POLITECNICO
MILANO 1863

CodeKataBattle project Angelo Attivissimo,
Isaia Belardinelli, Carlo Chiodaroli

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Angelo Attivissimo, Isaia Belardinelli, Carlo Chiodaroli
Version:	1.1
Date:	07-January-2024
Download page:	https://github.com/Angelo7672/AttivissimoBelardinelliChiodaroli
Copyright:	Copyright © 2024, Angelo Attivissimo, Isaia Belardinelli, Carlo Chiodaroli – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

Computer Science is a field where theory and practice are tightly coupled and both very important.

In fact in theory it can be easily studied by reading books or following courses on specific topics, but in reality being able to practice the subject is the best way to consolidate studies from pure erudition to useful knowledge.

For this reason CodeKataBattle is a Platform that aims to help Students improve their skills by training and competing in Tournaments.

The word "Kata" is Japanese word from the world of Judo, which denotes a meticulously planned pattern of martial arts moves meant to be practiced and repeated with the aim of mastering it; and in this scope is where this Platform got its inspiration.

Students can use the Platform to compare themselves to one another while gradually improving little by little. Moreover, coworking with other people become a moment of enriching personal knowledge with other people experience.

Throughout the entire CodeKataBattle experience, Students are under the guidance of Educators, who organize and oversee competitions in order to grade and award Students on the basis of their work.

1.1 Purpose

The Platform's goal is to provide Students with hands-on experience in software development.

The system ought to be able to discern between the two categories of Users (Students and Educators) upon login. Users can look up and view the profiles of other Users on the Platform, this allows them to get to know about other people and collaborate here (maybe in a Tournament as a Team) and there (possibly in the real world as colleagues).

By applying their knowledge of one or more programming languages to solve puzzles or create whole products from scratch, Users can hone their software development skills. In order to consistently improve their code, adhere to the Kata's philosophy and train their coworking capabilities, Students will be able to compete against one another in Teams, and can improve their code until the end of the Tournament.

In the Platform, competitions take form of Tournaments, which are managed by Educators that design Battles (with their test cases) and lay out requirements for earning Badges.

The Platform will evaluate the code written by Students via automatic revision that will examine if tests are passed, time of execution, security and reliability. Additionally, also the Educators could participate in the evaluation process of a code Kata Battle of a Team of Students.

Ultimately, the Platform's automated processes, by means of test cases, or Educators themselves assess Teams' work, and the most deserving Students, which pass the respective requirements, may also receive special achievement Badges.
When a Student's profile is displayed, Educators and Students alike can view the Badges that have been accrued.

Additionally, an Educator involved in a Tournament is allowed to add additional Educators to it, who can assist in all aspects on managing the Tournament, like the development of Battles and scoring Teams.

1.1.1 Goals

Platform Goals

GP1: Allow Users to subscribe to the Platform
In order to interact with the Platform and its Users, Students and Educators have to be allowed to register. The satisfaction of this goal will allow them to be identified.

GP2: Allow Educators to create Tournaments
Battles are fought within a Tournament context, which is created by an Educator. So it is a goal of the Platform to allow them in this.

GP3: Allow Educators to invite other Educators to manage Tournaments
Allowing Educators to invite other Educators will empower them to help to manage the Tournament.

GP4: Allow Educators to create Battles
Both Educator that created the Tournament and the ones invited should be allowed to create Battles.

GP5: Allow Educators to define Badges
Badges are created to gamify the Platform and encourage Students to do their best. Educators are allowed to create Badges using different criteria.

GP6: Allow Educators to score Students
In order to make Students aware of their work Educators are allowed to assign scores on top of the one defined by the Platform itself.

GP7: Allow Students to get notificated about new Tournaments
To allow Student to participate, they should be notified of new Tournaments created.

GP8: Allow Students to subscribe to the Tournament
To participate in the Platform and being scored Students have to fight in "Kata Battles", which are defined within Tournament context.

GP9: Allow Students to invite other Students to partecipate to a Tournament
The purpose of the Platform is to improve Students' coding skills. This is done even through Teams that Students have to be allowed to autonomously form. To do so, they should be able to search for other Students within the Platform.

User Goals

GU1: Allow Users to do the login
Registered Users to be identified should be allowed to login in the Platform.

GU2: Allow Users to search Tournaments
Both Educators, to ask for access to Tournament in order to create Battles, and Students, to join them, Users should be allowed to search for Tournaments.

GU3: Allow Users to search for other Users
To collaborate with other Users, they have to be allowed to search for others, both Educators, to share Tournaments accesses and create Battles within it, and Students, to invite others to a Battle.

Student Goals

GS1: Allow Students to compete each other
The principle behind the Platform is to improve Students' skills, following "Kata Battle" method. This implies Students to compete with each other on their own or in Teams.

GS2: Allow Students to be rewarded for special achievement
Students should be allowed to receive Scores for their work. In addition to this special achievements are rewarded, according to criteria decided by Educators.

GS3: Allow Students to form and work in Team
To compete with each other, Students can form Teams. Working together they improve their collaboration skills.

GS4: Allow Students to have works evaluated
To be conscious of the quality of their work, Students have to be allowed to have their work evaluated.

Educator Goals

GE1: Allow Educators to create Tournament
Battles are created within a Tournament context, which Educators are allowed to create.

GE2: Allow Educators to create Code Kata Battle
Educators, both the one that created the Tournament and the ones who join it, should be allowed to create Battles, within Students will compete.

GE3: Allow Educators to create Badges
To reward Students for special achievement, Educators should be allowed to create Badges.

GE4: Allow Educators to evaluate Students
Scores, on top of the ones computed by the Platform itself, can be assigned by the Educator.

1.2 Scope

The Platform, which will be called CodeKataBattle, will have two User types: Educators and Students, and it will be reachable via a web app.

Users of all types are able to view each other's profiles. User Profiles will show the User details, as e-mail address and the handle of the repository management Platform. Moreover Students' profiles display their earned Badges.

Tournaments can be created by Educators, and this involves:

- the Educator sets the programming language/s of the Tournament
- the Educator inserts an outline of the competition and the instructions for the Students
- the Educator creates the Battle/s for the Tournament:
  - the Educator optionally can upload a partial code for the Battles
  - the Educator uploads the test cases for the Battle
  - the Educator defines the modality of evaluation and if in this Battle will be performed also a manual evaluation
- the Educator optionally can define Badges with attached rules
- the Educator optionally can add other Educators to the Tournament to help to manage it

- the Educator sets the deadline for subscribing to the Tournament
- the Educator sets the duration of the Tournament

After the creation of a Tournament, the Platform notifies all the Students that a new Tournament is available and, in the ones in which they are involved, Educators optionally can evaluate code written by Students in the Battle.

The main purpose of a Student in the Platform is to compete in Tournaments, every one of which all Students will have the opportunity to explore through the Platform.

A Student could subscribe to a Tournament by himself or by an invitation from a colleague. Optionally, shortly after subscribing to a Tournament, a Student can invite other Students to join to it together as a Team.

The Platform notifies all the participating Students that the Tournament has begun at the conclusion of the subscription period. The same thing will be done, including the final rank position, at the conclusion of the competition.

The Scores, that Students can receive, are split between the Battle score for a single code Kata Battle and the Tournament Score, which is the total of all the Battle Scores.

The Platform should automatically assess the code that Students wrote and update the Battle Score after an automatic evaluation; if an Educator-performed manual evaluation is available, the Platform should also take that into account when calculating the Battle Score.

The Platform, at the end of the Tournament, assign to each Student that satisfied criterias the correspondent Badge.

Code of all Battles will be handled on a third party repository manager like "GitHub" or "GitLab", where Educators are able to upload Battle code then Students to modify it and the Platform to run it. Given the abstraction level of the document, in the following repository manager will be known as Repository Management Platform.

1.2.1 Phenomena

Phenomenon	Who controls it?	Is shared?
User wants to improve his software developing skill	W	N
Educator wants to create a Battle	W	N
Educator wants to create a Tournament	W	N
Student forks the directory on Repository Manager Platform	W	N
Student creates a workflow Action on Repository Manager Platform	W	N
Student push the file on Repository Manager Platform	W	N
Student writes code	W	N
User registers to CodeKataBattle	W	Y
User logins in CodeKataBattle	W	Y
User searches for Tournaments	W	Y
User searches for Users	W	Y
Repository manager Platform action workflow notifies CodeKataBattle	W	Y
Educator creates Tournament	W	Y
Educator assigns optional manual evaluated Scores to Students	W	Y
Educator grants access to other Educators	W	Y
Educator creates Battles	W	Y
Educator creates a Badge	W	Y
Student invites other Students in a Team	W	Y
Student subscribes to a Tournament	W	Y
Student gets notified of new Tournament	M	N
Student gets notified of being invited to participate in a Team	M	N
Student gets notified of the end of a Tournament	M	N
Educator gets notified of being invited to manage a Tournament	M	N
Student gets Score	M	N
Student gets Badge	M	N
Educator gets notified about Student push	M	N

Phenomena table

1.3 Definitions

- **Platform**
  The complete "Code Kata Battle" Platform. This entity is inserted in order to represent system's behaviors in different situations, without explaining in details its implementation or workflow.

- **User**
  It is the generic User, Educators or Students depending on specific situation. It would be preferred to other definitions to model common behaviors and scenarios within the other two entities.

- **Student**
  Student is one of the main actors. His/Her scope is to improve its coding skills using the Platform. It will interact with it properly in order to have its work evaluated, to collaborate with other Students and to join Battles.

- **Educator**
  Educator is the other main actor of the Platform. His/Her scope is to create Tournaments and Battles in which Students can compete.

- **Battle**
  With this entity is described the single challenge Students have to face. It is the center around which all the Platform is build, and only through it would be achieved by Students the goal to improve their skills.

- **Team**
  The entity describes a set of Students, which are working in Team on a single Battle, or of Educators, which creates Battles in a Tournament.

- **Badge**
  With Badges Students are rewarded for special achievements. This entity is introduced to gamify the Platform.

- **Tournament**
  The Tournament is the context within Battles are created. A Tournament can host multiple Battles and represents the topic in which they take place.

- **Repository Manager Platform**
  The Repository manager Platform is a system which manages all code involved in Battles of all Students of all Tournaments.

- **Score**
  Through Score the Platform and/or an Educator evaluates Student's works. Each Student has its own Score history assigned, visible to other Students and Educators.

1.3.1 Repository Manager Platform specific definitions

These definitions describe the technical jargon used while talking about Repository Manager Platform functionalities.

- **Version Control System**
  A system that allows to save data in different versions keeping track of changes and authors.

- **Repository**
  A folder in which code is stored and managed by a Version Control System.

- **Action**
  Code that gets executed by the RMP on the repository once a certain condition is met (i.e. new upload, time of the day, ect...).

- **Commit**
  Act in which changes are saved in the Repository. Commits have details consisting in a short textual description known as message, and author data consisting in RMP handle and e-mail address.

- **Push**
  Act in which commits are uploaded to the RMP Platform.

- **Fork**
  Act of duplicating repositories created by others in a personal one.

1.4 Abbreviations and Acronyms

- **CKB**
  CodeKataBattle, the name of the Platform.
- **RMP**
  Repository Manager Platform
- **VCS**
  Version Control System

1.5 Revision History

- **Version 1.0** - 2023-12-22
  - First release
- **Version 1.1** - 2024-01-7
  - Tweaked UC1a - Student signs in to the Platform Sequence Diagram
  - Tweaked UC1b - Educator signs in to the Platform Sequence Diagram
  - Tweaked UC3 - Student subscribes to Tournament
  - Tweaked UC4 - User invites other User to collaborate
  - Tweaked UC7 - Educator creates Badge
  - Tweaked UC9 - Team gets code evaluated
  - Better Explained 3.3: Evaluation Requirements

1.6 Document Structure

- **Section 1: Introduction**
  A synopsis of the issue and necessary features is provided in this section.
  A list of definitions, acronyms, and abbreviations that appear in this document is also included.
  Lastly, there is the document's changelog, which includes the list of revisions and their content, and the document structure, which outlines the primary goals of each section.
- **Section 2: Overall Description**
  In this section is presented a high-level description of the Platform, showing mainly non-technical characteristics.
  These characteristics are: scenarios, where some informal descriptions of User-Platform interaction are given; general structure diagrams, where the main relationships between actors are described; functionalities and assumptions, that are described in a discursive manner.
- **Section 3: Specific Requirements**
  In this section are proposed again elements shown in section 2 in a more technical manner, with the objective of facilitating the real implementation of the Platform.
  The most important technical description given in this section is the one of the Use Cases, which are structured in tables followed by their respective sequence diagrams to better explain how scenarios are implemented.
  Moreover, are also described every kind of functional or non-functional requirements (like external interfaces, performance, design constraints and system attributes).

- **Section 4: Formal Analysis**
  In this section is checked the model of the Platform in the formal verification language Alloy. Specifically is shown the implementation in Alloy code of the model and relative constraints, then are presented views of the results of some executions to show the model soundness.
- **Section 5: Effort Spent**
  In this section is shown the effort spent for the production of this document, divided by author and section.
- **Section 6: References**
  In this section are listed all references to external sources cited during the document.

2 Overall Description

This section describes the perspective, functionalities, characteristics, and domain assumptions of CKB Platform, which is a complete system.
The rigorous description of the RMP is out of the scope of the document.

Of the RMP will be only described critical parts needed for the correct functioning of the Platform, allowing to choose those that satisfy them in a later stage.

2.1 Product Perspective

2.1.1 Scenarios

SC1 Student signs in to the Platform to participate to Tournaments
Alice is studying programming all by herself and wants to evaluate her work. She accesses the Platform website and is intrigued to try it out. Alice signs in as a Student giving her details like name, surname, e-mail, year of birth and RMP handle. Once done that she will receive updates on new open Tournaments. Moreover, she can log in and be presented with a list of Tournaments to which she can subscribe.

SC2 Student wants to participate in a Tournament
Dan, after studying a bit of Javascript wants to apply its studies in a project. In order to do so he wants to participate in a Tournament. He is already subscribed to the Platform, so he can just log in to it and search a suitable Tournament. After being presented of some valid choices, he clicks on one of them to see more details like description, Educators, subscription deadline and available Badges. If he likes it, he clicks a "Subscribe" button, otherwise goes back and continues to search.

SC3 Student wants to participate to Tournament with friends
Bob, a User of the Platform, got notified of the creation of an interesting Tournament in which he wants to participate with his friend Carlo. Bob logs in to the website and subscribes to the Tournament sending an invitation to Carlo, who gets notified of the invitation and, once logged in to the Platform, can choose to accept or to deny it; if he accepts Bob and Carlo result subscribed to the Tournament as a Team.

SC4 Educator signs in to the Platform to create Tournaments
Eve is a teacher that, in order to widen her didactic offer, wants to create some Tournaments for her Students. She accesses the Platform website and is intrigued to use it. Eve signs in as an Educator giving her details like name, surname, e-mail, year of birth, institution and RMP handle. Then she logs in and is presented with a list of managed Tournaments, empty for her, and the possibility to create new ones.

SC5 Educator wants to create a Tournament
Faith is a User of the Platform signed in as an Educator and wants to create a Tournament. She logs in to the website and is presented with the homepage, in which she can start to create the Tournament by clicking a specific button. After clicking it she can enter details like Tournament name, details (as overview, programming language/s and maximum and minimum number of Students for a Team), subscription deadline and duration, then click a "Create Tournament" button that will create the Tournament and bring her to the manager Tournament page.

SC6 Educator wants to manage a Tournament with a colleague
Grace is an Educator who is managing a Tournament, she finds that the work behind creating all Battles of the Tournament is a bit too daunting, so she wants help. To do so she goes to the

Tournament management page and by clicking the "add Educator" button she can send an invitation to manage the Tournament to her friend Heidi, another Educator registered to the Platform. Heidi can accept the invite and create new Battles and Badges for the Tournament and if it is required, also manual evaluate the Students' code.

SC7 Team wants to work on a Battle
After receiving notification of the Tournament's start, a Team that has subscribed to the competition wishes to focus on a Battle. After arriving at the Tournament page, to Isaac, a member of the Team, is shown a list of the Battles that are available. He selects one and is taken to a new page that includes his name, details, the Team scoreboard, and a link to the repository within the RMP. Since Isaac has already registered with the RMP, he is able to fork the repository and make an actual copy that is stored in its profile. He now needs to check each Team member's RMP handle in their profile and invite them all to join the Battle. All Team members have to accept the invitation, and after then they can start to work on the Battle. The owner of the forked repository has to put its link in the Battle details to signal the Platform of its existence and in other he has to set an automated workflow that informs the CBK Platform (through proper API calls) as soon as someone of the Team make a modification of the code in the repository.

SC8 Team wants to have work evaluated
A Team has worked on the Battle and wants to check how is the quality of his work. In order to do so they only need to commit their changes and push their work to the RMP, this will be able to alert the Platform of the new changes and will set off a pull of the files to be evaluated. After running all tests, these will generate points that will be combined to form a score. This score will be published near the name of the Team in the leaderboard of the Battle, and will update also the main Tournament score of that Team. Possible checks for Badge assignment can be also done, and if satisfied, Badges can be granted at the end of the Tournament.

SC9 Team wants to see how it is placed in the Tournament
Angelo, a Student participating in a Tournament in a Team, wants to know how well they are doing. To do so he can log in to the Platform and go to the page of the Tournament, on which he can see the leaderboard where he can find the overall Tournament score of the Team. Angelo can also check specific Battles visiting their specific page where the related Battle leaderboard is shown.

SC10 User wants to check out other User's profile
Michael, a User of the Platform, wants to find the profile of his friend Nick to check out on him. He logs in to the website in which he can search for other Users via a specific bar, there searches for Nick, and clicking on his name goes to the profile page. Here he can see his details like: User type, name, surname, RMP handle and Badges.

SC11 Educator wants to create a Battle
Jude oversees a Tournament as an Educator. She now wants to put her Students to the test by suggesting a Battle. When she opened the CKB Web App, she was taken to the page where she oversees the Tournaments she manages. Jude then chooses the one where he wants to start a Battle and clicks the "Create Battle" button. This opens a form for her to fill out with the Battle's name and an overview of the Battle. After supplying these details, she needs to include the Battle's RMP repository link and specify the Battle's evaluation modality. After filling out the form, she presses the "Submit" button. Now that the system has confirmed all the data, a new Battle has been created.

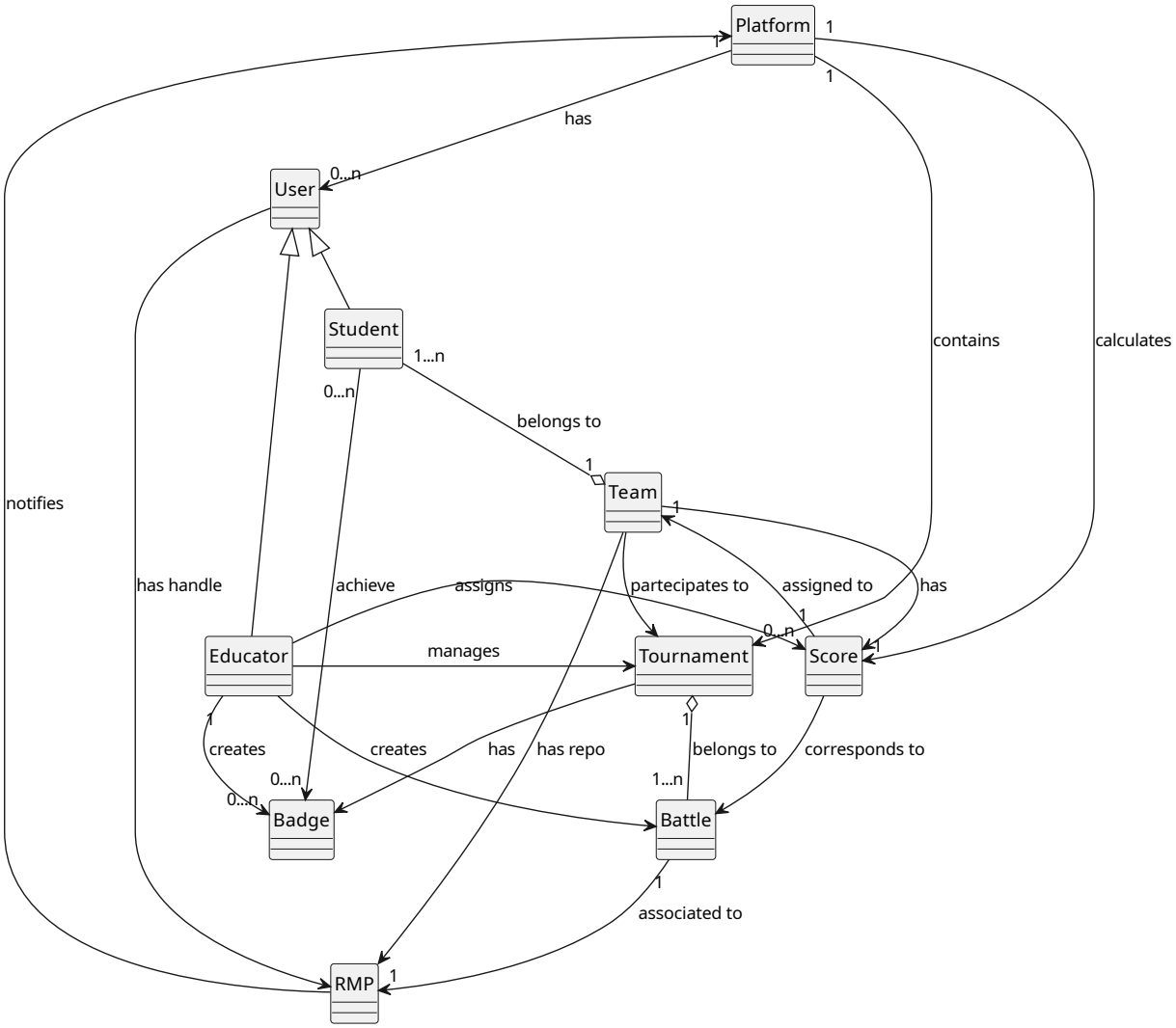
SC12 Educator wants to manually evaluate teams code
Gino is an Educator, and after checking the code of a Team on a battle on a tournament he manages, he wants to manually evaluate it. He opens the CKB Web App on his home page, goes to the tournament page to reach battle rankings, there finds the team he wants to evaluate and clicks on the "Add score" button. He receives a form to complete, which he provides by adding a score. Once the form is submitted, the score is updated on the respective entry on the ranking list.

SC13 Educator wants to create a Badge
Beniamino is an Educator, and he wants to design a Badge for a Tournament he oversees. He open the CKB Web App on his home page and go to the page with the managed Tournaments list. He clicks the "Add Badge" button after choosing the one for which he wanted to make the Badge. Beniamino receives a form to complete, which he provides by adding an image, a description, and the requirements that must be met in order to obtain this new Badge. Once the form is submitted, the Badge is created and made achievable during the Tournament.

SC14 Student gets awarded with Badge
Rupert, a Student of the Platform, has participated in a Tournament, writing code for the Battles. His code got evaluated and is in the scoreboard of the Platform. The Tournament has ended, so he can no more change his code. While checking his e-mails, he finds one email, sent from the Platform, that notifies him that he got awarded with a new Badge. Then curious, he goes to his profile page to see the Badge, where he finds it with a photo and a short description. After some time, Olivia, a User of the Platform wanting to find a Student to invite to a Team, for curiosity checks Rupert's profile page, seeing that he gained a lot of Badges during his time on the Platform, decides to try inviting him.

2.1.2 Domain Model

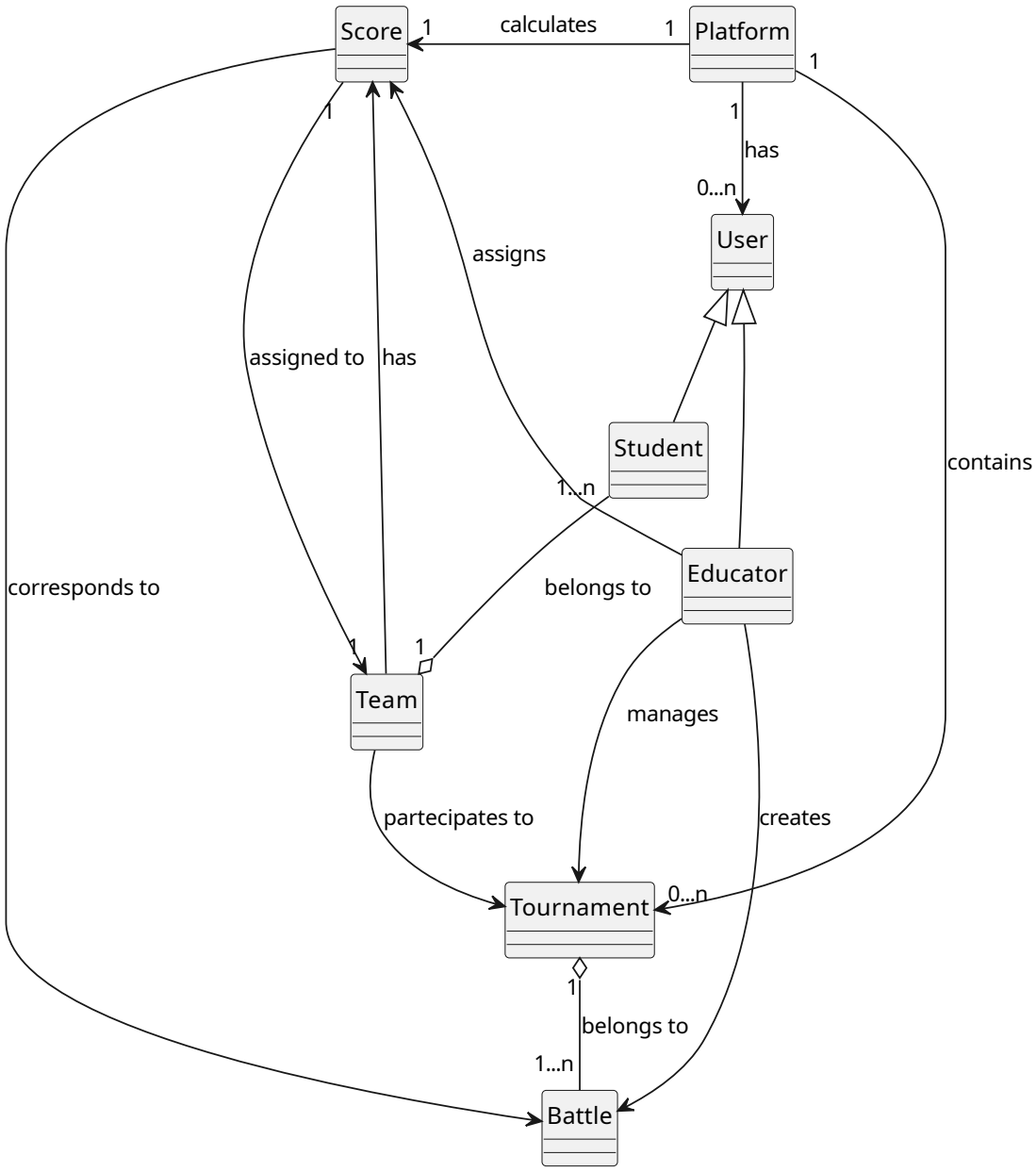
Class diagram This is an illustration of a high-level UML. The primary classes are described, along with the relationships that exist between them.



High-level UML diagram with main classes

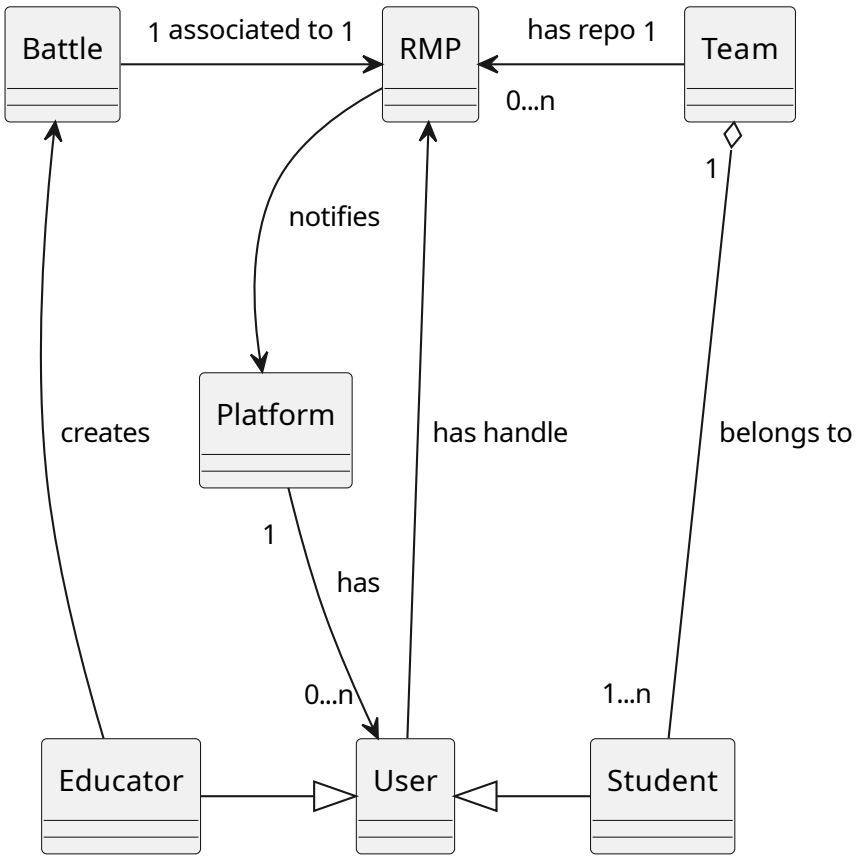
For compassion for the reader, this class diagram has been divided in three parts to facilitate fruition:

Platform Class Diagram Describes the core functionality of the Platform without caring about RMP and Badges.



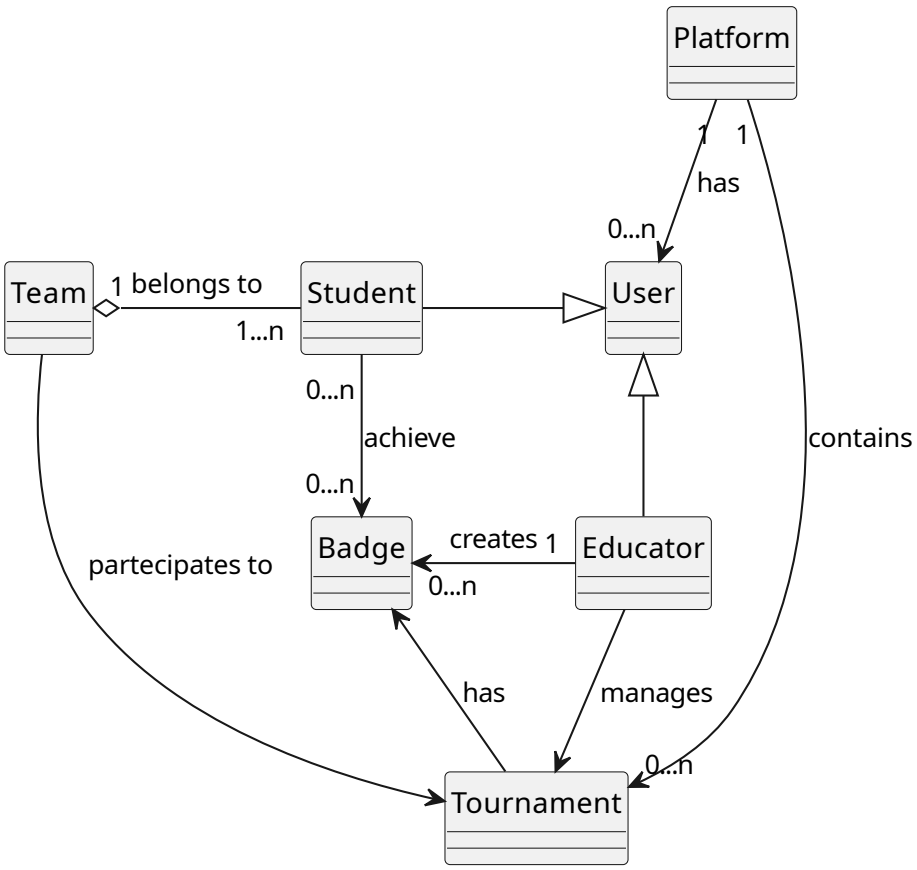
Platform Class Diagram

Platform relationship with RMP Class Diagram Describes which elements of the Platform have relations with RMP.



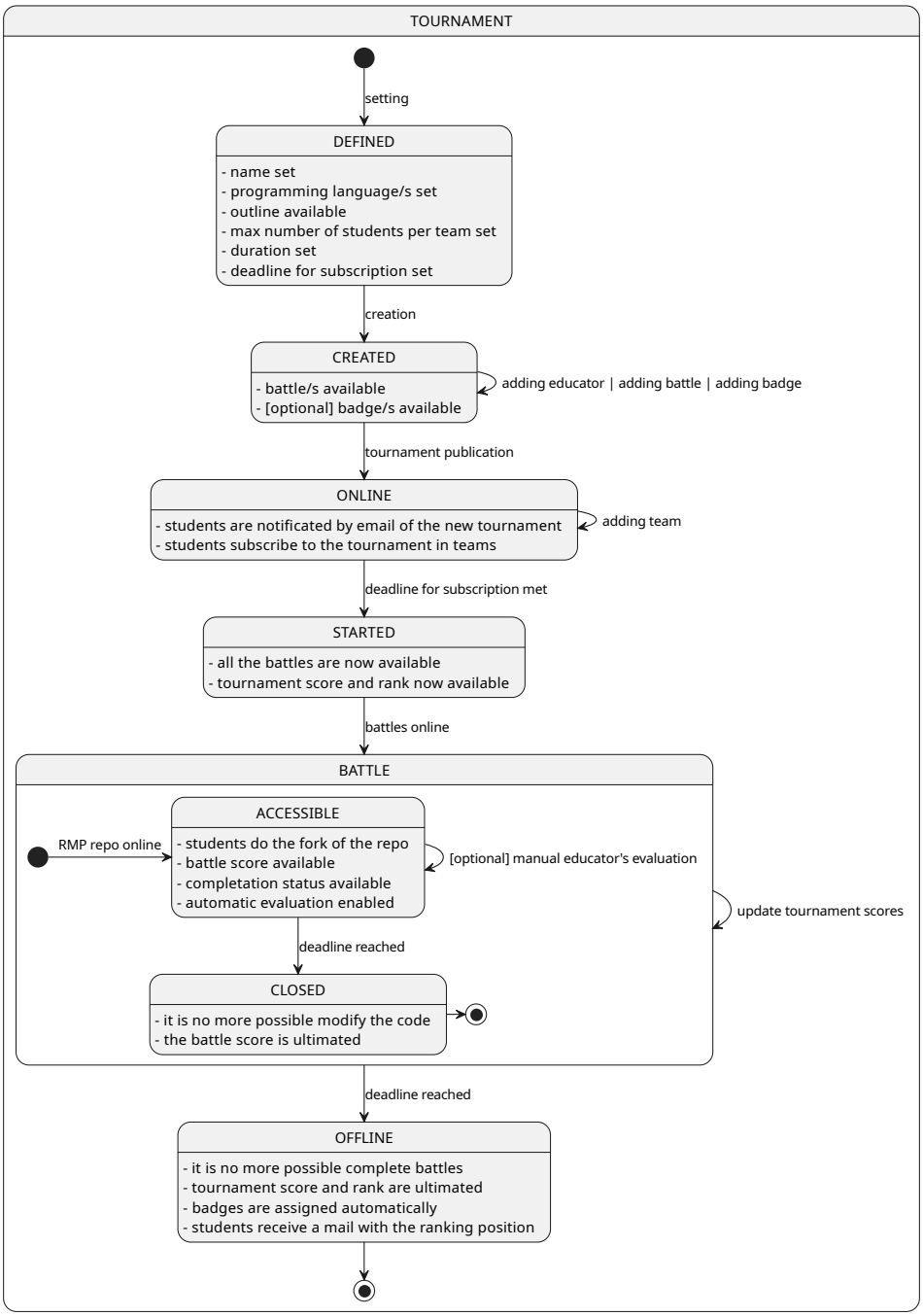
Platform relationship with RMP Class Diagram

Platform Gamification Class Diagram Describes how Badges work inside the Platform.



Platform Gamification Class Diagram

State diagram The different stages of the Tournament and the Battle are depicted in the state diagram in the figure.



State Diagram of the Tournament and the Battle

In the CREATED state it is possible to add Educators, Battles and Badges to the Tournament:

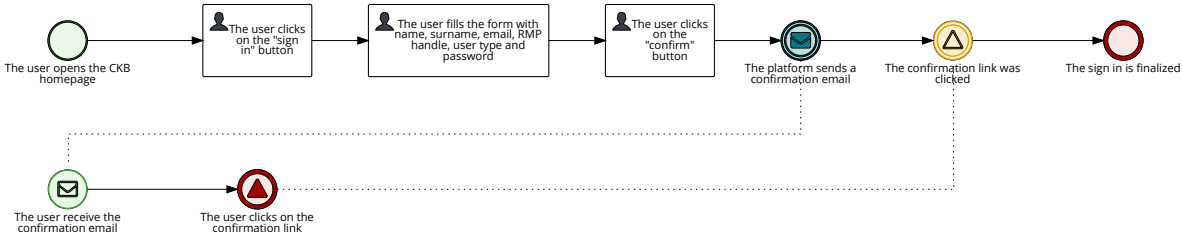
- Adding a Battle consists in provide a description for it, make the code and the test available and in defining the modality of evaluation between the automatic modality and the optional manual modality.
- Adding a Badge consists in giving a name for it and defining its requirements.

2.2 Product Functions

From the previous scenarios and the introduction above follow these Product Functions.

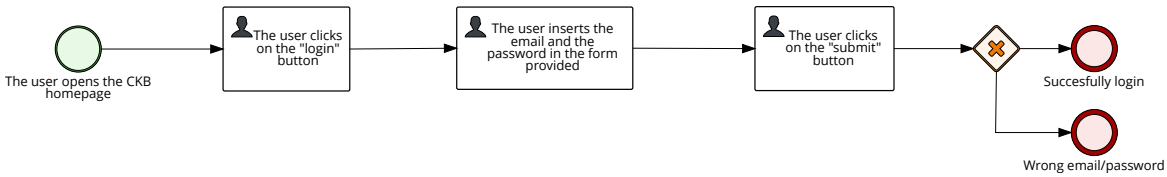
2.2.1 User Functions

Sign in Every person that wishes to sign in to the Platform can sign in via a specific form which allows to become a User of it. In the Platform the form requests: name, surname, e-mail address, RMP handle, User type and password; after clicking on confirmation button, the Platform sends a confirmation e-mail to the e-mail address given in the form with a button to finalize the subscription to the Platform. Sign in is finalized once the button is clicked.



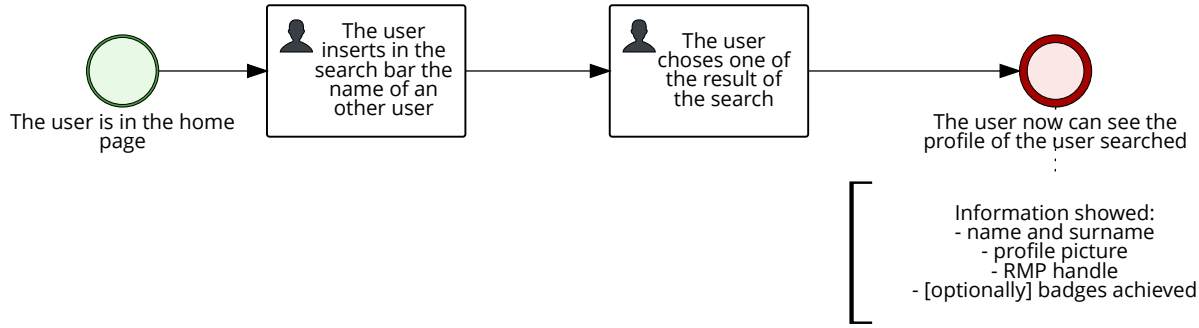
BPMN diagram of the sign in

Log in Every User needs to log in to the Platform to perform any activity in it. Log in is done via a form which has to be filled with e-mail address and password, if the pair corresponds to a User, the User will be logged in to the Platform as of his role (Student or Educator).



BPMN diagram of the log in

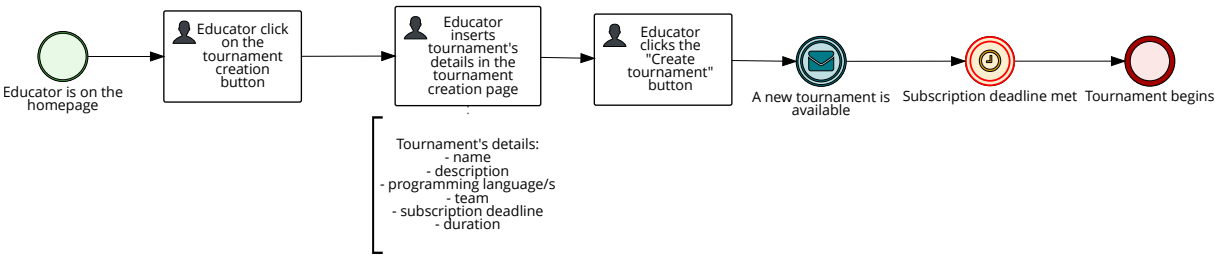
Search for other User Every User subscribed to the Platform, can search other Users by name, email or RMP handle in the search bar. If the User searched is a Student, on his profile page are shown also the Badge achieved.



BPMN diagram of searching a User

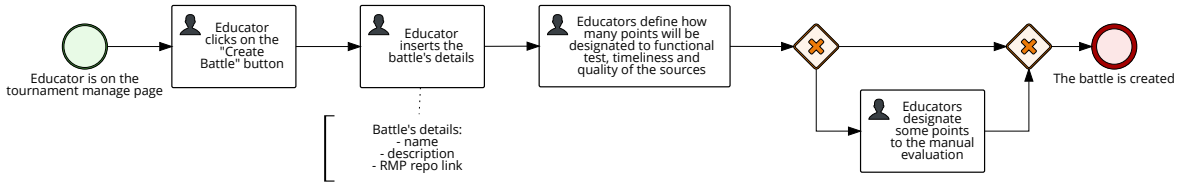
2.2.2 Educator Functions

Create a Tournament The page where the Tournament creation is displayed is accessed by clicking a specific button on the homepage.
There, Educators can enter the Tournament's name, description, programming language/s, minimum and maximum number of Students per Team, subscription deadline, and duration.
After the Educator clicks the "Create Tournament" button, Students are notified that a new Tournament is available. Students will see in the Tournament details also the Educators involved in it: in this way they can add them to their Battle's RMP directory.
When the deadline subscription is reached, the Tournament begin.



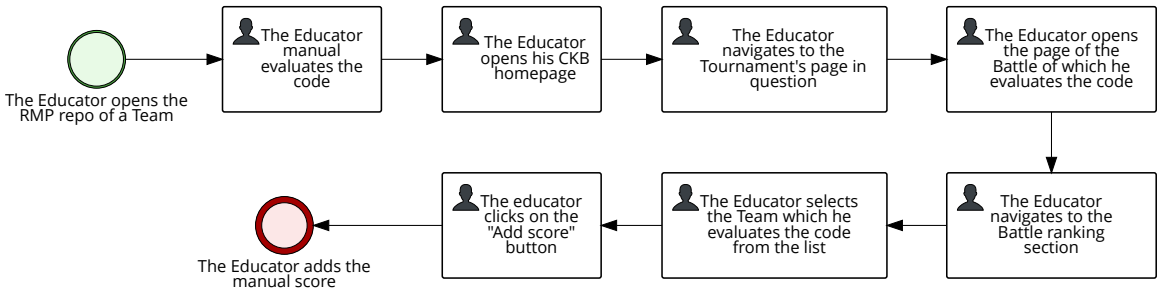
BPMN diagram of the creation of a Tournament

Create a Battle In the dedicated manage page that is provided to the Educators involved in a specific Tournament, there is a "Create Battle" button that an Educator can use to create Battles.
After that the Educator click on that button, he inserts in the Battle creation page the name, a description, the link to the RMP directory (the RMP directory contains all the test cases of the Battle and in some case also some programming code already available) and set how the Battle will be evaluated, defining how many points of the total 100 will go to the functional tests, to timeliness, to the quality level of the sources, and optionally he could designate some points also to the manual evaluation.
In any case the sum of the points that a Team could achieve from all this parameter must be 100.



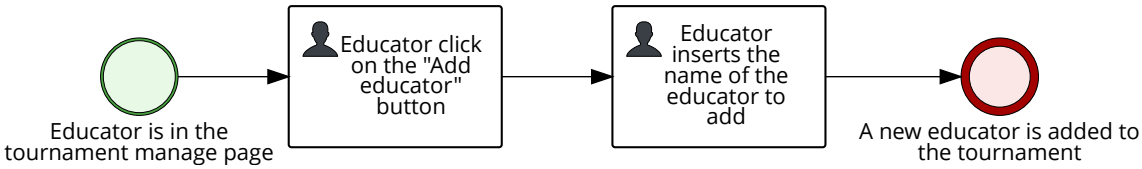
BPMN diagram of the creation of a Battle

Manual evaluation When necessary, the Educator must manually assess the code that the Teams have written.
In order to do this, he must complete his own manual evaluation visiting the RMP directory of a Battle of the Teams, where he must first be added by the Students. He must next navigate to the tournament page of the corresponding tournament, access the corresponding Battle page, and proceed to the Battle ranking section. There, he chooses the Team he evaluated and clicks the "Add score" button next to the Team's name to enter his manual score.



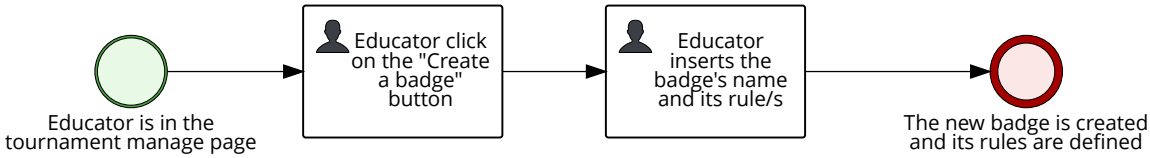
BPMN diagram of the manual evaluation

Add an Educator to a Tournament An Educator can add another Educator to a Tournament once he is on the appropriate management Tournament page. The Educator's addition will support the manual assessment of the Students' code.



BPMN diagram of the addition of a new Educator to the Tournament

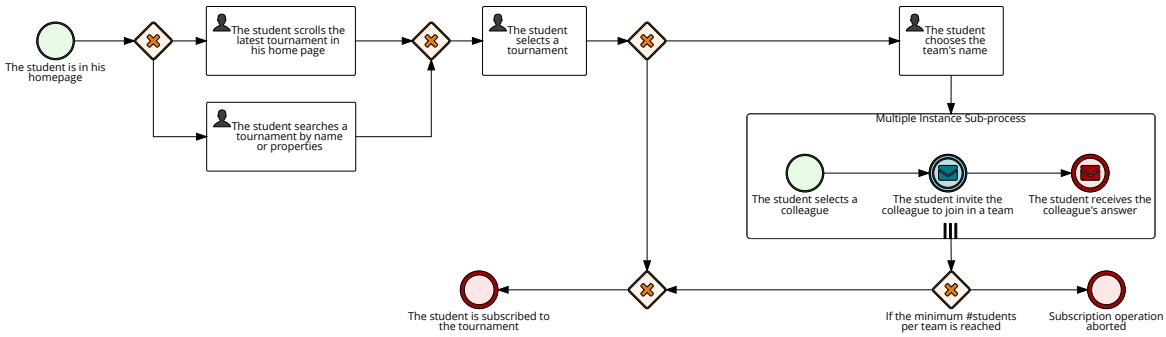
Create a Badge The Tournament management page allows an Educator to make a Badge. To accomplish this, he must click the "Create a Badge" button. This will open a creation form, which the Educator must fill out with the name of the Badge and the rule or rules that the Students must follow in order to receive the Badge.



BPMN diagram of the creation of a Badge

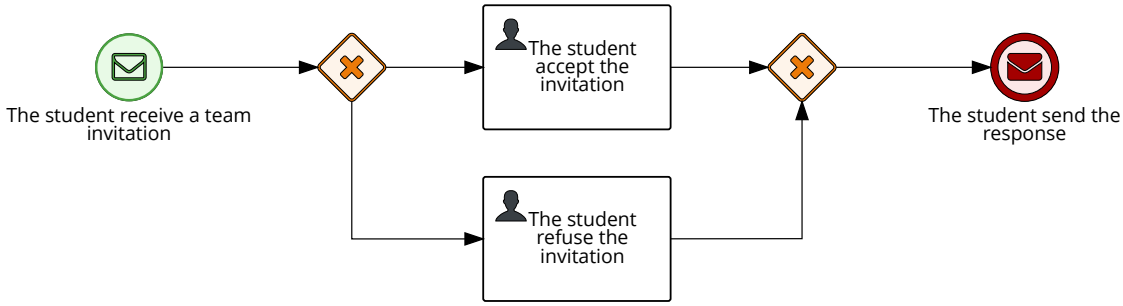
2.2.3 Student Functions

Search and subscribe to a Tournament A Student can search for Tournaments on the Platform from his homepage. He can view the most recent Tournaments added or search for them by name or properties. He can subscribe to a Tournament after selecting one, and each Tournament has a minimum and maximum number of Students signed up for each Team. While it is occasionally possible to compete alone, one of CKB's goals is to teach Students how to write code in a Team. As a result, Students will most likely join a Team when they subscribe to the Tournament, and it's possible that he will invite some of his friends to join a Team that he will name during the subscription process.



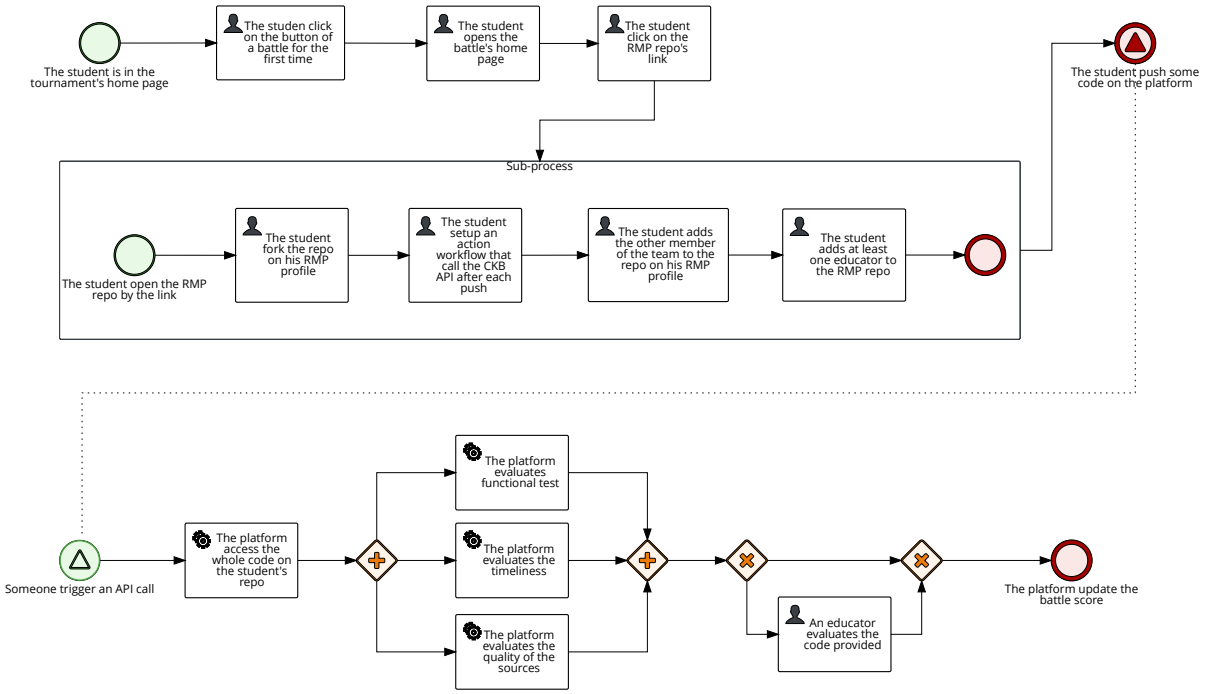
BPMN diagram of the subscription to a Tournament

Accept an invitation from another Student A Student can join to a Tournament by an invitation of another Student and compete together in a Team.



BPMN diagram of accept an invitation from another Student

Complete a Battle A Student can carry out Battles by clicking on the button of a particular Battle from the Tournament page once he has subscribed to it. Once the RMP directory has been configured with the fork and workflow action, he can write code locally to pass the test cases that are included in the Battle repository. The Platform will automatically evaluate the functional test, the timeliness and quality of the sources, and it will update the Battle score, that the Student can view from the Battle page, each time he pushes code into the repository. Optionally, an Educator might evaluate the written code as well.



BPMN diagram of Battle completion

2.3 User Characteristics

2.3.1 User Characteristics

A User is a normal person who intend to subscribe to CKB. It is able to access to the Platform by web app from any type of device with an internet connection.
Every User subscribed to the Platform is able to search for other Users using the search bar.

2.3.2 Educator Characteristics

An Educator is a User whose job it is to provide Students with rewards and practice materials for software development.

He designs coding Battles and software development Tournaments that simulate actual IT case problems for Students. In addition, he evaluates their work optionally and creates a special Badge for Students who successfully complete specific tasks.

2.3.3 Student Characteristics

A subscriber to CKB who wants to hone his skills in one or more programming languages or who wants to practice developing software in a Team is identified as a Student.

He can use the homepage to browse Tournaments and subscribe to them, but he may also want to use the search bar to look up Tournaments by name or just look them up by programming language. He has the option to invite other Students to join him in a Team that he will name in the subscription, or he can accept invitations to participate in Tournaments.

His Team will be ranked in the Tournament according to the total Battle score they earn during the coding Battle phase of the Tournament.

2.4 RMP Functions

The Repository Manager Platform is a Platform that allows to store code in a smart way RMP is third party system that is responsible for storing all Team code related to all Battles of all Tournaments, the code will be kept only for test purposes and discarded shortly after score assignment. To be useful to the Platform the RMP needs to have some necessary functionalities like:

1. RMP Users can search for other RMP Users by handle or e-mail
2. RMP Users can collaborate on shared repositories
3. RMP allows running actions and these can work with KCB Platform's API, technically needed to signal the Platform of new pushes on the repository.

2.5 Assumptions, Dependencies and Constraints

2.5.1 Assumptions

D1: User has provided valid credentials - RMP handle, e-mail, name and surname

D2: User is registered to the RMP Platform

D3: User has an email

D4: User has a functioning device capable of internet connection

D5: User has a functioning internet connection

D6: User can locally work on project files

D7: User use Platform ethically

D8: Educators correctly set the repository for each Battle

D9: Educators create Tournaments adequate for the Platform

D10: Educators allow Students to fork Battle repository

D11: Teams fork repository once for Battle

D12: Teams, once forked repository, invite at least one educator to join the repository

D13: Teams work on their forked repositories

D14: Students don't allow Students out of their Team access to modify code in their repository

D15: Students allow Tournament Educators access to their Team repository

D16: Teams declare coherent links to their Battle repositories

D17: RMP is able to notify the Platform once every push

D18: RMP is able to reliably store Battle source code

D19: RMP API is accessible to third party applications

D20: RMP bases itself on version control system activate E

2.5.2 Dependencies

Repository Manager Platform The Repository Manager Platform is a Platform that allows to store code in a smart way using a VCS. RMP is assumed to be a third party system that is responsible for storing all Team code related to all Battles of all Tournaments, the code will be mainly kept on the RMP and the CKB Platform will take it only for test purposes and discard it shortly after score assignment. To be useful to the CKB Platform the RMP needs to have some necessary characteristics that have been listed as domain assumptions. An example of compatible RMP is GitHub.

2.5.3 Constraints

Security Constraints Given her nature, this Platform is more vulnerable to security attacks than other ones. For this reason, especially in Battle code evaluation, the Platform needs to implement state of the art security policies in order to protect User data and safety.

Privacy Polices Personal User data and historical User data shall be stored and used only for legitimate use internal to the Platform. Public to the Platform User data has to be accessible only via the Platform. Finally, none of the User data has to be used for commercial purposes, and its use has to be done in compliance with, to the User, local privacy policies.

Platform Limitations User is required to use internet connection to access the Platform, download code repository and upload code for evaluation. Out of these actions, Users can work on Battles locally without the need of internet connections, if not specifically stated otherwise in the Battle description.
Battle computational size and complexity has to be sized in order to be feasible to be coded and executed locally by Students.
Platform does not allow for live code editing and execution, for this reason Users have to be able to edit and run code locally.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

In order to access the Platform the crucial interface needed by the User is the one provided by a web browser. In fact, it is sufficient to reach the CKB's URL to start with log in or sign up operations, described in the scenarios, or, after authentication, interact with the Platform. The Platform software, in fact, is a Web App. Instead, to join a Battle a RMP link would be required.

3.1.2 Hardware Interfaces

Users have to provide themselves with a device able to access internet. It would be sufficient that it is equipped with a Wi-Fi and/or Ethernet interface. Of course would be crucial that it provides adequate components to allow Users to interact with the Platform, showing its interfaces and collecting inputs.

3.1.3 Software Interfaces

As defined above, a web browser is the only software needed to access the Platform. The interfaces that have to be supported are the ones defined by the Web Page rendering. About RMP, the Platform software run on the server would be equipped with appropriate interfaces to interact with the RMP provided by the Student or the Team in the steps described in previous sections of this document.

3.1.4 Communication Interfaces

Communication Interfaces needed are the one necessary to access the internet. So for the User, TCP and HTTP interfaces would be crucial to reach the server on which the Platform runs, while for the CKB's app it is fundamental to interact with RMP and with E-mail Provider. For this last communication the protocol that would be used is the SMTP, as for RMP the same ones described above for Client-Server interaction.

3.2 Functional Requirements

Here follows a list of the Platform functional requirements:

R1 The Platform allows Users to sign in to the Platform itself either as Student either Educator.

R2 The Platform allows Users to log in to the Platform.

R3 The Platform shows Students the list of available Tournaments.

R4 The Platform allows registered Students to search for Tournaments.

R5 The Platform allows registered Students to subscribe to Tournaments.

R6 The Platform allows registered Students to participate to Tournaments as a Team.

R7 The Platform allows registered Students to invite other Students to join a Team.

R8 The Platform allows registered Educators to create Tournaments.

R9 The Platform allows registered Educators to create Battles in the context of a Tournament.

R10 The Platform allows registered Educators to manually assign scores to Teams in a context of a Battle in a Tournament managed by them.

R11 The Platform allows registered Educators to create Badges in the context of a Tournament.

R12 The Platform assigns a Battle Score to Teams' work.

R13 The Platform provides a Teams' ranking based on the Tournament Score within a Tournament context.

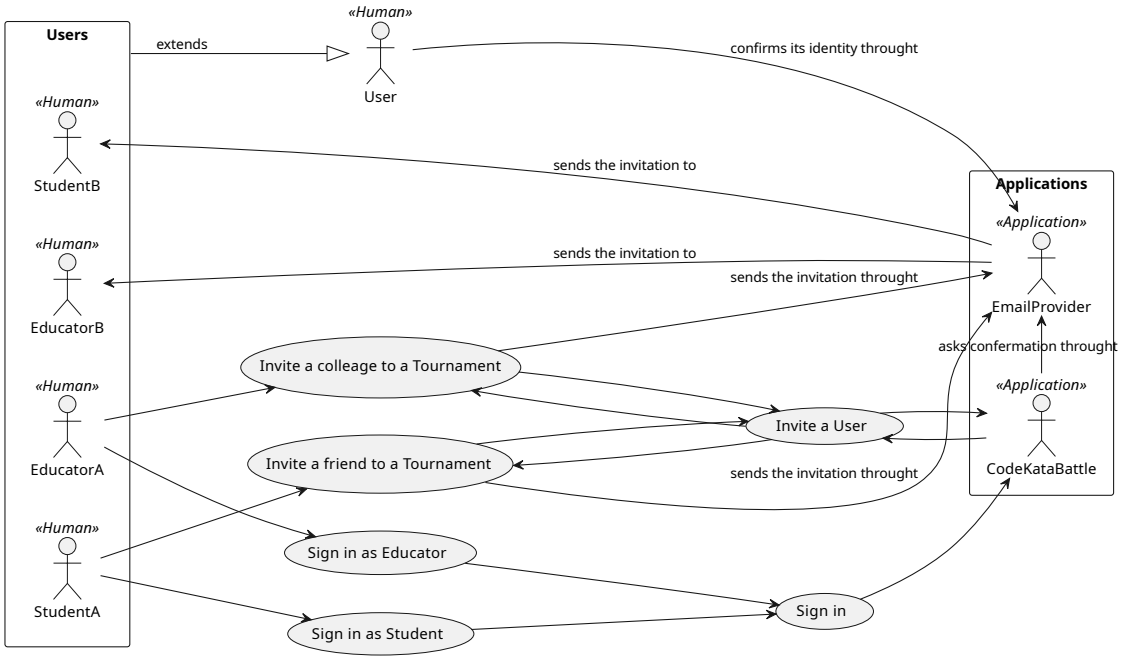
R14 The Platform allows registered Educators to add other Educators to a Tournament.

R15 The Platform allows Users to search for other Users.

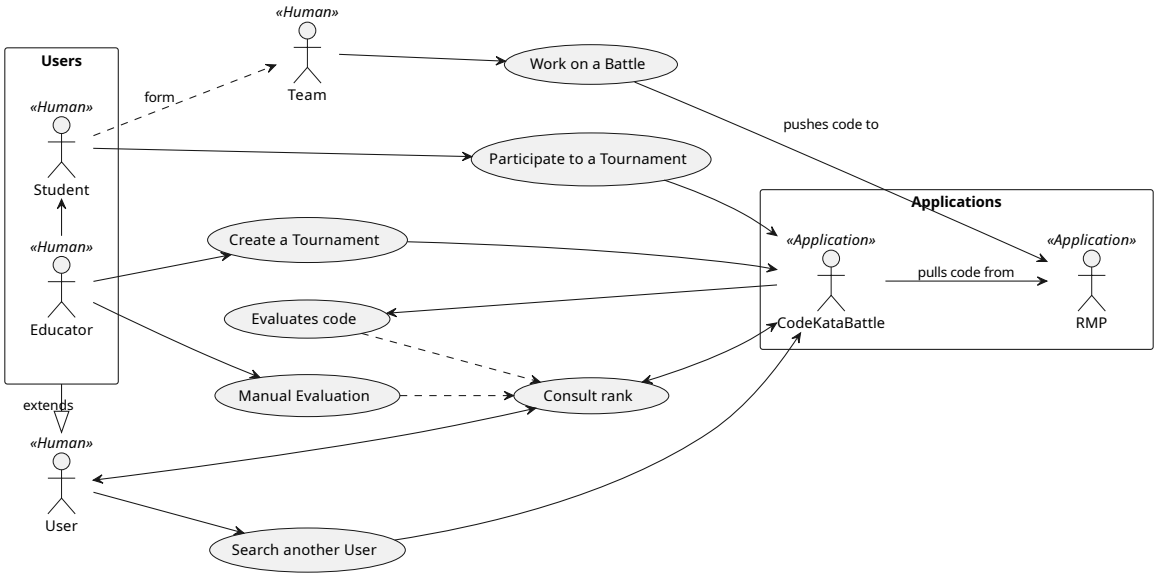
R16 The Platform interacts properly with different RMPs to acquire the latest versions of code uploaded by related Teams.

R17 The Platform awards Badges to deserving Students

3.2.1 Use case diagrams



Use case diagram of the login and the invitation



Use case diagram of the Battle and the Tournament

3.2.2 Use cases and associated sequence diagrams

Here follow Use Case tables followed by respective sequence diagrams, to be followed with the "Scenario - Use case - Requirement Mapping" Table, which shows the relationship between Scenarios, Use Cases and Requirements.

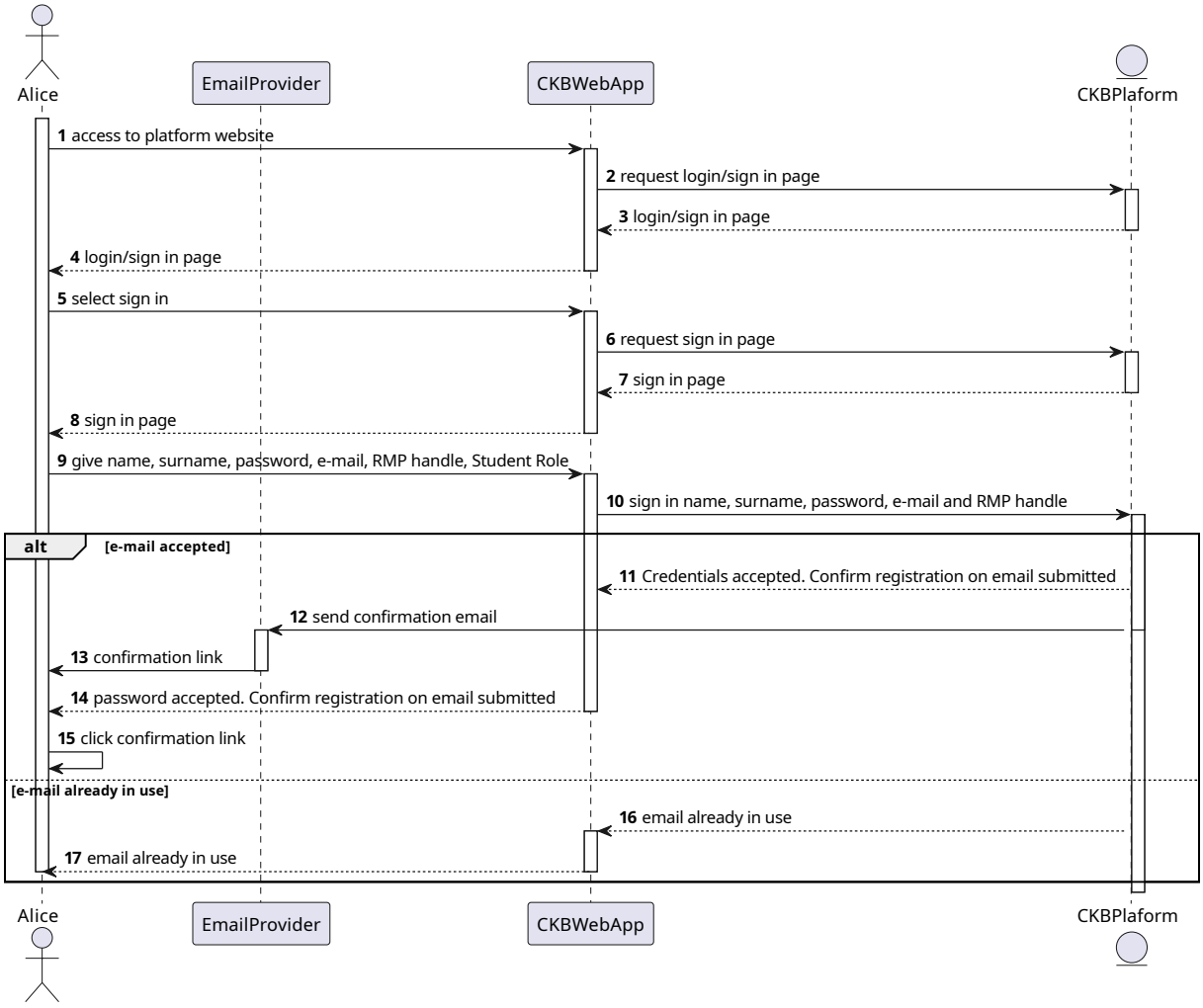
UC1 - User signs in to the Platform

Name	User signs in to the Platform
Actors	User
Entry Condition	User has a valid e-mail address and valid RMP handle
Event Flow	1 At Homepage click on "Sign in" button 2 System shows User the registration form 3 User fills form with data, caring to choose his role accordingly 4 User clicks on "Submit" button 5 Platform saves submitted information 6 Platform sends e-mail confirmation link to User 7 User confirms e-mail by clicking confirmation e-mail
Exit Condition	User correctly registered in the Platform, User needs to login to use the Platform
Exception	User provides an Email already in use.
Special Requirement	-

"UC1 - User signs in to the Platform" is a generalization of:
"UC1a - Student signs in to the Platform" and
"UC1b - Educator signs in to the Platform".

UC1a - Student signs in to the Platform

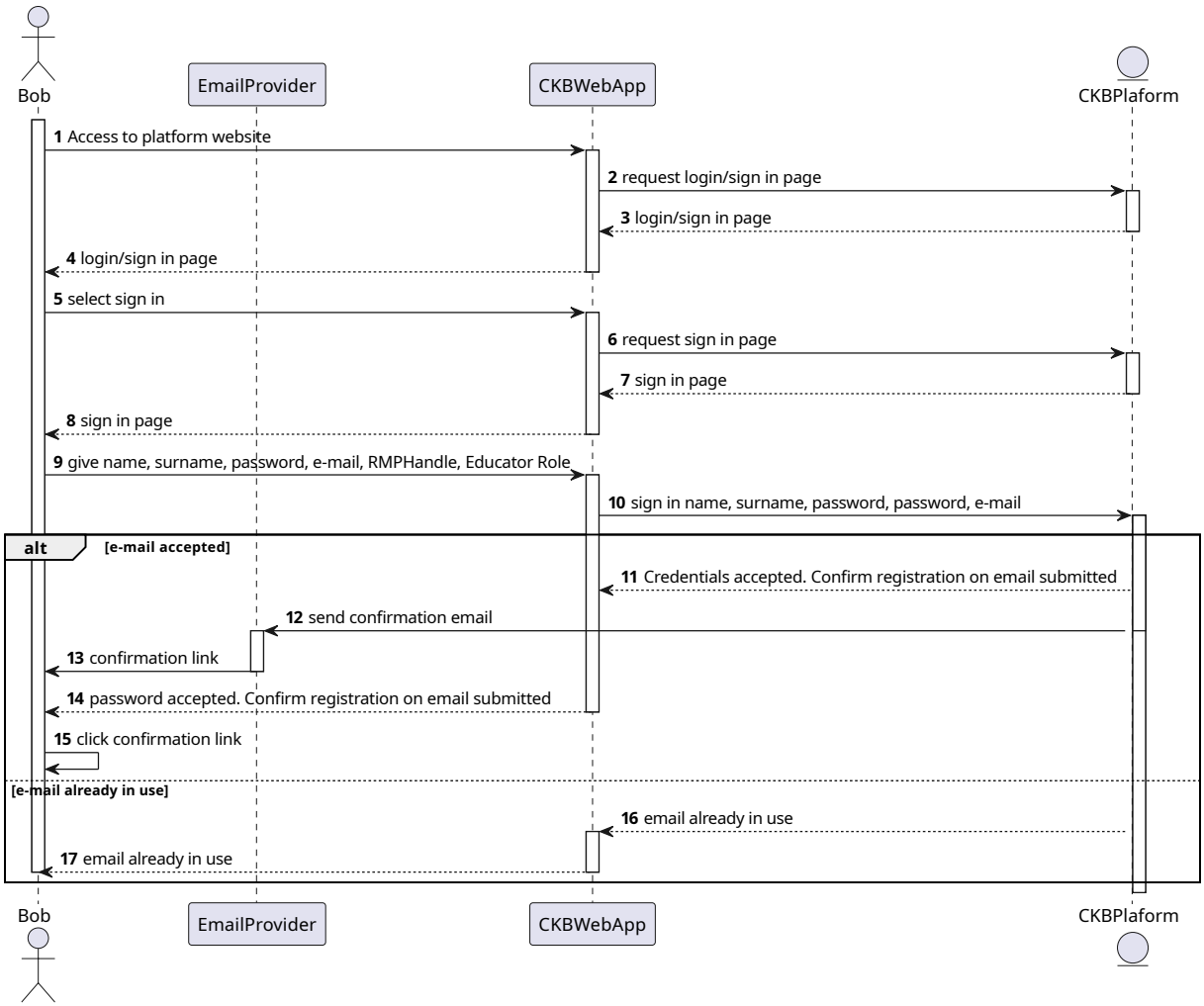
Name	Student signs in to the Platform
Actors	Student
Entry Condition	Same as UC1 - User signs in to the Platform
Event Flow	Same as UC1 - User signs in to the Platform with exception of 3 Student chooses Student role while filling in form
Exit Condition	Same as UC1 - User signs in to the Platform
Exception	Same as UC1 - User signs in to the Platform
Special Requirement	Same as UC1 - User signs in to the Platform



UC1, UC1a, Sequence Diagram - Student signs in to the Platform

UC1b - Educator signs in to the Platform

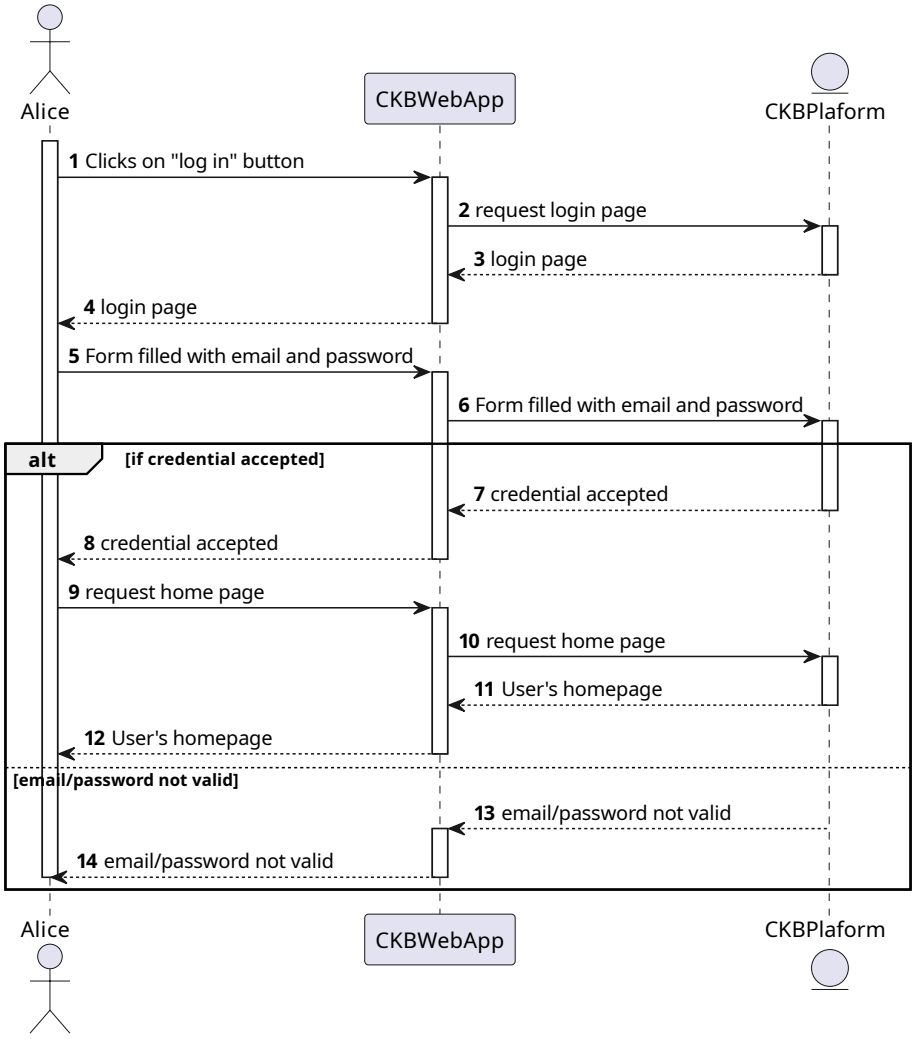
Name	Educator signs in to the Platform
Actors	Educator
Entry Condition	Same as UC1 - User signs in to the Platform
Event Flow	Same as UC1 - User signs in to the Platform with exception of 3 Educator chooses Educator role while filling in form
Exit Condition	Same as UC1 - User signs in to the Platform
Exception	Same as UC1 - User signs in to the Platform
Special Requirement	Same as UC1 - User signs in to the Platform



UC1, UC1b Sequence Diagram - Educator signs in to the Platform

UC2 - User logs in to the Platform

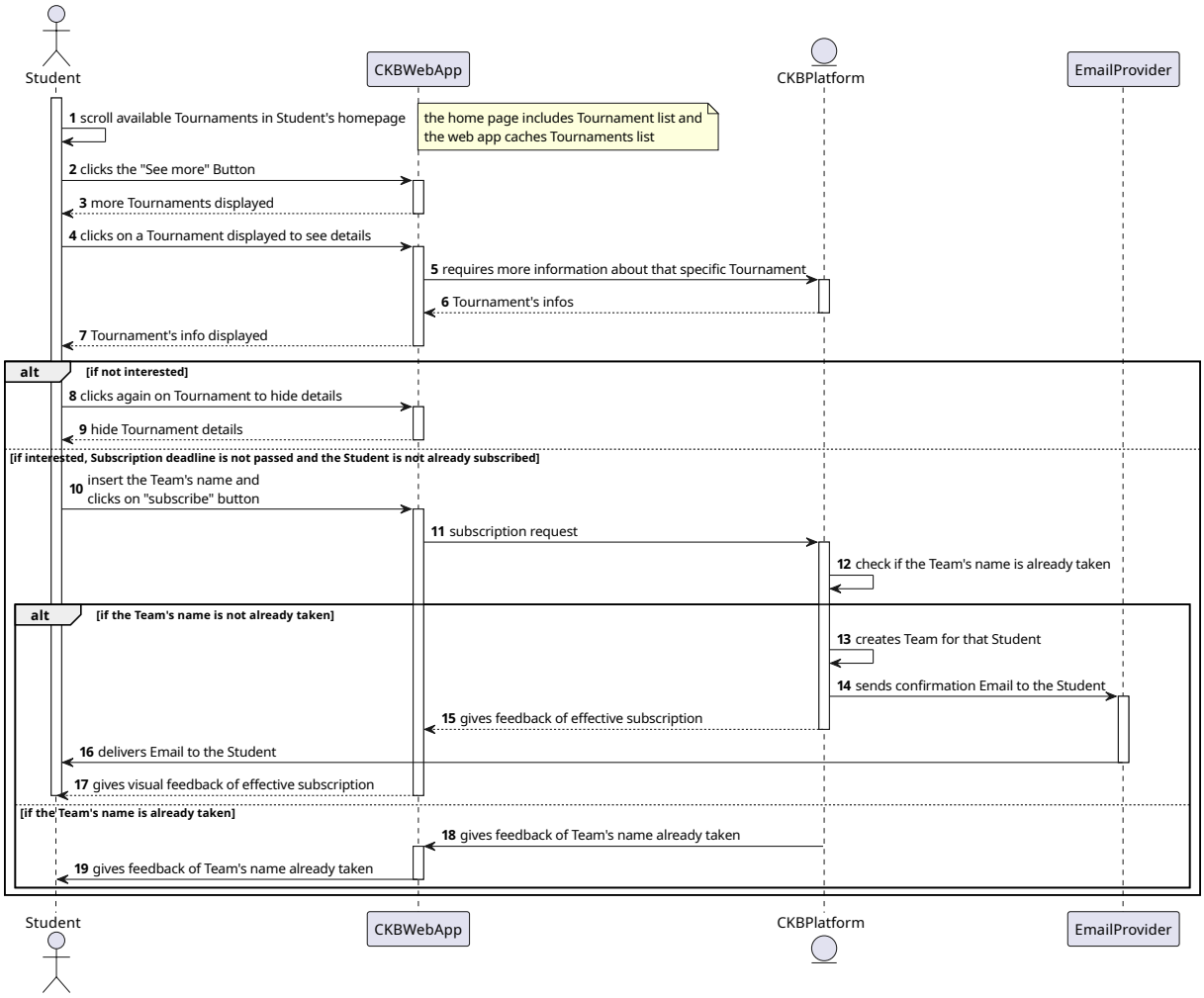
Name	User logs in to the Platform
Actors	User
Entry Condition	User has valid credentials corresponding to an account
Event Flow	1 User is on web app welcome page 2 User clicks on "Log in" button 3 Platform shows User login form 4 User fills the form accordingly 5 User clicks on login button 6 Platform shows User's homepage
Exit Condition	User finds in a valid session in his homepage free to do whatever he/she can do in the Platform
Exception	User fills form with wrong User data, Platform visually notifies about the error
Special Requirement	-



UC2 Sequence Diagram - User logs in to the Platform

UC3 - Student subscribes to Tournament

Name	Student subscribes to Tournament
Actors	Student
Entry Condition	Tournament is in "ONLINE" state and Student is logged in to the Platform
Event Flow	1 Student is in its home page and is presented with a list of available Tournaments 2 Student scrolls through the list of available Tournaments 3 Student can ask to see more Tournaments via the "See more" button 4 Student clicks on Tournament entry to see Tournament details 5 Student clicks again on Tournament if not interested to hide details 6 Student inserts the Team's name and clicks on "subscribe" button to subscribe to Tournament 7 Platform registers Tournament subscription 8 Platform creates new Team for that Student 9 Platform sends confirmation e-mail to Student 10 Platform gives visual feedback on web page of effective subscription
Exit Condition	Student is subscribed to Tournament and knows it
Exception	Subscription deadline is passed, Student cannot subscribe to Tournament. If a Student is already subscribed to a Tournament, he/she cannot click "subscribe" button.
Special Requirement	-



UC3 Sequence Diagram - Student subscribes to the Tournament

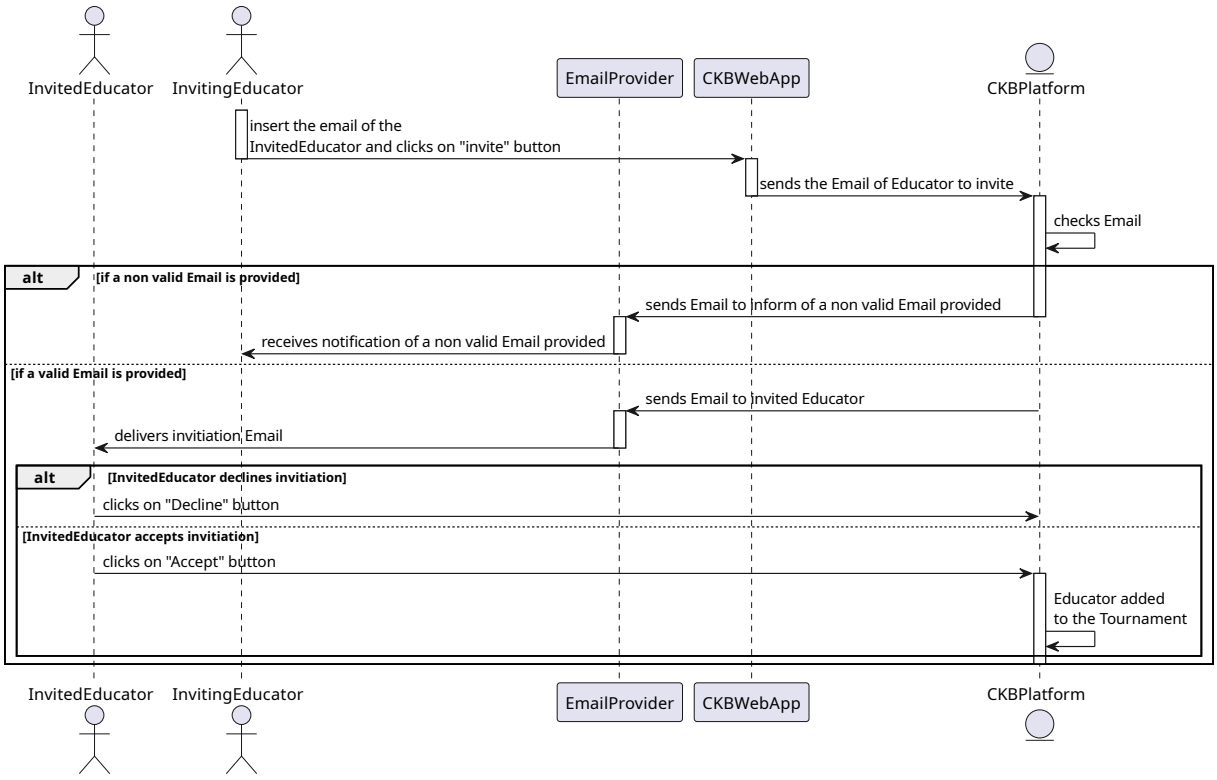
UC4 - User invites other User to collaborate

Name	User invites other User to collaborate
Actors	Inviting User, invited User
Entry Condition	Inviting User knows email of invited User Tournament is in the right state to invite User
Event Flow	1 Inviting User insert the Invited User email and clicks on "invite" button 2 System sends invite e-mail notification to invited User 3 Invited User clicks on "Accept" button to accept invitation 4 Invited User clicks on "Decline" button to decline invitation, or ignores it
Exit Condition	If invited User accepts invitation, invited User can collaborate with inviting one, else as nothing happened
Exception	Invited User e-mail address can be: non-existent, non-registered, of User of different role. In this case Inviting User get's notified of the error via e-mail.
Special Requirement	-

"UC4 - User invites other User to collaborate" is a generalization of:
"UC4a - Educator invites other Educator to co-manage Tournament" and
"UC4b - Student invites other Student to participate in the Tournament as a Team".

UC4a - Educator invites other Educator to co-manage Tournament

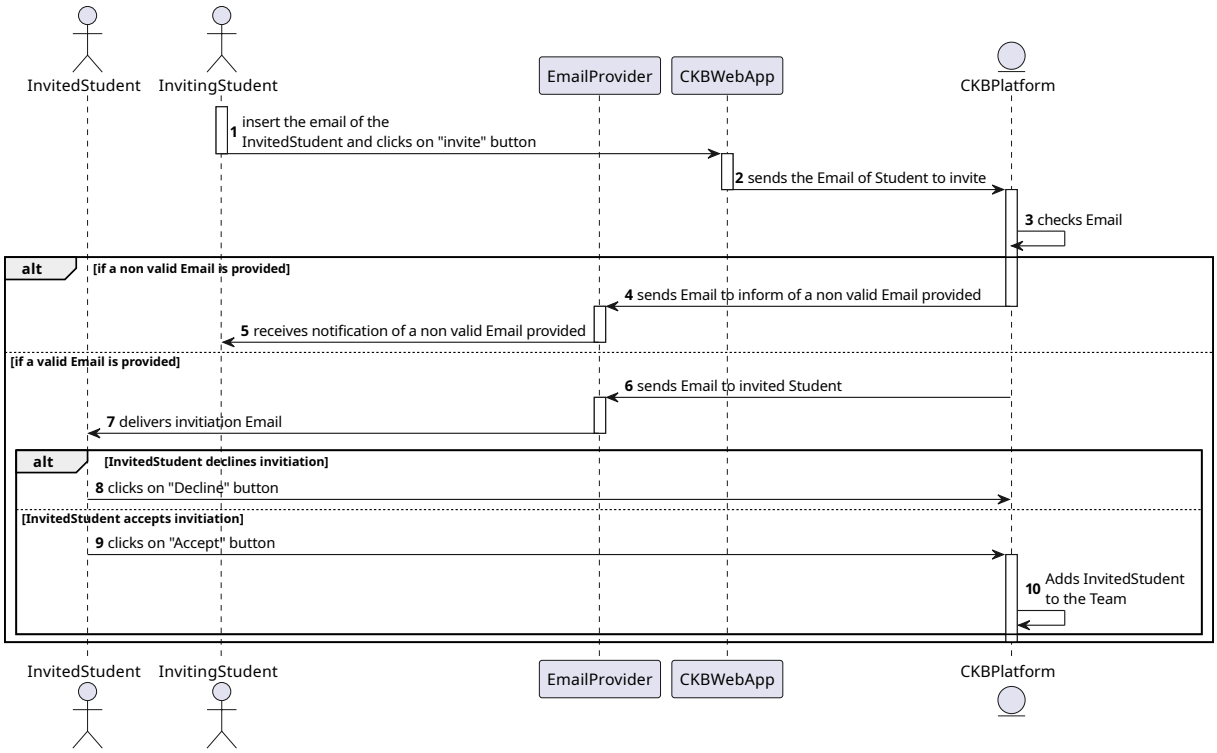
Name	Educator invites other Educator to co-manage Tournament
Actors	Inviting Educator, invited Educator
Entry Condition	Same as UC4 - User invites other User to collaborate Inviting Educator is managing a Tournament Tournament is in "CREATED" state
Event Flow	1 Educator is in the managing page of the Tournament Event flow follows father's UC4 - User invites other User to collaborate event flow
Exit Condition	Invited Educator can manage the Tournament
Exception	Adding to UC4 - User invites other User to collaborate, the email can be of an user that is already managing that same tournament.
Special Requirement	-



UC4, UC4a Sequence Diagram - Educator invites Educator to collaborate

UC4b - Student invites other Student to participate in the Tournament as a Team

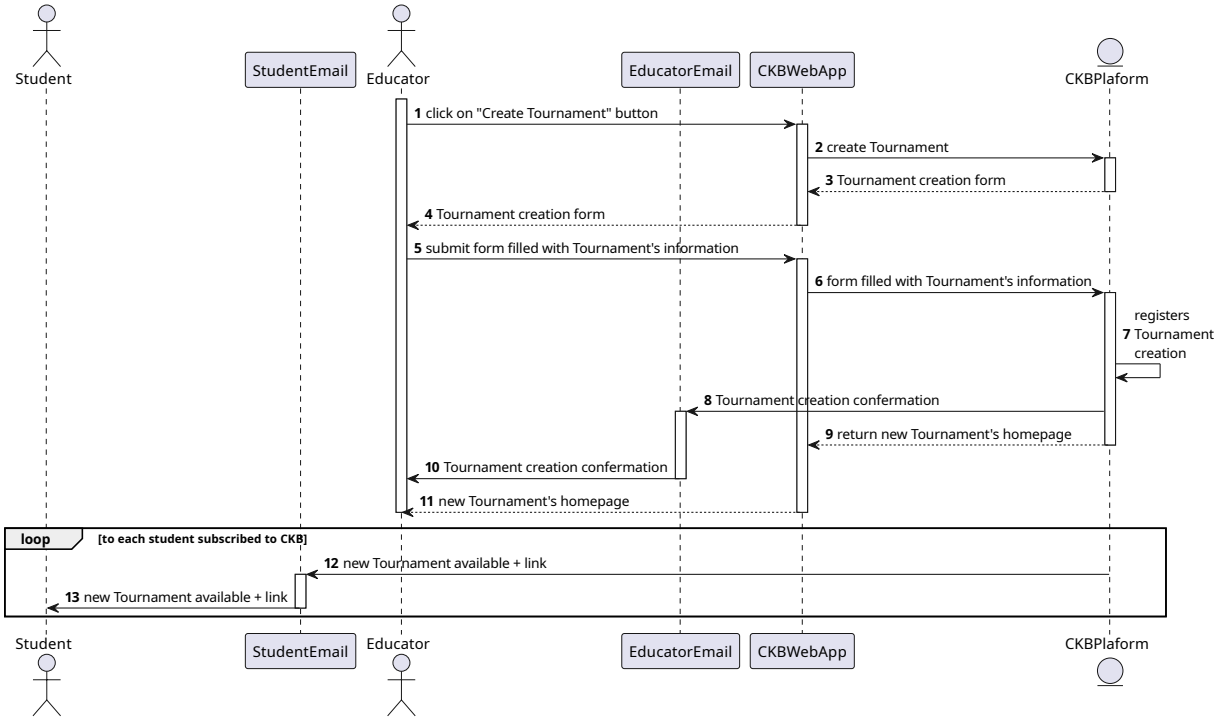
Name	Student invites other Student to participate in the Tournament as a Team
Actors	Inviting Student, invited Student
Entry Condition	Same as UC4 - User invites other User to collaborate Inviting Student is part of a team subscribed to a Tournament Tournament is in "ONLINE" state
Event Flow	1 Student is in the Tournament detail page Event flow follows father's UC4 - User invites other User to collaborate event flow
Exit Condition	If invited Student accepts invitation, invited User is part of inviting User's team
Exception	Same as UC4 - User invites other User to collaborate, with the verification of no participance to the Team in which he/she is invited or any other Team
Special Requirement	-



UC4, UC4b Sequence Diagram - Student invites Student to collaborate

UC5 - Educator creates Tournament

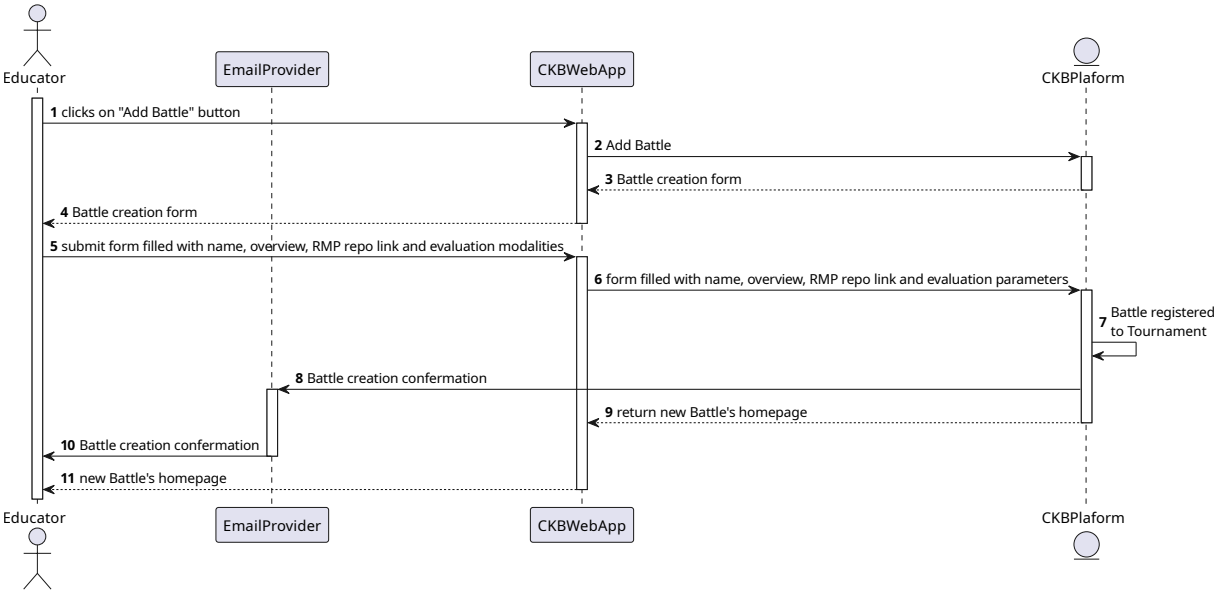
Name	Educator creates Tournament
Actors	Educator
Entry Condition	Educator has a clear plan of how to run the Tournament
Event Flow	1 Educator is in its homepage 2 Educator clicks on "Create Tournament" button 3 Platform shows Tournament creation form to the Educator 4 Educator fills in the form 5 Educator clicks on "Submit" button 6 Platform registers Tournament creation 7 Platform sends e-mail notification of success to Educator 8 Platform shows the new Tournament's homepage to the Educator 9 Platform sends e-mail notification of Tournament existence to Students
Exit Condition	Tournament is created and available for Students to subscribe to it
Exception	-
Special Requirement	-



UC5 Sequence Diagram - Educator creates Tournament

UC6 - Educator adds Battle to Tournament

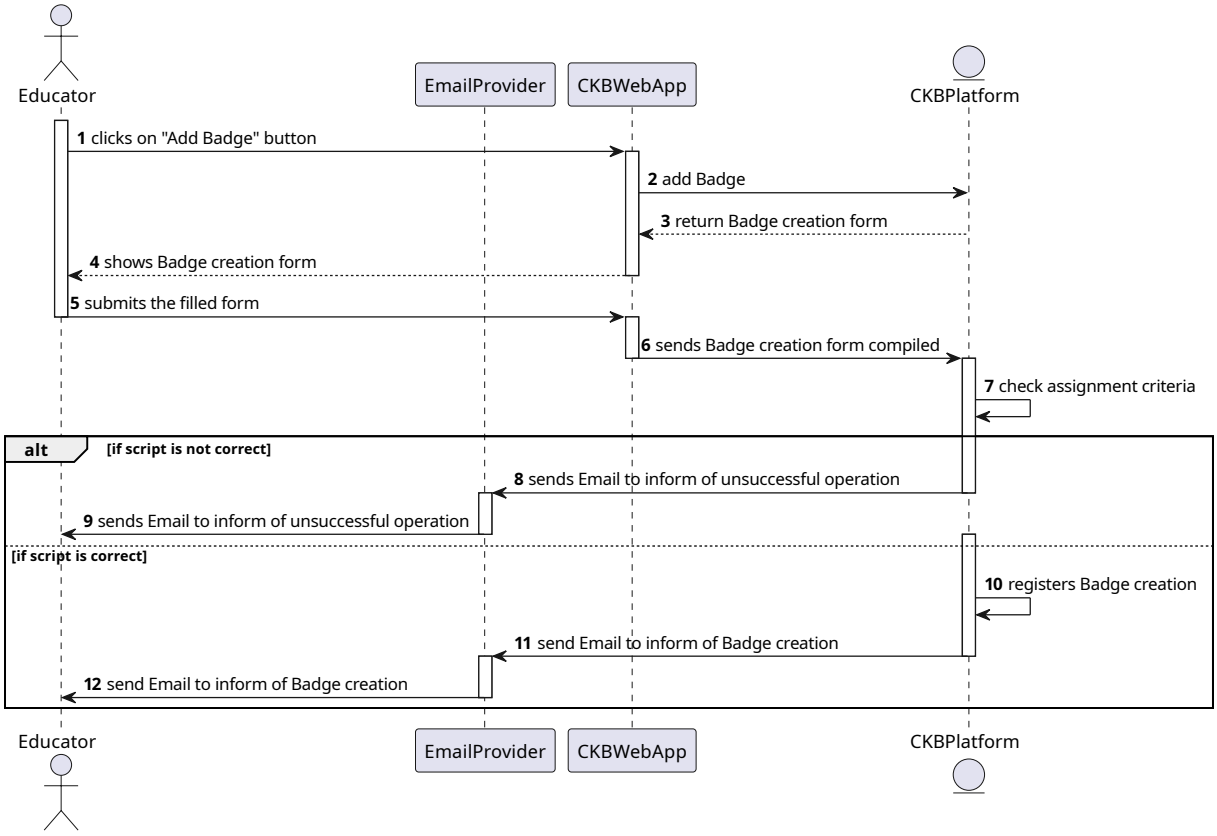
Name	Educator adds Battle to Tournament
Actors	Educator
Entry Condition	Educator manages a Tournament Tournament is in "CREATED" state Educator has prepared: - Repository to be available for Teams to fork - Repository containing evaluation test suite
Event Flow	1 Educator is in Tournament management page 2 Educator clicks on "Add Battle" button 3 Platform shows a Battle creation form 4 Educator fills the form with link to repository in RMP 5 Educator clicks on "Submit" button 6 Platform registers Battle addition to Tournament 7 Platform sends confirmation e-mail to the Educator 8 Platform shows the new Battle's homepage to the Educator
Exit Condition	Platform is ready to test Battles and code is available for Students to fork
Exception	-
Special Requirement	-



UC6 Sequence Diagram - Educator adds a Battle to the Tournament

UC7 - Educator creates Badge

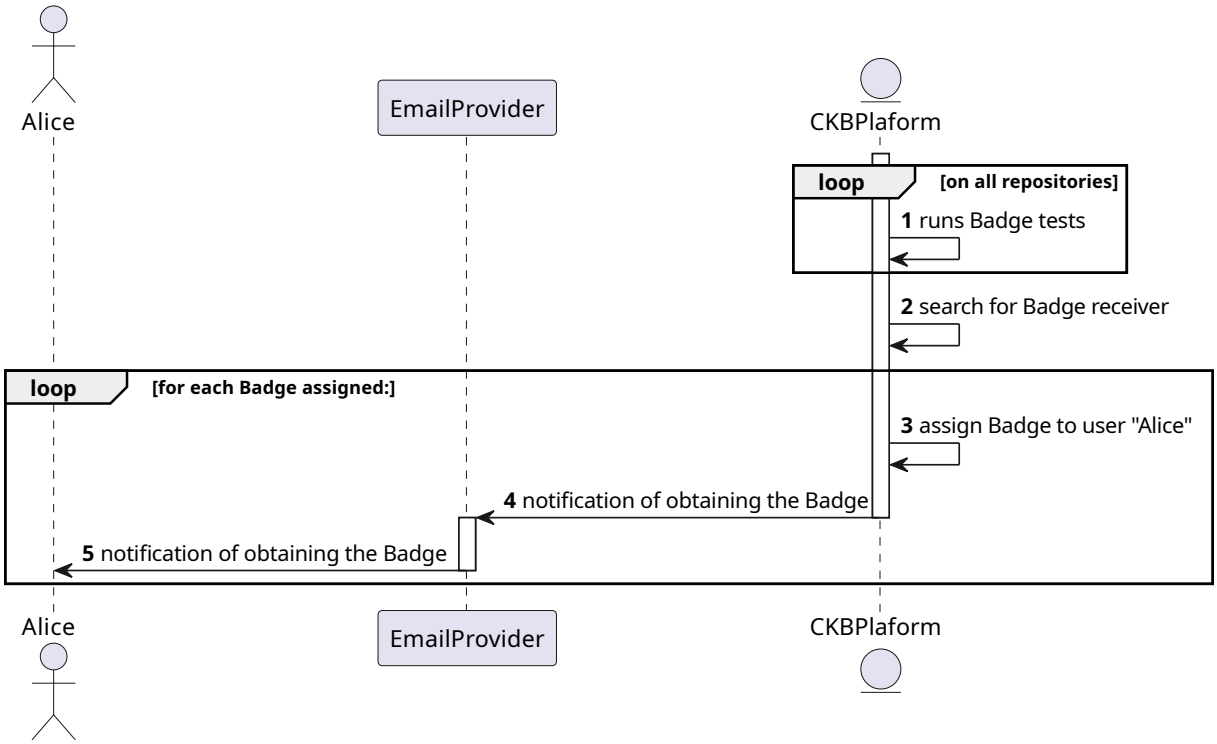
Name	Educator creates Badge
Actors	Educator
Entry Condition	Educator is managing a Tournament Tournament is in "CREATED" state
Event Flow	1 Educator is in Tournament management page 2 Educator clicks on "Add Badge" button 3 Platform shows Badge creation form to Educator 4 Educator fills the form with Badge image, description and assignment criteria 5 Educator clicks on "Submit" button 6 Platform checks Badge requirements script correctness 7 Platform registers Badge creation 8 Platform sends a confirmation e-mail to Educator
Exit Condition	Every User is able to see the existence of the Badge in the Tournament detail page
Exception	If requirements script isn't correct, Badge information is dropped and a notification e-mail is sent to the Educator describing the issue
Special Requirement	-



UC7 Sequence Diagram - Educator creates Badge

UC8 - Student gets awarded with Badge

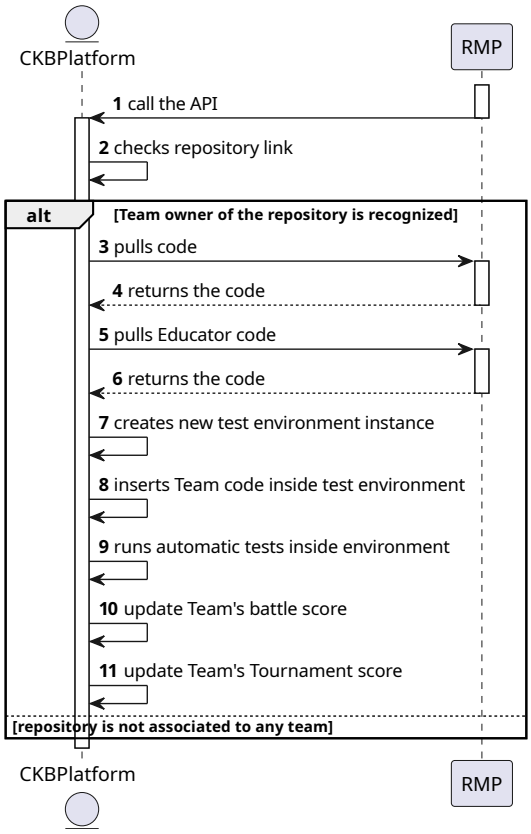
Name	Student gets awarded with Badge
Actors	Student
Entry Condition	Student is part of a Team, and has contributed to Team's repository Tournament is in "OFFLINE" state
Event Flow	1 System runs Badge tests on all repositories 2 Badge tests give a list of RMP handles and e-mail addresses to which assign corresponding Badge 3 System searches for corresponding accounts 4 System assigns Badges to each User found 5 System sends e-mail notification to chosen Students
Exit Condition	Random Platform User can see Badge in the specific Student profile page
Exception	-
Special Requirement	Commit User data is consistent with CKB Platform User data



UC8 Sequence Diagram - Student gets awarded with Badge

UC9 - Team gets code evaluated

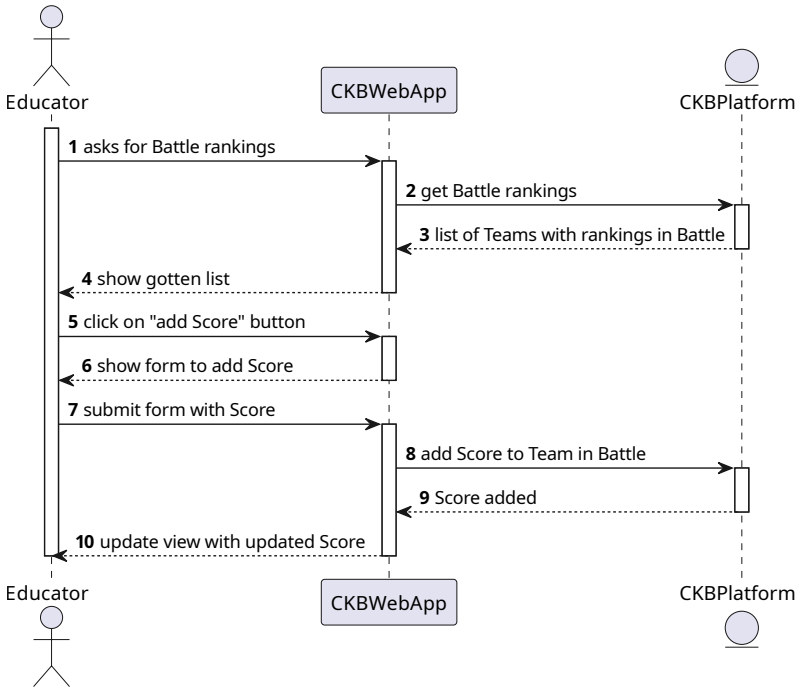
Name	Team gets code evaluated
Actors	RMP
Entry Condition	Team Student pushes code to RMP Tournament is in "STARTED" state Battle is in "ACCESSIBLE" state
Event Flow	1 RMP call CKB's API to notify Platform of an update to repository 2 Platform checks the repository link and recognize the Team 3 Platform pulls the code from repository 4 Platform pulls the code from the educator repository 5 Platform creates new test environment instance 6 Platform inserts Team and Educator code inside newly created test environment 7 Platform runs automatic tests inside environment 8 Test ends with a new Battle score which gets assigned to corresponding Team 9 Platform updates Team's Tournament score
Exit Condition	Users are able to see newly created score inside Battle leaderboards
Exception	Repository is not associated to any Team of the Tournament: tests are not run
Special Requirement	Automatic action is run shortly after new push to the RMP



UC9 Sequence Diagram - Team gets code awarded

UC10 - Team gets code evaluated by Educator

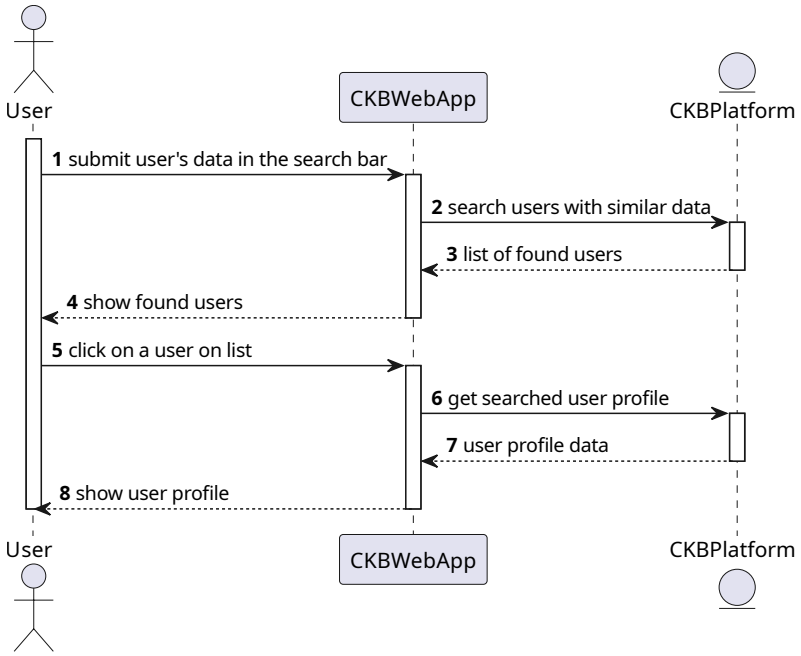
Name	Educator manually evaluates code
Actors	Educator
Entry Condition	Tournament is in "STARTED" state Battle is in "ACCESSIBLE" state
Event Flow	1 Educator goes to the Tournament management page 2 Educator goes to the Battle page and look for the Battle ranking list 3 In the row associated to the score of the Team, Platform shows "Add score" button 4 Educator clicks on "Add score" button 5 Platform shows add score form to Educator 6 Educator fills form with his score 7 Educator clicks on "Submit" button 8 Platform registers the score 9 Platform updates Team's Tournament score
Exit Condition	Users are able to see newly created score inside battle leaderboards
Exception	-
Special Requirement	Educator has been invited to the RMPrepo, and can see the code of the team, and has seen it before scoring the students



UC10 Sequence Diagram - Educator manually evaluates code

UC11 - User searches for other Users

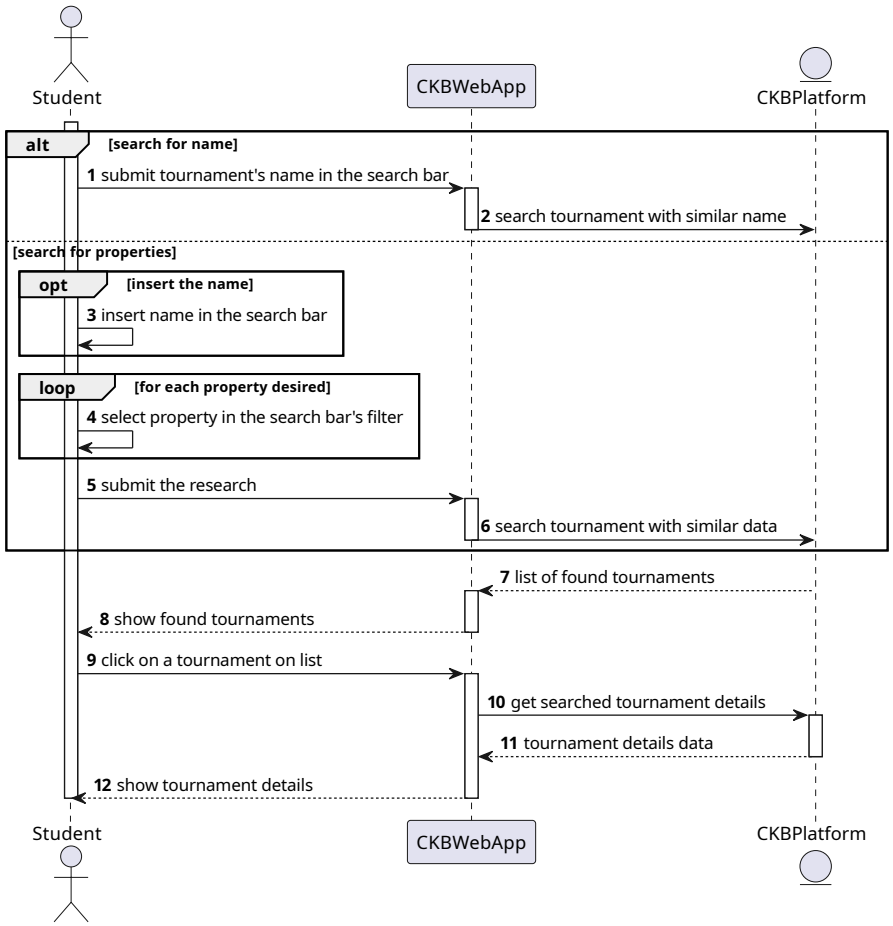
Name	User searches for other Users
Actors	User
Entry Condition	User is in his homepage
Event Flow	1 Platform shows in User's homepage a search bar to search for Users 2 User writes in search bar a string representing one of some other User's data (like name, surname, e-mail address or RMP handle) 3 User clicks on "Search" button 4 Platform shows a list of Users with data compatible to the string given by the User 5 Platform keeps showing search bar to the User to be able to start another search 6 User clicks on a User on the list to see details
Exit Condition	User can look other User's profile
Exception	-
Special Requirement	-



UC11 Sequence Diagram - User searches for other Users

UC12 - Student searches for a Tournament

Name	Student searches for a Tournament
Actors	Student
Entry Condition	Student is in his homepage
Event Flow	1 Platform shows in Student's homepage a search bar to search for Tournaments 2 Student writes in search bar a string representing the name of the Tournament 3 Student alternatively can filter Tournaments by selecting specific properties of it in the filter of the search bar 3 Student clicks on "Search" button 4 Platform shows a list of Tournaments with data compatible to the string or properties given by the Student 5 Platform keeps showing search bar to the Student to be able to start another search 6 Student clicks on a Tournament on the list to see details
Exit Condition	Student can look Tournament details
Exception	-
Special Requirement	-



UC12 Sequence Diagram - Student searches for a Tournament

3.2.3 Scenario - Use case - Requirement Mapping

Scenario	Use Case	Requirement
SC1, **SC4**	UC1 - User signs in to the Platform	**R1**
SC1	UC2 - User logs in to the Platform	**R2**
SC2	UC3 - Student subscribes to Tournament	**R2**, **R3**, **R5**, **R6**
SC3, **SC6**	UC4 - User invites other User to collaborate	**R6**, **R7**, **R14**
SC6	UC4a - Educator invites other Educator to co-manage Tournament	**R14**
SC3	UC4b - Student invites other Student to participate in the Tournament as a Team	**R6**, **R7**
SC5	UC5 - Educator creates Tournament	**R8**
SC11	UC6 - Educator adds Battle to Tournament	**R9**
SC13	UC7 - Educator creates Badge	**R11**
SC14	UC8 - Student gets awarded with Badge	**R17**
SC8, **SC9**	UC9 - Team gets code evaluated	**R12**, **R13**, **R16**
SC12	UC10 - Team gets code evaluated by Educator	**R10**
SC14	UC11 - User searches for other Users	**R15**
SC2	UC12 - Student searches for a Tournament	**R4**

3.3 Evaluation Requirements

The Platform perform two types of evaluation: the Battle evaluation and the Badge Evaluation.

3.3.1 Battle Evaluation

This Evaluation is performed automatically by the platform itself. The RMP Action signals the Platform of a new code update, which then proceeds to pull student's code and to pull also educator's code. This double pull is done to prevent students to tamper with tests, by keeping students code from the student's repository and merging it with educator's code from educator's repository. Once tests are finished, these return a value that is used as the score of the team in that battle. Finally, the shortly obtained score is used to update tournament score.

3.3.2 Badge Evaluation

This Evaluation is performed automatically by the platform once the Tournament reaches "OFFLINE" state. Badges can have assignment criteria based on factors different from once based purely on code (like maximum amount of commit, or maximum amount of points scored in a single battle). This brings the necessity to define syntactic rules so that the Educator can easily write them, and the platform can easily evaluate them.

3.4 Performance Requirements

The Platform, that can be described as a Web App, has to be able to manage multiple Users requests, constantly update rankings for related Tournaments, perform tests on provided code and assign related Score, all at the same time.

So, in order to satisfy all Requirements defined above, providing a good time response for each task, the Platform should have good time performance. Being a Web App, this implies a high concurrent programming implementation, to manage multiple requests at the same time, and a rapid related Data Storage System.

3.5 Design Constraints

3.5.1 Standard Compliance

First of all, the Platform has to respect privacy standards, such as GDPR for EU countries or similar regulations, in order to protect personal information provided by Users when they sign in, their code and identity across the Web App.

It is also important to respect internet protocols standards and their rules, in order to make the app run and communicate properly. Other standards are related to UI accessibility features and servers' power consumption.

3.5.2 Hardware Limitations

Hardware limitations in the RASD context are referred to User's machine capabilities and server's limits.

In fact, the devices used by both Students and Educators are constrained by memory capacity, CPU capabilities and GPU computational power. The first is essential to memorize the file to upload in RMP, run the software to access internet , to manage the Web App data itself and allow User to receive Emails. The second is fundamental to run the connection tasks and the site, receiving and elaborating inputs for the User and CKB's responses. Finally, GPU is essential to render UI and allows Users to read Emails via a proper Email Client App.

On server side, instead, is important for our App to have an organized storage to preserve all necessary data and enough computational power in our CPUs to manage multiple requests for each machine that composes CKB's server.

In fact, to satisfy all possible inputs from all possible Users, formulating for each one a response, and meet requirements, is critical to have a multiple machine architecture on our servers.

To complete, a multiple internet connection ports is fundamental, to make the Platform available on the internet and make it capable of comunication with Users, RMPs and Email Provider.

3.6 Software System Attributes

3.6.1 Reliability

The Platform described in this document should be an always available Web App.

Downtimes are meant to be as less as possible, and the overall system, in order to prevent evaluation, ranking, registration, search problems, should be designed with robustness in mind and with built-in mechanisms to manage situations that could compromise information, materials and commands assigned to the Platform. All of this performed transparently to the Users as much as possible.

3.6.2 Availability

The Platform should be able as much as possible, 24/7, 365 days per year.

The minimum availability rate should be 99%. An inferior percentage could compromise educational activities potentially damaging Students.

3.6.3 Security

Both Users' devices and servers communications must always take place only via encrypted channels, using appropriate cryptographic protocols such as SSL.

All operations and contributions to the Platform, except automated tasks performed on server side communicating with RMP and Email Provider, must be always explicitly approved by the Users.

3.6.4 Maintainability

In order to extend as much as possible the Platform's life cycle it should be designed with modularity in mind.

This approach will guarantee the interchanging of old modules to update or to substitute, possibly maintaining the same interfaces and core functionalities, at minimum cost.

3.6.5 Portability

The Platform, intended to be build, would be used independently of the OS provided with the machine.

In fact CKB can be accessed simply using just a web browser, on which maximum compatibility the developers should concentrate.

On User side, it would be sufficient to guarantee portability to build the Web App using common internet standards and protocols to make it usable on any web browser.

About RPM portability, the Platform should be provided with the right interfaces and modules to communicate with the largest amount of RMPs available, in order to acquire and run code evaluating it.

4 Formal Analysis Using Alloy

In this section we provide the Alloy model of the Platform. This model is useful to check the consistency of the system and to verify that the requirements are satisfied. Given the limitations of the modelling language, this model represents statically the Platform and Users, showing their relationships and the constraints that they must satisfy. The state of the system that is represented is the one at which all represented Tournaments have reached the state of "STARTED" or "CLOSED", so where Tournament Badges can be assigned, Teams got their code evaluated, and the Tournament score has been computed. For reducing the complexity some details about Battles and Tournaments have been omitted (like subscription deadline, programming language and duration) since the focal point of view of this model is evaluating how the main entities of the Platform, adding also the RMP, relate between each other in the perspective of the Tournament, Battle and code.
As for how is structured an Alloy model, firstly will be shown the signatures, then the facts, assertions (followed by check results) and predicates (followed by model found).

4.1 Signatures

```
sig Name{}

sig Surname{}

sig Email{}

sig RMPHandle{
    rmp: one RMP,
}

abstract sig User {
    name: one Name,
    surname: one Surname,
    email: disj one Email,
    rmpHandle: disj one RMPHandle
}

sig Educator extends User{}

sig Student extends User{
    badges: set Badge
}

one sig Platform{   //the platform is unique
    students: set Student,
    educators: set Educator,
    tournaments: set Tournament
}

sig Team{
    students: some Student,
    tournamentScore: one TournamentScore,
    battleScore: set BattleScore,
    repos: some RMPRepo
}

abstract sig Score{
    points: one Int
}{
    points ≥ 0      //the score is positive
}

sig BattleScore extends Score{}

sig TournamentScore extends Score{}

sig RMPRepo{
    rmp: one RMP
```



```
}

sig Battle{
    repo: disj one RMPRepo,
    scores: some BattleScore
}

sig Rules{}

sig Badge{
    educator: one Educator,
    name: one Name,
    rules: disj one Rules
}

sig Tournament{
    battles: set Battle,
    badges: set Badge,
    educators: some Educator,
    teams: some Team,
    scores: some TournamentScore
}

sig RMP{}
```

4.2 Facts

```
// User facts

//a surname always has a user
fact SurnameBelongsToUser{
    all s: Surname | s in User.surname
}

//an email always has a user
fact EmailBelongsToUser{
    all e: Email | e in User.email
}

//a RMP handle always has a user
fact RMPHandleBelongsToUser{
    all e: RMPHandle | e in User.rmpHandle
}

//there cannot exist a rmp handle shared between to users
fact RMPHandleIsPersonal{
    all disj u1, u2: User | u1.rmpHandle ≠ u2.rmpHandle
}

//all the students, educators and tournaments belongs to the platform
fact AllConnectedToPlatform {
    all s: Student | s in Platform.students
    all e: Educator | e in Platform.educators
    all t: Tournament | t in Platform.tournaments
}

//there cannot exists a badge with the same name of a user
fact peopleNameNoObjectName{
    all u:User, b:Badge | u.name ≠ b.name
}

// Tournament facts

//all tournaments have at least one battle
fact tournamentHasAtLeastOneBattle{
    all t: Tournament | #t.battles > 0
}

//there cannot exists a student in two different teams in the same tournament
fact NoStudentIsInTwoTeamsInSameTournament {
    all to: Tournament | all disj t1, t2 :to.teams | all s1 : t1.students | all s2 : t2.
        ↪ students | s1 ≠ s2
```



```
}

//a team can exists only in the context of a tournament
fact TeamIsPartOfOneTournament{
    all t: Team | one to: Tournament | t in to.teams
}

//get all RMP repos present in the context of a tournament, the team ones and Battle ones
fun allReposFromTournament[t: Tournament]: set RMPRepo{
    {r: RMPRepo | r in t.battles.repo} + {r: RMPRepo | r in t.teams.repos}
}

//get all RMP handles present in the context of a tournament, the student ones and
    ↪ educator ones
fun allRmpHandlesFromTournament[t: Tournament]: set RMPHandle{
    {r: RMPHandle | r in t.educators.rmpHandle} + {r: RMPHandle | r in t.teams.students.
        ↪ rmpHandle}
}

// all tournament repos and rmp handles are in the same rmp
fact uniqueRMPinATournament{
    all to: Tournament | all r :allReposFromTournament[to]| all h :
        ↪ allRmpHandlesFromTournament[to]| r.rmp = h.rmp
}

//a RMP repo belongs to only a team
fact allTeamsDontShareRepos{
    all disj t1, t2 :Team | all r1: t1.repos | all r2: t2.repos | r1 ≠ r2
}

//the RMP repo of the battle isn't use from the teams
fact noTeamWorksOnBattleRepo{
    all t: Team | all b: Battle | all r: t.repos | r ≠ b.repo
}

//a RMP repo belongs to a Team or to a Battle
fact noRepoAlone{
    all r: RMPRepo | (r in Team.repos) or (r in Battle.repo)
}

// Battle facts

//there cannot exists a battle that isn't part of a tournament
fact battleIsPartOfOneTournament{
    all b1: Battle | one t1: Tournament | b1 in t1.battles
}

// Score facts

//a tournament score is associated uniquely to a team
fact TeamTournamentScoreIsUnique{
    all t: TournamentScore | t in Team.tournamentScore
    all disj t1, t2: Team | t1.tournamentScore ≠ t2.tournamentScore
}

//there cannot exists a battle score without a team
fact allBattlescoreBelongstoOneTeam{
    all bs:BattleScore | one t:Team | bs in t.battleScore
}

//there cannot exists a battle score without a battle
fact allBattlescoreBelongstoOneBattle{
    all bs:BattleScore | one b:Battle | bs in b.scores
}

//a battle score is uniquely associated to only one battle and only one team
fact allBattlescoreBelongstoOneTeamAndOneBattle{
    all bs:BattleScore | one to: Tournament | one t:to.teams | one b:to.battles | bs in t
        ↪ .battleScore and bs in b.scores
}

//there cannot exists a tournament score without a team
```



```
fact allTournamentscoreBelongstoOneTeam{
    all ts:TournamentScore | one t:Team | ts in t.tournamentScore
}

//there cannot exists a tournament score without a battle
fact allTournamentscoreBelongstoOneBattle{
    all ts:TournamentScore | one to:Tournament | ts in to.scores
}

//a tournament score is uniquely associated to only one tournament and only one team
fact allTournamentscoreBelongstoOneTeamAndOneBattle{
    all ts:TournamentScore | one to: Tournament | one t:to.teams | ts in t.
        ↪ tournamentScore and ts in to.scores
}

//there cannot exists a team without a tournament score
fact allTeamsHaveATournamentScore{
    all t:Team | one ts:TournamentScore | ts in t.tournamentScore
}

//each team in a tournament has a battle score for each battle in the same tournament
fact allBattleTeamCoupleHaveABattleScore{
    all to: Tournament | all t:to.teams | all b:to.battles | one bs:BattleScore| bs in t.
        ↪ battleScore and bs in b.scores
}

//the tournament score of a team is equal to the sum of all the battle score of that team
fact tournamentScoreIsSumOfBattleScores{
    all to: Tournament | all t:to.teams | all ts: t.tournamentScore | ts.points = sum[t.
        ↪ battleScore.points]
}

// Badge facts

//there cannot exists a set of rules without a badge connected
fact noAloneDescriptions{
    all r: Rules | r in Badge.rules
}

//there cannot exists a name without a badge or user
fact noAloneNames{
    all n: Name | (n in User.name) or (n in Badge.name)
}

//all the badge belongs to tournaments
fact noBadgeWithoutTournament{
    all b: Badge | b in Tournament.badges
}
```

4.3 Assertions

```
assert newTournament{
    no t:Tournament | t not in Platform.tournaments
}
check newTournament for 10  //VALID

//GP4: Allow Educators to create battles
assert newBattle{
    no b:Battle | all t:Tournament | b not in t.battles
}
check newBattle for 6 //VALID

//GP5: Allow Educators to create badges
assert newBadge{
    (no b:Badge | all t:Tournament | b not in t.badges) or (all b: Badge | one e:
        ↪ Educator | e = b.educator)
}
check newBadge for 6 //VALID

//GS2: Allow Students to be rewarded for special achievement
assert studentReceivesSpecialAchievements{
    all s: Student | #s.badges ≥ 0
```



```
}
check studentReceivesSpecialAchievements for 6 //VALID

//GS4: Allow Students to have work evaluated
assert haveWorkEvaluated{
    all to: Tournament | all t: to.teams | #t.tournamentScore = 1 and #t.battleScore = #
        ↪ to.battles
}
check haveWorkEvaluated for 6 //VALID

assert noBattlesHaveSameRepo{
    all disj b1, b2: Battle | b1.repo ≠ b2.repo
}
check noBattlesHaveSameRepo for 6 //VALID
```

Here the assertion results:

```
Executing "Check newTournament for 10"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20 Mode=batch
95882 vars. 2450 primary vars. 240333 clauses. 322ms.
No counterexample found. Assertion may be valid. 54ms.

Executing "Check newBattle for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=batch
121684 vars. 3416 primary vars. 299209 clauses. 57ms.
No counterexample found. Assertion may be valid. 6ms.

Executing "Check newBadge for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=batch
147484 vars. 4388 primary vars. 358199 clauses. 56ms.
No counterexample found. Assertion may be valid. 6ms.

Executing "Check studentReceivesSpecialAchievements for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=batch
147484 vars. 4388 primary vars. 358199 clauses. 49ms.
No counterexample found. Assertion may be valid. 0ms.

Executing "Check haveWorkEvaluated for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=batch
173443 vars. 5360 primary vars. 417705 clauses. 60ms.
No counterexample found. Assertion may be valid. 108ms.

Executing "Check noBattlesHaveSameRepo for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=batch
199347 vars. 6332 primary vars. 476904 clauses. 83ms.
No counterexample found. Assertion may be valid. 14ms.

6 commands were executed. The results are:
#1: No counterexample found. newTournament may be valid.
#2: No counterexample found. newBattle may be valid.
#3: No counterexample found. newBadge may be valid.
#4: No counterexample found. studentReceivesSpecialAchievements may be valid.
#5: No counterexample found. haveWorkEvaluated may be valid.
#6: No counterexample found. noBattlesHaveSameRepo may be valid.
```


4.4 Predicates

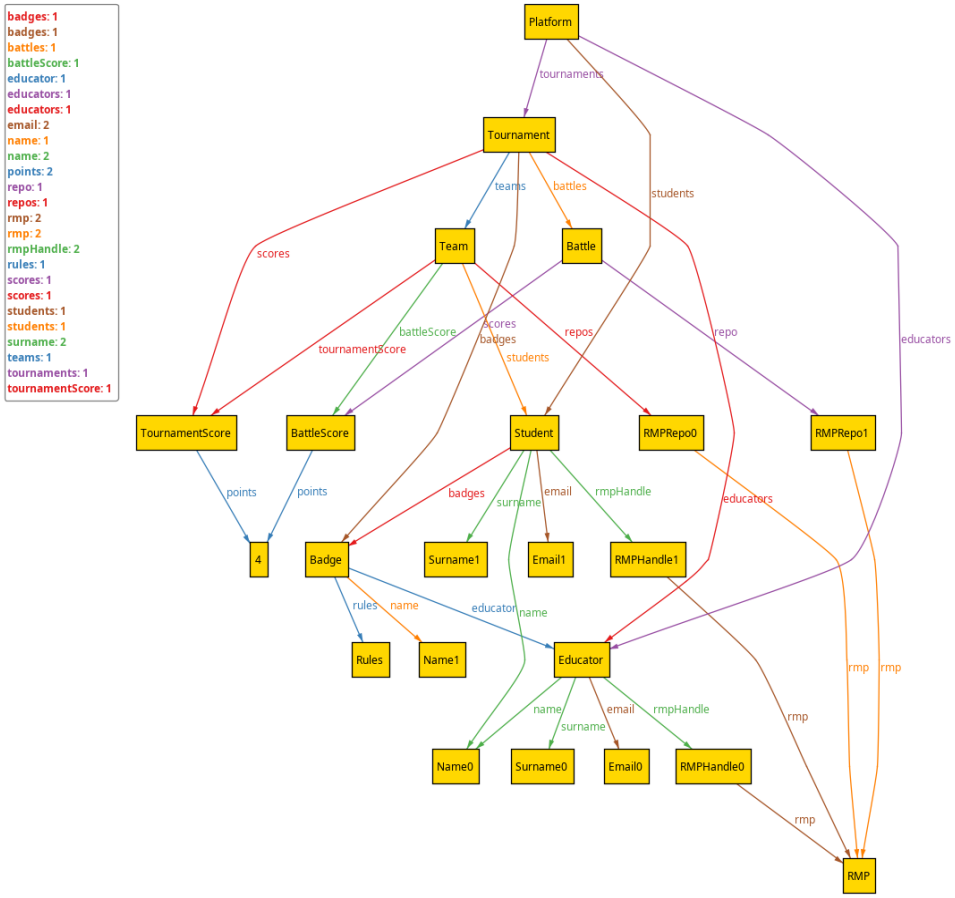
The predicates here are used to represent the system in different states to highlight different aspects of its behavior.

4.4.1 World 1

This is a simple world, where all entities are present once. The scope of this world is to show relationships between entities.

```
pred world1{
    #Platform.students = 1
    #Platform.educators = 1
    #Platform.tournaments = 1
    #Tournament = 1
    #Tournament.educators = 1
    #Team = 1
    #Battle = 1
    #Educator = 1
    #Student = 1
    #Student.badges = 1
    #Badge = 1
    #RMP = 1
}

run world1 for 13
```



Alloy predicate World 1

4.4.2 World 2

This is a complex world, where only one Tournament is considered. The scope of this world is to see how the model behaves with more entities.

```
pred world2{
    #Tournament = 1
    #Tournament.educators > 1
    #Team = 3
    #Battle = 3
    #Educator = 3
    #Student = 5
    #Student.badges = 1
    #Badge = 2
    #RMP = 1
}

run world2 for 13
```

Since the graph representation becomes very large, here is presented the table representation of the world.

this/Battle	repo	scores
Battle0	RMPRepo5	BattleScore6
		BattleScore7
		BattleScore8
Battle1	RMPRepo4	BattleScore3
		BattleScore4
		BattleScore5
Battle2	RMPRepo3	BattleScore0
		BattleScore1
		BattleScore2

this/Educator	Educator0	Educator1	Educator2

this/Student	badges
Student0	
Student1	
Student2	
Student3	
Student4	Badge0

this/Name
Name0
Name1

this/Badge	educator	name	rules
Badge0	Educator2	Name1	Rules1
Badge1	Educator2	Name1	Rules0

this/RMPRepo	rmp
RMPRepo0	RMP0
RMPRepo1	RMP0
RMPRepo2	RMP0
RMPRepo3	RMP0
RMPRepo4	RMP0

RMPRepo5	RMP0

this/User	name	surname	email	rmpHandle
Educator0	Name0	Surname0	Email2	RMPHandle2
Student0	Name0	Surname0	Email7	RMPHandle7
Educator1	Name0	Surname0	Email1	RMPHandle1
Student1	Name0	Surname0	Email6	RMPHandle6
Educator2	Name0	Surname0	Email0	RMPHandle0
Student2	Name0	Surname0	Email5	RMPHandle5
Student3	Name0	Surname0	Email4	RMPHandle4
Student4	Name0	Surname0	Email3	RMPHandle3

this/RMPHandle	rmp
RMPHandle0	RMP0
RMPHandle1	RMP0
RMPHandle2	RMP0
RMPHandle3	RMP0
RMPHandle4	RMP0
RMPHandle5	RMP0
RMPHandle6	RMP0
RMPHandle7	RMP0

this/Email	Email0	Email1	Email2	Email3	Email4	Email5	Email6	Email7	-1

this/Rules
Rules0
Rules1

//For space reasons in the following table BattlScore = BattleScore

this/BattlScore	BattlScore0	BattlScore1	BattlScore2	BattlScore3	BattlScore4	BattlScore5	BattlScore6	BattlScore7	BattlScore8	-1

this/Tournament	battles	badges	educators	teams	scores
Tournament0	Battle0	Badge0	Educator0	Team0	TournamentScore0
	Battle1	Badge1	Educator2	Team1	TournamentScore1
	Battle2			Team2	TournamentScore2

this/Team	students	tournamentScore	battleScore	repos
Team0	Student2	TournamentScore0	BattleScore2	RMPRepo2
			BattleScore4	
			BattleScore6	
Team1	Student1	TournamentScore2	BattleScore1	RMPRepo1
			BattleScore3	
			BattleScore8	
Team2	Student0	TournamentScore1	BattleScore0	RMPRepo0
			BattleScore5	
			BattleScore7	

this/Score	points
BattleScore0	0

TournamentScore0	0
BattleScore1	0
TournamentScore1	0
BattleScore2	0
TournamentScore2	0
BattleScore3	0
BattleScore4	0
BattleScore5	0
BattleScore6	0
BattleScore7	0
BattleScore8	0

this / TournamentScore	TournamentScore0	TournamentScore1	TournamentScore2	-1

this / Platform	students	educators	tournaments
Platform0	Student0	Educator0	Tournament0
	Student1	Educator1	
	Student2	Educator2	
	Student3		
	Student4		

4.4.3 World 3

This is an even more complex world, where more Tournaments are considered. The scope of this world is similar to the one of World 2, but with more Tournaments;

```
pred world3{
    #Tournament = 4
    #Team = 5
    #Battle = 5
    #Educator = 3
    #Student = 7
    #Badge = 3
}

run world3 for 20
```

For the same reason as before, here is presented the table representation.

this/Badge	educator	name	rules
Badge0	Educator2	Name1	Rules2
Badge1	Educator1	Name1	Rules1
Badge2	Educator0	Name1	Rules0

this/Educator	Educator0	Educator1	Educator2

this/Name
Name0
Name1

this/Email	Email0	Email1	Email2	Email3	Email4	Email5	Email6	Email7	Email8	Email9

this/RMPRepo	rmp
RMPRepo0	RMP0
RMPRepo1	RMP0
RMPRepo2	RMP0
RMPRepo3	RMP0
RMPRepo4	RMP0
RMPRepo5	RMP0
RMPRepo6	RMP0
RMPRepo7	RMP0
RMPRepo8	RMP0
RMPRepo9	RMP0

this/Tournament	battles	badges	educators	teams	scores
Tournament0	Battle1	Badge2	Educator0	Team2	TournamentScore3
Tournament1	Battle2	Badge1	Educator0	Team3	TournamentScore0
	Battle3				
Tournament2	Battle4	Badge1	Educator0	Team4	TournamentScore4
Tournament3	Battle0	Badge0	Educator0	Team0	TournamentScore1
				Team1	TournamentScore2

this/Platform	students	educators	tournaments
Platform0	Student0	Educator0	Tournament0
	Student1	Educator1	Tournament1

	Student2	Educator2	Tournament2
	Student3		Tournament3
	Student4		
	Student5		
	Student6		

this/Student	badges
Student0	
Student1	
Student2	
Student3	
Student4	
Student5	
Student6	

this/BattleScore	BattleScore0	BattleScore1	BattleScore2	BattleScore3	BattleScore4	BattleScore5	-1

this/Battle	repo	scores
Battle0	RMPRepo9	BattleScore4
		BattleScore5
Battle1	RMPRepo8	BattleScore3
Battle2	RMPRepo7	BattleScore2
Battle3	RMPRepo6	BattleScore1
Battle4	RMPRepo5	BattleScore0

this/TournamentScore	TournamentScore0	TournamentScore1	TournamentScore2	TournamentScore3	TournamentScore4	-1

this/Team	students	tournamentScore	battleScore	repos
Team0	Student1	TournamentScore1	BattleScore5	RMPRepo2
Team1	Student0	TournamentScore2	BattleScore4	RMPRepo4
Team2	Student0	TournamentScore3	BattleScore3	RMPRepo1
Team3	Student0	TournamentScore0	BattleScore1	RMPRepo3
			BattleScore2	
Team4	Student0	TournamentScore4	BattleScore0	RMPRepo0

this/User	name	surname	email	rmpHandle
Educator0	Name0	Surname0	Email2	RMPHandle6
Student0	Name0	Surname0	Email9	RMPHandle5
Educator1	Name0	Surname0	Email1	RMPHandle8
Student1	Name0	Surname0	Email8	RMPHandle7
Educator2	Name0	Surname0	Email0	RMPHandle9
Student2	Name0	Surname0	Email7	RMPHandle2
Student3	Name0	Surname0	Email6	RMPHandle3
Student4	Name0	Surname0	Email5	RMPHandle0
Student5	Name0	Surname0	Email4	RMPHandle1
Student6	Name0	Surname0	Email3	RMPHandle4

this/Score	points
BattleScore0	0


```
+----------------+------+
| TournamentScore0 |0     |
+----------------+------+
| BattleScore1     |0     |
+----------------+------+
| TournamentScore1 |0     |
+----------------+------+
| BattleScore2     |0     |
+----------------+------+
| TournamentScore2 |0     |
+----------------+------+
| BattleScore3     |0     |
+----------------+------+
| TournamentScore3 |0     |
+----------------+------+
| BattleScore4     |0     |
+----------------+------+
| TournamentScore4 |0     |
+----------------+------+
| BattleScore5     |0     |
+----------------+------+

+--------------+----+
| this/RMPHandle | rmp  |
+--------------+----+
| RMPHandle0     | RMP0 |
+--------------+----+
| RMPHandle1     | RMP0 |
+--------------+----+
| RMPHandle2     | RMP0 |
+--------------+----+
| RMPHandle3     | RMP0 |
+--------------+----+
| RMPHandle4     | RMP0 |
+--------------+----+
| RMPHandle5     | RMP0 |
+--------------+----+
| RMPHandle6     | RMP0 |
+--------------+----+
| RMPHandle7     | RMP0 |
+--------------+----+
| RMPHandle8     | RMP0 |
+--------------+----+
| RMPHandle9     | RMP0 |
+--------------+----+

+----------+------+------+------+-1
| this/Rules | Rules0 | Rules1 | Rules2 |
+----------+------+------+------+
```


5 Effort Spent

Name	Section 1	Section 2	Section 3	Section 4
Angelo Attivissimo	10 h	12 h	11 h	6 h
Isaia Belardinelli	10 h	8 h	11 h	6 h
Carlo Chiodaroli	7 h	7 h	13 h	10 h

6 References

[1] Specification Document: RASD and DD Assignment A.Y. 2023-24

[2] Alloy official Documentation: https://alloytools.org/documentation.html

[3] UML official specification: https://www.omg.org/spec/UML/

[4] BPMN official specifiction: https://www.omg.org/spec/BPMN/2.0/

