POLITECNICO DI MILANO

Software Engineering 2

CodeKataBattle - Improve your programming skills online

Requirements Analysis
and Specification Document

Authors:

Nicolò Giallongo - 10764261

Giovanni Orciuolo - 10994077

Giuseppe Vitello - 10766482

Table of Contents

1 - Introduction........ 3
1.1 - Purpose........ 3
1.1.1 - Goals........ 3
1.2 - Scope........ 5
1.2.1 - World Phenomena........ 10
1.2.2 - Shared Phenomena........ 10
1.3 - Definitions, Acronyms, Abbreviations........ 11
1.4 - Revision History........ 13
1.5 - Reference Documents........ 13
1.6 - Document Structure........ 14
2 - Overall Description........ 15
2.1 - Product Perspective........ 15
2.1.1 - Class Diagram........ 15
2.1.2 - State Diagrams........ 16
2.1.3 - Scenarios........ 17
2.2 - Product Functions........ 21
2.3 - User Characteristics........ 25
2.4 - Domain Assumptions........ 26
3 - Specific Requirements........ 27
3.1 - External Interface Requirements........ 27
3.1.1 - User Interfaces........ 27
3.1.2 - Hardware Interfaces........ 27
3.1.3 - Software Interfaces........ 27
3.1.4 - Communication Interfaces........ 28
3.2 - Functional Requirements........ 28
3.2.1 - Students........ 28
3.2.2 - Educators........ 29
3.2.3 - Other........ 32
3.2.4 - Mapping on Goals........ 34
3.2.5 - Use Case Diagrams........ 36
3.2.6 - Use Cases........ 39
3.3 - Performance Requirements........ 82
3.4 - Design Constraints........ 82
3.4.1 - Standards Compliance........ 82
3.4.2 - Hardware Limitations........ 82
3.5 - Software System Attributes........ 82
3.5.1 - Reliability........ 82
3.5.2 - Availability........ 82
3.5.3 - Security........ 82

3.5.4 - Maintainability.. 83
3.5.5 - Portability.. 83
4 - Formal Analysis (Alloy)...83
4.1 - Alloy Model..83
4.2 - Simulation.. 86
5 - Effort Spent.. 87
6 - References.. 87

1 - Introduction

1.1 - Purpose

The purpose of the CKB application is to provide students and educators with a platform to improve their programming skills by taking part in friendly competitions (called “Tournaments”) where they can resolve various programming problems (called “Katas”) in their preferred language of choice (e.g. Java or Python).

The platform relies heavily upon GitHub in order to store solutions and to run tests and other activities on each push. As such, the users are required to use their GitHub account to access the application (or register one if they don’t have it).

1.1.1 - Goals

Student Goals

ID	Description
SG1	Students that are IU receive a notification whenever a new tournament is created.
SG2	Students receive a notification whenever a new battle is created in a tournament in which they participate.
SG3	Users involved in a tournament can see the leaderboard of its battles change.
SG4	Students can join the battle directly or by invitation from other students that participate in the tournament, until the EBD is reached and if there is a spot available.
SG5	Students in a Battle can invite other Students in a Team if the EBD hasn’t been reached yet and if the Team is not full.

SG6	If the CD is before the ETD, and the MNS isn't reached, Students can join a Tournament in three different ways: ● via notification; ● via link; ● by selecting the Tournament on the website (by searching it or finding it on a User profile).

Educator Goals

ID	**Description**
EG1	Educators can create Tournaments and invite TCs.
EG2	An authorized Educator (TC or OCT) can create a Battle in a Tournament, uploading the corresponding kata, and configuring OME and MAE.
EG3	When the FBD is reached, the Educator that created the Battle can perform manual evaluation, if the option is enabled. Once this phase is terminated, all the users participating in the battle get notified and the leaderboard of the Tournament is updated.
EG4	Educators can create badges for their Tournaments. Badges get assigned to Students based on a specific set of rules.
EG5	Educators that are OCT can ban a Student from their Tournament.

Other Goals (related to System itself, GitHub, and Users in general)

ID	**Description**
OG1.1	For each battle, when EBD is reached, the System creates a GitHub Repository, that every Student must fork and set up an automated

	workflow through GitHub Actions (which invokes the system each time new commits are pushed to the main branch).
OG1.2	When invoked, the System pulls the latest version and analyzes it to calculate the score.
OG2	All the Users can see the list of ongoing Tournaments as well as the corresponding Tournament leaderboard.
OG3	Users can send and receive friend requests from other Users, and accept or refuse them. The friend request notification is sent regardless of the privacy options of the recipient.
OG4	As soon as a Tournament ends, the subscribed Students are notified to see the final tournament leaderboard.
OG5	Users can decide from whom they receive notifications about Tournaments: *Everyone*, *Friends Only* or *Nothing*.
OG6	User profile shows all the badges they have gained.
OG7	Users must authenticate using their GitHub account to use the platform.
OG8	A User involved in a Tournament (in any way possible) can visualize all the information about the said Tournament (including participants, ongoing battles and their rank, TCs, ETD and FTD)

1.2 - Scope

The following is the product description provided by the Product Owner (PO):

CodeKataBattle (CKB) is a new platform that helps students improve their software development skills by training with peers on code katas.

Educators use the platform to challenge students by creating code kata battles in which teams of students can compete against each other, thus proving (and improving) their skills. A code kata battle is essentially a programming exercise in a programming language of choice (e.g., Java, Python). The exercise includes a brief textual description and a software project with build automation scripts (e.g. a Gradle project in case of Java sources) that contains a set of test cases that the program must pass, but without the program implementation. Students are asked to complete the project with their code. In particular, groups of students participating in a battle are expected to follow a test-first approach and develop a solution that passes the required tests. Groups deliver their solution to the platform (by the end of the battle). At the end of the battle, the platform assigns scores to groups to create a competition rank.

Each battle created by an educator belongs to a specific tournament. Tournaments are created by an educator who can then grant to other colleagues the permission to create battles within the context of a specific tournament. When a new tournament is created, all students subscribed to the CKB platform are notified and they can subscribe by a given deadline. If they subscribe, they are notified of all upcoming battles created within that tournament. To create a new battle, an educator uses the CKB platform to perform the following steps:

- upload the code kata (description and software project, including test cases and build automation scripts),
- set minimum and maximum number of students per group,
- set a registration deadline,
- set a final submission deadline,
- set additional configurations for scoring (see further details in the following).

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 his/her own or by inviting other students (respecting the minimum and maximum number of students per group set for that battle). When the registration deadline expires, the platform creates a GitHub repository containing the code kata and then sends the link to all students who are members of subscribed teams. At this point, students can start working on the project. 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 students push a new commit into the main branch of their repository. Thus, each push before the deadline triggers the CKB platform, which pulls the latest sources, analyzes them, and runs the tests on the corresponding executables to calculate and update the battle score of the team.

The score is a natural number between 0 and 100 determined by considering some mandatory factors evaluated in a fully automated way, and optional factors evaluated manually by educators.

Mandatory automated evaluation includes:

- functional aspects, measured in terms of number of test cases that pass out of all test cases (the higher the better);
- timeliness, measured in terms of time passed between the registration deadline and the last commit (the lower the better);
- quality level of the sources, extracted through static analysis tools that consider multiple aspects such as security, reliability, and maintainability (the higher the better). Aspects are selected by the educator at battle creation time.

Optional manual evaluation includes:

- personal score assigned by the educator, who checks and evaluates the work done by students (the higher the better).

The CKB platform automatically updates the battle score of a team as soon as new push actions on GitHub are performed. So, both students and educators involved in the battle can see the current rank evolving during the battle. When the submission deadline expires, there is a consolidation stage in which, if manual evaluation is required, the educator uses the CKB platform to go through the sources produced by each team to assign his/her score. Once the consolidation stage finishes, all students participating in the battle are notified when the final battle rank becomes available.

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. This information is available for all students and educators subscribed to the CKB platform, that is, 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.

The CKB platform also includes gamification badges: elements in the form of rewards that represent the achievements of individual students. Badges are defined by educators when they create a tournament. Each badge has a title (e.g., “top committer”) and one or more rules (i.e., simple Boolean properties) that must be fulfilled to achieve the badge. Thus, each badge is assigned to one or more students depending on the rules checked at the end of the tournament. The following are examples of possible badges:

- Title: Tournament Participant

        - Rules: `{ tot_attended_battles > 0 }`

o Rules: { tot_attended_battles > 0 }

- Title: Top Committer
    - Rules: `{ tot_commits_student == max_tot_commits }`

Where `tot_attended_battles`, `tot_commits_student` and `max_tot_commits` are predefined variables that represent, respectively, the total number of battles the student has been involved in, the number of commits carried out by the student and the maximum total number of commits considering all students.

As mentioned above, educators can create new badges and define new rules as well as new variables associated with them. Variables can represent any piece of information available in CKB relevant for scoring. It is up to you to identify such information and a simple language for creating new variables, rules, and badges.
Badges can be visualized by all users. In particular, both students and educators can see collected badges when they visualize the profile of a student.

Completion of product description

In this section the scope is narrowed down with some constraints and top-level design decisions, in order to complete production description from PO by eliminating any ambiguity, thus making further developments leaner. Refer to *Section 1.3* to understand acronyms and abbreviations.

1. Login and Sign-up functionalities are only available through GitHub;
2. Users can become friends in the CodeKata platform;
3. Users can decide their preferred policy for notifications:
    - *Everyone*: receive notification from every user;
    - *Friends Only*: receive notifications only from friends and subscribed Tournaments (and their Battles too);
    - *Nothing*: receive notifications only from subscribed Tournaments (and their Battles too);
4. Friend requests notifications are sent regardless of notification policy;
5. While creating a new Tournament, the creator is required to insert the following information: MNS, EDT and FDT, TPL. It is also possible to invite other users as TC, and to add badges (either by using the default badges provided by the system or creating new badges). Badges are assigned to the students at the end of the tournament;
6. All the Tournaments are accessible through a public link (shared by OCT and TCs): by using this method, even those who have not received the notification can view and join a Tournament;

7. Tournaments where the TPL is *Open* can be accessed by any User, and the notification for its creation is sent to all IUs;
8. Tournaments where the TPL is *Friends Only* can only be accessed by the OCT friend, but only those who are IUs receive the notification for its creation. Moreover, the Tournament public link will show tournament details and allow the viewing User to join only if they are friends with the OCT;
9. Tournaments where the TPL is “Private" can be accessed by any User with the Tournament’s link, and the OCT receives a notification to admit or reject the User. Moreover, the notification for its creation is not sent.
10. When the OCT wants to add a coordinator, this User receives a notification (respecting their notification policy) that must be accepted before the ETD;
11. The number of TCs for a given Tournament has a hard limit of 5;
12. The creator of a Battle must insert the following information: EDB and FDB (both editable by the creator);
13. The Users in a Battle can send invitations to join their Team only to the students in the Tournament (even those who are already in a Team), but can do so only if their Team is not full.
14. If a Student accepts an invitation in Team A, and they are already in another Team B, they leave B and join A.
15. If a Battle terminated (either manually or by reaching FDB) and OME is enabled:
    a. Battle participants receive a notification to inform them that it is not possible to submit anymore;
    b. Battle Creator receives a notification to inform them that they must perform the manual evaluation before the FTD;
    c. When they finish performing evaluation, another notification is sent to Battle participants, inviting them to see the final updated leaderboard;
    d. If FTD is reached and the Battle Creator doesn't perform manual evaluation, the system ignores manual evaluation and uses the current leaderboard to calculate final Tournament ranking;
16. One CKB User can be both a Student (e.g. they can join Tournament and more generally perform all the features dedicated to Students) and an Educator (can create Tournaments and more generally perform all the features dedicated to Educators);
17. OCT and their TCs can’t participate in their own Tournament’s Battles as Students.
18. A CKB profile of a User shows tournaments created by them or where they are a TC, and the Tournaments in which they participated.
19. Tournaments are accessible by searching them in the website or in a User profile respecting the same policy of the link.

1.2.1 - World Phenomena

ID	Description
WP1	Student wants to improve their programming skills.
WP2	Educator wants to put their students on a programming trial.
WP3	Educator wants to track their students objectives through custom achievements.
WP4	Educator wants to know the programming knowledge progress of their students.
WP5	Student wants to compete with their peers in programming competitions.
WP6	Student commits the code for a Battle on the right GitHub repository.
WP7	Static analysis tools perform evaluation of the code on a quality level.

1.2.2 - Shared Phenomena

World-Controlled Shared Phenomena

ID	Description
WCP1	User register on the website, inserting all the requested information.
WCP2	User logs in on the website, inserting all the requested information.
WCP3	Educators create Tournaments.
WCP4	Educators create Battles within Tournaments.
WCP5	GitHub notifies the platform when a new version of the code is committed by a student.
WCP6	Student subscribes to Tournaments.
WCP7	Student participate in Battles.
WCP8	Educator creates badges.
WCP9	Student invites other students to join a battle.
WCP10	Educator performs manual evaluation, if enabled.
WCP11	User opens a User profile.

WCP12	User opens the Tournament leaderboard.
WCP13	Student opens the leaderboard of a Battle in which they participate.
WCP14	Educator opens the leaderboard of a Battle they have created.
WCP15	User visualizes a notification.

Machine-Controlled Shared Phenomena

ID	**Description**
MCP1	The System sends notification to IUs whenever a new Tournament is created.
MCP2	The System sends notification to SST whenever a new battle is created in the Tournament.
MCP3	The System grants the access to their account to an authenticated user.
MCP4	The System creates a repository for a battle when the EBD expires.
MCP5	The System pulls the code of a GitHub repository when notified of a commit.
MCP6	The System sends notification to a Student when they are invited by another student to participate in a Battle.
MCP7	The System shows Users profile.
MCP8	The System shows the tournament leaderboard.
MCP9	The System shows the battle rank to the participant and the creator.
MCP10	The System calls the Static Analysis Tools enabled by the Battle's creator.
MCP11	The System sends all notifications when it is necessary.

1.3 - Definitions, Acronyms, Abbreviations

In order to reduce ambiguity as much as possible, this document heavily uses acronyms and abbreviations to refer to entities and concepts in the CKB system. This section is aimed at defining each of these acronyms and abbreviations in a precise and concise way.

Acronym	Definition
IU	Interested Student: a Student U is Interested by a notification N: ● if U has selected the option to receive notification from all the users. ● if U has selected the option to receive notification only from their friend and the user who has sent N is a friend of U. ● N is a notification of a Tournament in which U is subscribed.
SST	Students Subscribed to a Tournament.
OCT	Original Tournament Creator.
TC	Tournament Coordinator, appointed by the OCT.
CDT	Current DateTime.
ETD	Enrollment Tournament Deadline. After this deadline, it is not possible to join the Tournament as a Student.
FTD	Final Tournament Deadline. After this deadline, the Tournament is ended and the final leaderboard is made available.
MNS	Maximum Number of Subscribers. It is the maximum number of Students that can subscribe to a specific Tournament.
EBD	Enrollment Battle Deadline. After this deadline, it is not possible to join the Battle as a Student.
FBD	Final Battle Deadline. After this deadline, the Battle is ended and the final leaderboard is made available. Moreover, the Tournament related to the Battle is updated with the new leaderboard.
OME	Optional Manual Evaluation. Personal score assigned by the Educator, who checks and evaluates the work done by students (the higher the better).
MAE	Mandatory Automated Evaluation, which includes: ● Functional aspects, measured in terms of number of test cases that pass out of all test cases (the higher the better); ● Timeliness, measured in terms of time passed between the registration deadline and the last commit (the lower the better); ● Quality level of the sources, extracted through Static Analysis Tools that consider multiple aspects such as security,

	reliability, and maintainability (the higher the better). Aspects are selected by the educator at battle creation time.
DAS	Directly Accepted Student: a Student <u>U</u> is directly accepted in a Tournament <u>T</u> if: ● <u>T</u> is public; ● <u>T</u> is friends only and <u>U</u> is a friend of the OCT of <u>T</u>.
EB	End Battle: ● If OME is not required, EB occurs when the FBD is reached. ● If OME is required, EB is after the Educator performs it, or when the FTD is reached.
SAT	Static Analysis Tools.

1.4 - Revision History

Version	Date	Changelog
1.0	04/12/2023	First presentable version of the document.
1.1	17/12/2023	Finalized Alloy definitions.
1.2	19/12/2023	Adjusted format of functional requirements, refined use case diagrams.
1.3	20/12/2023	Add Alloy simulations, add Effort Spent section.
1.4	04/01/2024	Fix Domain Class Diagram with corrections coming from DD reasoning.
1.5	29/01/2024	Remove ER6.
1.6	14/02/2024	Complete section 3.3

1.5 - Reference Documents

- The specification document Assignment RDD AY 2023-2024.pdf

1.6 - Document Structure

This document is divided in 4 main parts:

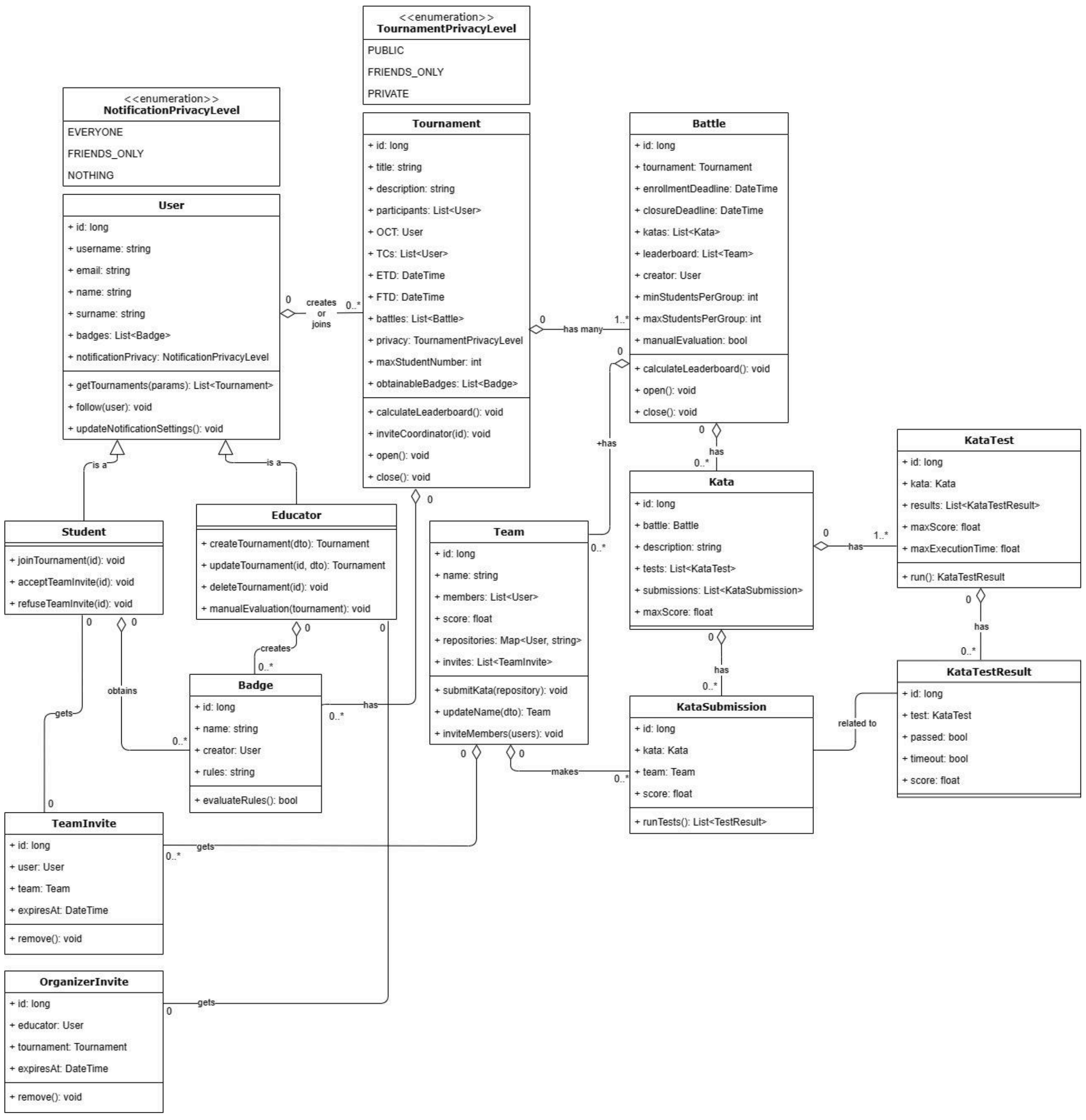
1. **Introduction**: introduces the project statement from Product Owner (PO), and expands upon it by defining goals and purposes (scope) along with a short analysis of world and shared phenomena. The objective is to understand the problem as better as possible.
2. **Overall Description**: includes scenarios and a domain model (class diagram and state diagrams), along with domain assumptions, constraints and an analysis of user characteristics.
3. **Specific Requirements**: follows from the previous part by delving deeper into the functional and non-functional requirements, along with performance requirements, design constraints and a clear definition of the software system attributes.
4. **Formal Analysis (Alloy)**: introduces the Alloy tool and the main objectives driving the formal modeling activity, as well as a description of the model itself, what can be proved with it, and why what is proved is important given the problem at hand. It also includes some example worlds simulations.
5. **Effort Spent**: contains information about the number of hours each group member has worked on this document.
6. **References**: contains information about the resources used to redact this document.

2 - Overall Description

2.1 - Product Perspective

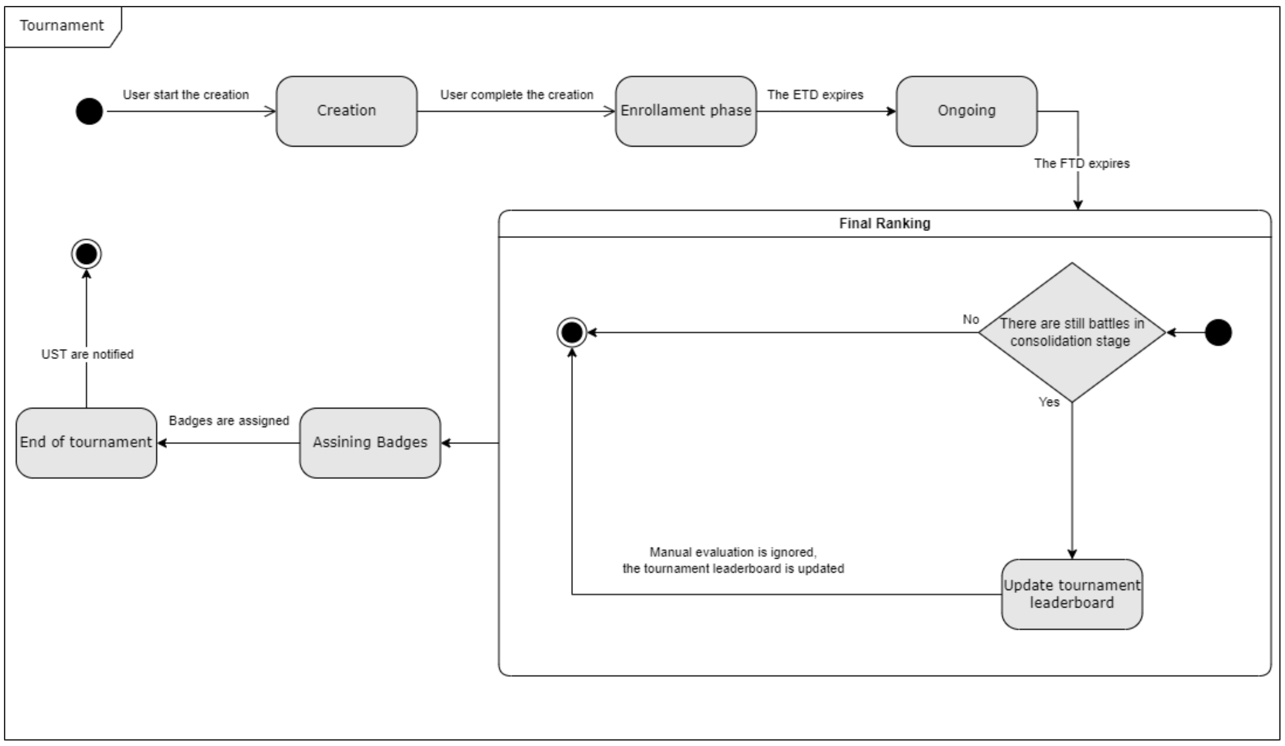
2.1.1 - Class Diagram

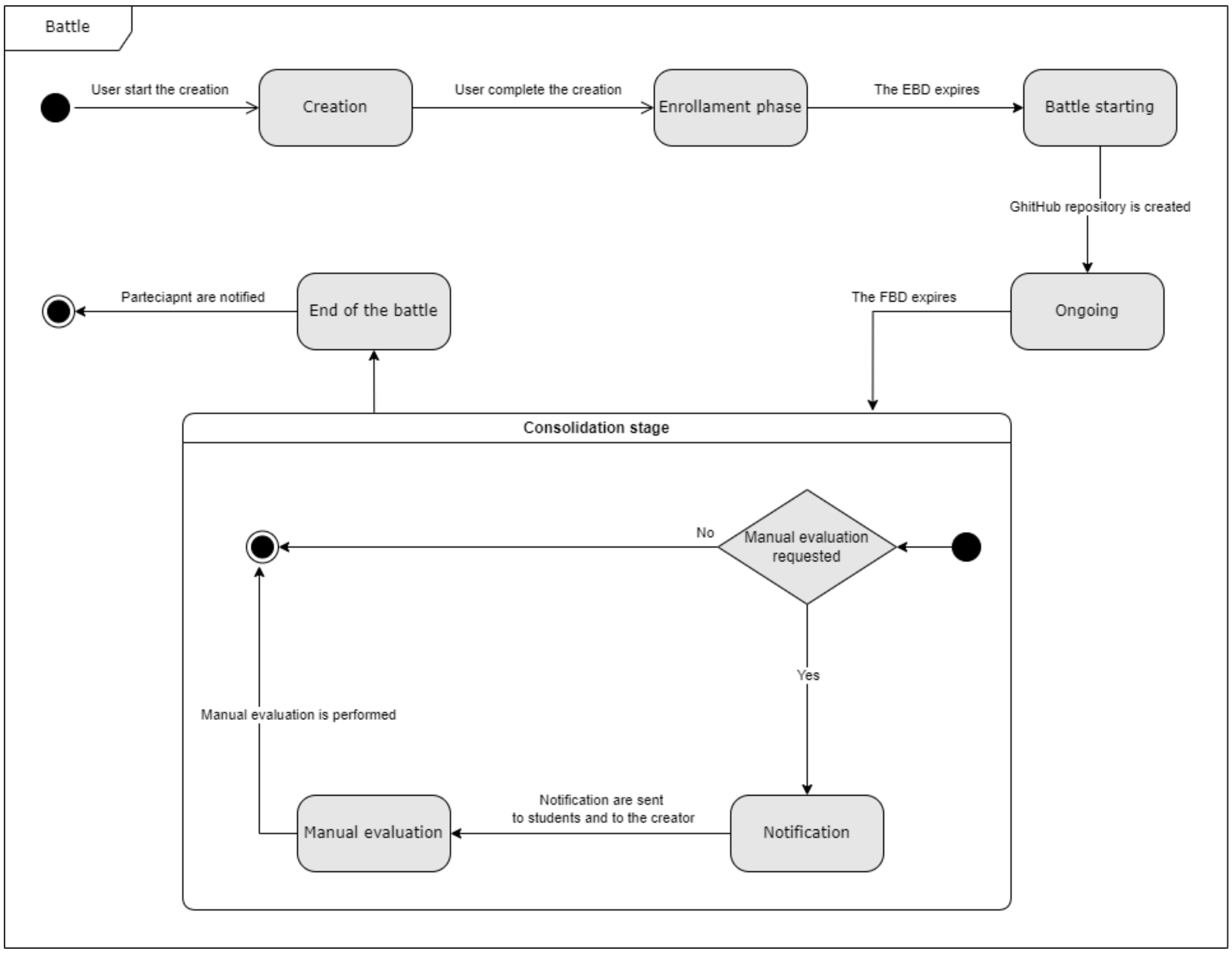
The diagram below describes the classes involved in the CKB system, along with their attributes, functionalities and the relations between them.



2.1.2 - State Diagrams

The two main components of CKB system that can be represented by state diagrams are tournaments and battles:





2.1.3 - Scenarios

For a realistic formulation, there is a first part called “MetaScenario”.
MetaScenario is a union of scenarios in sequence, in this way it is created a story that has the goal to explain in clear modality a realistic example of the application’s use.
After that there is a second part called “Single Scenarios” with different scenarios independent of MetaScenario such that it can complete all principal situations for the application’s use.

MetaScenario

Mario , Angelo, Sabino: professors of a high school that want create a challenge for the school;

Professors= Mario, Angelo and Sabino;

“Programming Challenge HS”: it is the name for the Tournament created for the challenge;

All the other names are the students subscribed in the “Programming Challenge HS”.

S1. Let’s try CodeKataBattle

Mario, a computer science professor of a high school, wants to create a programming challenge for his students. The other computer science professors of the school are Angelo and Sabino, so Mario talks to them.

They are interested in the idea and Mario suggests using CodeKataBattle platform.

Therefore, they enter the website and sign up, but it’s necessary to have a GitHub account. Angelo and Sabino own one, but not Mario.

So Mario creates a GitHub account, and all the professors sign up on CodeKataBattle’s website inserting all the necessary information.

S2. Becoming friends

Mario, Angelo and Sabino own a CodeKataBattle account now.

Visiting the website a little, there are many Tournaments organized by other people. The Professors can also see the Tournament leaderboards, and the profiles of the Tournament’s participants.

In the user profile page it is presented the possibility to send a friend request.

So, Mario searches Giovanni and Sabino user profiles to send them a friend request.

They receive a notification to accept or decline the request. Obviously they accept it, and now Mario is friends with Giovanni and Sabino on CKB.

S3. Creation of the Tournament

Angelo wants to create a Tournament now.

So he creates a Tournament naming “Programming Challenge HS”, sets it to private, inserts all the required information to create it, noticing the possibility to add or create badges.

He reads that Tournament’s Badges are a title assigned to the students through the rules. He can add a default Badge or create it himself.

He adds 3 default badges, first, second and third place titles, they are obvious rules.

He also creates 3 badges:

1. The best Python programmer: assigned to the student who has the most points in the Python battle at the end of a tournament.

2. The best C programmer: assigned to the student who has the most points in the C battle at the end of a tournament.
3. The best Java programmer: assigned to the student who has the most points in the Java battle at the end of a tournament.

He completes the creation, granting the privileges to create battles to Sabino and Mario, and the enrollment phase starts.

S4. Students join the Tournament

Each Professor explains at their class the Tournament named “Programming Challenge MHS”, how to create a GitHub account, how to sign up on Codekata's website and the link to access at the Tournament.

The students follow all the steps and they use the link to enter in the Tournament. Prof. Angelo will admit everyone in the tournament.

S5. Creation of a battle

Sabino starts with a first challenge of the Tournament, so he logins in his CKB account, enters in the Tournament named “Programming Challenge MHS” Tournament and creates a battle, inserting all the required information.

Once the battle has been created, start the enrollment phase. Giuseppe, like all the students subscribed at the Tournament, receives a notification to access the battle, he can see all the information of the Battle. He notices that the number of students for Team greater than one, it’s four, so he wants to create a Team with their friends Nicolò, Giovanni and Michele. Nicolò and Giovanni immediately accept the invitation unlike Michele, who can't accept the invitation because the EBD is expired.

Arianna, like Giuseppe, receives a notification to access the battle. She invites Maurizo ,Stefano, Simone and Mattia (SSTs). Stefano, Simone and Mattia accept the invitation first, unlike Maurizio who can’t access because the Team is full.

After the EDB the battle really begins, GitHub repository is created, the students can fork it and set up with the GitHub Action.

S6. During the battle

Nicolò, Giuseppe and Giovanni participated at a Battle of the “Programming Challenge MHS” and they are in a Team.

Giuseppe, after having written his version of the solution, is satisfied, so he wants to push the version on the GitHub repository.

After that, he accesses CKB website to see the points scored, he notices he totaled a score of 10/100, he is the last on the leaderboard. He remembers that in the last version he has scored 60/100, so he pushes again the last version to return to 60/100.

S7. End of the battle

Prof. Sabino, during the creation of the battle in “Programming Challenge MHS”, has selected OME.

Arianna (SST) receives a notification to inform her that no one can push, because the battle is terminated.

Prof Sabino receives a notification inviting him to perform manual evaluation.

So he accesses the leaderboard, sees the last version and the repository’s link for the first Team, with 94/100, he checks the repository. He likes the code, because it’s commented very well, and the code is written by the student in a Team in equity quantity, so he decides to assign additional 5 points to each student in the Team, the final score is 99/100 for each student in the Team. Prof. Sabino continues with all the other students.

When he has terminated he finally closes the battle. So the system notifies all students to see the final battle leaderboard and the Tournament leaderboard updated.

S8. End of a tournament

“Programming Challenge MHS” Tournament is unfortunately terminated, so all the students and professors receive a notification to see the final Tournament rank.

Giovanni, after a notification, accesses on the CodeKata website to see the final rank, he is very happy because is the first ranked, entering in the his profile find new Badges:

“First place”, ”Best Python programming” and ”Best Java programming”, so he decides to tell Giuseppe and Nicolò.

Even Giuseppe sees his profile with new Badge “Third place”, at the end Nicolò sees his profile with new Badges “Second place”, ”Best C programming” and ”Best Java programming”.

Giovanni and Nicolò notice they have scored the same total points in the Java battles, so they both have the badge.

Single Scenarios

In this part scenarios are independent by the precedent ,they need to complete all the principal application uses not covered by the precedent scenarios.

S9. Server maintenance

Luciano has a CKB account, he wants access on the website to see new subscribers on his tournaments, he enters on the website, but it is in maintenance. The website says any deadline will be increased by the duration of the maintenance, there is even an estimated time of the maintenance.

S10. GitHub is down

Prof. Mario receives a notification by CKB, the notification says GitHub is down, the notification invites him to increase the time for his Battle and Tournament because the subscribed Students can’t push their code for evaluation.

S11. Missed notification

Luigi and Mario both have a CKB account.

Luigi wants to create a Tournament, adding Mario as TC, but Mario doesn't receive a notification due his policy notification.

S12. Player ban

Riccardo is a OCT of CodeKataBattle platform.

Riccardo, seeing the list of participants in his Tournament, notices that one of them has an offensive name: “OffensiveName”. So Riccardo bans “OffensiveName” from the tournament.

If “OffensiveName” has a Team in a battle, teammates will lose a member.

S13. Francesco is receiving too many notifications.

After four months, CKB platform has 1 million accounts, the website has been very successful.

Francesco, accessing his account, notices a very high number of notifications because he receives any type of notification, so he decides to limit the number, selecting the policy “receive notifications only from your friends”.

2.2 - Product Functions

PF1. Access to the site

CKB’s services are usable via a web application. To access the application the user needs to be registered. Registration and login will occur via GitHub, so the user needs to have a GitHub account. Once the login is completed, the user can start to navigate the website.

On registration, the user will create the account. The system will ask the user to insert some additional information: name*, surname*, birth date*, school/university, profile image. Some of this information can be added by the user later. Then the system provides the user a list of suggested friends (based on their list of GitHub followers, or Users with the same school/university field), and then asks the User to choose whether to receive notification only from friends, from all the users or from nobody. This option can be modified later.

When the user wants to access their profile, they just have to login with GitHub, and the access to the profile will be granted. The profile will have two sections: the student sections, from which a user can join tournaments and battles, and the educator section, from which a user can create tournaments and battles.

User’s profile displays all the basic information inserted by the user upon registration, and some other information like the badges they obtained in various tournaments, and all the ongoing tournaments (and respective leaderboards) in which they are involved. This information can be accessed from every user who visits the profile.

PF2. Tournament functions

Tournaments are one of the main functionalities of CKB. They are essentially containers in which various battles take place, and the main goal of a battle is in fact to accumulate points to get a better position in the tournament final ranking.

To create a tournament a user must go to the educator section of their profile, and start the creation of a tournament. The tournament creation form is opened. The user will be asked to enter the name* of the tournament, a cover image and a brief description. Then the user will set ETD, FTD and MNS of the tournament.
With the function “Add other educators” the user will be able to select, either from their friends or by searching them manually, other users to whom grant the privileges of TC. All the users selected by the OCT will receive an invitation to become TC of the tournament, which they can accept or refuse. There is an upper limit for the number of TC in a tournament, defined by the system, based on the server capacity. When this number is reached for a tournament, it won’t be possible to accept the invitation to become a TC anymore. The TCs will be allowed to create battles, but they won’t be able to modify the options of the tournament.
With the subfunction “Add badges” the user will be able to add existing badges to the tournament, or to create new ones. Finally the user can decide the level of privacy of the tournament. There are three levels of privacy: public, friends only and private.

If the tournament is public, every IU receives the notification of the tournament, and can join freely in every way, as long as the number of users subscribed to the tournament is lesser than the MNS.

If the tournament is friends only, only the user’s friends receive the notification of the tournament, and can join freely, as long as the number of users subscribed to the tournament is less than the MNS. Whenever someone who isn't a friend of the OCT tries to join the tournament via link or website the OCT gets notified and can decide either to grant them access or not.

If the tournament is private, no one receives notification. Whenever someone tries to join the tournament via link or website the OCT gets notified and can decide either to grant them access or not.

In all three cases the OCT will receive a link that can be used to access the tournament. This link can be copied and sent by the user to people they want to join the tournament.
When they have finished the user confirms the creation of the tournament and the system notifies all the IU.

During a tournament, several battles can occur. At the end of each battle, the system updates the current score of each participant to the tournament and its leaderboard.

As soon as the FTD expires the tournament ends. If there are any battles still in the consolidation stage (so waiting for manual evaluation to be performed) the system ignores manual evaluation, and uses the current battle score to update the tournament leaderboard. Then the system checks the rules of the badges added to the tournament and decides whether or not to assign the badge to a certain UST. Finally, the system notify all the USTs that the tournament is finished and the final rank is available.

PF3. Create a Badge

Badges are special achievements that a user can obtain in a tournament. There are a few default badges, but an OCT can create custom badges for their tournaments.

This function is accessible during the creation of a tournament. When the user adds badges to the tournament, they can decide to create new badges. In this case the user will select the option of creating a badge, and will access the badges editor. The user will insert the name of the badges and a brief description, will choose the images to represent the badge and will specify the rule to be satisfied to get the badge.

PF4. Create a Battle

Battles are the main functionality of CKB. In a battle various users compete to find the best solution for one or more code katas.

Battles are created by a user within a tournament. The user must be OCT or TC of a tournament to create a battle. The user goes in the educator section of their profile and selects the tournament in which they want to create the battle. They then select the create battle option, and the form of battle creation opens. The user is asked to perform the following step:

- Insert the name of the battle;
- Upload one or more code kata (description and software project, including test cases and build automation scripts);
- Set minimum and maximum number of users per group;
- Set the EBD;
- Set a final FBD;
- Set additional configurations for scoring (the aspects to consider for the quality level evaluation and if manual evaluation is required).

Then the user will confirm the creation of the battle and the system notifies all the UST.

PF4. Participating in a Battle

Users in a tournament can participate in all its battles. Each participant in the battle is part of one group. Minimum and maximum number of users per group are defined upon creation of the battle.
When a user decides to join in a battle, a new group is created and the user is added there. The user can then decide to invite other USTs to their group. This function is available while the user is joining a battle. The system will ask the user if they want to send the request to join their group to other USTs. The user can filter the USTs (for example to see only their friends) or can directly search between the USTs the ones who want to invite. The user can send as many invites as they want, but once the maximum number of students per group is reached, no one else will be able to join through the invite. Those who join a battle through the invitation of another UST, no group will be created: they will be added to the group of the user who invited them. If a user that already has a group accepts the invitation to join another group, they will leave the current group and be added to the new one. All these operations are available until the EBD expires.

As soon as the EBD expires, the battle begins. No user will be able to join the battle and send or accept invitations to a group. Groups who have not reached the minimum number of users requested, are disqualified from the battle.

At this point the system creates the GitHub repository for a battle. Is up to the participants of the battle to fork this repository and push their solutions there. At every push on one of the forked repositories, the system pulls the code, calculates the score according to the evaluation options of the battle, and assigns this score to the group of which the user who pushed the code belongs. Moreover, the system

updates the link of the last version of the group, substituting it with the link of the repository where the push occurred.
When the FBD expires, the system ignores all the successive pushes. If manual evaluation is required, the system sends the notification to the battle creator, and awaits for the manual evaluation to be performed before sending the end battle notification to all the participating users. Otherwise it sends the end battle notification immediately.

PF5. Optional Manual Evaluation

The creator of a battle can decide to perform manual evaluation on the last version of the code written by a group. This function is available only if the manual evaluation was selected upon creation of the battle. In that case, when the FBD expires, the system will notify the participant to the battle, informing them that it is no longer possible to push their code, and the creator of the battle, asking him to perform manual evaluation. When the creator accesses the battle to perform manual evaluation, the system will show them the list of all groups, each with the link to the github repository in which the last push happened. The user can read the code, insert the amount of points to assign to the group, then confirm and go to the next group. When they have finished, the user confirms the evaluation, and the system updates the leaderboard and terminates the battle.

Manual evaluation can be performed until the FTD of the tournament in which the battle takes place expires. When the FTD of a tournament expires, manual evaluation is ignored for all the battles in said tournament which are still in the consolidation stage. In that case, the current score of the battles are used to update the final tournament leaderboard.

PF6. Adding Friends

Users can send friend requests to other users. They can filter the possible friend using various criteria, or simply search them. The system also provides suggested friends, based on various criteria (same school/university, friends on GitHub, friends of friends).

2.3 - User Characteristics

There are essentially two kinds of users that interact with the System: the Student and the Educator.

- **Student**
  A Student is a person who participates in tournaments alone or with other teammates (which are also Students), with the objective to improve their programming skills. They need to own a device which is able to connect to the

internet (a laptop or desktop PC, for example) and login with GitHub (create a GitHub account if they haven't already) in order to use all the features of the System.

- **Educator**
  An Educator is a person who creates tournaments alone or with other organizers (which are also Educators) and upload code katas into them. They need to own a device which is able to connect to the internet (a laptop or desktop PC, for example) and login with GitHub (create a GitHub account if they haven't already) in order to use all the features of the System.

An User in the System can be both a Student and an Educator.

2.4 - Domain Assumptions

ID	Description	Related Goals
D1	Users have a device with an internet connection to interact with the System.	SG1, SG2, SG3, SG4, SG5, SG6, SG7, EG1, EG2, EG3, EG4, EG5, OG2, OG3, OG4, OG5, OG6, OG7, OG8
D2	GitHub system is working correctly and is not offline.	OG1
D3	GitHub notifies the System correctly when a new commit is pushed.	OG1
D4	GitHub allows the System to pull the latest version of the code from the repository.	OG1
D5	Users know the basics of Git and can use the GitHub platform correctly.	OG1

D6	All the notifications sent by the System are delivered to the User in a reasonable amount of time.	SG1, SG2, SG4, SG5, SG6, SG7, EG1, EG3, OG3, OG4
D7	The SAT provides correct analysis in relation to the setting provided by the User.	OG1, SG3
D8	Users can create GitHub Accounts.	OG7

3 - Specific Requirements

3.1 - External Interface Requirements

3.1.1 - User Interfaces

CKB users operate through a web-based interface, either on desktop or on their smartphone. When they interact with the external GitHub system, they do so using the GitHub user interface, which is also web-based.

The web application should be easy to use for both students (to submit their solutions and visualize their ranking) and educators (to create new tournaments, let their students join and upload code katas).

3.1.2 - Hardware Interfaces

Users are required to have access to a device capable of connecting to the internet (e.g. smartphone, laptop, desktop PC) in order to interact with the System, which is a web application.

3.1.3 - Software Interfaces

The System requires to connect to the following software interfaces in order to work correctly:

- GitHub API: in order to implement login with GitHub.

3.1.4 - Communication Interfaces

The System needs a constant active internet connection in order to guarantee availability to every user, regardless of their location. It also needs to implement authentication protocol with GitHub API in order to guarantee "Login with Github" functionality.

3.2 - Functional Requirements

3.2.1 - Students

ID	Description	Related Goals
SR1	The System notifies all IUs when a tournament is created.	SG1
SR2	The System notifies all UST when a new Tournament's Battle is created.	SG2
SR3	The System creates a score rank for each Tournament's Battle, where the Team's score is the maximum score Students managed to accomplish with their last submission.	SG3
SR4	The System allows Students to join in a Battle from its Tournament, if EBD is not expired and there are available spots in three different ways: ● [SR4.1] via notification (from the system or from another student) ; ● [SR7.2] via website, accessing it from the tournament page;	SG4
SR5	The System notifies all SST when a new Battle is created.	SG4

SR6	The System allows SST to send an invitation to another SST to join a Battle within their Team, if the EBD is not expired and the Team is not full.	SG4, SG5
SR7	The System allows Students to join a Tournament if the CD is before the ETD, and the MNS isn't reached, in three different ways: ● [SR7.1] via notification. In this case the student is automatically accepted; ● [SR7.2] via website (by searching it or finding it on a user profile). In this case the user is accepted only if DAS; ● [SR7.3] via link. In this case the user is accepted only if DAS.	SG6
SR8	As soon as EB occurs the student receives a notification to see the final Battle leaderboard.	EG3
SR9	The system does not allow a Student to join a Team if it is full.	SG4, SG5
SR10	If a Student accepts an invitation in a Team A, and he is already in another Team B, they leave B and enter in A.	SG4, SG5
SR11	The System allows a Student to see the following information about a tournament in which they are subscribed: participants, ongoing battles and their leaderboard, TCs, ETD and FTD.	OG9

3.2.2 - Educators

ID	**Description**	**Related Goals**

ER1	The System allows an Educator to create a Tournament specifying the ETD, FTD and MNS.	EG1
ER2	The System allows an Educator to modify ETD and FTD of a Tournament if they are the OCT for that Tournament.	EG1
ER3	Upon the creation of a Tournament , the System allows an Educator to send the invitation to another Educator to become TC in the Tournament.	EG1
ER4	Upon the creation of a Tournament, The System allows an Educator to define the Badges that can be obtained in their own Tournament.	EG1
ER5	Upon the creation of a Tournament, The System allows an Educator to create new Badges, specifying the rules to obtain them.	EG4
ER7	The System allows an Educator to create Battles in their own Tournaments or in other Tournaments where they are the TC, if the maximum limit of contemporary Battles in the Tournament imposed by the System isn’t reached.	EG2
ER8	Upon the creation of a Battle, The System allows an Educator to upload a kata to a battle belonging to their own tournament or to a tournament in which they are TC.	EG2
ER9	Upon the creation of a Battle, The System allows Educators to set the EBD and FBD to their Battles, as	EG2

	well as the maximum and minimum number of students per team.	
ER10	The System disallows an Educator to violate this condition: CD<=ETD<=EBD<FBD<=FTD.	EG1, EG2
ER11	In a Battle, after EBD expires: if OME has been selected, participants receive a notification to inform them that it isn't possible to push new submissions anymore.	EG3
ER12	Upon the creation of a Battle, the System allows Educators to set additional configuration for scoring for that Battle.	EG2
ER13	After EBD is reached for a Battle where OME is enabled, the System sends a notification to the Educator that created the Battle, inviting them to perform manual evaluation.	EG3
ER14	The System sends a notification to OCT to admit or reject a Student that isn't DAS and wants to join the Tournament.	SG6
ER15	The System shows Tournament's link to their OCT.	SG6
ER16	The System allows an Educator that is OCT to ban a Student from their Tournament.	EG5
ER17	The System disallows an Educator to join a Tournament in which they are OCT or TC.	

ER18	The System allows an Educator to accept or reject the invitation to become TC in a Tournament.	EG1
ER20	The System allows an Educator to see the following information about a tournament in which they are OCT or TC: participants, ongoing battles and their rank, TC's, ETD and FTD	OG8
ER21	Upon the creation of a Tournament, the System allows an Educator to select the privacy level of the Tournament among the following options: ● Public (everyone can join); ● Friends Only (only friends of OCT can join); ● Private (it is only possible to join by link, Tournament is always hidden from user profiles);	EG1
ER22	The System allows an Educator to modify the EBD and FBD of a battle they have created.	EG2

3.2.3 - Other

ID	Description	Related Goals
OR1	The System creates a GitHub repository to be forked and used by the Users to push their code.	OG1.1
OR2	The System allows all Users subscribed to the platform to see the list of ongoing tournaments and the corresponding leaderboards.	OG2
OR3	As soon as the FTD is reached, the SST, OCT and TCs receive a notification to see the final tournament leaderboard.	OG4

OR4	The System allows any User to send friend requests to any other User, regardless of their privacy policy.	OG3
OR5	The System allows Users to accept or refuse friend requests from any other User.	OG3
OR6	The System allows Users to decide to receive notification about Tournaments: ● From all Users in CKB; ● Only from friends; ● From no one.	OG5
OR7	The System allows Users to visualize collected badges and other relevant information when they visit a User's profile.	OG6
OR8	The System allows a Users to create a profile on the platform only using their GitHub account.	OG7
OR9	The System allows Users to login in their profile only via GitHub.	OG7
OR10	The System calculates the score, applying MAE, for each Team from a Tournament's Battle as soon as the Student makes a push to their forked repository.	SG3, OG1.2
OR11	When a Battle is terminated (with MAE or not), the System updates the Tournament leaderboard by summing for each Student their Team's score in the Battle.	OG2

OR12	When the FTD of a Tournament is reached, the System assigns the Badges to Students who respect the Badge’s rules.	EG4
OR13	The System disallows an Educator to become TC of a Tournament if that Tournament has a number of TCs equal to the maximum number of TCs imposed by the System.	EG1
OR14	The System disallows OCT to set a MNS that exceeds the maximum value of MNS imposed by the system.	EG1
OR15	The System delivers notifications related to a Tournament only to the IUs.	SG1
OR16	As soon as the FTD is reached, if there are Battles that are not ended yet due to OME being enabled not performed yet, the System updates the final Tournament leaderboard using the current battle leaderboard (and by doing so, ignoring the OME option), with the same modality of {OR11}	OG4
OR17	The System executes a pull operation on the GitHub repository created by a Student when GitHub notifies it of a new push on that repository. The System associates the pulled version to the Team which the Student is part of.	OG1

3.2.4 - Mapping on Goals

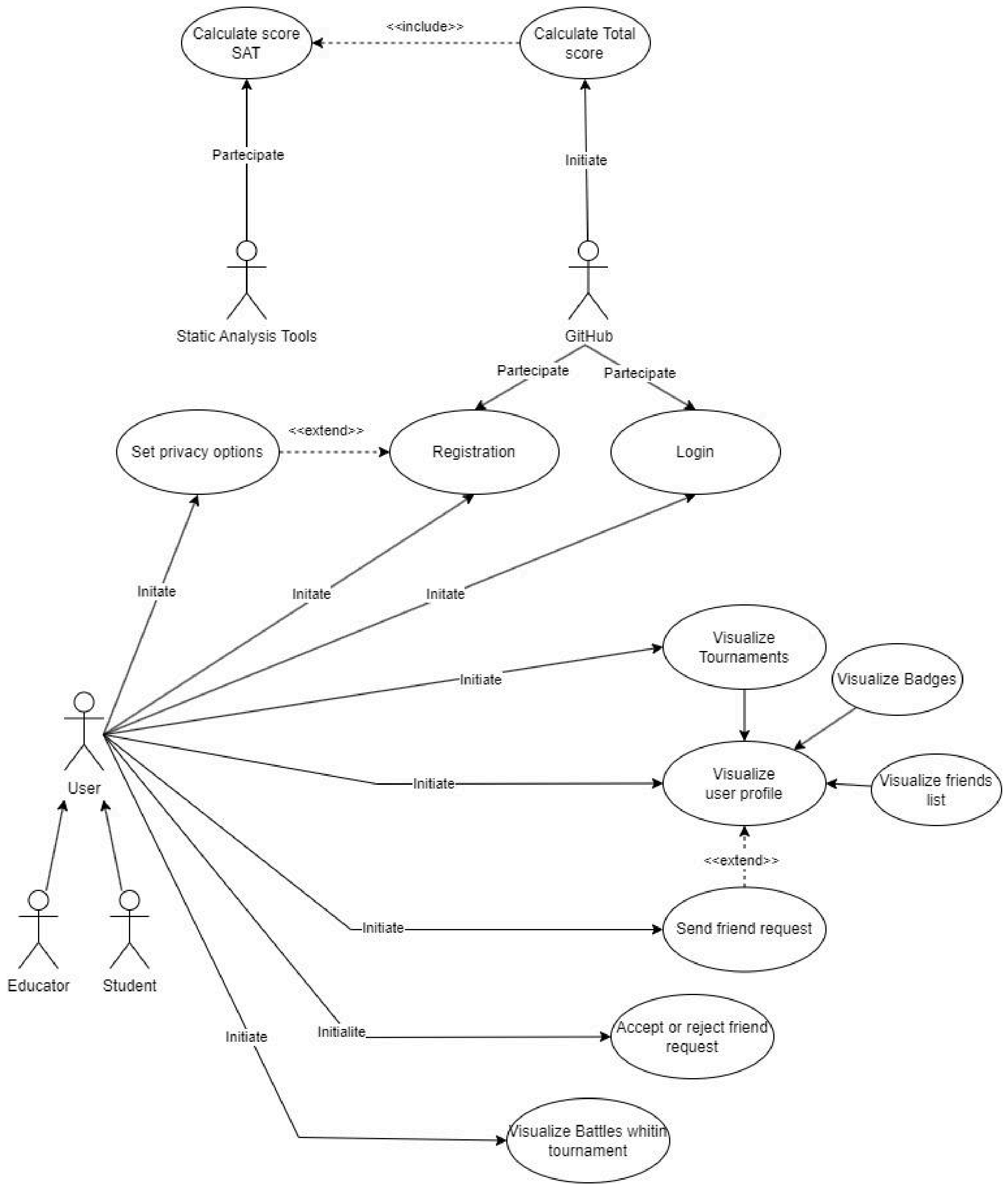
This section shows how the relation R∧D |= G holds through a traceability matrix that associates domain assumptions and requirements to each goal.

Goal	**Requirements**	**Domain Assumption**

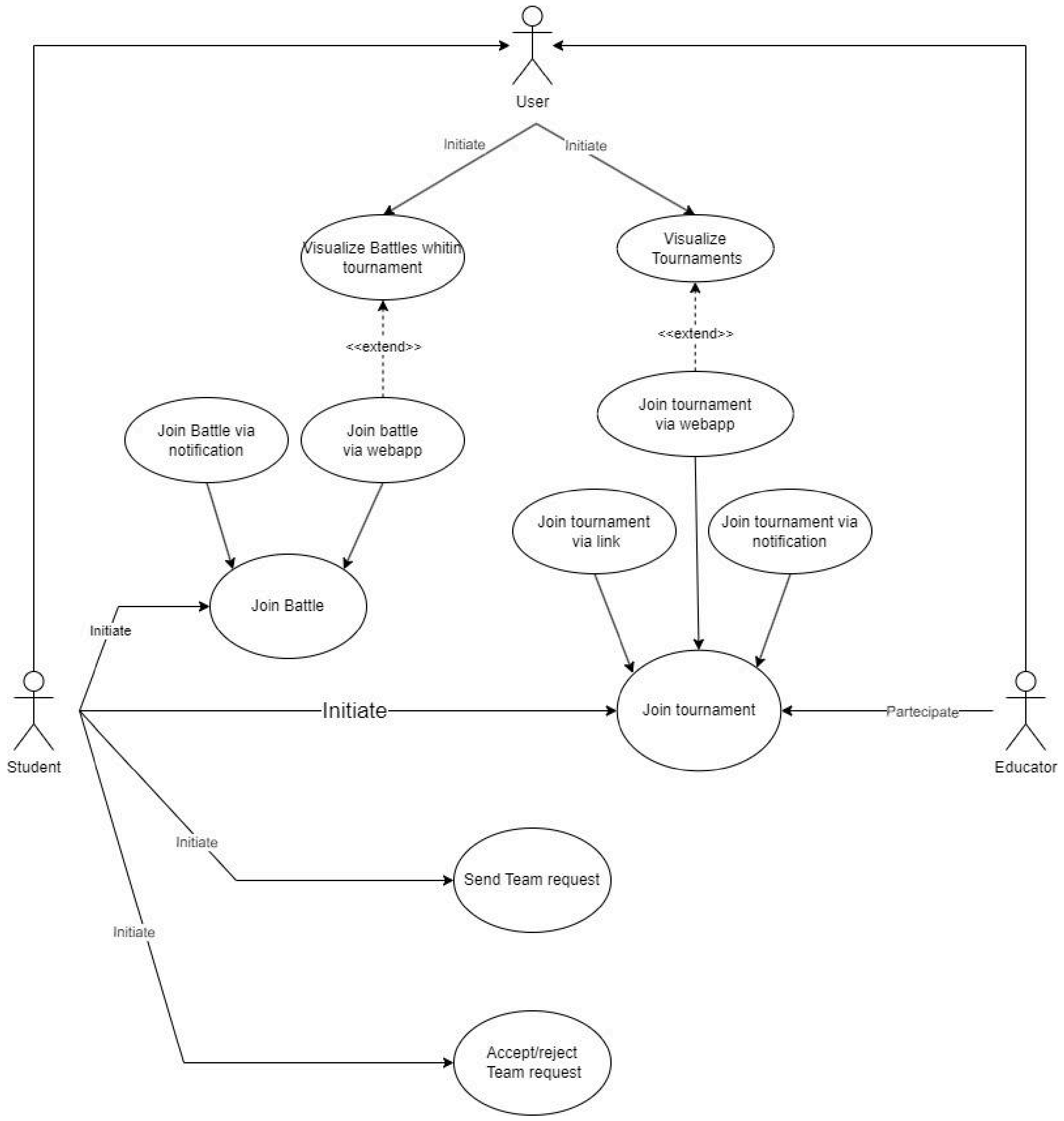
SG1	SR1, OR15	D1, D6
SG2	SR2	D1, D6
SG3	SR3, OR10	D1, D7
SG4	SR4, SR5, SR6, SR9, SR10	D1, D6
SG5	SR6, SR9, SR10	D1, D6
SG6	SR7, ER15, ER14	D1, D6
EG1	ER1, ER2, ER3, ER4, ER10, ER18, OR13, ER21, OR14	D1, D6
EG2	ER7, ER8, ER9, ER10, ER12, ER22	D1
EG3	ER13, SR8, ER11	D1, D6
EG4	ER5, ER6, OR12	D1
EG5	ER16	D1
OG1	OR1, OR10, OR17	D2, D3, D4, D5, D7
OG2	OR2, OR11	D1
OG3	OR4, OR5	D1, D6
OG4	OR3, OR16	D1, D6
OG5	OR6	D1
OG6	OR7	D1
OG7	OR8, OR9	D1, D8
OG8	SR11, ER20	D1

3.2.5 - Use Case Diagrams

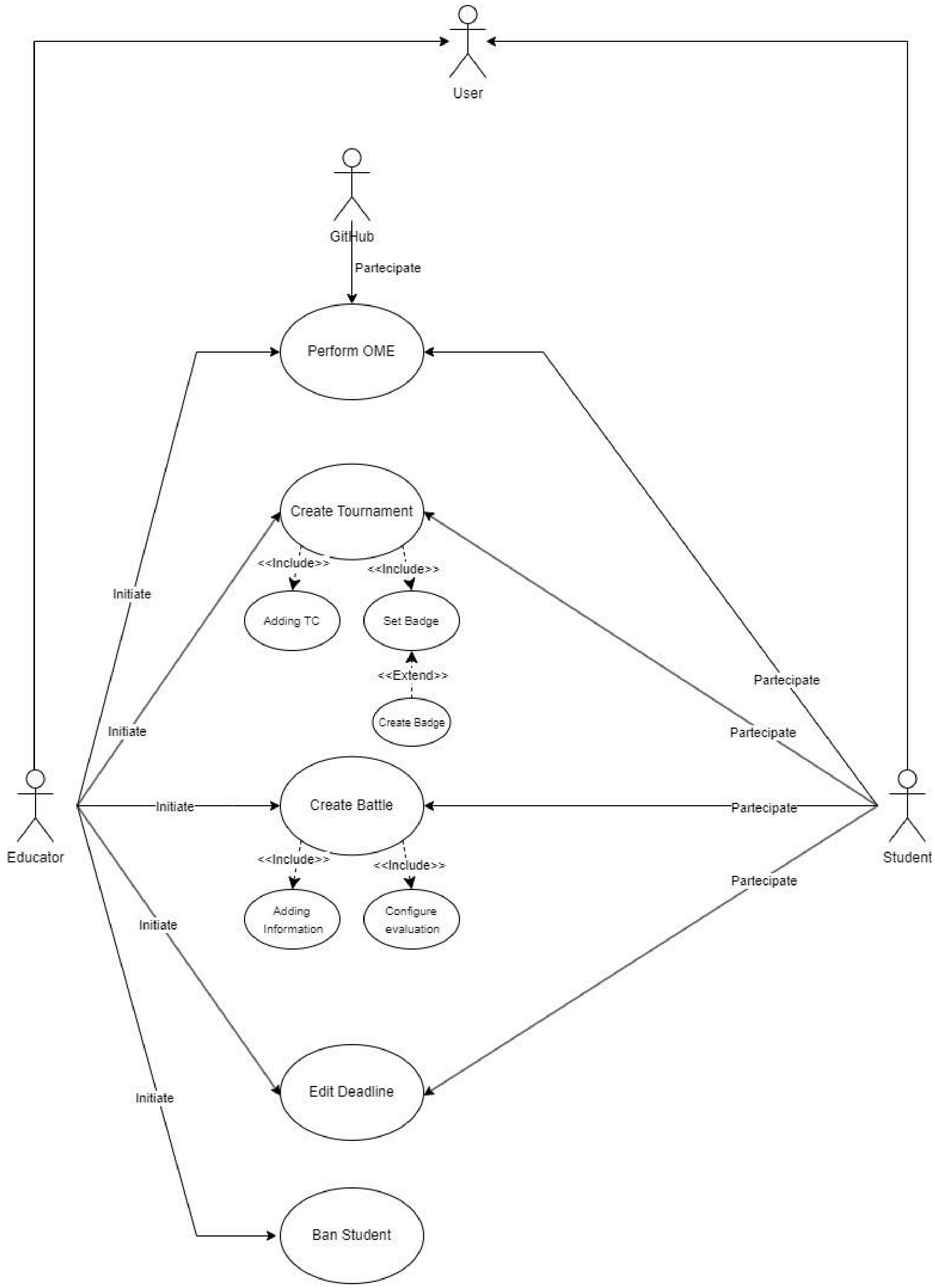
User



Student


User
Initiate
Initiate
Visualize Battles whitin tournament
Visualize Tournaments
<<extend>>
<<extend>>
Join Battle via notification
Join battle via webapp
Join tournament via webapp
Join tournament via link
Join tournament via notification
Join Battle
Initiate
Initiate
Join tournament
Partecipate
Student
Educator
Initiate
Send Team request
Initiate
Accept/reject Team request

Educator


User
GitHub
Partecipate
Perform OME
Create Tournament
<<Include>>
<<Include>>
Adding TC
Set Badge
<<Extend>>
Create Badge
Initiate
Initiate
Initiate
Initiate
Initiate
Educator
Create Battle
<<Include>>
<<Include>>
Adding Information
Configure evaluation
Partecipate
Partecipate
Partecipate
Partecipate
Student
Edit Deadline
Ban Student

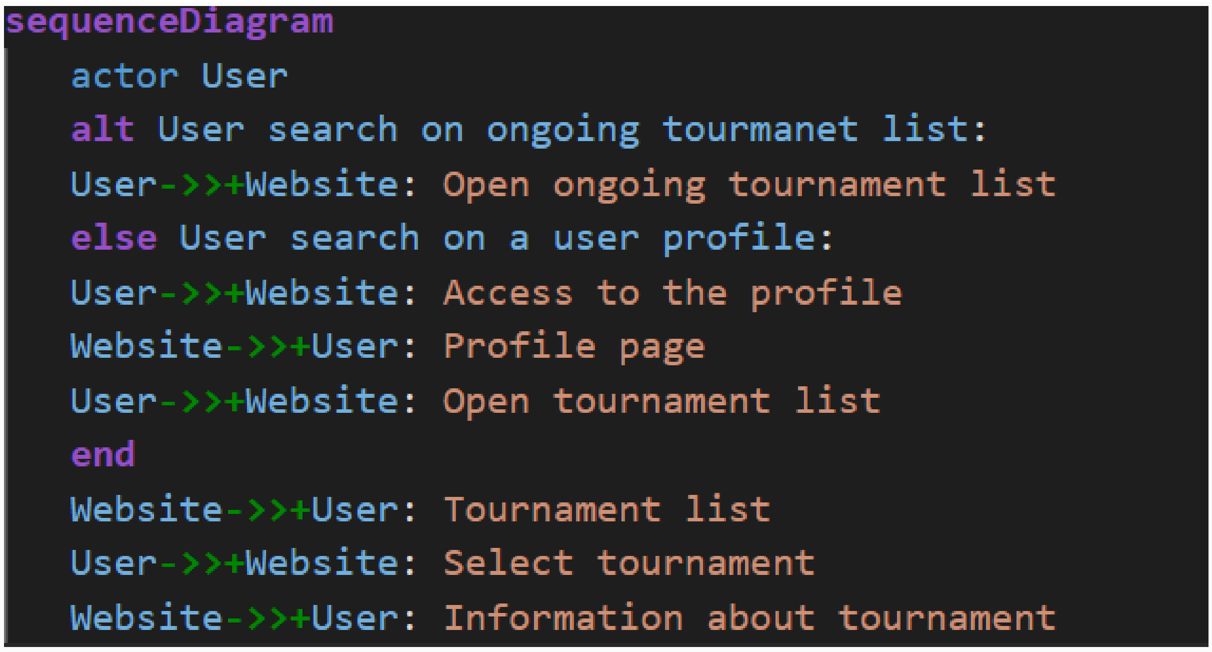
3.2.6 - Use Cases

In this section, there are listed use cases. For each one, there is:

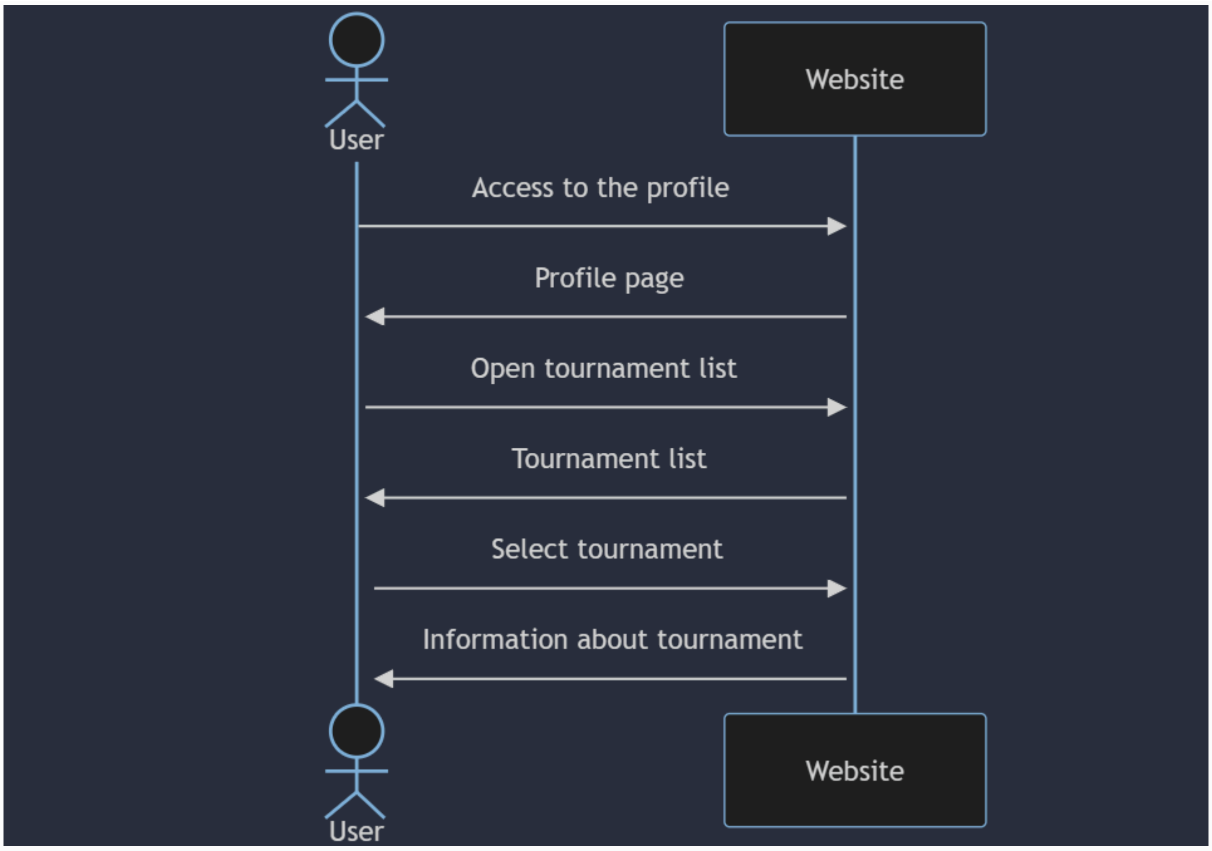
- a table with name, actor(s), entry condition(s), event flow, exit condition(s) and alternative flows;
- sequence diagram in textual form;
- sequence diagram rendered in graphical form;

UC1. Visualize tournaments

Name	Visualize tournaments
Actor	User
Entry Condition	User is registered and logged in on the website
Event Flow	1a. The user goes on their profile page. 2. The user opens the list of tournaments on the profile page. 3. The user opens the tournament page of the tournament they want to visualize. 4. The user visualizes all the information about that tournament and if OCT visualizes the tournament link too.
Exit Condition	The user goes back to their profile page.
Alternative flows	1b. The user opens the ongoing tournament page. In this case step 2 is skipped. 1c.The user opens another user’s profile page.

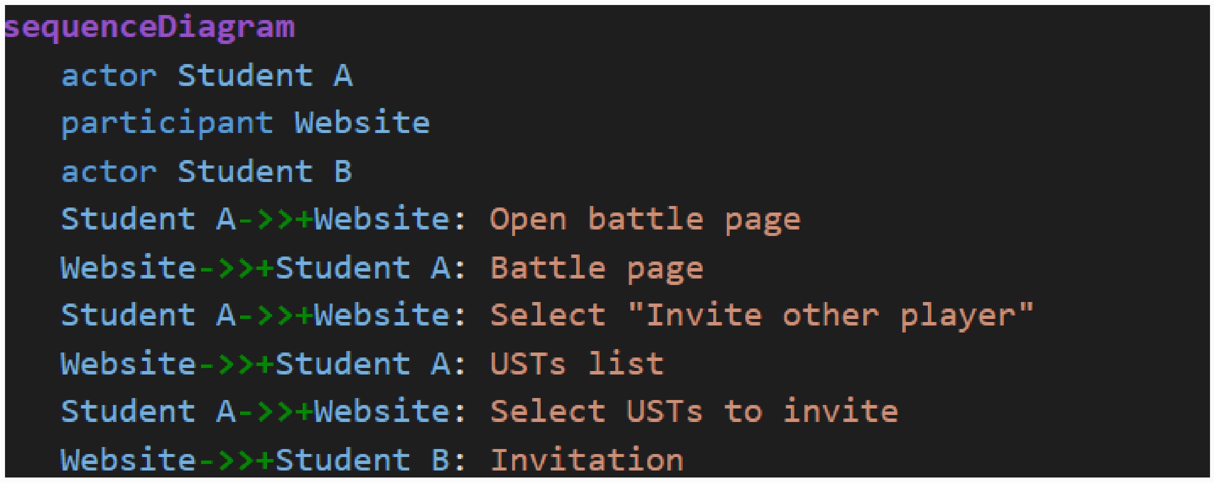


```
sequenceDiagram
   actor User
   alt User search on ongoing tourmanet list:
   User->>+Website: Open ongoing tournament list
   else User search on a user profile:
   User->>+Website: Access to the profile
   Website->>+User: Profile page
   User->>+Website: Open tournament list
   end
   Website->>+User: Tournament list
   User->>+Website: Select tournament
   Website->>+User: Information about tournament
```

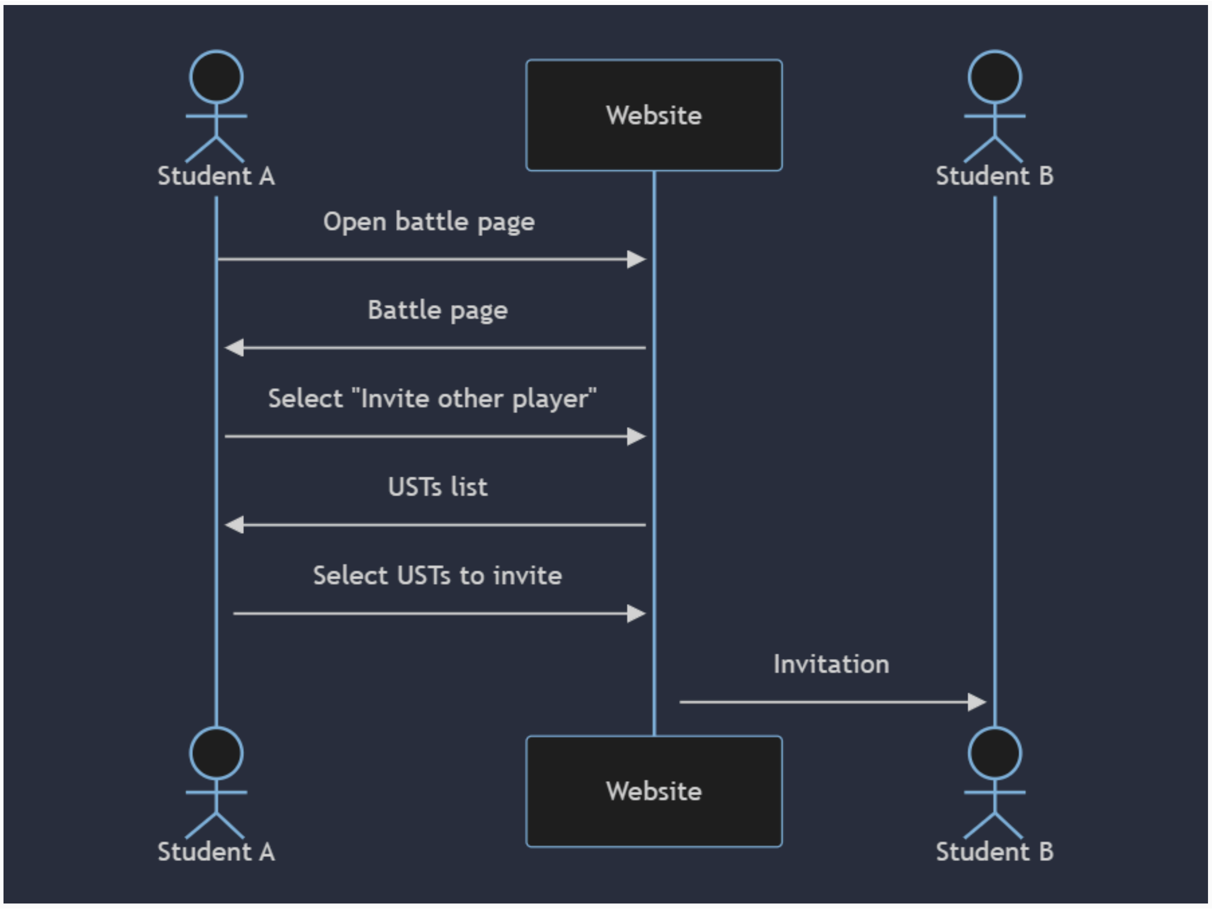


UC2. Send team request

Name	Send team request
Actor	Student
Entry Condition	Student is part of a team in a battle and the EBD is not expired
Event Flow	1. The student goes on the battle page. 2. The student selects the “Invite other player” option. 3. The student select the other students they want to invite among the USTs. 4. A notification with the invitation to join the student’s team is sent to the selected USTs.
Exit Condition	The student is no longer interested in inviting other students.



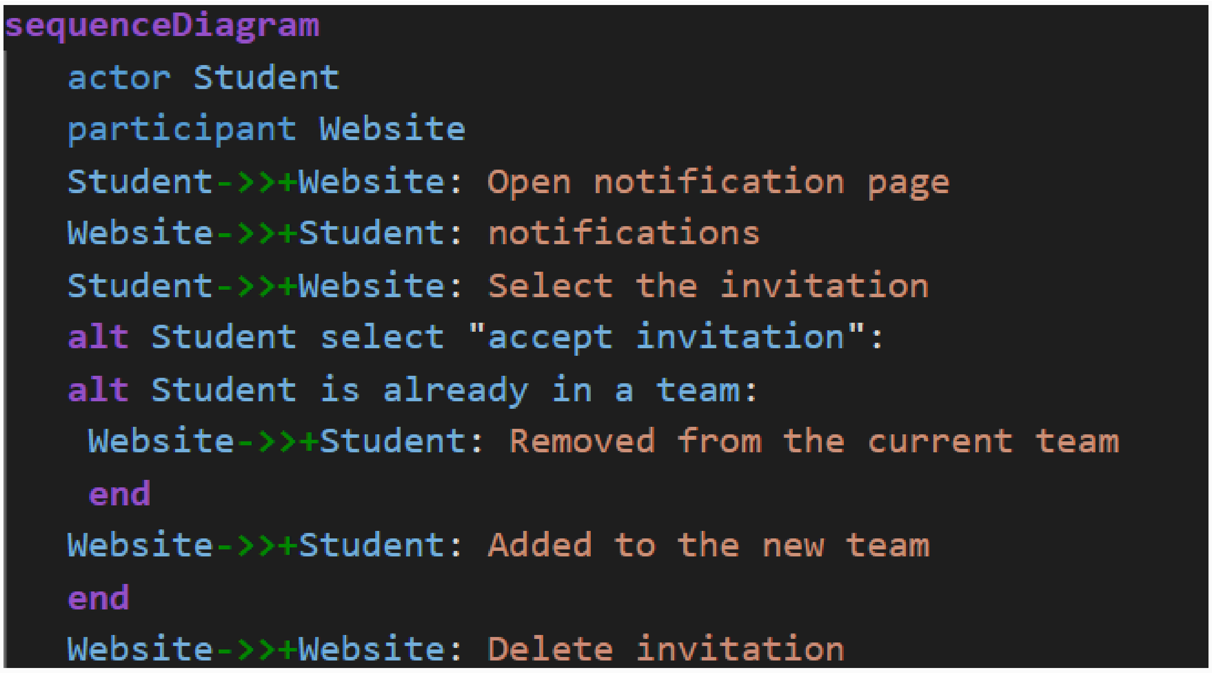
```
sequenceDiagram
   actor Student A
   participant Website
   actor Student B
   Student A->>+Website: Open battle page
   Website->>+Student A: Battle page
   Student A->>+Website: Select "Invite other player"
   Website->>+Student A: USTs list
   Student A->>+Website: Select USTs to invite
   Website->>+Student B: Invitation
```



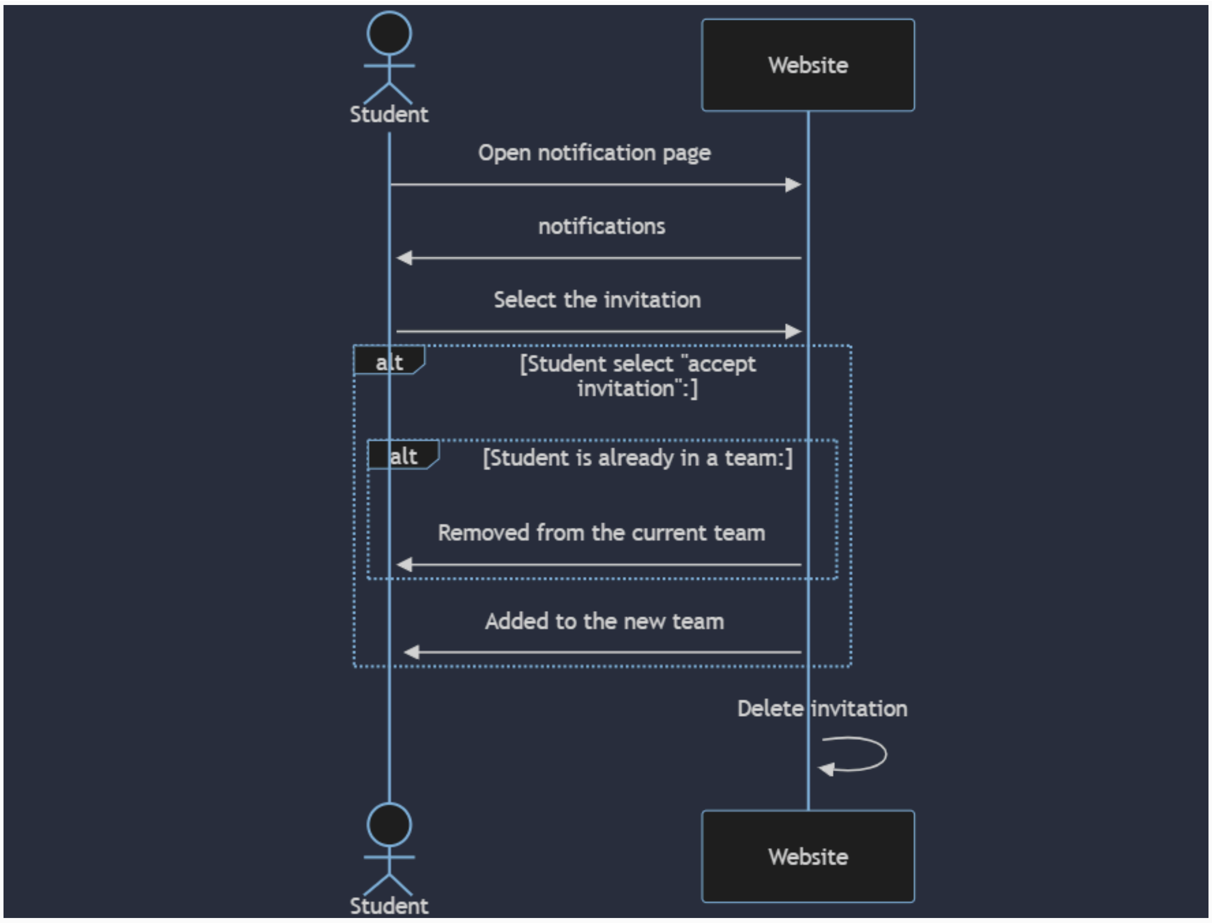
UC3. Accept/reject team request

Name	Accept/reject team request
Actor	Student
Entry Condition	Student is subscribed to a tournament.

Event Flow	1. The student opens their notification page. 2. The student selects the notification with the label “Invite to join a battle team”. 3a. The student selects the accept option. 4a. The student is not part of a team for the battle in which they are invited, so they’re directly added to the team they are invited in.
Exit Condition	The student has successfully accepted or rejected the invitation.
Exceptions	The team in which the student has been invited is full. When the student accepts the invitation they do not join the team and an error message occurs. The EBD for that battle is expired. When the student accepts the invitation they do not join the team and an error message occurs.
Alternate Flows	3b. The student selects the reject option. 4b. The student does not join the team and the operation terminates. 4c. The student is already part of a battle team for the battle in which they are invited, so they leave the former team and join the new one.

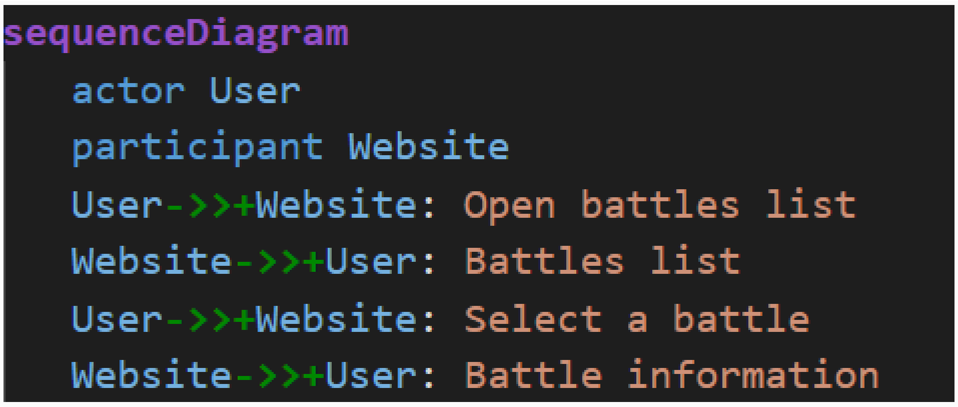


```
sequenceDiagram
   actor Student
   participant Website
   Student->>+Website: Open notification page
   Website->>+Student: notifications
   Student->>+Website: Select the invitation
   alt Student select "accept invitation":
   alt Student is already in a team:
    Website->>+Student: Removed from the current team
    end
   Website->>+Student: Added to the new team
   end
   Website->>+Website: Delete invitation
```

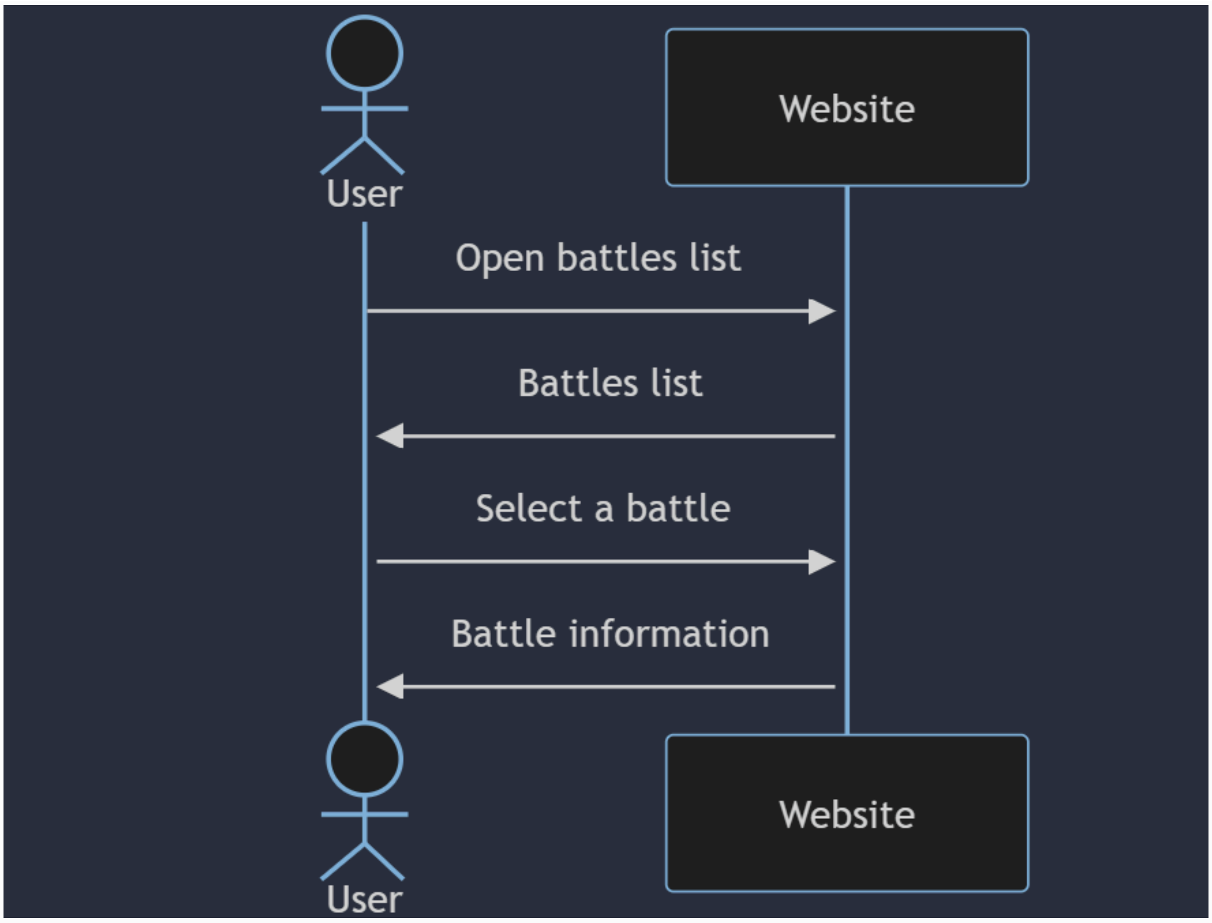


UC4. Visualize Battles within tournament

Name	Visualize Battles within tournament
Actor	User
Entry Condition	User has opened the tournament page of a tournament in which they are involved
Event Flow	1. The user opens the list of tournament battles 2. The user opens the battle page of the battle they want to visualize. 3. The user visualizes all the information about that battle.
Exit Condition	The user goes back to the tournament page.



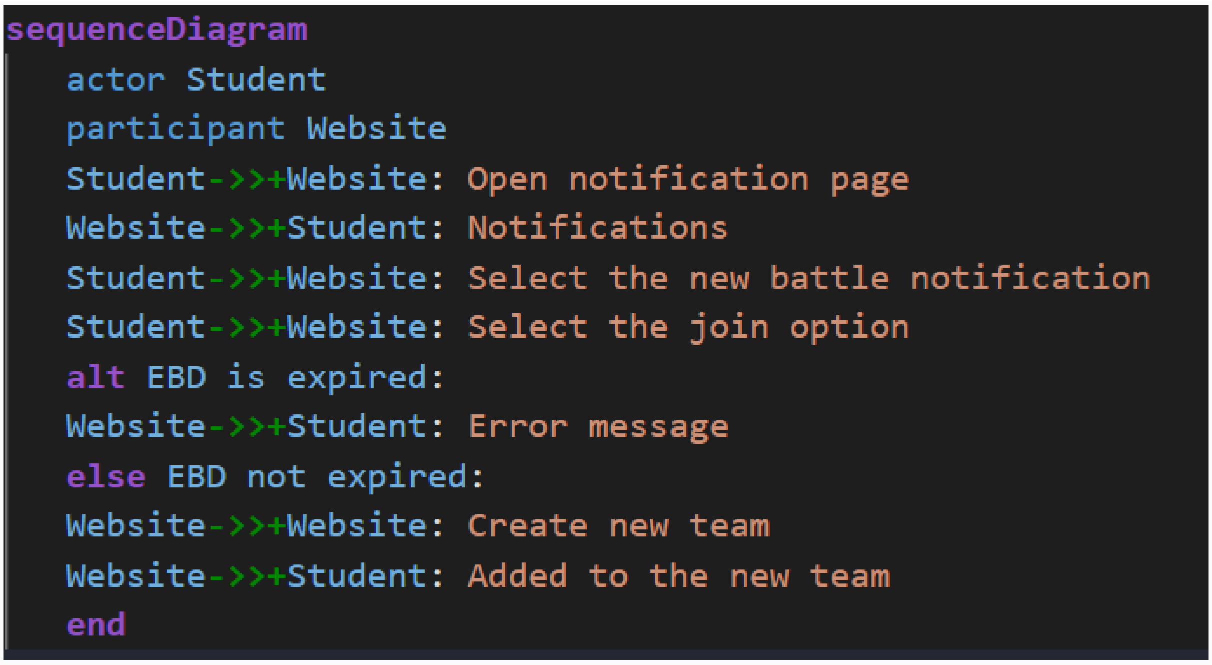
```
sequenceDiagram
    actor User
    participant Website
    User->>+Website: Open battles list
    Website->>+User: Battles list
    User->>+Website: Select a battle
    Website->>+User: Battle information
```



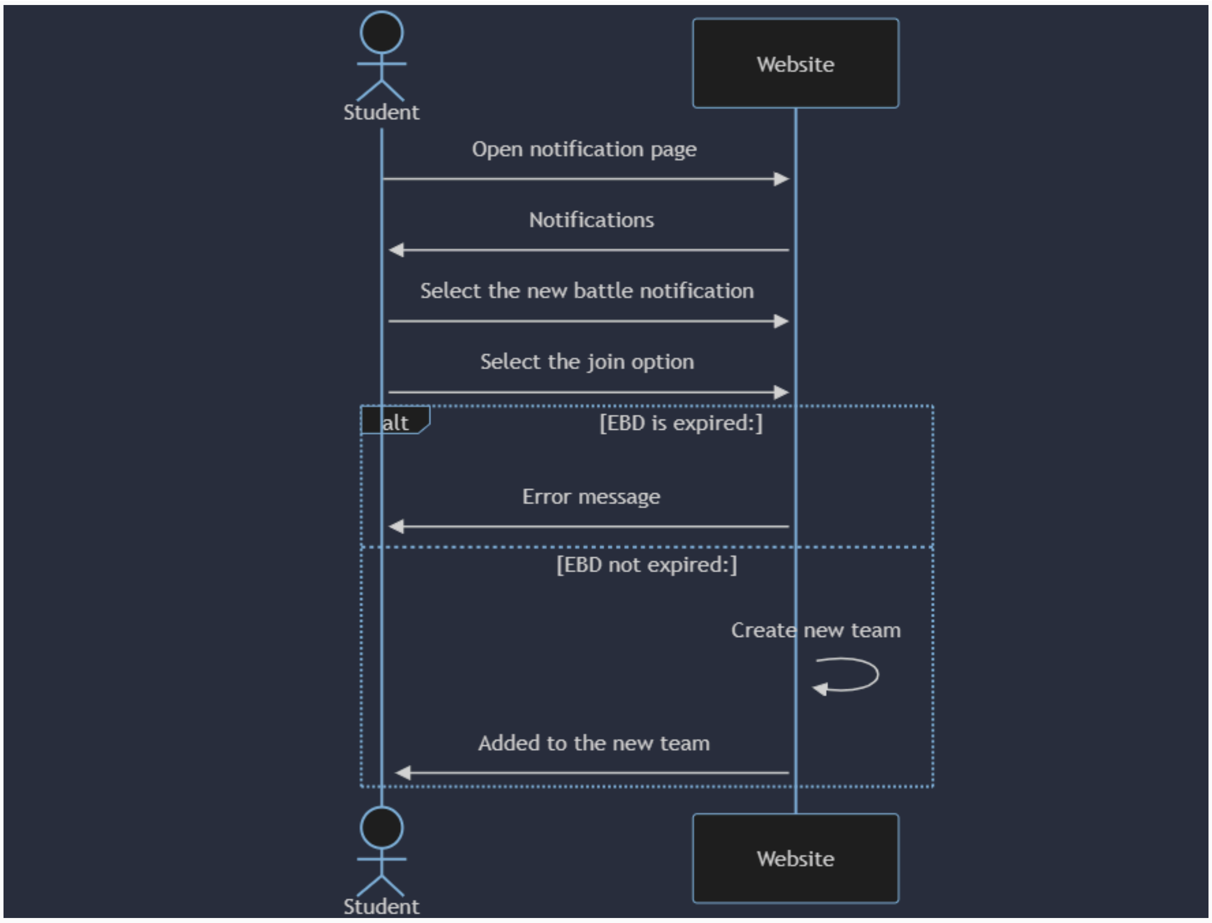
UC5. Join battle via notification

Name	Join battle via notification
Actor	Student
Entry Condition	Students are subscribed to a tournament.

Event Flow	1. The student opens their notification page. 2. The student selects the notification with the label “New battle created”. 3. The student selects the join option. 4. A new team is created and the student joins it.
Exit Condition	The student successfully joins the battle.
Exceptions	The EBD for that battle is expired. When the student tries to join the battle an error message occurs.



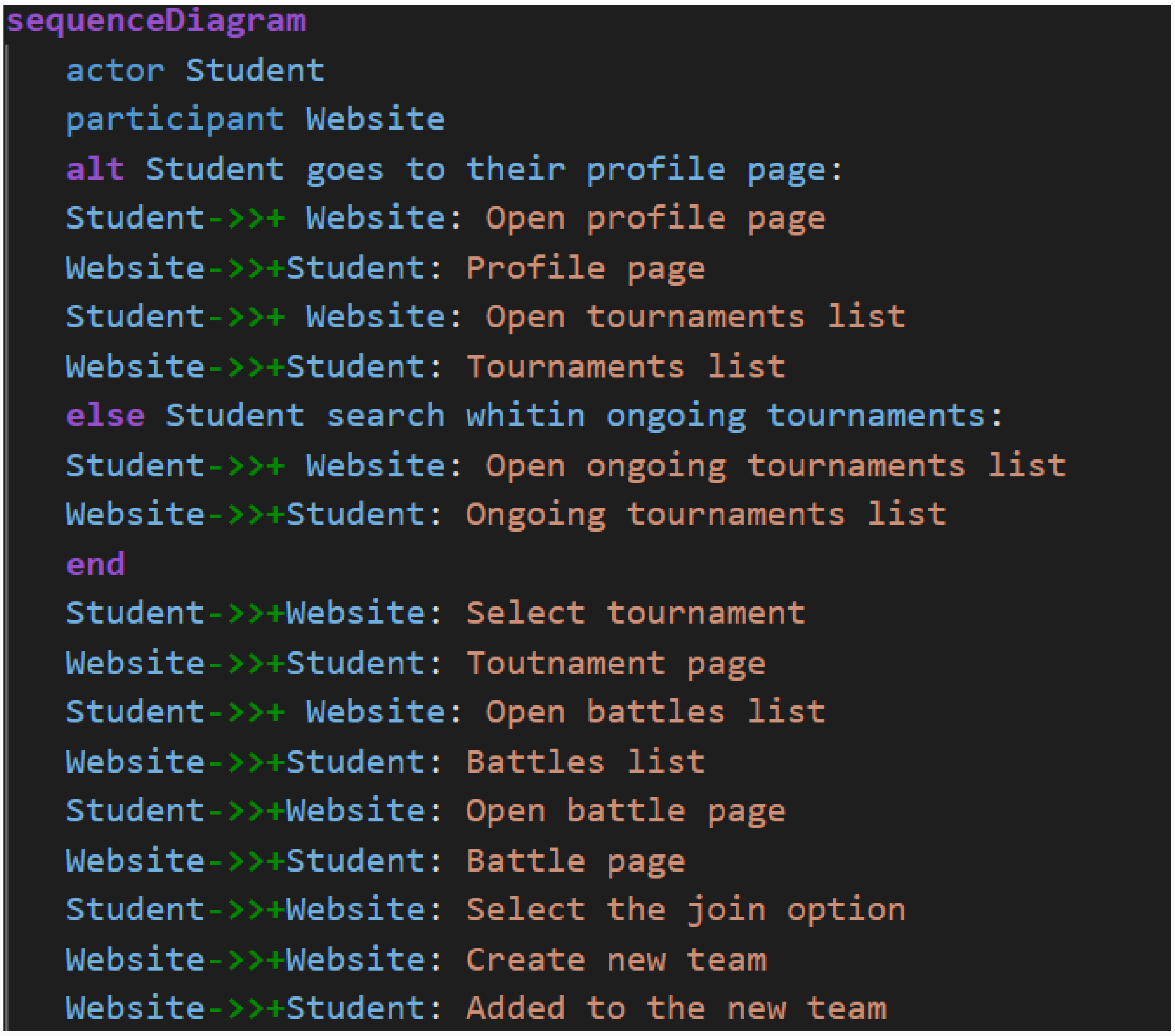
```
sequenceDiagram
    actor Student
    participant Website
    Student->>+Website: Open notification page
    Website->>+Student: Notifications
    Student->>+Website: Select the new battle notification
    Student->>+Website: Select the join option
    alt EBD is expired:
    Website->>+Student: Error message
    else EBD not expired:
    Website->>+Website: Create new team
    Website->>+Student: Added to the new team
    end
```



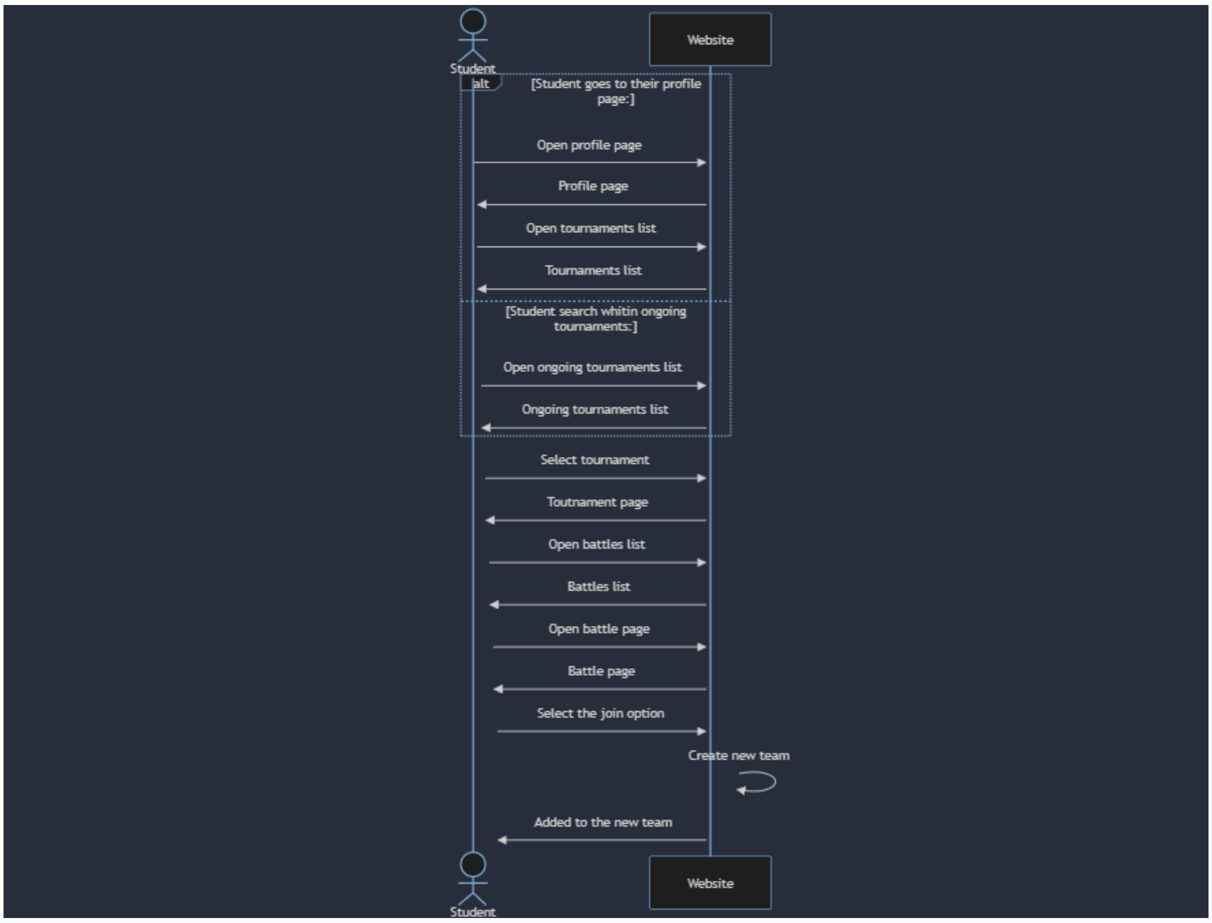
UC6. Join battle via webapp

Name	Join battle via webapp
Actor	Student
Entry Condition	Students are subscribed to a tournament.
Event Flow	1a. The student goes on their profile page. 2a. The student opens the list of tournaments on their profile page. 3. The students open the tournament page of the tournament where the battle they want to join was created. 4. The student opens the list of tournament battles. 5. The student opens the battle page of the battle they want to join. 6. The student selects the join option. 7. A new team is created and the student joins it.
Exit Condition	The student successfully joins the battle.
Exceptions	The EBD for that battle is expired. When the student tries to join the battle an error message occurs.

Alternate Flows	1b. The student searches the tournament manually among the ongoing tournaments. 2b. The students select from the results of the research the tournament in which they're interested.



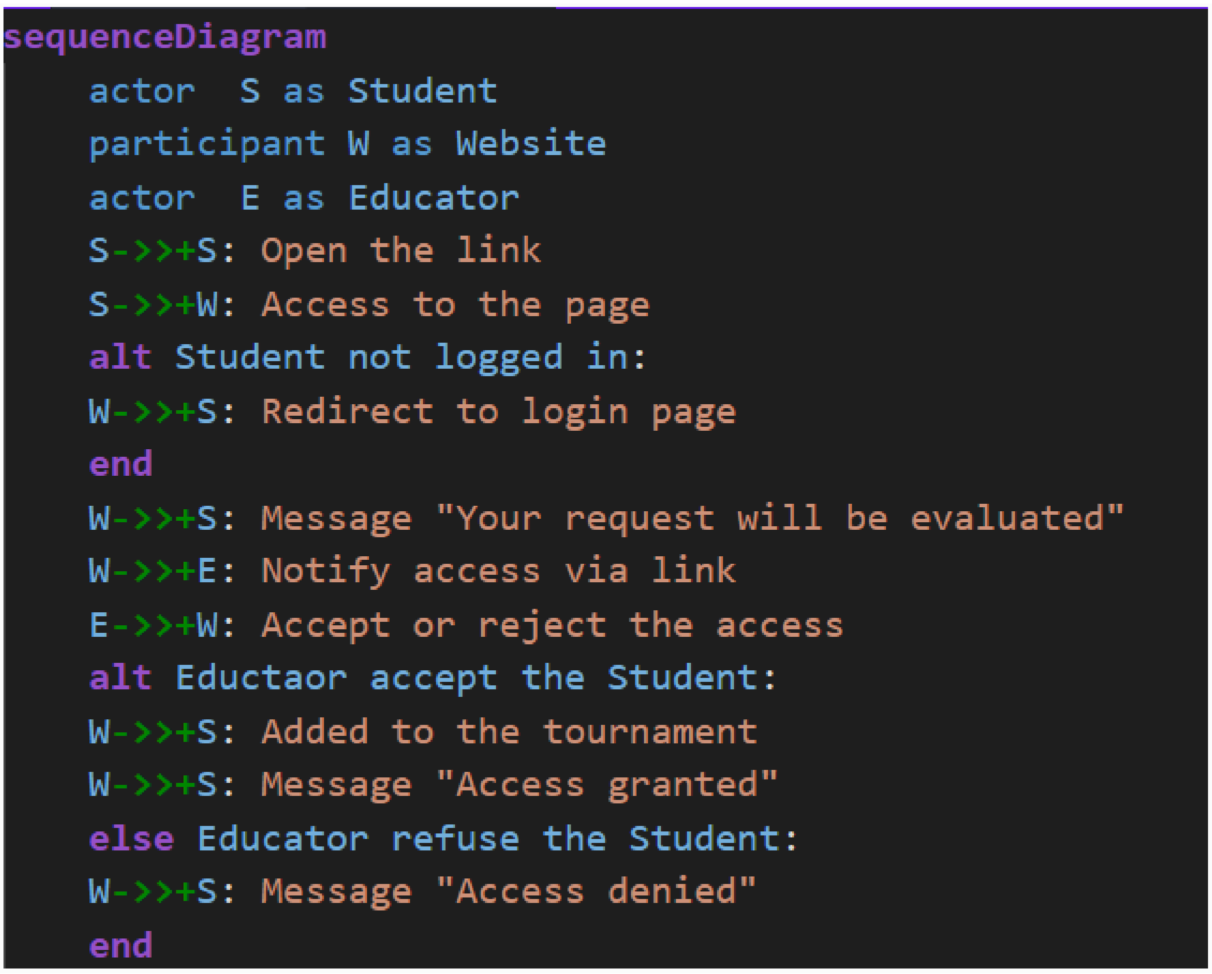
```
sequenceDiagram
    actor Student
    participant Website
    alt Student goes to their profile page:
    Student->>+ Website: Open profile page
    Website->>+Student: Profile page
    Student->>+ Website: Open tournaments list
    Website->>+Student: Tournaments list
    else Student search whitin ongoing tournaments:
    Student->>+ Website: Open ongoing tournaments list
    Website->>+Student: Ongoing tournaments list
    end
    Student->>+Website: Select tournament
    Website->>+Student: Toutnament page
    Student->>+ Website: Open battles list
    Website->>+Student: Battles list
    Student->>+Website: Open battle page
    Website->>+Student: Battle page
    Student->>+Website: Select the join option
    Website->>+Website: Create new team
    Website->>+Student: Added to the new team
```



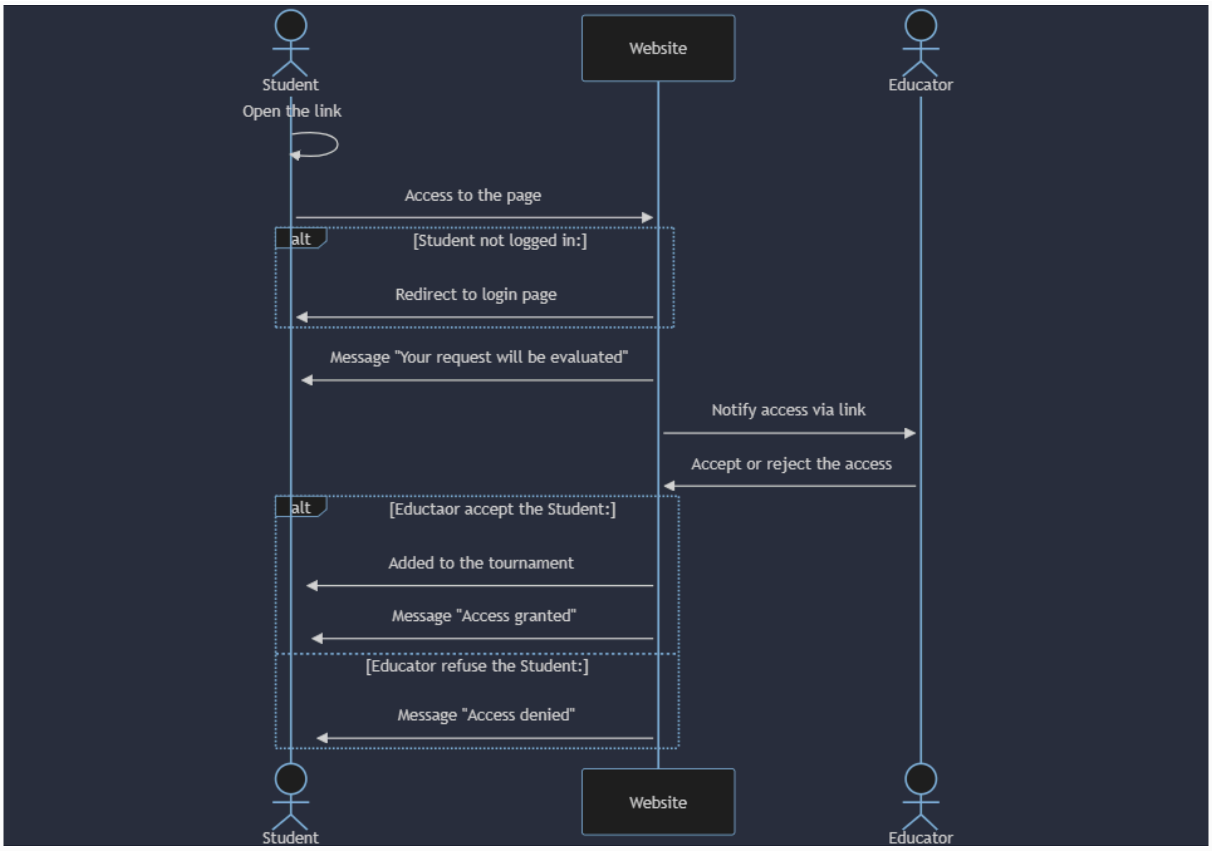
UC7. Join tournament via link

Name	Join tournament via link
Actor	Student, Educator (the OCT of the tournament)
Entry Condition	Student has received the link from outside the application.
Event Flow	1. The student opens the link given by the educator (outside the application). 2a. The student is logged in the site, the website page is opened. 3. A message informs the student that their request is being evaluated from the OCT. 4. The system informs the OCT that someone is trying to join their tournament directly. 5a. The OCT accepts the user. 6a. The student joins the tournament and is notified that their request has been accepted.
Exit Condition	The user either successfully joins the tournament or is rejected by the OCT.

Exceptions	The ETD for that tournament has expired. When the student tries to join the tournament an error message occurs. The MNS is already reached. When the student tries to join the tournament an error message occurs.
Alternate Flows	2b. The student is not logged in the site, so they're asked to login via GitHub. Once logged, the website page is opened. 5c. The OCT rejects the student. 6c. The student doesn't join the tournament and is notified that their request has been rejected.

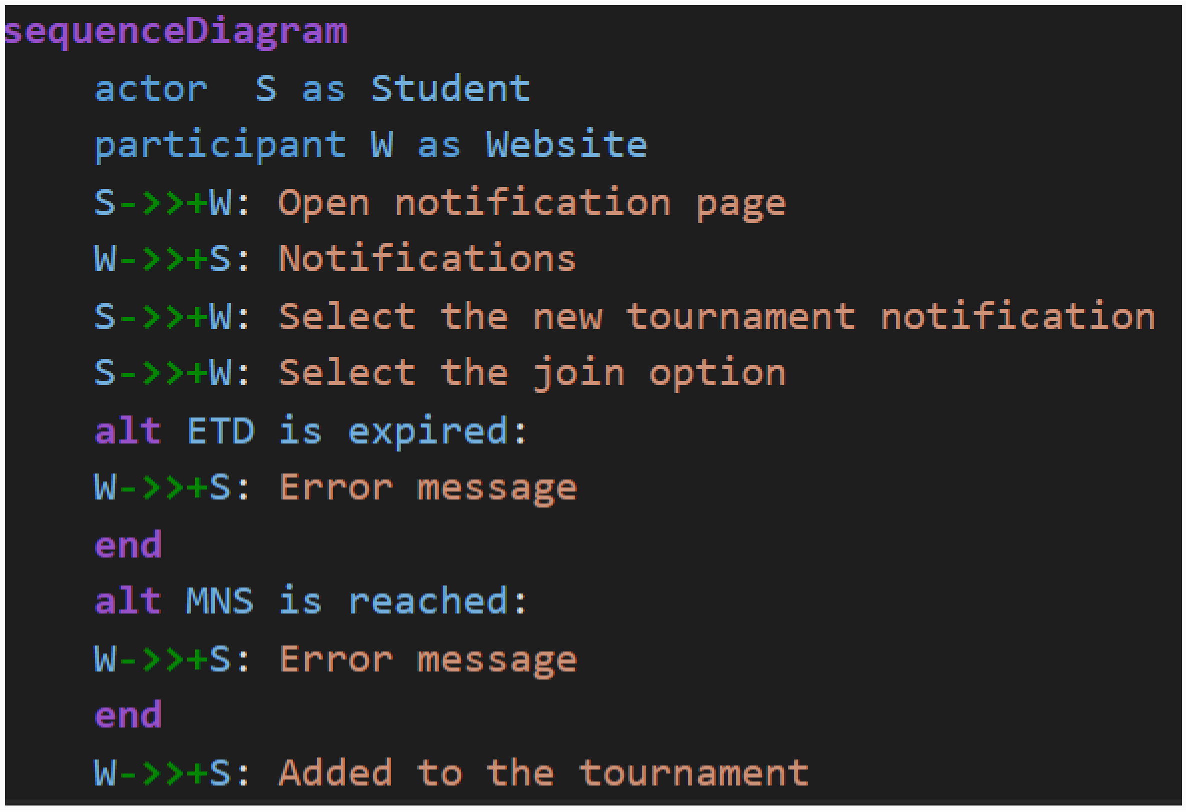


```
sequenceDiagram
    actor  S as Student
    participant W as Website
    actor  E as Educator
    S->>+S: Open the link
    S->>+W: Access to the page
    alt Student not logged in:
    W->>+S: Redirect to login page
    end
    W->>+S: Message "Your request will be evaluated"
    W->>+E: Notify access via link
    E->>+W: Accept or reject the access
    alt Eductaor accept the Student:
    W->>+S: Added to the tournament
    W->>+S: Message "Access granted"
    else Educator refuse the Student:
    W->>+S: Message "Access denied"
    end
```

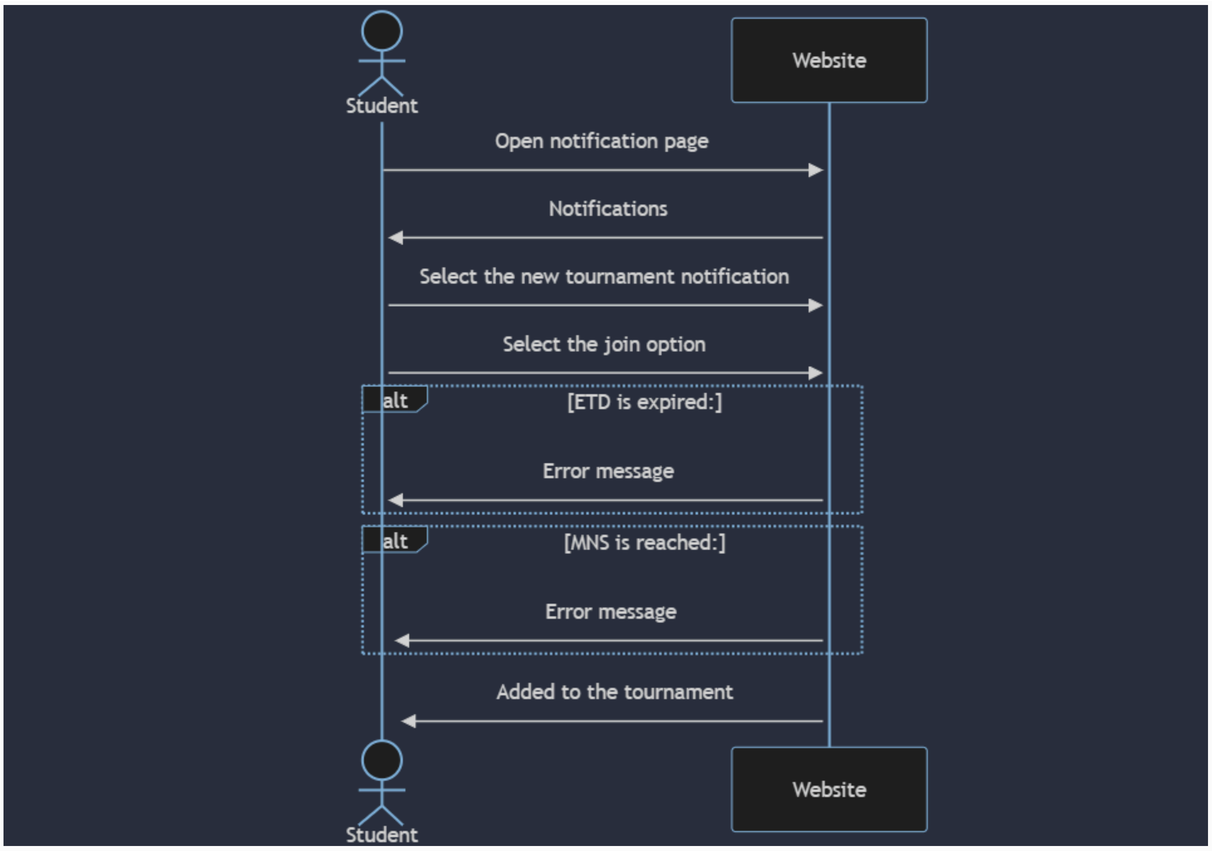


UC8. Join tournament via notification

Name	Join the tournament via notification.
Actor	Student
Entry Condition	Student is registered and logged in on the website
Event Flow	1. The student opens their notification page. 2. The student selects the notification with the label “New tournament created”. 3. The student selects the join option. 4. The student joins the tournament.
Exit Condition	The student successfully joins the tournament.
Exceptions	The ETD for that tournament has expired. When the student tries to join the tournament an error message occurs. The MNS is already reached. When the student tries to join the tournament an error message occurs.

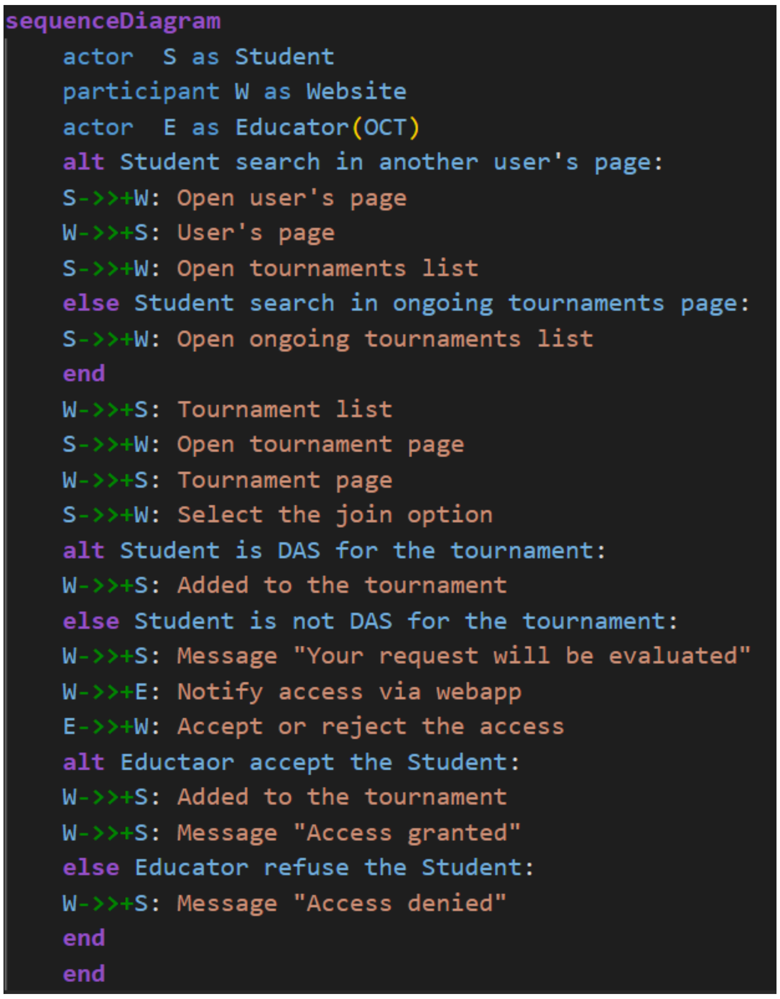


```
sequenceDiagram
    actor  S as Student
    participant W as Website
    S->>+W: Open notification page
    W->>+S: Notifications
    S->>+W: Select the new tournament notification
    S->>+W: Select the join option
    alt ETD is expired:
    W->>+S: Error message
    end
    alt MNS is reached:
    W->>+S: Error message
    end
    W->>+S: Added to the tournament
```

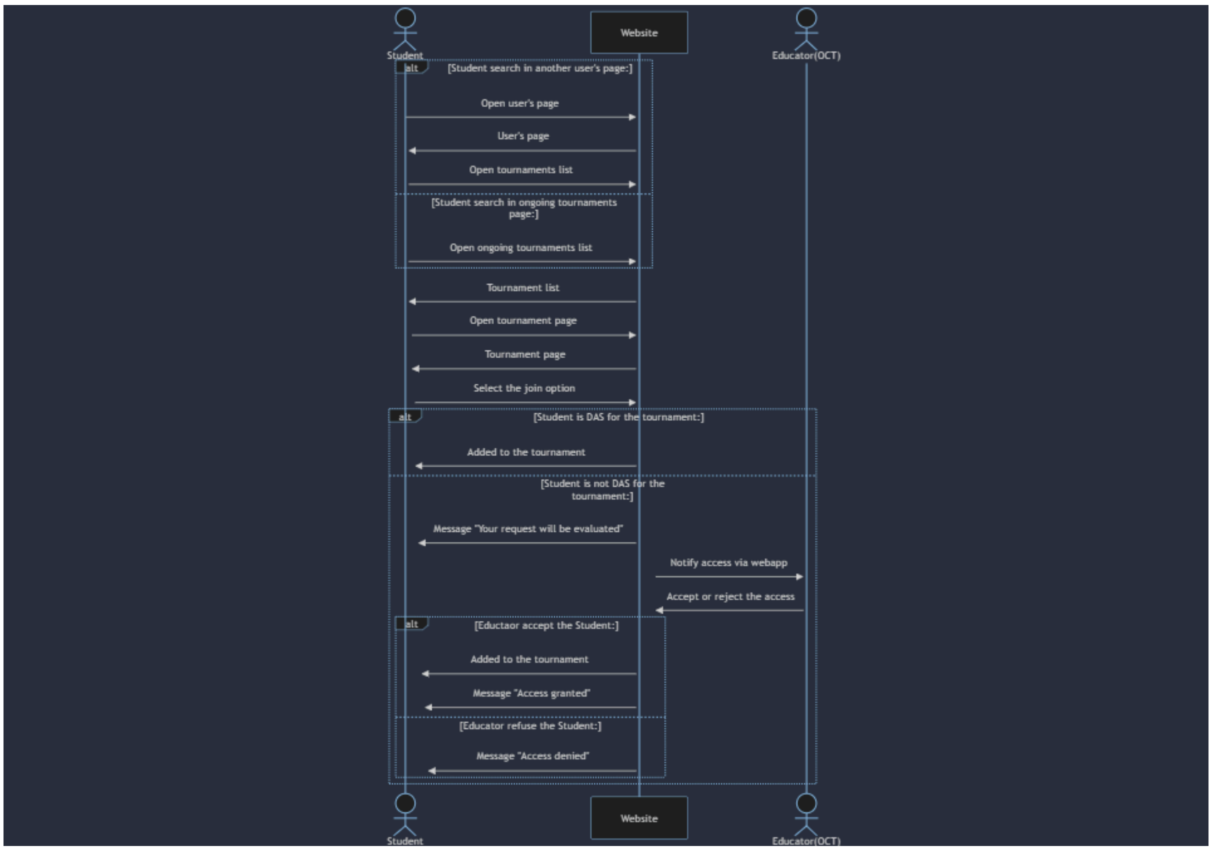


UC9. Join tournament via webapp

Name	Join tournament via webapp
Actor	Student, Educator (the OCT of the tournament)
Entry Condition	Student is registered and logged in on the website
Event Flow	1a. The student opens the tournament list of another user. 2. The student opens the tournament page of the tournament they want to join. 3. The student select the join option 4a. The student is DAS for the tournament, so they automatically joins.
Exit Condition	The user either successfully joins the tournament or is rejected by the OCT.
Exceptions	The ETD for that tournament has expired. When the student tries to join the tournament an error message occurs. The MNS is already reached. When the student tries to join the tournament an error message occurs.
Alternate Flows	1b. The student searches the tournament manually among the ongoing tournaments. 4c. The student is not DAS for the tournament, so they must be accepted by the OCT. 5c. The system informs the OCT that someone is trying to join their tournament directly. 6c. The OCT accepts the student. 7c. The student joins the tournament and is notified that their request has been accepted. 4d. The student is not DAS for the tournament, so they must be accepted by the OCT. 5d. The system informs the OCT that someone is trying to join their tournament directly. 6d. The OCT rejects the student. 7d. The student doesn't join the tournament and is notified that their request has been rejected.



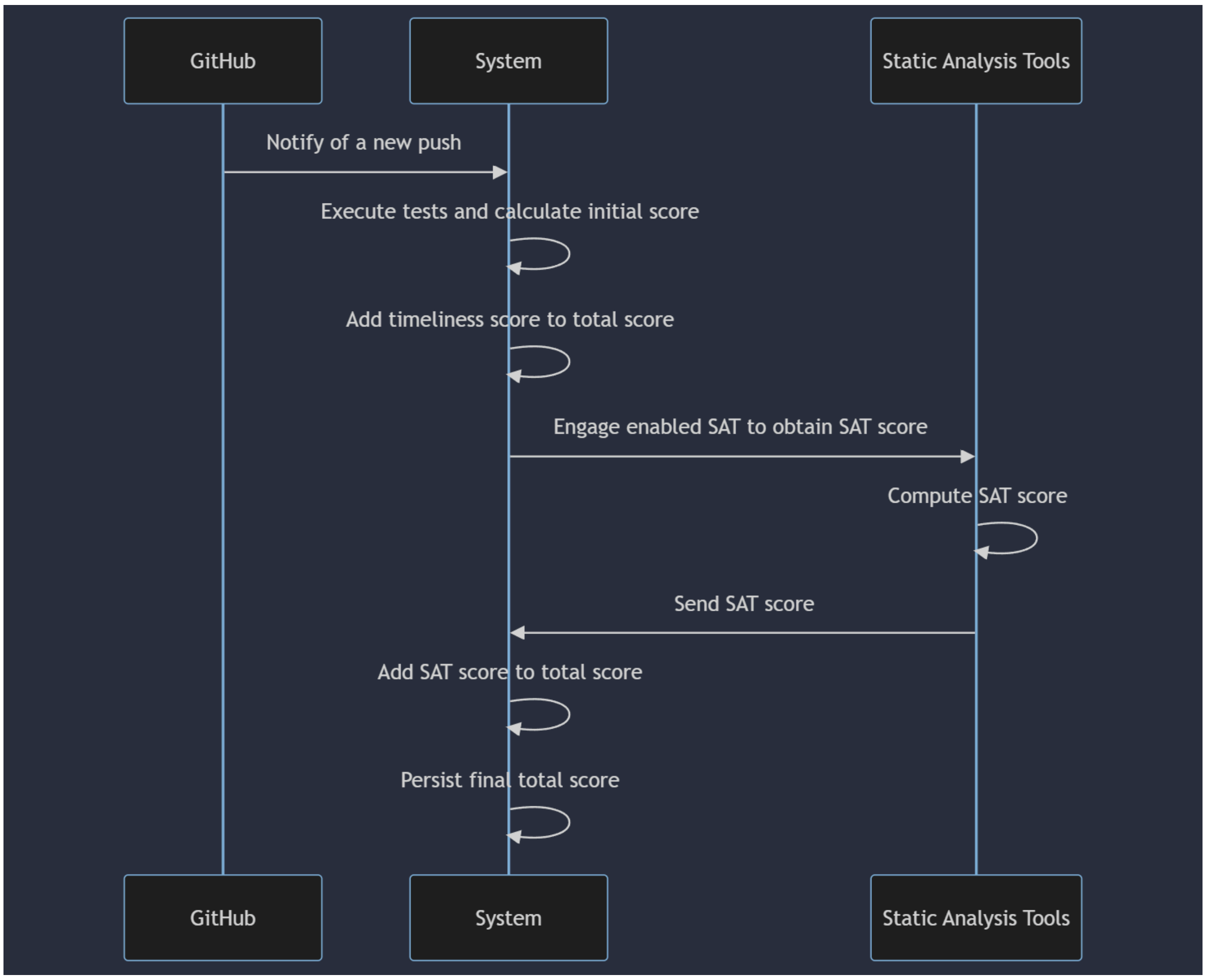
```
sequenceDiagram
    actor  S as Student
    participant W as Website
    actor  E as Educator(OCT)
    alt Student search in another user's page:
    S->>+W: Open user's page
    W->>+S: User's page
    S->>+W: Open tournaments list
    else Student search in ongoing tournaments page:
    S->>+W: Open ongoing tournaments list
    end
    W->>+S: Tournament list
    S->>+W: Open tournament page
    W->>+S: Tournament page
    S->>+W: Select the join option
    alt Student is DAS for the tournament:
    W->>+S: Added to the tournament
    else Student is not DAS for the tournament:
    W->>+S: Message "Your request will be evaluated"
    W->>+E: Notify access via webapp
    E->>+W: Accept or reject the access
    alt Eductaor accept the Student:
    W->>+S: Added to the tournament
    W->>+S: Message "Access granted"
    else Educator refuse the Student:
    W->>+S: Message "Access denied"
    end
    end
```


Student
Website
Educator(OCT)
alt
[Student search in another user's page:]
Open user's page
User's page
Open tournaments list
[Student search in ongoing tournaments page:]
Open ongoing tournaments list
Tournament list
Open tournament page
Tournament page
Select the join option
alt
[Student is DAS for the tournament:]
Added to the tournament
[Student is not DAS for the tournament:]
Message "Your request will be evaluated"
Notify access via webapp
Accept or reject the access
alt
[Eductaor accept the Student:]
Added to the tournament
Message "Access granted"
[Educator refuse the Student:]
Message "Access denied"
Student
Website
Educator(OCT)

UC10. Calculate total score

Name	Calculate total score
Actor	GitHub, System
Entry Condition	New commits are pushed to the main branch
Event Flow	1. GitHub Actions notifies the System of a new push, passing the latest version of the code with battle id. 2. System starts to compute the score by considering: - Number of test cases that pass out of all the test cases (the higher the better) - Timeliness score measured in terms of time passed between the registration deadline and the last commit (the lower the better) 3. System engages enabled SAT for that specific battle to obtain SAT score. 4. SAT score is added to the total score. The total score is persisted.
Exit Condition	Total score is calculated and persisted correctly.
Exceptions	1. System is not available to receive the notification from GitHub. 2. System errors out while executing tests.

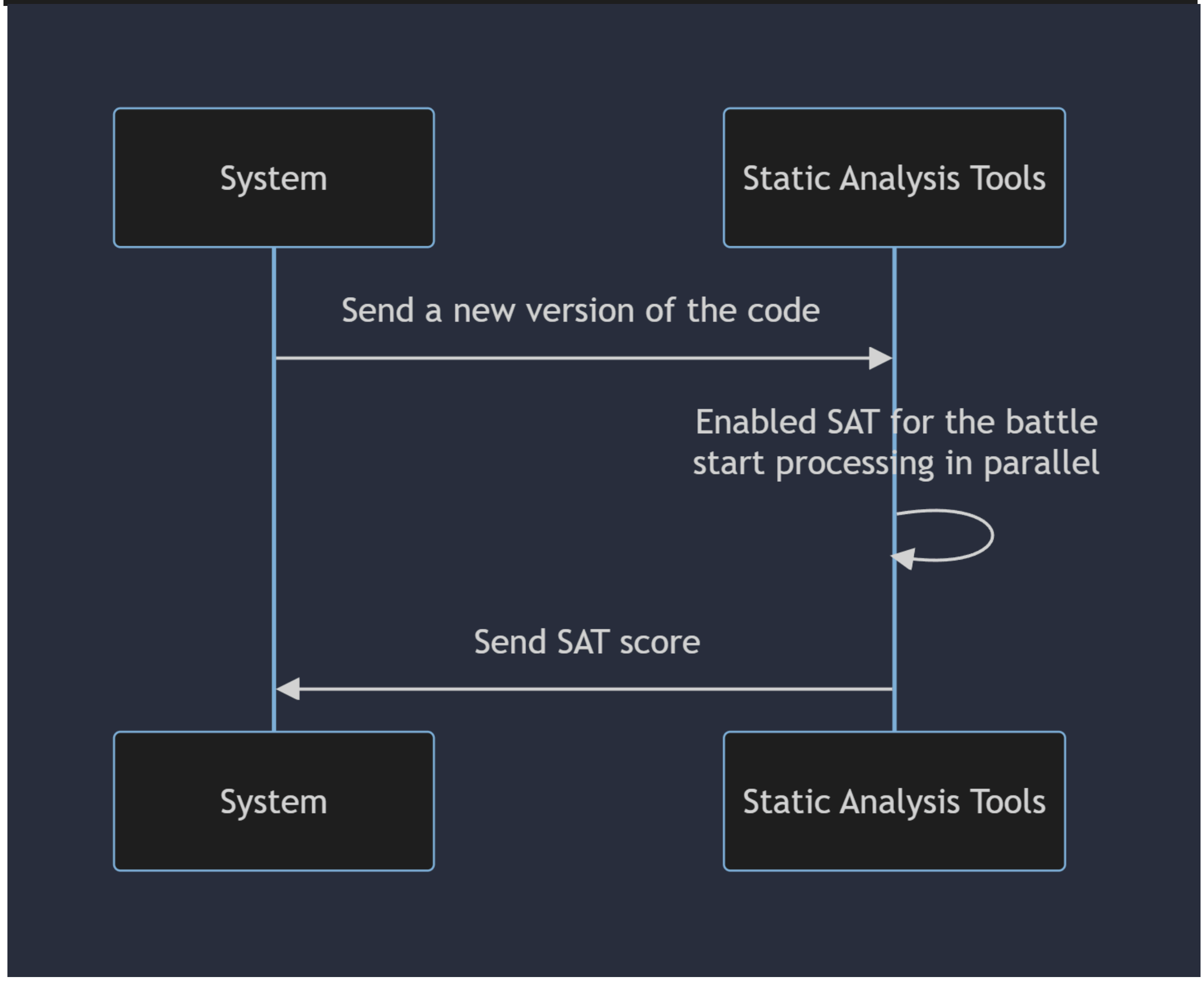
```
sequenceDiagram
    participant GH as GitHub
    participant S as System
    participant SAT as Static Analysis Tools
    GH->>S: Notify of a new push
    S->>S: Execute tests and calculate initial score
    S->>S: Add timeliness score to total score
    S->>SAT: Engage enabled SAT to obtain SAT score
    SAT->>SAT: Compute SAT score
    SAT->>S: Send SAT score
    S->>S: Add SAT score to total score
    S->>S: Persist final total score
```


GitHub
System
Static Analysis Tools
Notify of a new push
Execute tests and calculate initial score
Add timeliness score to total score
Engage enabled SAT to obtain SAT score
Compute SAT score
Send SAT score
Add SAT score to total score
Persist final total score
GitHub
System
Static Analysis Tools

UC11. Calculate SAT score

Name	Calculate SAT score
Actor	Static Analysis Tools, System
Entry Condition	System engages SAT
Event Flow	1. The SAT receive a new version of the code from System. 2. The SAT enabled for that battle are engaged and start the analysis process in parallel. 3. SAT provide their scores to the System.
Exit Condition	All of the engaged SAT completed the analysis process.
Exceptions	One or more of the engaged SAT errors out during the analysis process. In this case, only the SAT which completed the process are considered in score calculation.

```
sequenceDiagram
    participant S as System
    participant SAT as Static Analysis Tools
    S->>SAT: Send a new version of the code
       SAT->>SAT:  Enabled  SAT  for  the  battle<br>start  processing  in
parallel
    SAT->>S: Send SAT score
```

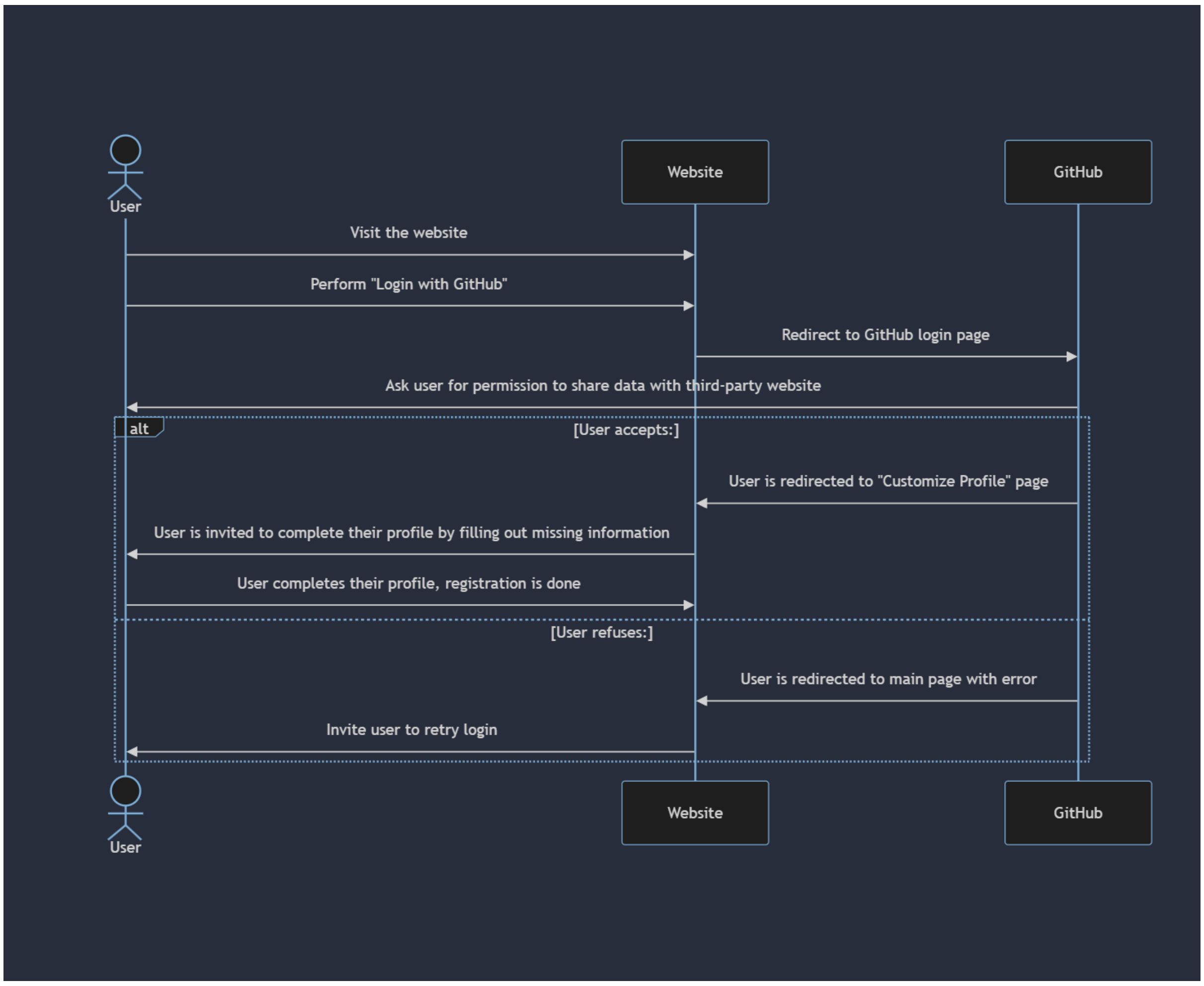

System
Static Analysis Tools
Send a new version of the code
Enabled SAT for the battle
start processing in parallel
Send SAT score
System
Static Analysis Tools

UC12. Registration

Name	Registration
Actor	User, GitHub
Entry Condition	User wants to register to the application.
Event Flow	1. User visits the website through a browser. 2. User performs “Login with GitHub” through the main page. 3. GitHub asks the user for permission to share their personal data. 4. User accepts, and it is redirected to the website, to the Customize Profile page.

	5. User is invited to set their privacy options, and to complete their profile by adding missing information (such as first name and last name).
Exit Condition	User has completed the registration flow correctly.
Exceptions	1. User interrupts the flow by closing the browser 2. User refuses to give permission to register with GitHub. User is redirected to the web-app, registration is aborted. 3. GitHub service is unavailable.

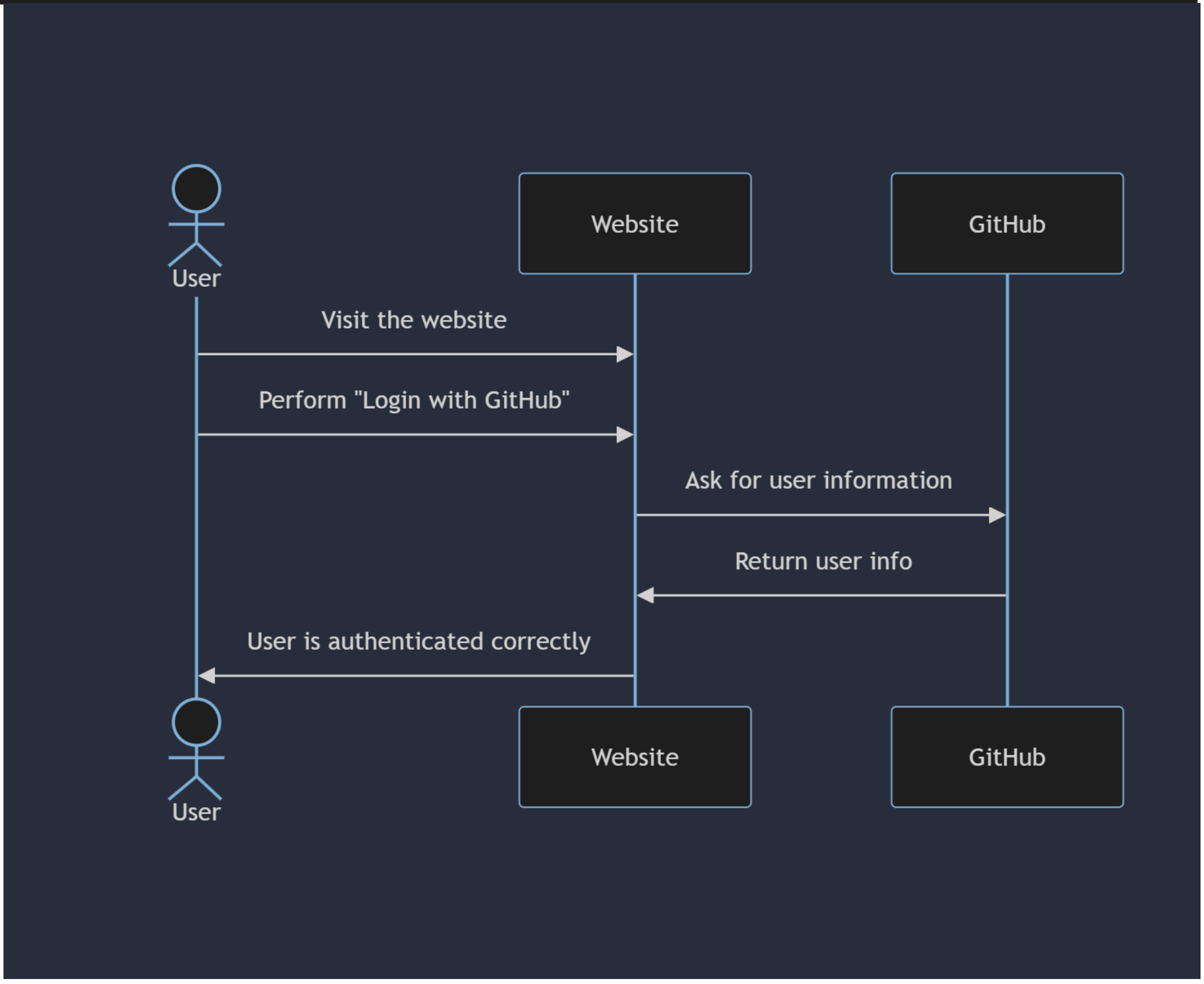
```
sequenceDiagram
    actor U as User
    participant W as Website
    participant GH as GitHub
    U->>W: Visit the website
    U->>W: Perform "Login with GitHub"
    W->>GH: Redirect to GitHub login page
      GH->>U: Ask user for permission to share data with third-party
website
    alt User accepts:
    GH->>W: User is redirected to "Customize Profile" page
     W->>U: User is invited to complete their profile by filling out
missing information
    U->>W: User completes their profile, registration is done
    else User refuses:
    GH->>W: User is redirected to main page with error
    W->>U: Invite user to retry login
    end
```


User
Website
GitHub
Visit the website
Perform "Login with GitHub"
Redirect to GitHub login page
Ask user for permission to share data with third-party website
alt
[User accepts:]
User is redirected to "Customize Profile" page
User is invited to complete their profile by filling out missing information
User completes their profile, registration is done
[User refuses:]
User is redirected to main page with error
Invite user to retry login
User
Website
GitHub

UC13. Login

Name	Login
Actor	User, GitHub
Entry Condition	User is already registered and its access token is missing or has expired.
Event Flow	1. User visits the website through a browser. 2. User clicks on the “Login with GitHub button”. 3. User is redirected to the website, login is successful.
Exit Condition	User is logged in correctly.
Exceptions	1. User interrupts the flow by closing the browser. 2. GitHub service is unavailable.

```
sequenceDiagram
    actor U as User
    participant W as Website
    participant GH as GitHub
    U->>W: Visit the website
    U->>W: Perform "Login with GitHub"
    W->>GH: Ask for user information
    GH->>W: Return user info
    W->>U: User is authenticated correctly
```

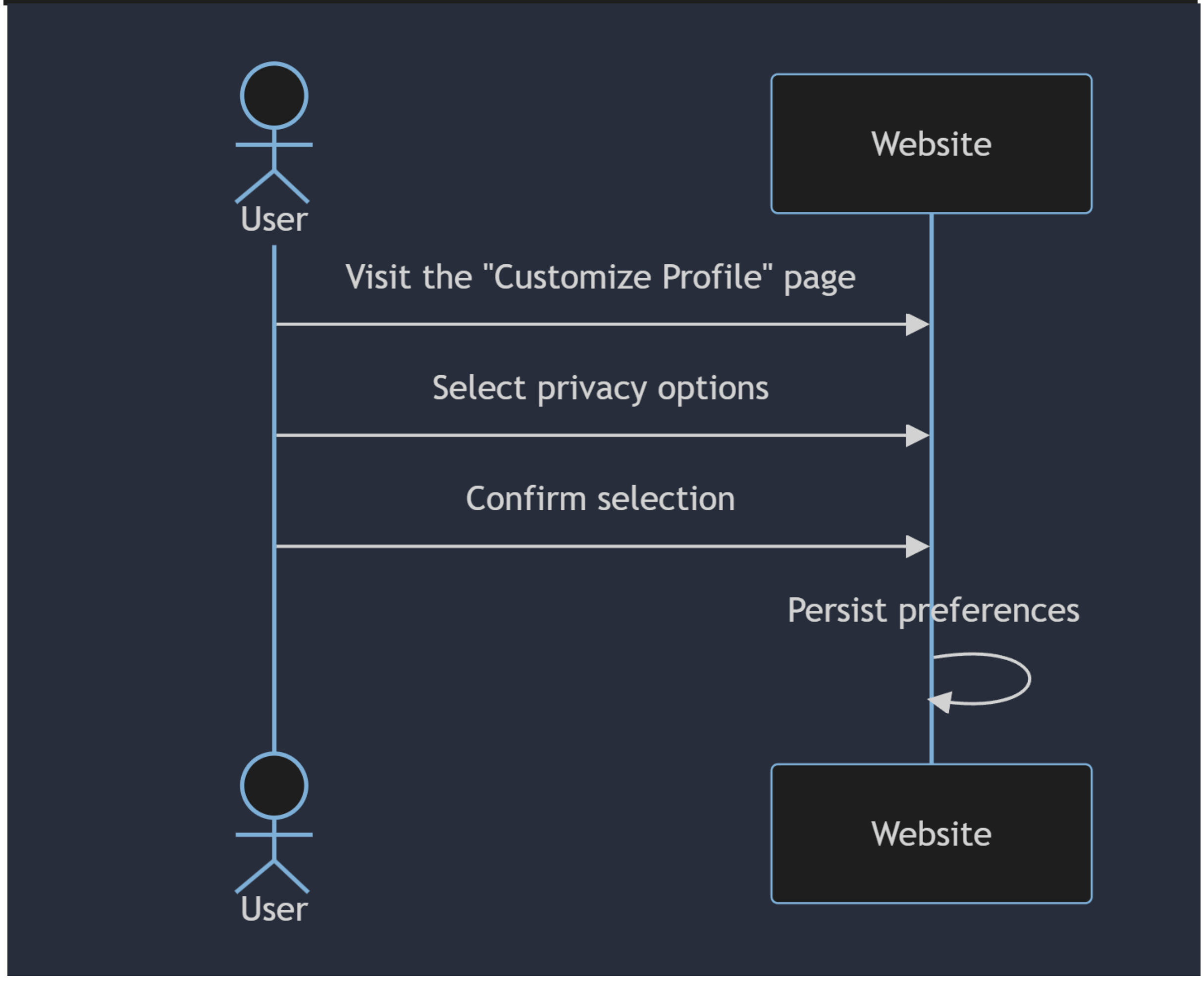


UC14. Set privacy options

Name	Set privacy options
Actor	User
Entry Condition	User is registered and logged in on the website.

Event Flow	1. User visits the Customize Profile page. 2. User selects from which subset of users to receive notifications: Everyone, Friends Only, No one. 3. User confirms their selection by clicking the Save button.
Exit Condition	User has correctly selected an option, and their selection is correctly persisted.

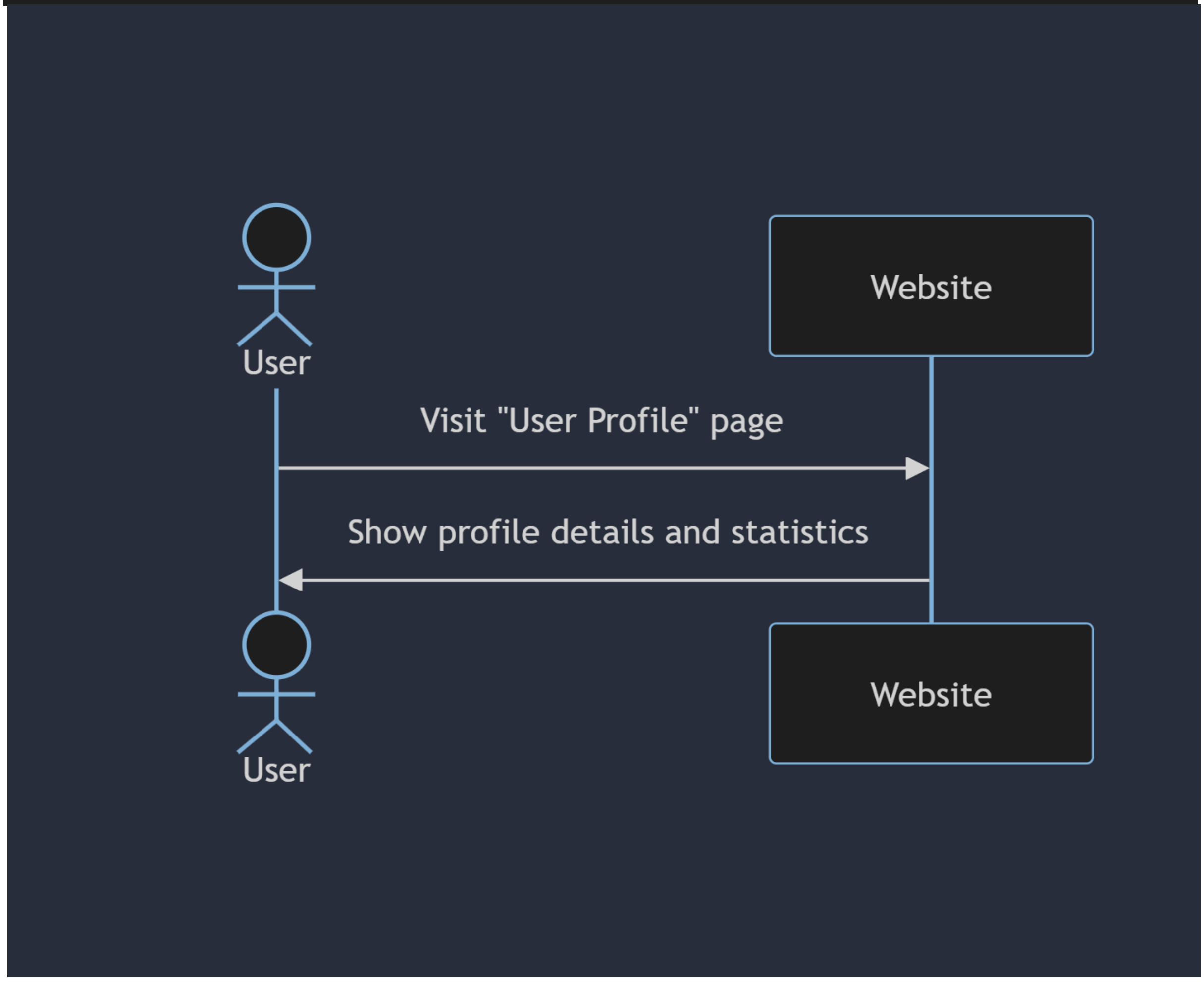
```
sequenceDiagram
    actor U as User
    participant W as Website
    U->>W: Visit the "Customize Profile" page
    U->>W: Select privacy options
    U->>W: Confirm selection
    W->>W: Persist preferences
```



UC15. Visualize user profile

Name	Visualize user profile
Actor	User
Entry Condition	User is registered and logged in on the website.
Event Flow	1. User visits their own Profile page or a Profile page of another user. 2. User can see profile details and some statistics such as: number of tournaments in which they participated, preferred programming language, badges obtained by the user etc.
Exit Condition	User has correctly landed on the Profile page.

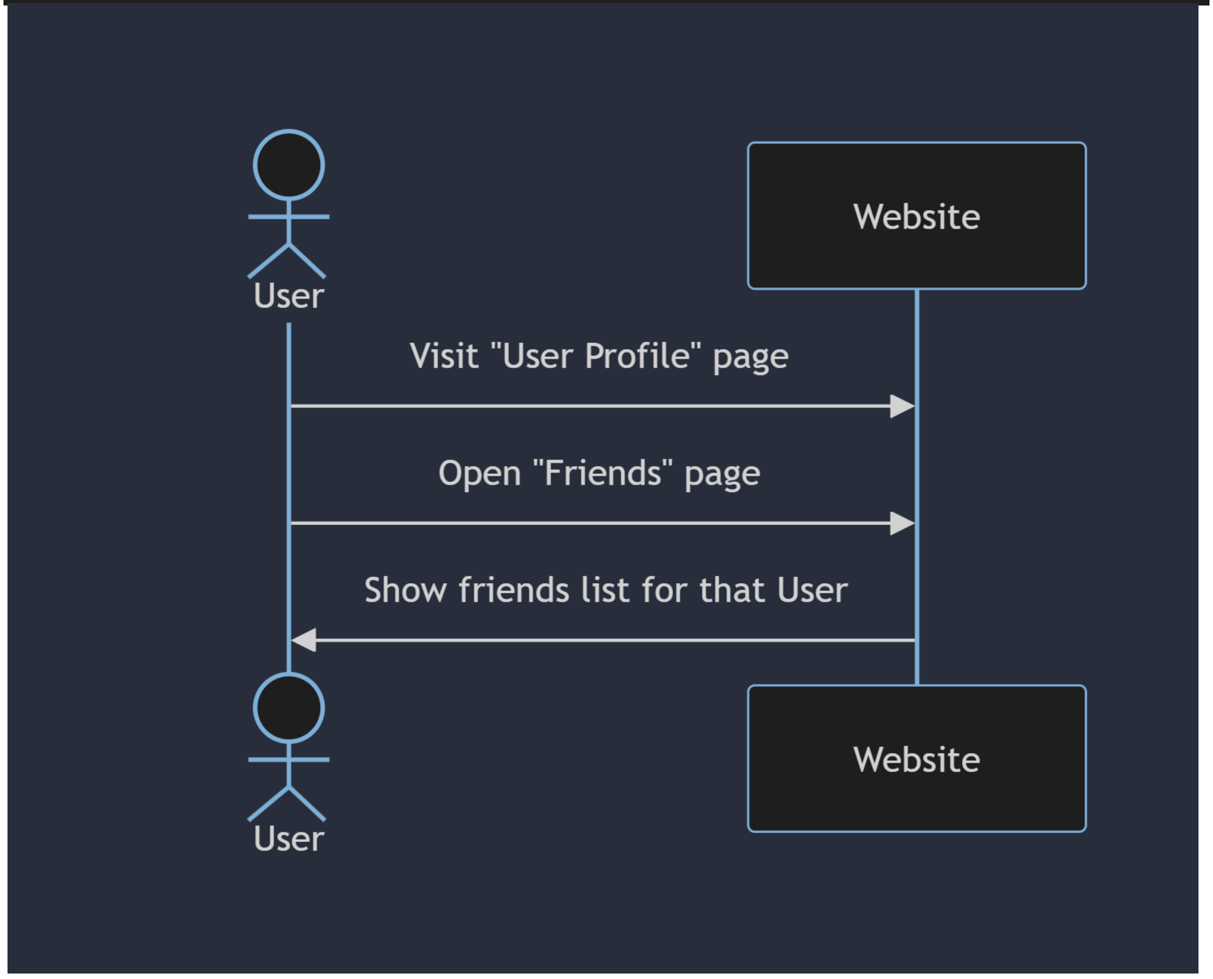
```
sequenceDiagram
    actor U as User
    participant W as Website
    U->>W: Visit "User Profile" page
    W->>U: Show profile details and statistics
```


User
Website
Visit "User Profile" page
Show profile details and statistics
User
Website

UC16. Visualize friends list

Name	Visualize friends list
Actor	User
Entry Condition	User is registered and logged in on the website.
Event Flow	1. User visits their own Profile page or a Profile page of another user. 2. User opens the “Friends” page on the Profile. 3. User can see the list of friends of that user.

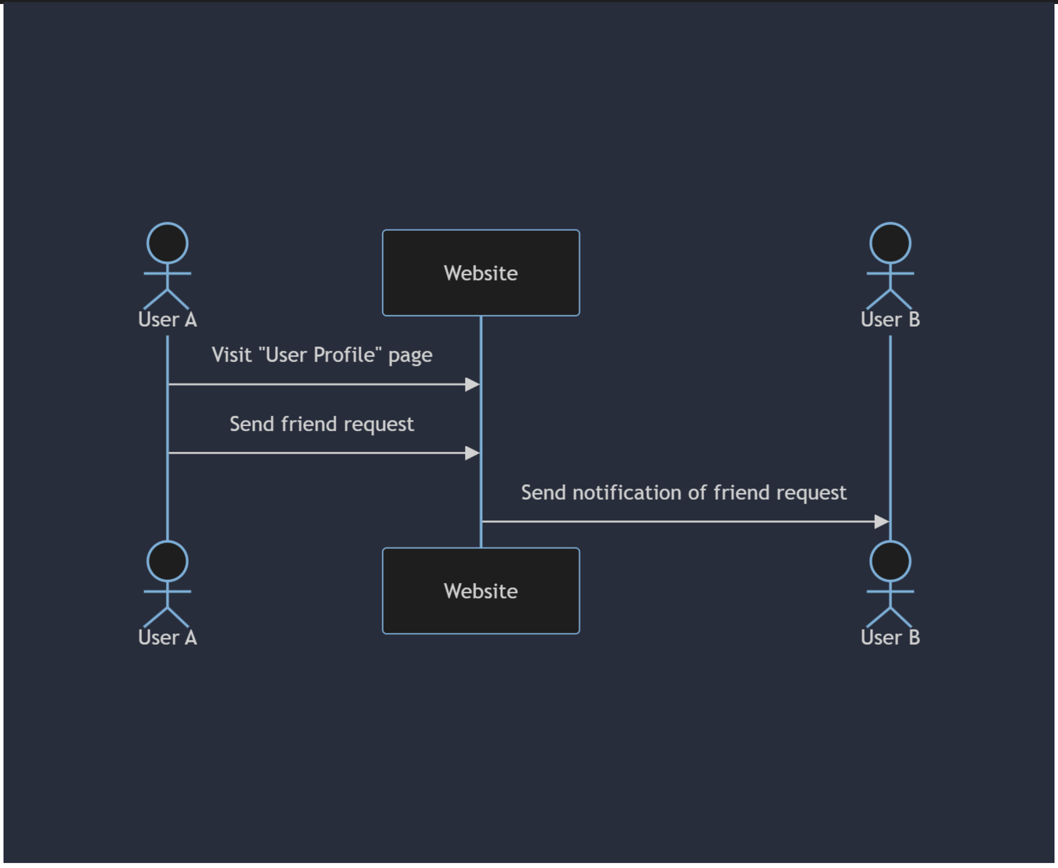
```
sequenceDiagram
    actor U as User
    participant W as Website
    U->>W: Visit "User Profile" page
    U->>W: Open "Friends" page
    W->>U: Show friends list for that User
```



UC17. Send friend request

Name	Send friend request
Actor	User
Entry Condition	User is visualizing a User Profile page.
Event Flow	1. User visualizes the Profile page of the user which they are interested in. 2. User sends them a friend request, if they are not friends already. 3. System sends a notification to the invited User.
Exit Condition	Friend request is sent successfully

```
sequenceDiagram
    actor UA as User A
    participant W as Website
    actor UB as User B
    UA->>W: Visit "User Profile" page
    UA->>W: Send friend request
    W->>UB: Send notification of friend request
```

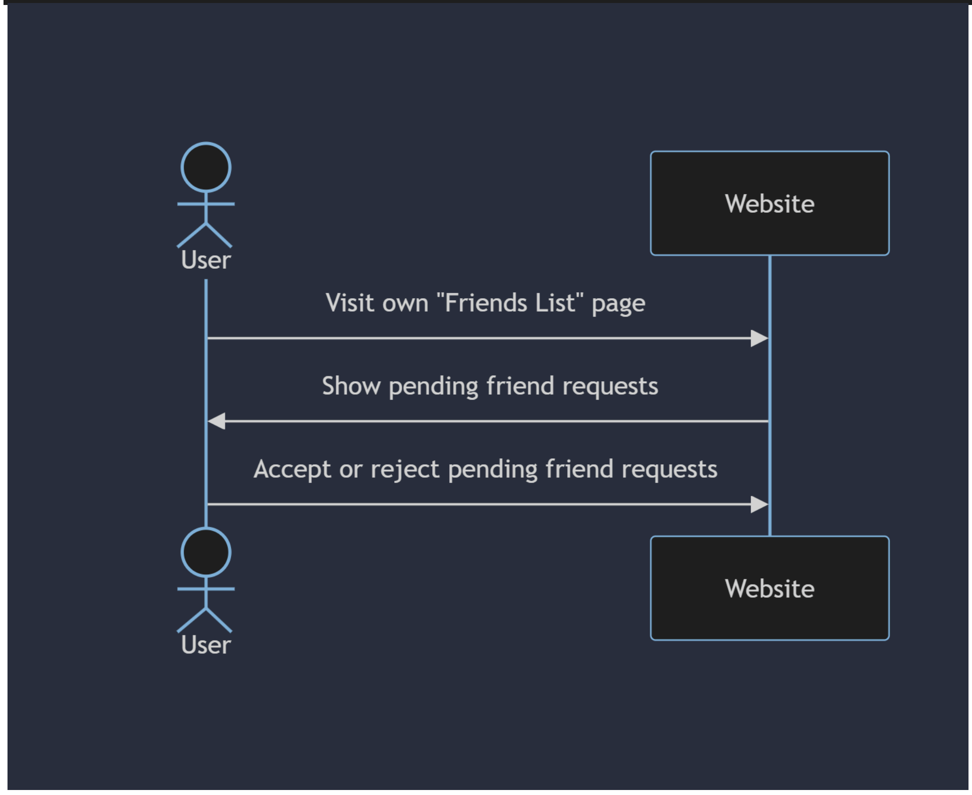


User A
Website
User B
Visit "User Profile" page
Send friend request
Send notification of friend request
User A
Website
User B

UC18. Accept or reject friend request

Name	Accept or reject friend request
Actor	User
Entry Condition	1. User is registered and logged in on the website. 2. User has 1 or more pending friend requests.
Event Flow	1. User goes to list of pending friend requests 2. User accepts or rejects pending friend requests.
Exit Condition	User does not see the pending friend request anymore.

```
sequenceDiagram
    actor U as User
    participant W as Website
    U->>W: Visit own "Friends List" page
    W->>U: Show pending friend requests
    U->>W: Accept or reject pending friend requests
```


User
Website
Visit own "Friends List" page
Show pending friend requests
Accept or reject pending friend requests
User
Website

UC19. Create tournament

Name	Create tournament
Actor	Educator, Student
Entry Condition	Educator is registered and logged in on the website.
Event Flow	1. Educator opens the Create Tournament section. 2. Educator fills the form with information related to the tournament, such as: name, deadlines, policies, etc. 3. (Optional) Educator performs “Adding TC”. 4. (Optional) Educator performs “Set Badge”. 5. Educator confirms the Tournament's creation, and they become the OCT. 6. IUs receive the notification about the creation of the Tournament. 7. OCT sees the Tournament page.
Exit Condition	Tournament is created correctly.
Exceptions	Educator aborts the operation because they have changed their mind.

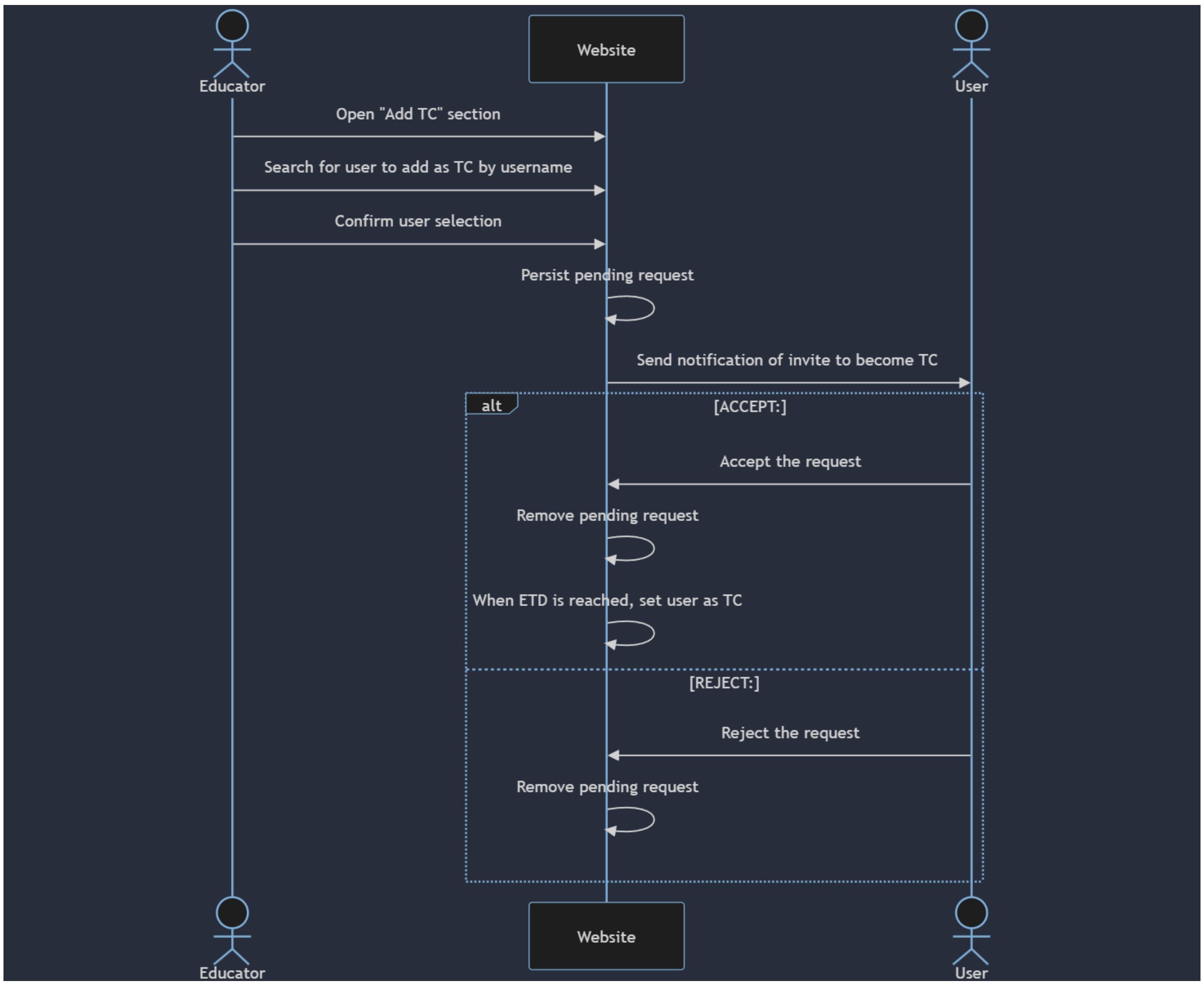
```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor S as Student
    E->>W: Open "Create Tournament" page
    E->>W: Fill form with information related<br>to the tournament
    E->>W: [OPTIONAL] Perform "Adding TC"
    E->>W: [OPTIONAL] Perform "Set Badge"
    E->>W: Confirm Tournament's creation became the OCT
    W->>S: Send "New Tournament" notification to IUs
    W->>E: Show "Tournament Details" page for newly-created tournament
```

UC20. Adding TC

Name	Adding TC
Actor	Educator
Entry Condition	Educator is performing “Create Tournament”.

Event Flow	1. Educator is creating a Tournament. 2. Educator opens the “Add TC” section. 3. Educator searches for the User “U” which they want to add as TC. 4. U receives the notification to become a Coordinator for the Tournament. 5. U accepts the request (before the ETD is reached) and becomes a Coordinator of the Tournament. 6. Educator can perform step 3 as many times as they want. 7. When the ETD is reached, the System assigns all invited Educators (which accepted the request) as TCs, and does not allow any new User to become TC.
Alternative Flow	In Step 3, the Educator can pick the User “U” from their friends list. In Step 5, the invited User “U” can reject the request. They won’t become TC after the ETD.
Exit Condition	The TCs are added correctly in the Tournament after ETD.
Exceptions	1. A user that the creator wants to add as TC doesn't receive the notification due to their notifications policy, so they can’t accept the request. 2. Educator aborts the operation because they have changed their mind.

```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor U as User
    E->>W: Open "Add TC" section
    E->>W: Search for user to add as TC by username
    E->>W: Confirm user selection
    W->>W: Persist pending request
    W->>U: Send notification of invite to become TC
    alt ACCEPT:
    U->>W: Accept the request
    W->>W: Remove pending request
    W->>W: When ETD is reached, set user as TC
    else REJECT:
    U->>W: Reject the request
    W->>W: Remove pending request
    end
```

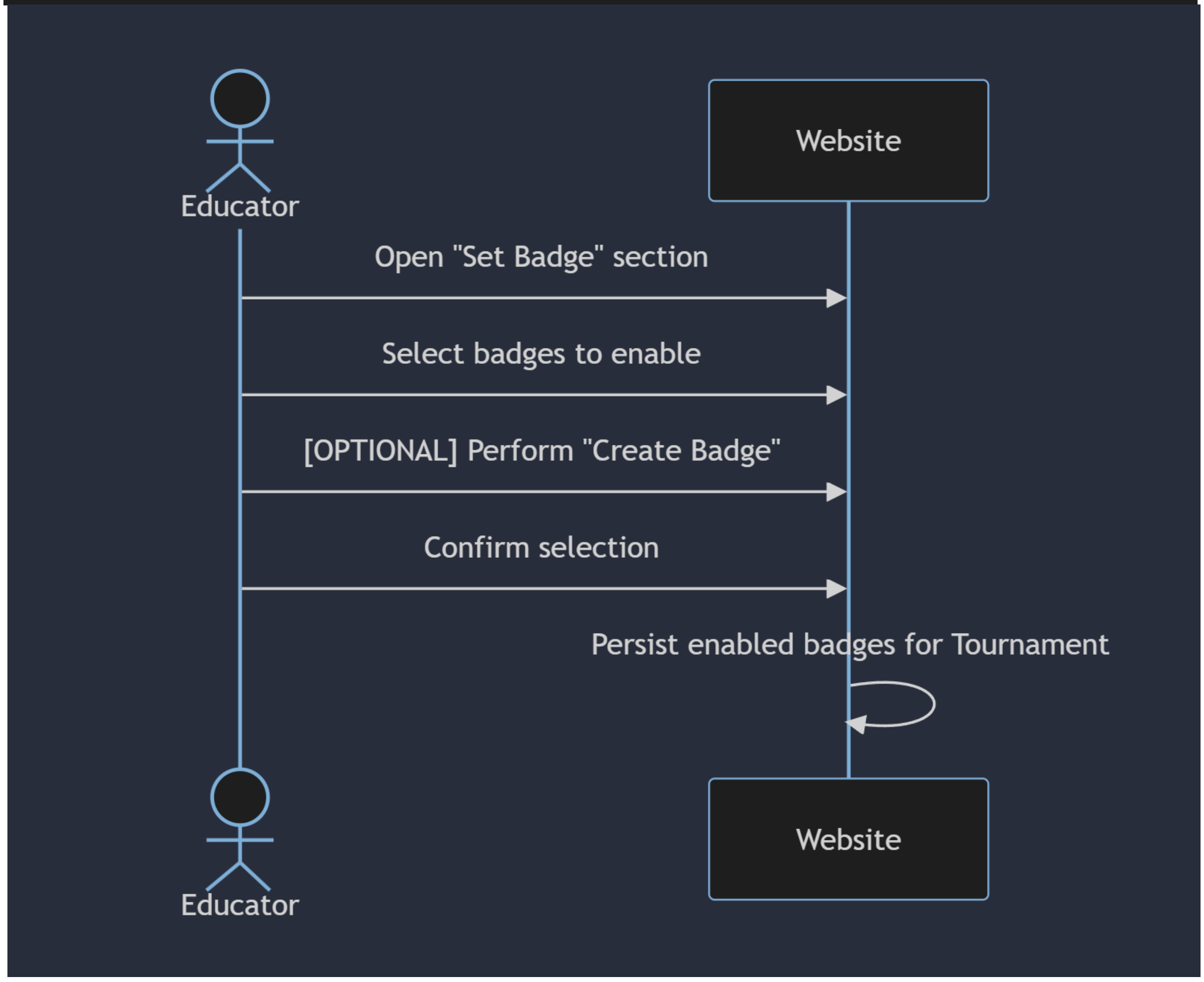


UC21. Set Badge

Name	Set Badge
Actor	Educator
Entry Condition	Educator is performing “Create Tournament”.
Event Flow	1. Educator is creating a Tournament. 2. Educator opens the “Set Badge” section. 3. Educator selects which badges to enable for the Tournament from the available list. The list includes default badges and custom badges (previously created, if present). 4. (Optional) Educator performs “Create Badge” as many times as they want, creating and adding custom badges to the Tournament. 5. Educator confirms the selected badges and closes the section.
Exit Condition	Badges are selected correctly for the Tournament.

Exceptions	Educator aborts the operation because they have changed their mind.

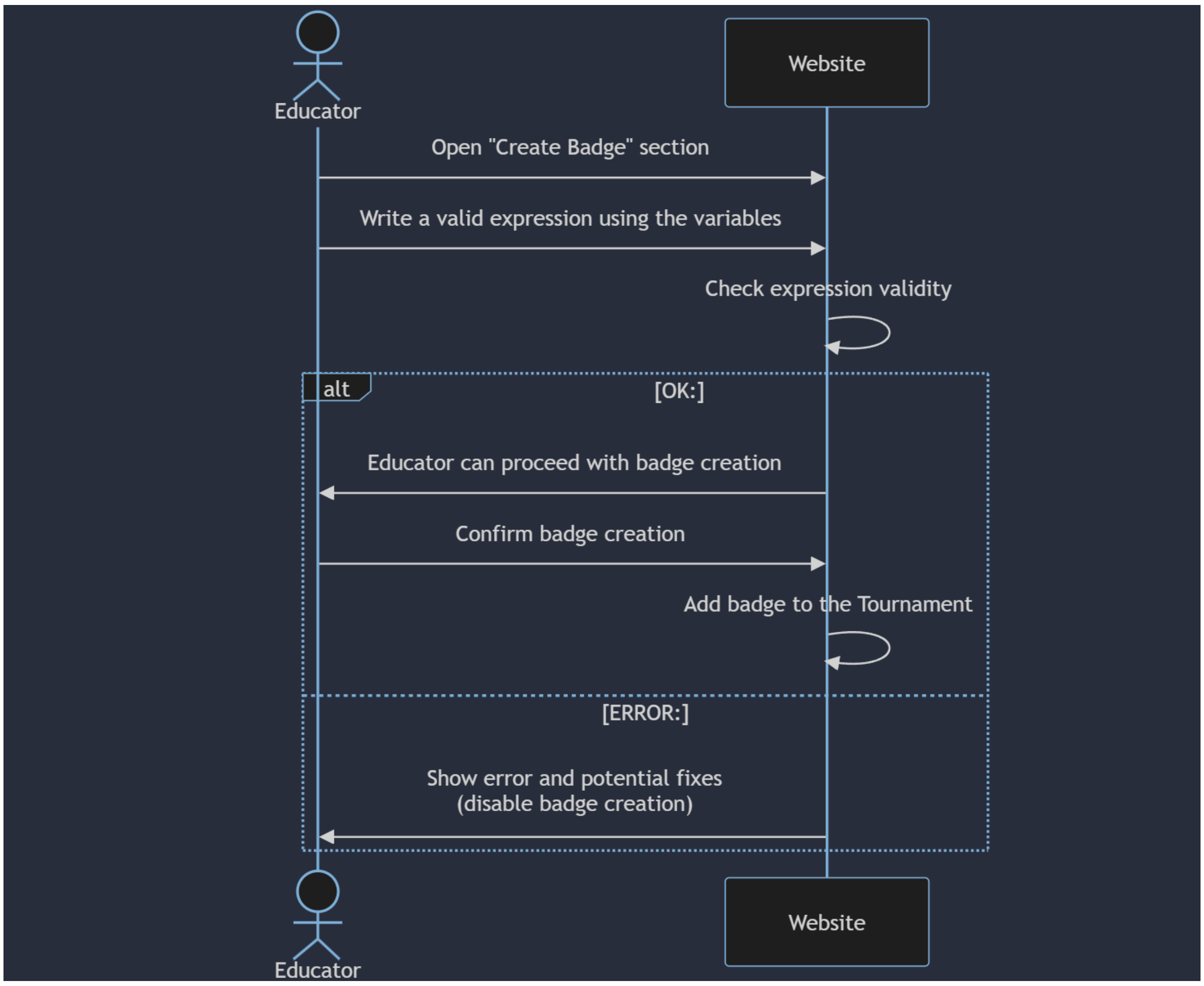
```
sequenceDiagram
    actor E as Educator
    participant W as Website
    E->>W: Open "Set Badge" section
    E->>W: Select badges to enable
    E->>W: [OPTIONAL] Perform "Create Badge"
    E->>W: Confirm selection
    W->>W: Persist enabled badges for Tournament
```



UC22. Create Badge

Name	Create Badge
Actor	Educator
Entry Condition	Educator is performing “Create Tournament”, and is in the “Set Badge” section.
Event Flow	1. Educator opens “Create Badge” section. 2. Educator writes a valid expression using the provided variables to determine Badge conditions. 3. Educator can chain multiple expressions together. 4. Educator confirms the expressions and creates the Badge.
Exit Condition	The Badge is created correctly and added to the Tournament.
Exceptions	1. Expression is not valid, the Educator is informed about the error and potential fixes.<br> 2. Educator aborts the operation because they have changed their mind.

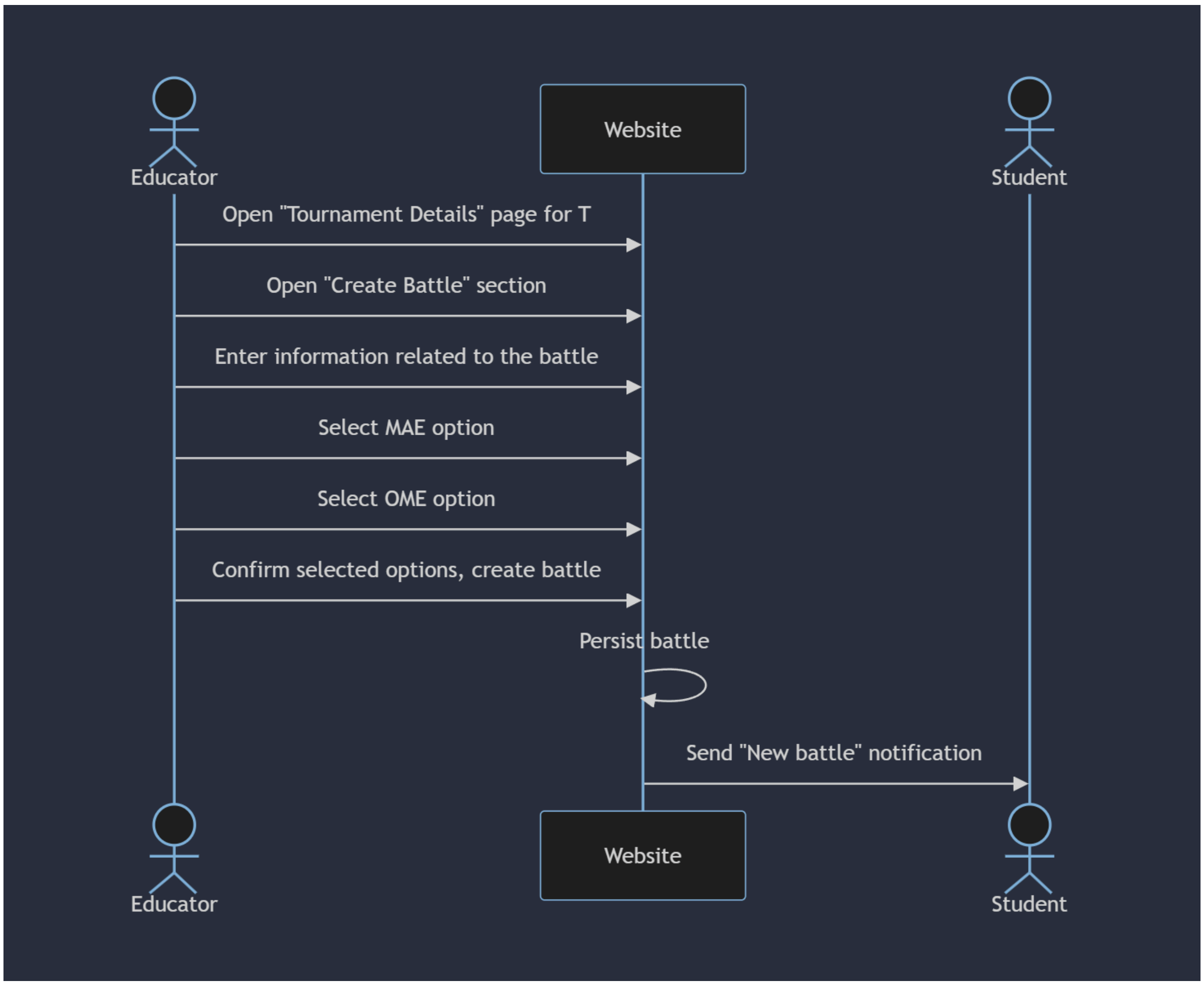
```
sequenceDiagram
    actor E as Educator
    participant W as Website
    E->>W: Open "Create Badge" section
    E->>W: Write a valid expression using the variables
    W->>W: Check expression validity
    alt OK:
    W->>E: Educator can proceed with badge creation
    E->>W: Confirm badge creation
    W->>W: Add badge to the Tournament
    else ERROR:
    W->> E: Show error and potential fixes<br>(disable badge creation)
    end
```


Educator
Website
Open "Create Badge" section
Write a valid expression using the variables
Check expression validity
alt
[OK:]
Educator can proceed with badge creation
Confirm badge creation
Add badge to the Tournament
[ERROR:]
Show error and potential fixes
(disable badge creation)
Educator
Website

UC23. Create Battle

Name	Create Battle
Actor	Educator, Student
Entry Condition	Educator E is OCT or TC of Tournament T.
Event Flow	1. E opens the “Tournament Details” page for T. 2. E opens the “Create Battle” section. 3. E enters the information related to the battle such as: name, max number of students per team, deadline, katas. 4. E selects MAE option. 5. E selects OME option. 6. E confirms selected options and creates the battle. 7. IUs receive “New Battle” notification.
Exit Condition	The battle is created correctly.
Exceptions	1. Tournament is closed, as such it is not possible to create new battles. 2. E aborts the operation because they have changed their mind.

```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor S as Student
    E->>W: Open "Tournament Details" page for T
    E->>W: Open "Create Battle" section
    E->>W: Enter information related to the battle
    E->>W: Select MAE option
    E->>W: Select OME option
    E->>W: Confirm selected options, create battle
    W->>W: Persist battle
    W->>S: Send "New battle" notification
```

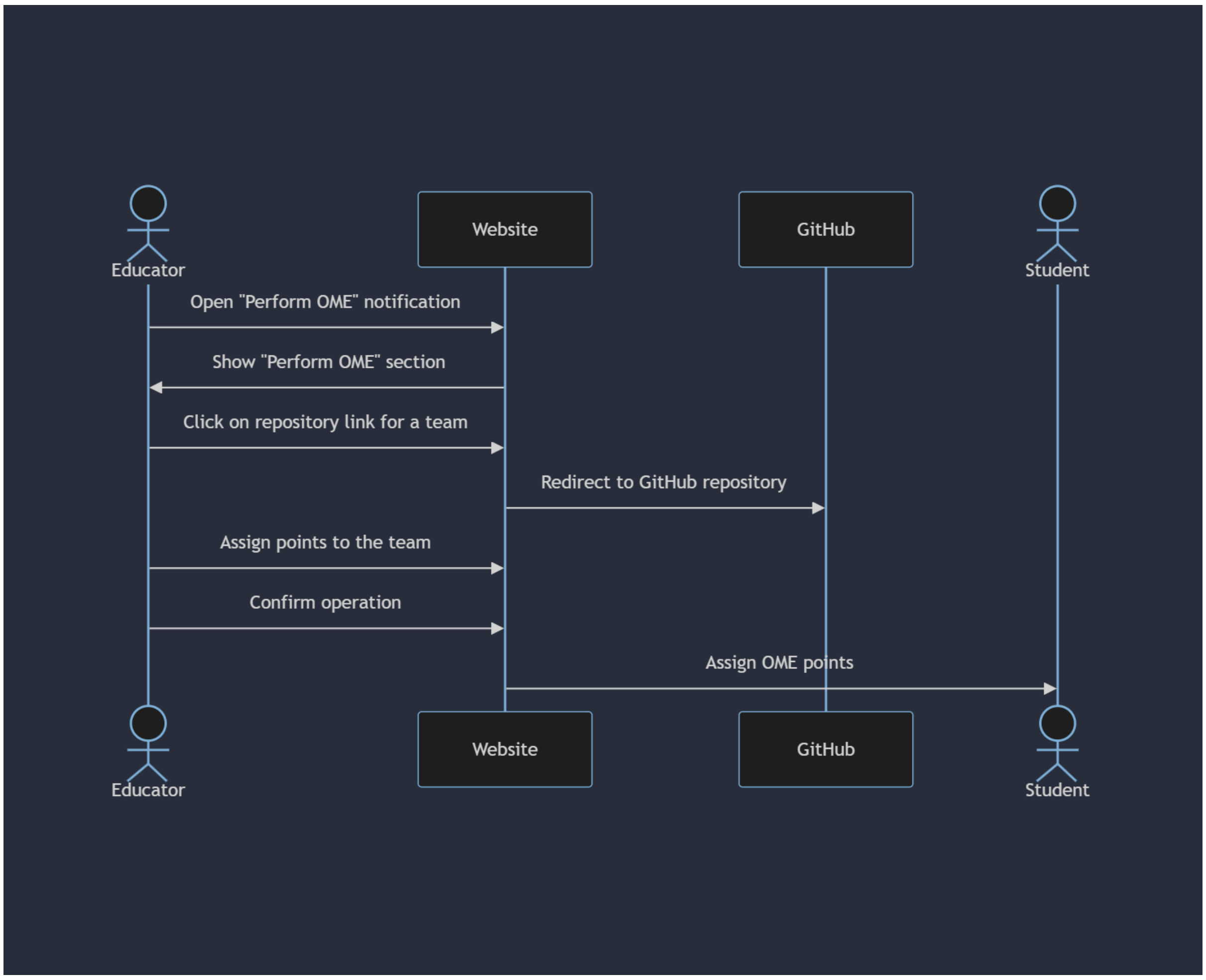

Educator
Website
Student
Open "Tournament Details" page for T
Open "Create Battle" section
Enter information related to the battle
Select MAE option
Select OME option
Confirm selected options, create battle
Persist battle
Send "New battle" notification
Educator
Website
Student

UC24. Perform OME

Name	Perform OME
Actor	Educator, Student, GitHub
Entry Condition	An Educator E is the creator of the Battle, and receives the notification to perform manual evaluation.

Event Flow	1. E clicks on the notification. 2. The “Perform OME” section is showed. 3. E sees a list of all Teams in the Battle and their GitHub repository’s link. 4. E click the repository link of a Team to see their code submission. 5. E assigns the points at the Team. 6. E can perform steps 4 and 5 for each Team in the battle. 7. E confirms the operation. 8. System adds OME points to every Student.
Exit Condition	The OME is done for the battle, and the final leaderboard is correctly updated.
Exceptions	1. The FTD is expired, so the OME can’t be done. OME score is considered to be 0. 2. E aborts the operation because they have changed their mind.

```
sequenceDiagram
    actor E as Educator
    participant W as Website
    participant GH as GitHub
    actor S as Student
    E->>W: Open "Perform OME" notification
    W->>E: Show "Perform OME" section
    E->>W: Click on repository link for a team
    W->>GH: Redirect to GitHub repository
    E->>W: Assign points to the team
    E->>W: Confirm operation
    W->>S: Assign OME points
```

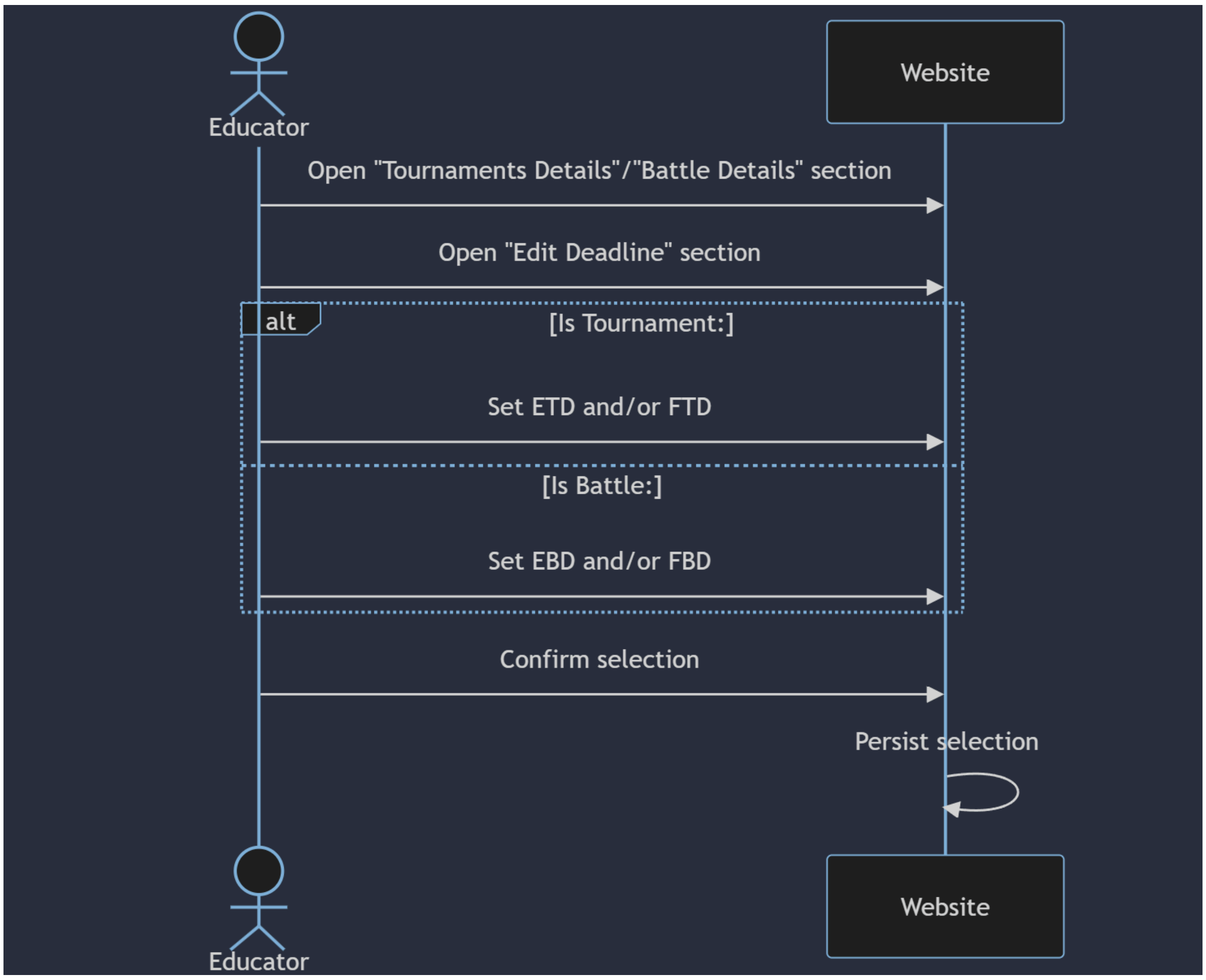

Educator
Website
GitHub
Student
Open "Perform OME" notification
Show "Perform OME" section
Click on repository link for a team
Redirect to GitHub repository
Assign points to the team
Confirm operation
Assign OME points
Educator
Website
GitHub
Student

UC25. Edit Deadline

Name	Edit Deadline
Actor	Educator
Entry Condition	An Educator E is the creator of a Tournament or Battle.
Event Flow	1. E is in the “Tournament Details”/”Battle Details” section. 2. E opens the “Edit Deadline” section. 2a. If E is in “Battle Details”, E can edit EBD and/or FBD. 2b. If E is in “Tournament Details”, E can edit ETD and/or FTD. 3. E confirms the selection. 4.All Students involved receive the notification. 4a. If E edit Battle Deadline Students involved are the Students subscribed in the Battle

	4b.If E edit Tournament Deadline Students involved are the Students subscribed in the Tournament
Exit Condition	The Deadline is updated correctly.
Exceptions	1. The Educator violates this condition: CD<=ETD<=EBD<FBD<FTD. The System disallows them to confirm options. 2. E aborts the operation because they have changed their mind.

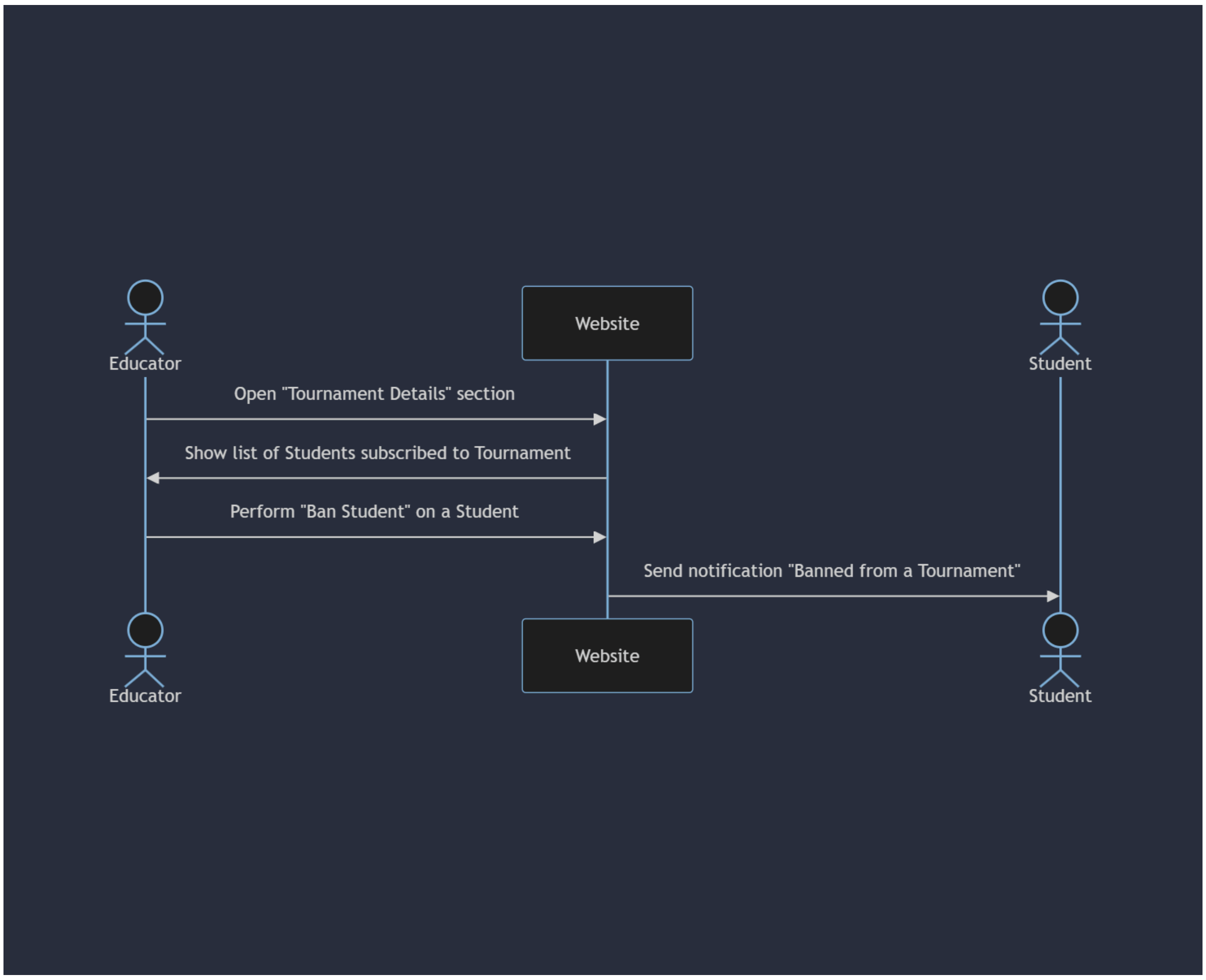
```
sequenceDiagram
    actor E as Educator
    participant W as Website
    E->>W: Open "Tournaments Details"/"Battle Details" section
    E->>W: Open "Edit Deadline" section
    alt Is Tournament:
    E->>W: Set ETD and/or FTD
    else Is Battle:
    E->>W: Set EBD and/or FBD
    end
    E->>W: Confirm selection
    W->>W: Persist selection
```



UC26. Ban student

Name	Ban Student
Actor	Educator, Student
Entry Condition	An Educator E is the creator of a Tournament.
Event Flow	1. E is in the “Tournament Details” section. 2. E opens the list of Students subscribed to the Tournament. 3. E selects a Student and performs the “Ban Student” function. 4. Banned Student receives a notification informing them of the ban.
Exit Condition	Banned Student can not access the Tournament anymore and submit code in katas.
Exceptions	E aborts the operation because they have changed their mind.

```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor S as Student
    E->>W: Open "Tournament Details" section
    W->>E: Show list of Students subscribed to Tournament
    E->>W: Perform "Ban Student" on a Student
    W->>S: Send notification "Banned from a Tournament"
```

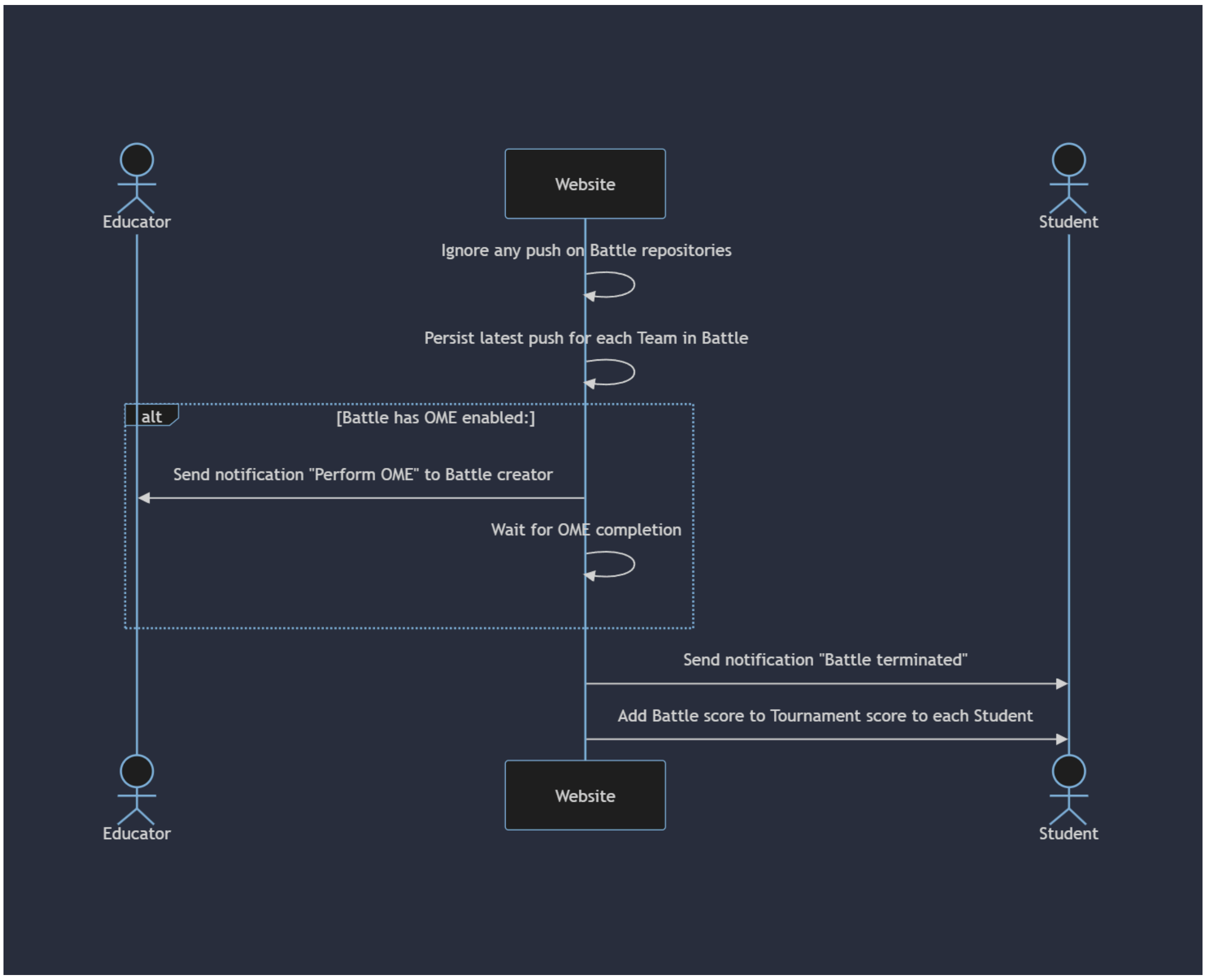

Educator
Website
Student
Open "Tournament Details" section
Show list of Students subscribed to Tournament
Perform "Ban Student" on a Student
Send notification "Banned from a Tournament"
Educator
Website
Student

UC27. FBD is expired

Name	FBD is expired
Actor	Educator, Student, System
Entry Condition	The FBD of a Battle has expired.

Event Flow	1. System ignores any push in the Battle's Repository. 2. System persists the last push for each Team in the Battle. 3. If the Battle have OME: 3a. The Creator of the Battle "cb" receives a notification to "Perform OME". 3b. The Students in a Battle receive a notification to inform them that the FBD is expired, so new pushes will be ignored, they need to attend the "Perform OME" by the "cb" or the FDT expires, and it is possible to see the temporary leaderboard of the Battle. 4. When the "cb" "Perform OME" or the FDT is expired the Battle is Terminated. 5. For each Student in a Battle's Team, the System adds to their Tournament score the score obtained by the Team in the Battle.
Exit Condition	The Battle becomes "Terminated".
Alternative Flows	The Battle doesn't have OME, so step 3 is skipped and the Battle is Terminated after step 2.
Exceptions	E aborts the operation because they have changed their mind.

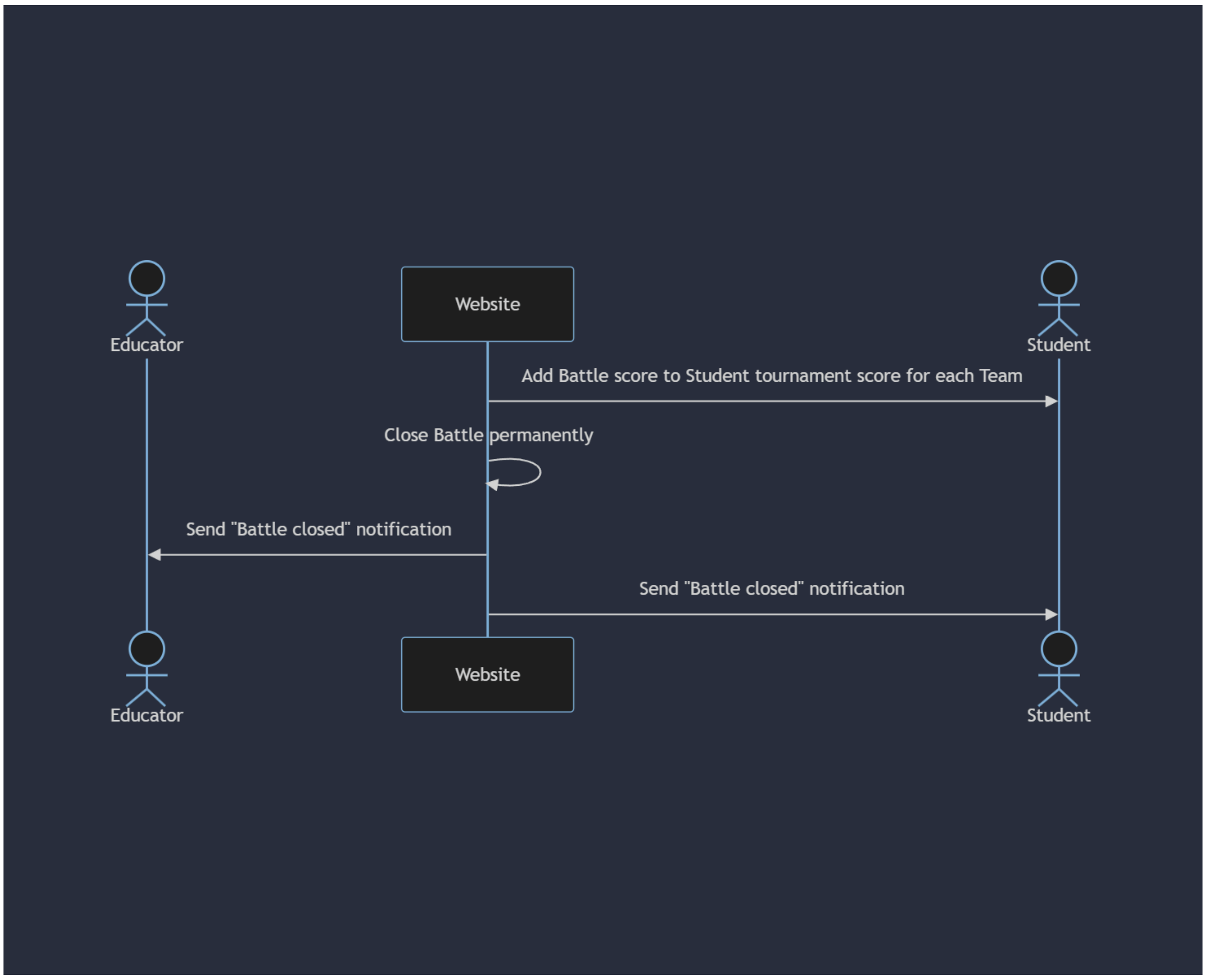
```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor S as Student
    W->>W: Ignore any push on Battle repositories
    W->>W: Persist latest push for each Team in Battle
    alt Battle has OME enabled:
    W->>E: Send notification "Perform OME" to Battle creator
    W->>W: Wait for OME completion
    end
    W->>S: Send notification "Battle terminated"
    W->>S: Add Battle score to Tournament score to each Student
```


Educator
Website
Student
Ignore any push on Battle repositories
Persist latest push for each Team in Battle
alt
[Battle has OME enabled:]
Send notification "Perform OME" to Battle creator
Wait for OME completion
Send notification "Battle terminated"
Add Battle score to Tournament score to each Student
Educator
Website
Student

UC28. Battle is Terminated

Name	Battle is Terminated
Actor	Educator, Student
Entry Condition	A Battle b is Terminated or the FTD is expired.
Event Flow	1. For each Student in a Battle's Team, the System adds their Team Battle' score. This score is the maximum score obtained. 2. The System closes the Battle permanently. 3. The System sends notification to the Educator, who has created the Battle, and to the Students that were in the Battle, to see the final leaderboard of the battle, and the updated leaderboard of the Tournament.
Exit Condition	The Battle became "Closed", and the Leaderboards (of the battle and of the tournament) are correct.

```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor S as Student
    W->>S: Add Battle score to Student tournament score for each Team
    W->>W: Close Battle permanently
    W->>E: Send "Battle closed" notification
    W->>S: Send "Battle closed" notification
```



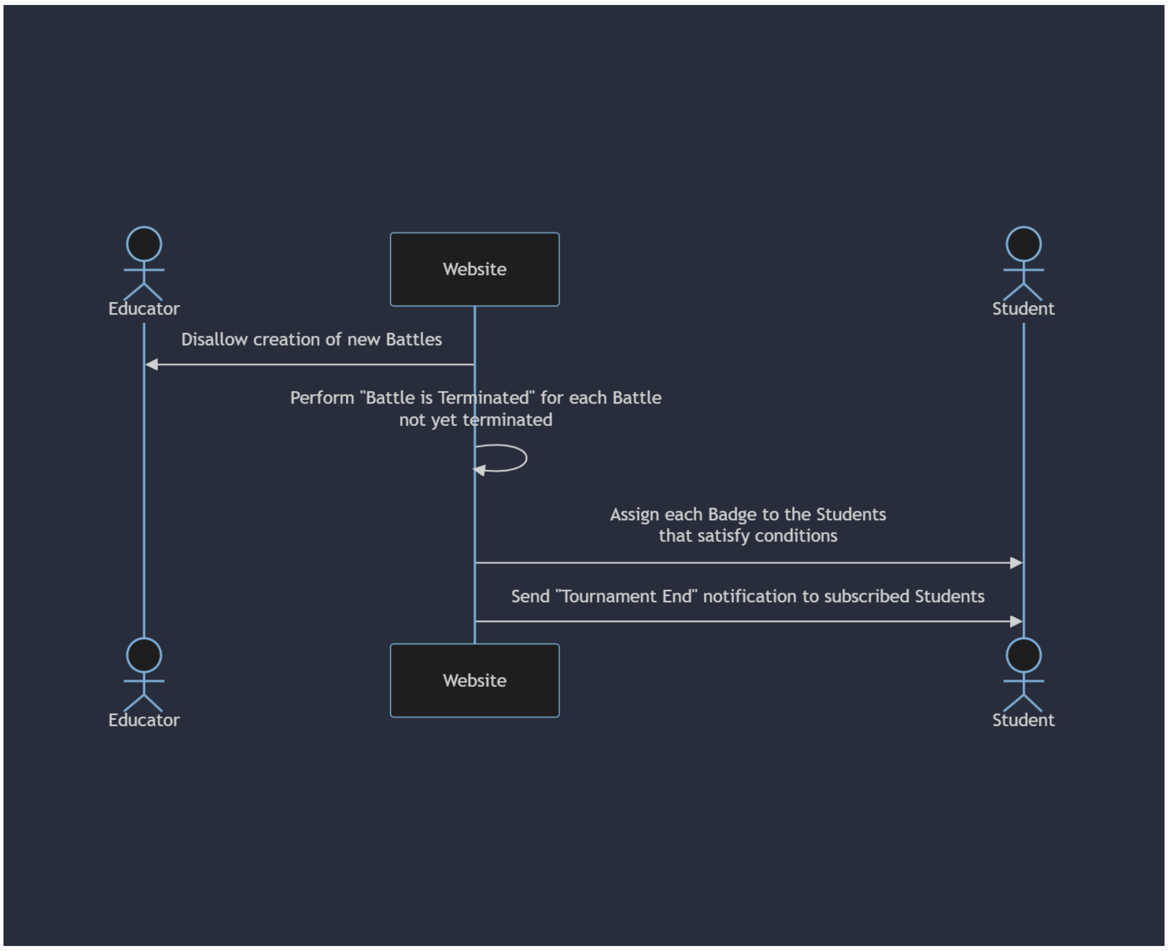
Educator
Website
Student
Add Battle score to Student tournament score for each Team
Close Battle permanently
Send "Battle closed" notification
Send "Battle closed" notification
Educator
Website
Student

UC29. End Tournament

Name	End Tournament
Actor	Educator, Student
Entry Condition	The FTD of a Tournament has expired

Event Flow	1. The System disallows creation of new Battles for the Tournament. 2. The System performs “Battle is Terminated” for each Battle that isn’t already terminated due to OME not performed yet. 3. The System assigns each Badge to the Students that satisfy the conditions. 4. The System sends a notification to see the final leaderboard to each Student, to the creator of the Tournament and to the TCs in the Tournament.
Exit Condition	The Badges are assigned correctly and the Tournament is closed correctly.

```
sequenceDiagram
    actor E as Educator
    participant W as Website
    actor S as Student
    W->>E: Disallow creation of new Battles
     W->>W: Perform "Battle is Terminated" for each Battle<br>not yet
terminated
    W->>S: Assign each Badge to the Students<br>that satisfy conditions
    W->>S: Send "Tournament End" notification to subscribed Students
```


Educator
Website
Student
Disallow creation of new Battles
Perform "Battle is Terminated" for each Battle not yet terminated
Assign each Badge to the Students that satisfy conditions
Send "Tournament End" notification to subscribed Students
Educator
Website
Student

3.3 - Performance Requirements

The System has to guarantee good performance in order to serve a large number of concurrent requests and submissions. This imposes a time limit on the request execution which must be no greater than 500 ms. The System must also handle cases where the connectivity is limited or absent.

3.4 - Design Constraints

3.4.1 - Standards Compliance

The entire CodeKataBattle System must respect all the laws regarding privacy and data treatment and exchange with third parties (GitHub): to work in Europe, it must respect the EU GDPR.

The User must always give their permission prior to sharing their personal information with CodeKataBattle (specifically through the “Login with GitHub” functionality). The User must be able to request account deletion in order to adhere with GDPR.

3.4.2 - Hardware Limitations

CodeKataBattle can be used both from a desktop computer or from a smartphone.

3.5 - Software System Attributes

3.5.1 - Reliability

The CodeKataBattle system does not deal with critical user operations (e.g. payment processing) and exists purely as a platform to engage in some fun competitions between programmers. So, considering this, it is considered reasonable to have a failure rate between 0.1% and 1%, as to be a high-quality reliable system.

3.5.2 - Availability

Based on a reasoning similar to Section 3.5.1, the CodeKataBattle system must guarantee 99.9% of uptime by contract.

3.5.3 - Security

The CodeKataBattle system stores personal information of its Users. As such, it is critical to maintain state-of-the-art standards in terms of cybersecurity.

Data exchange must be performed using HTTPS tunnels, and the database chosen for system implementation must support encryption-at-rest in order to store user data in an encrypted manner.

3.5.4 - Maintainability

The CodeKataBattle system shall be divided into different modules depending on the desired functionalities to increase maintainability as much as possible. Moreover, every single functionality must be thoroughly documented to avoid knowledge silos.

3.5.5 - Portability

The CodeKataBattle system is not directly tied to the hardware of its User, since it runs on all major browsers. As such, portability is not a concern.

4 - Formal Analysis (Alloy)

4.1 - Alloy Model

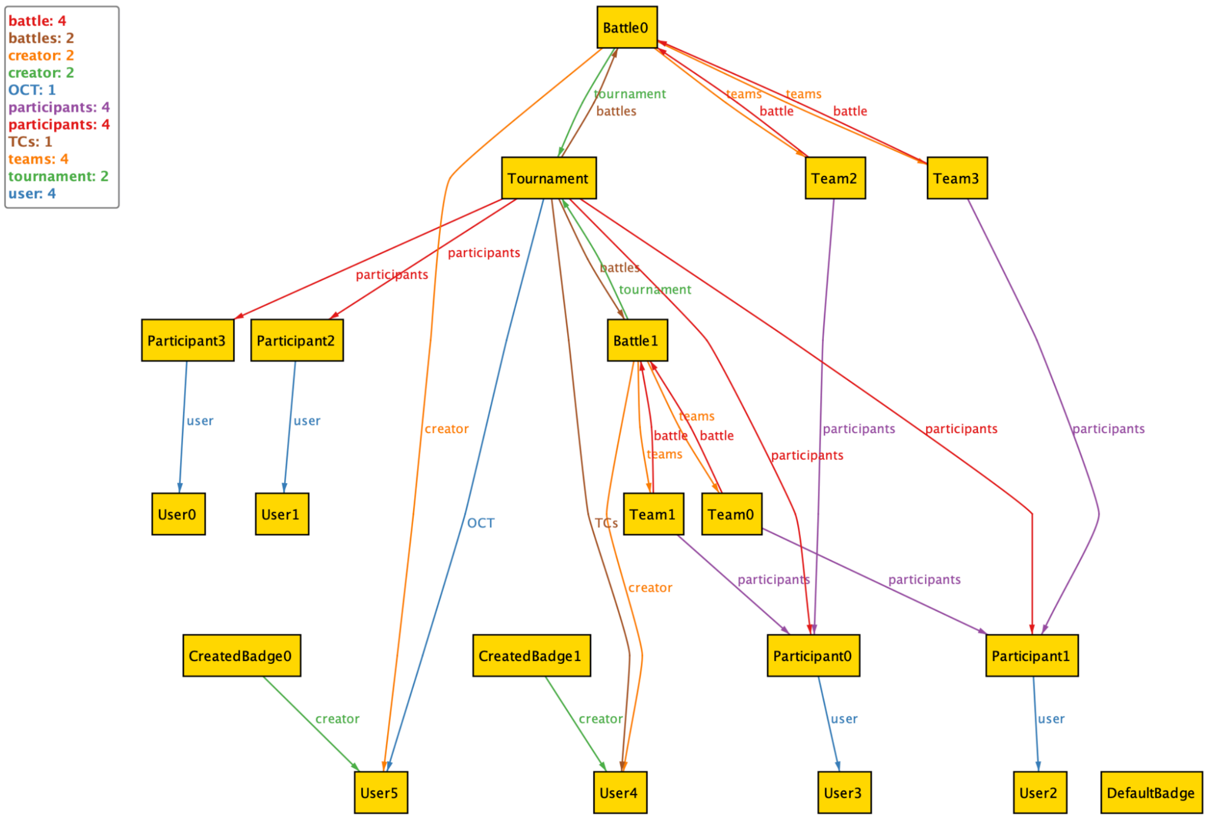
This section presents the model built with Alloy to represent the world in which the CodeKataBattle system operates.
The objective of the formal analysis is to verify the correctness and the completeness of relations, and to check some simulated worlds visually in order to guarantee exactness.

```
abstract sig Badge {
}
sig DefaultBadge extends Badge {
}
sig CreatedBadge extends Badge {
     creator: one User
}
sig User {
     badges: set Badge,
     friends: set User
}

sig Participant {
     user: one User
}

sig Team {
     participants: some Participant,
     battle: one Battle
```

```
}

sig Battle {
     tournament: one Tournament,
     creator: one User,
     teams: set Team,
}
sig Tournament {
     badges: set Badge,
     participants: set Participant,
     OCT: one User,
     TCs: set User,
     battles: set Battle,
}{
     OCT not in TCs
}

// A OCT or TC can not be a Participant
fact CoordinatorNotParticipant{
     all t:Tournament, p: Participant |
     (p in t.participants) implies ((p.user != t.OCT) and
(p.user not in t.TCs))
}

// Bidirectional relation Battle-Tournament
fact BattleLinkTournament {
     all t:Tournament, b: Battle | (b in t.battles) iff
(b.tournament=t)
}

// Creator of a Battle must be OCT or TC
fact BattleCreator {
     all t:Tournament, b: Battle | (b in t.battles) implies
(b.creator in (t.OCT+t.TCs))
}

// Participants of Battle must be participants of Tournament
fact BattleParticipant{
     all t:Tournament, b:Battle | (b in t.battles) implies (
b.teams.participants in t.participants )
}

// Badge must be created by the OCT
fact BadgeCreation {
     all t: Tournament, b: Badge | (b in t.badges) implies
((b=CreatedBadge) iff (b.creator=t.OCT))
}

// If a User has a badge, it comes from a specific Tournament
fact AssignedBadge {
     all u: User, b: Badge| some t:Tournament | (b in
u.badges) implies ((b in t.badges)and(u in
t.participants.user))
}

// Bidirectional relation between Team and Battle
fact BattleLinkTeam{
     all t:Team, b:Battle | (t in b.teams) iff (t.battle = b)
}
// There can’t be the same participant in two different Teams
for a Battle
fact BattleTeam {
     all b:Battle| no disj t1,t2: Team | some p:Participant |
     (t1 in b.teams and t2 in b.teams) and
     (p in t1.participants and p in t2.participants)
}

// A User can participate only one time to a Tournament
fact PerticipantTournament {
     all t:Tournament| no disj p1,p2: Participant | some
u:User |
     (p1 in t.participants and p2 in t.participants) and
     ((u = p1.user) and (u = p2.user))
}

// Symmetric relation between friends
fact Friendlink {
     all u1,u2: User | (u2 in u1.friends) iff (u1 in
u2.friends)
}

// Non-reflective relation between friends
fact NotSelfFriends {
     all u: User | u not in u.friends
}

// Participant must exist in a Tournament
fact ParticipantInTournament{
     all p:Participant | one t:Tournament | (p in
t.participants)
}
```

4.2 - Simulation

```
pred show []{
     #Badge=3
     #User=6
     #Participant=4
     #Team=4
     #Tournament=1
     #Battle=2
}
run show for 10
```



5 - Effort Spent

Group Member	Effort Spent (hours)
Nicolò Giallongo	Introduction: 12h Overall Description: 13h Specific Requirements: 15h Formal Analysis (Alloy): 11h
Giovanni Orciuolo	Introduction: 13h Overall Description: 12h Specific Requirements: 24h Formal Analysis (Alloy): 2h
Giuseppe Vitello	Introduction: 12h Overall Description: 12h Specific Requirements: 16h Formal Analysis (Alloy): 11h

6 - References

- Version Control System: GitHub (https://github.com)
- State diagrams made with: Draw.io (https://app.diagrams.net)
- Domain class diagram made with: Draw.io (https://app.diagrams.net)
- Use case diagrams made with: Draw.io (https://app.diagrams.net)
- Use case sequence diagrams made with: Mermaid (https://mermaid.js.org)
- Document written with: Google Documents (https://docs.google.com)
- Formal Analysis: Alloy (https://alloytools.org)