less students during the RegistrationPhase


```
battle_status: 1
max_members: 1
min_members: 1
team_students: 2
teams_registered: 2
tournament_battles: 1
```

- The submissions done during the SubmissionPhase persist also when the Battle is closed


```
battle_status: 1
battles: 1
max_members: 1
min_members: 1
solutions: 2
submissions: 2
team_students: 2
teams_registered: 2
```

5. Effort Spent

Tommaso Fellegara

Introduction	4
Overall Description	2
Specific Requirements	8
Formal Analysis Using Alloy	5
miscellaneous	5

Manuela Marenghi

Introduction	5
Overall Description	1
Specific Requirements	7
Formal Analysis Using Alloy	1
miscellaneous	8

Cattani Luca

Introduction	5
Overall Description	5
Specific Requirements	12
Formal Analysis Using Alloy	1
miscellaneous	6

6. References

- IEEE standard for RASD documents: https://ieeexplore.ieee.org/document/8559686
- State machine, use case and class diagrams made with: https://app.diagrams.net/
- Sequence diagrams made with: https://sequencediagram.org/
- User Interface mockups made with: https://www.figma.com/