POLITECNICO
MILANO 1863

Code Kata Battle

Requirements Analysis and Specification Document

first version - 22/12/2023

Students:
Léandre Le Bizec (10990212)
Reza Ghadiri (10951889)

Software Engineering 2
2023-2024

Contents

1 Introduction

1.1 Purpose

This document is intended to guide the development process for Code Kata Battle (CKB): a platform allowing users to train and improve their programming skills. CKB platform enables users to join tournaments and improve their abilities by engaging in kata battles—exercises crafted and shared by other users. These kata battles, posted in tournaments, provide users with opportunities to elevate their scores through active participation.

This specification will address problem scope, goals and constraints, but also give an abstract (i.e non- architectural) view of the system, as long as indicating the intended use cases.

1.1.1 Goals

The table 1 lists the objectives of our platform

Table 1: Goals

Id	**description**
G1	An user must be able to take part in a tournament.
G2	An user must be able to take part in a kata battle alone or with a team and submit his solution.
G3	An user must be able to create a tournament and to manage it.
G4	An organiser of a tournament must be able to create kata battles, adding rules about the number of participants, deadlines and how to compute score of participants taking part.
G5	The platform must be able to analyze users' code, run test and assign them a score and a ranking within the kata battle and the tournament.

1.2 Scope

Defining world and shared phenomena within the scope is crucial for system development, facilitating seamless integration with external elements and enabling the system to respond effectively to unobservable events. This categorization informs logic, guides user experience design, aids in system state management, and ensures comprehensive testing, resulting in a robust and adaptable system aligned with user expectations and responsive to changing circumstances.

1.2.1 World Phenomena

The table 2 lists the different world phenomena, i.e. the phenomena that the machine cannot observe.

1.2.2 Shared Phenomena

The table 3 lists the different shared phenomena, i.e. the phenomena that are observed by the machine and the world.

Table 2: World Phenomena

Id	description
WP1	An user makes modifications on the GitHub repository without pushing
WP2	An user locally launches tests on his code

Table 3: Shared Phenomena

Id	description
SP1	An user sign up
SP2	An user sign in
SP3	An user creates an account
SP4	An user creates a tournament
SP5	An user creates a KB
SP6	An user invites other users to join a tournament
SP7	An user joins a tournament
SP8	An user starts a KB
SP9	An user pushes its code with git
SP10	An user takes a look on other solutions
SP11	An user takes a look on the ranking
SP12	An user manages the settings of its account
SP13	A mail is sent by CKB

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

1.3.2 Acronyms & abbreviations

- RASD: Requirements Analysis and Specification Document
- KB: Kata Battle
- CKB: Code Kata Battle
- G: Goal
- WP: World Phenomena
- SP: Shared Phenomena
- DM: Domain assumption

1.4 Revision history

- **22/12/2023** : first version

1.5 Reference Documents

- The specification document ”Assignment RDD AY 2023-2024.pdf”.
- Slides of Software Engineering 2 course on WeBeep;

1.6 Document structure

This document is composed of six sections, detailed below.

- In the 1st section the problem is introduced together with the associated goals of the project. Additionally the scope of the project is specified along with the various phenomena occurring. Lastly, the necessary information to read the report is presented, such as definitions and abbreviations;
- Section 2 contains an overall description of the system, including a detailing of its users and main functions. Moreover there is the class diagram, descriptions of several scenarios, some state charts and finally the domain assumptions made in this report;
- In section 3 the requirements on the system is specified. This includes functional requirements, non-functional requirements and requirements on external interfaces. Furthermore use cases are described, with accompanying use case and sequence diagrams. Section three also contains mappings of functional requirements to the goals of the system, and to the use cases;
- Section 4 contains a formal analysis with the help of Alloy. Together with the Alloy code, the analysis objective is described;
- In section 5 there is a presentation of the project members total effort spent;
- Section 6 contains the references used.

2 Overall Description

The aim of this section is to provide a nuanced understanding of the project's contextual framework, functional scenarios, and structural representations essential for its development.

2.1 Product perspective

Within the realm of the product perspective, this subsection delves into pivotal aspects, including scenarios, class diagrams, and state charts.

2.1.1 Scenarios

Let's start by describing the application's usage scenarios.

S1: An user want to visit and/or use CKB platform for the first time
Lorenzo, a computer science professor at Politecnico di Milano, discovered CKB platform and, impressed by positive feedback, opted to integrate it into his algorithm course. Upon accessing the platform's homepage and navigating to the user area, he created an account by entering his personal details. Paolo, a student in Lorenzo's course, followed the same account creation process to participate in the activities proposed by the teacher.

S2: An user want to connect to CKB platform

Upon validating their account through the platform's confirmation email, Lorenzo and Paolo can login by entering their username and password in the designated user area, gaining access to their personal space. Upon accessing their personal page post-login, Lorenzo and Paolo can:

- Search for available tournaments to participate in;
- Access the current tournament if already joined.
- Create a new tournament;
- Manage tournaments they organize, if applicable;

S3: An organizer want to create a tournament

Alice, a computer content creator on the Internet, wants to create a tournament for her followers to take part in. From her personal space, she selects the "create a tournament" option. Once on the tournament creation window, Alice follows the steps below:

- Add KBs to the tournament;
- Set a start and end date for the tournament;
- If Alice wishes, she can add other users to become tournament organizers, so that they can :
  - Add KBs to the tournament;
  - help Alice manage the tournament.

In the last stage, Alice selects the visibility of her tournament: public, allowing any user to join, or private, requiring an invitation link or manual addition by Alice for participation.

S4: An organizer want to create a KB in a tournament

Emmanuela, a computer science teacher, has the ability to create a KB for her students' training by selecting the option within a tournament. Upon logging in, she can function as an organizer by creating or managing a tournament. Alternatively, participating in a tournament allows her to take on the role of a participant, contributing to the enhancement of her computer skills. Every user of the platform have the possibility to become an organizer by creating tournament or a participant by joining a tournament. Once Emmanuela click on create a KB in the tournament management window, she will need to provide several information:

- a description of the exercise;
- a set of tests along with a build to compile and run the tests on students' code;
- specify the minimum and maximum number of participants;
- set a registration deadline;
- establish a deadline for submitting final solutions;
- configure the scoring parameters.

Once every element have been completed, the kata battle is created and saved in the tournament.

S5: An organizer want to invite an user to become an organizer of the tournament

Daniele, seeking assistance from Teo in managing his tournament, has two options on the event management page:

- Send Teo a clickable link through any communication channel, Teo becomes an organizer upon clicking the link and logging into his account;
- Add Teo's username to the list of organizers, automatically including him and notifying him via email.

S6: An organizer want to invite users as participants to his tournament

Maria, having completed the addition of desired KBs to her tournament, want to invite her students to participate. She can do it using two methods:

- Distribute a clickable link through any communication channel; students become participants upon clicking the link and logging into their accounts;
- Add students' usernames to the participant list, automatically including them in the tournament and notifying them via email.

If Maria chooses not to restrict her tournament to her students, she can also opt to make it public.

S7: A participant want to do a KB

After joining a tournament by clicking on a link, Leo wants to take part in a KB. Once on the tournament participation page, he can choose between several KBs. He decides to start with the first one, as the organizer has decided to classify his KBs in order of increasing difficulty. He then clicks on KB. A new window opens, offering two choices:

- Register for the KB on his own;
- Register for the KB as a group.

As Leo has no group, he decides to register for the KB on his own. Before the KB begins, CKB platform will create teams to meet the KB participation requirements.

S8: A participant want to create a team

Mario, Lisa, and Pablo, participants in a tournament, decide to collaborate on a KB. Mario takes the initiative to enter the team into a KB. Clicking on "Participate in KB" he then selects "Register as a team" in the new window, where he provides the names of his colleagues. In case the team lacks sufficient members to meet the KB constraints, the platform will automatically supplement the team. Conversely, if there are excess members, Mario will promptly be notified of this.

S9: The participation constraints for a KB are not met before its beginning

Upon the initiation of a KB, the platform must ensure that participation constraints are met. If the platform fails to resolve the constraint problem for any reason (unsolvable problem given the constraints or a bug), the KB's commencement will be postponed, requiring organizer intervention to address the conflict. To mitigate such issues, the platform will conduct a resolution simulation before the start of a KB, notifying the organizers in the event of anticipated conflicts to prevent any delays in the KB's commencement.

S10: A participant is notified that a KB is starting

A KB reaches its start date, with all constraints satisfied. A GitHub repository containing the KB is automatically created and sent via email to KB participants. Upon receiving the link, Giulia can now join the GitHub repository. Before starting work, she needs to establish an automatic workflow through GitHub Actions that will notify CKB platform. This workflow enables real-time automatic updates of all students' scores. Once this workflow is in place, Giulia can begin working on the KB.

S11: A participant want to send its solution

Alexandro has successfully validated all the tests for a KB. To submit his final solution to CKB platform, he initiates a Git push, automatically updating the platform through the workflow he had previously set up.

S12: A participant want to read the solution of others student

Andrea has completed a KB and wishes to review solutions offered by other students to gain insights into how his peers approached the challenging problem. To do so, he logs into CKB platform, navigates to the tournament page, and clicks on the KB he has just completed with a 100% score. Since Andrea has achieved a full completion, a new tab titled "Student Solutions" becomes accessible. In this tab, he can now examine the pushes made by students who have successfully validated 100% of the tests.

S13: An user want to know the ranking of a tournament

Emma wants to check the rankings of different teams in a KB and her personal ranking in the ongoing tournament. To do this, she visits CKB platform, navigates to the current tournament window, and can view the various KB rankings by clicking on them and accessing the "Ranking" tab. Alternatively, she can check the overall ranking by going to the "Ranking" tab in the tournament window. For each ranking, she can access team rankings or individual rankings.

S14: An user want to manage its account

Sofia wants to update her account details. She can do so by clicking on the profile icon in the settings, where she can modify her information or choose to delete her account.

2.1.2 Class diagram

The purpose of a class diagram is to provide a visual representation of the structure and relationships within a system or application. Class diagrams depict the classes, attributes, methods, and associations among different objects in a system, offering a concise overview of the system's architecture. They serve as a communication tool for developers, designers, and stakeholders to understand the static structure of a system, aiding in design, analysis, and documentation throughout the software development lifecycle.

The figure 1 shows the class diagram for CKB platform.

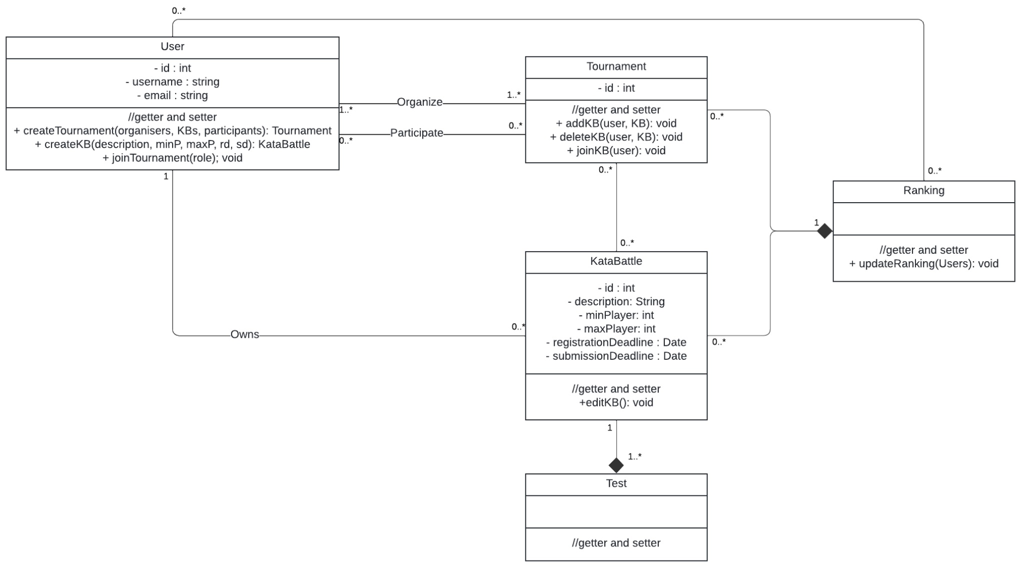


Figure 1: Class Diagram of CKB platform

2.1.3 State charts

In this section we present state charts for the most complex aspects of the system. The most trivial ones have been left out for the sake of brevity. The purpose of these state diagrams is to define the various states of an object during a complex query. Developers can consult these diagrams to verify the functionality of a query, while customers can ensure that all required resources are available for the seamless execution of complex queries.

Join a tournament

The figure 2 shows the state chart for joining a tournament.

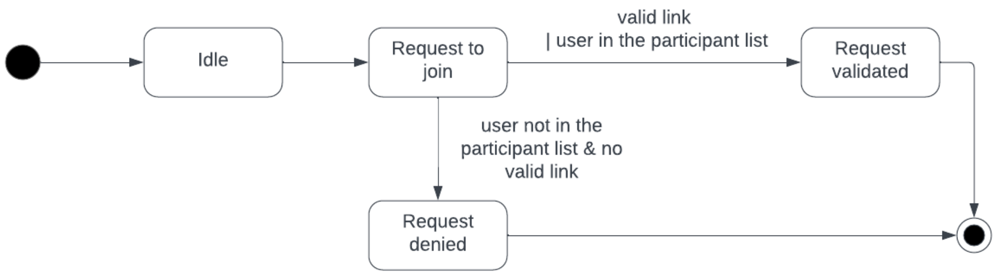

Idle
Request to join
valid link | user in the participant list
Request validated
user not in the participant list & no valid link
Request denied

Figure 2: State chart for joining a the tournament

When a user wishes to join a tournament via his personal space, a request is issued to check that this user does indeed have the right to access this tournament (i.e. that he has clicked on an invitation link or that he has been manually added to the list of participants by the organizer). If this request is validated, the user can access the tournament, otherwise he/she cannot.

Start a KB

The figure 3 shows the state chart for starting a KB.

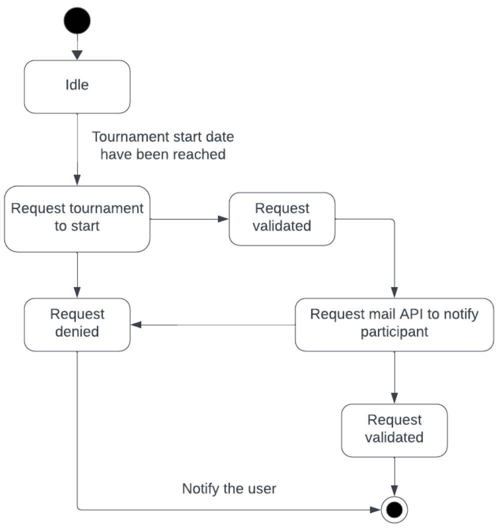

Idle
Tournament start date have been reached
Request tournament to start
Request validated
Request denied
Request mail API to notify participant
Request validated
Notify the user

Figure 3: State chart for starting a KB

Once the KB start date has been reached, the platform communicates with the GithHub API to create a repository associated with the KB. Once the repository creation has been validated, the platform communicates with a mail API to send repository links to users. If an error occurs at any point, the process is stopped, KB organizers are notified and action is required on their part to restart the process.

Start a tournament

The figure 4 shows the state chart for starting a tournament.

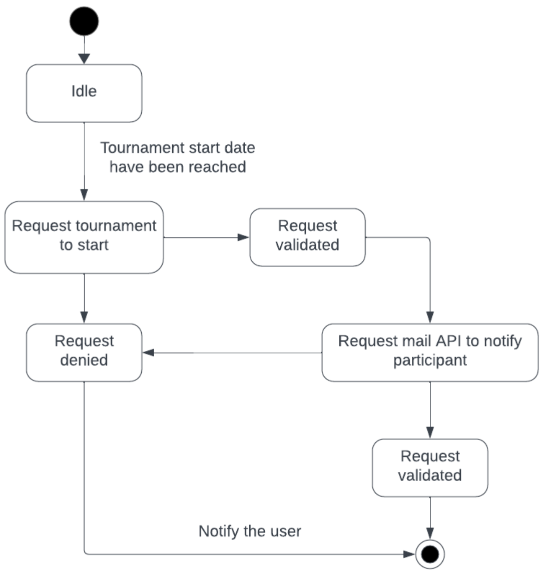

Idle
Tournament start date have been reached
Request tournament to start
Request validated
Request denied
Request mail API to notify participant
Request validated
Notify the user

Figure 4: State chart for starting a tournament

Once the tournament start date has been reached, the platform communicates with a mail API to notify participants of the tournament beginning. If an error occurs at any point, the process is stopped, KB organizers are notified and action is required on their part to restart the process.

Receive and compute the score of a KB and of the tournament
The figure 5 shows the state chart for the computation score process.

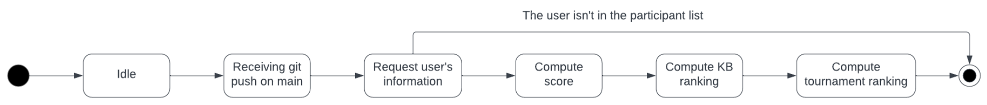

The user isn't in the participant list
Idle
Receiving git push on main
Request user's information
Compute score
Compute KB ranking
Compute tournament ranking

Figure 5: State chart for the computation score process

When a user pushes his or her code onto the hand panel, a notification is sent to CKB platform via the feed set up by the user. This notification triggers the user to check the platform's coast. If the user is indeed participating in this KB in this tournament, then the platform recovers the score calculation method for the KB in question and calculates the score. The calculated score is then used to update the ranking of this KB. The KB ranking and updated scores are then used to calculate the overall tournament ranking.

2.2 Product functions

Here we present the main functions of CKB platform in more detail.

Register and Login
A user will be able to create an account by providing an e-mail address, a password and a username. They can then use this information to log on to the platform and access their personal space.

Create a tournament
Any user will be able to create a tournament for which he or she will then be an organizer. To create a tournament, you need to provide start and end dates, and choose a tournament name. Once the tournament has been created, it is possible to add KBs, organizers and participants, either manually or by sending a link.

Create a KB in a tournament
Any user can create a KB from a tournament. When creating a KB, the user will be asked for :

- a description of the exercise;
- a set of tests and a compilation to compile and run the tests on the students students;
- to specify the minimum and maximum number of participants;
- to configure scoring parameters;
- a registration deadline;

- a start date;
- to set a deadline for submission of final solutions;

The KB will be automatically saved in the tournament.

Join a tournament

Any user can join a tournament, simply by clicking on a valid invitation link, which has the effect of adding the user to the list of participants, or of being added to the list of participants by an organizer. When a user subsequently wishes to access the tournament, he/she will simply be checked to see if he/she is on the list of participants.

Start a tournament When the start date of the tournament is reached, the tournament became open and accessible for every registered participant. A mail is sent to every registered participant.

Start a KB When the start date of the tournament is reached, the linked GitHub repository is created. The KB is now accessible through this link. A mail included a link to this repository is sent to every registered participant of the KB.

Play a KB

Any tournament participant can initiate a KB. To do so, when the user is on the tournament window, they can start a KB. It is then verified that the user launching the KB is on the tournament's participant list.

2.3 User characteristics

CKB platform offers a single user type, which can be divided into two distinct categories:

- **Organizer**: An organizer's role is to create KBs within a given tournament and manage that tournament.
- **Participant**: A participant's role involves joining tournaments and honing their programming skills.

Each user has the flexibility to be either an organizer or a participant, a distinction determined by the specific tournament in which they are involved. It is possible for a user to be an organizer in Tournament A and a participant in Tournament B.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

The table 4 lists the different domain assumption, i.e. the descriptive assertions assumed to hold in the world.

Table 4: Domain assumption

Id	**description**
DA1	The user provide correct information
DA2	The user has a GitHub account
DA3	The user know how to use git and use it on his computer
DA4	The GitHub API is working well
DA5	The mail API is working well
DA6	The user establishes a connection between the GitHub repository created by CKB platform and CKB platform, enabling automatic analysis of each push to the main branch.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

In this section of the document, the user interface for views of users is presented. The platform is just available as a web application. Since we have two different types of users, Students, and Educators, the app needs two different views. Students can see the information about each tournament and join battles and subscribe to tournaments. Educators can create tournaments and battles. Also, both type of users are able to see the ranks of each battle and tournament and profiles of other students(Including badges).

3.1.2 Hardware Interfaces

The Hardware Interface is the same for both types of users mentioned in section [number of section]:

- For access and use of the platforms users need a device that has a web browser and internet connection.

3.1.3 Software Interfaces

The platform requires some software interfaces to be available and all parts to be active:

- Github API is required to create repositories and test the code of the users.
- Email API for sending emails to users and making them aware of the time of the end of the battles, tournaments, final rankings, and invitations from other users.

Hence the platform is using GitHub as a place to get the answers, students need to have or create a GitHub account.

3.1.4 Communication Interfaces

Internet connection is required for using the platform from all user devices and also for the creation of the battles and giving answers to them the system uses the GitHub API.

3.2 Functional Requirements

Table 5: Functional Requirements - part 1

Id	**description**
R1	The system allows users to register.
R2	The system allows registered users to log in.
R3	The system allows users to see the list of unrestricted tournaments.
R4	The system allows users to create a tournament.
R5	The system allows users to create a battle for their tournament.
R6	The system allows users who are organizers of a tournament to grant other users to organizer for that tournament.
R7	The system notifies users about the new tournament created.
R8	The system allows users to subscribe to a tournament
R9	The system allows users to join the battles with free spaces before the registration deadline.
R10	The system notifies subscribers of a tournament about new battles of that tournament.
R11	The system allows participants to create teams by sending invitations for battles that are not full.
R12	The system allows users to join the free battles by invitation before the registration deadline.
R13	The system allows users who are subscribers of a battle to see the rank of that battle.
R14	The system Creates the GitHub repository containing the code kata for the battle.
R15	The systems send the link of the repository to all student subscribers of tournaments after the tournament started
R16	The system triggers each commit of the repositories of participants in the battle using API during the time of that battle.
R17	The system analyzes and runs the tests with the code of participants from their GitHub repository after each push.
R18	The system gives the score, a number between 0 and 100, to the participants after each push on their repository before the submission deadline.
R19	The system allows organizers to set configurations for scoring.
R20	The system calculates and generates the rank of each battle
R21	The system allows organizers to check and score the answers manually.

Table 6: Functional Requirements - part 2

Id	**description**
R22	The system updates the score of each team, including singletons, after any push
R23	The system allows participants and organizers to see the rank during the battle
R24	The system notifies participants when the final rank becomes available.
R25	The system allows participants and organizers who subscribed to see the rank of the tournament.
R26	The system allows organizers to close a tournament.
R27	The system Notifies participants from closing the tournament.

3.2.1 Mapping on Goals

Table 7: Mapping on goals

Goal	**Domain assumptions**	**Requirements**
G1	D1 D2 D3	R1 R2 R8
G2	D2 D3	R1 R2 R9 R11 R12
G3	D1 D5	R4 R5 R6 R26
G4	D1 D4 D5 D6	R5 R19 R21
G5	D4 D6	R16 R17 R18 R20

3.2.2 Use cases

Table 8: Register user

Name	Register User
Actors	User, Email provider
Entry condition	The user doesn't have an account on the platform.
Event flow	1. User opens the platform. 2. Clicks on the register button 3. system shows a form to fill 4. user fills the form 5. system checks the information is in the correct format 6. system sends an email to the email that the user inserted in the form 7. user click on the link in the email 8. system actives the user account 9. system redirects the user to the login page
Exit condition	The registration has been successful and the data of the user has been saved in the database and the account is now active.
Exception	The data that the user inserted is not valid, the system shows a message to correct the data. The link of email confirmation is expired, and the system redirects the user to the register page.

Table 9: Login user

Name	Login user
Actors	User
Entry condition	The user isn't logged in on the platform.
Event flow	1. user opens the platform 2. Clicks on the login button 3. system shows a form to fill 4. user fills the form 5. system checks the information is in the correct format 6. system checks if a user with that information exists or not 7. system allows the user to log in to the account 8. system redirects the user to the profile
Exit condition	The login has been successful and the user is now logged in.
Exception	The data that the user inserted is not valid or the account does not exist in the system. the system shows a message to correct the data.

Table 10: Create tournament

Name	Create tournament
Actors	User
Entry condition	The user presses create new tournament button
Event flow	1. system shows a form to the user 2. user fills the form 3. user clicks on the create button 4. system creates a new tournament and saves it in the database 5. system notifies all users about new tournaments by an email
Exit condition	The system shows a message to the organizer that tells tournament creation was successful

Table 11: Create Battle

Name	Create Battle
Actors	User, GitHub API
Entry condition	The organizer presses create new battle button
Event flow	1. system shows a form to the organizer 2. organizer upload the code kata(description and software project, including test cases and build automation scripts) 3. organizer sets minimum and maximum number of students per group 4. organizer sets a registration and final submission deadline for the battle 5. organizer sets the additional configuration for scoring 6. organizer click on the create button 7. system creates a new battle and saves it in the database 8. calls GitHub API and creates a repository for the battle 9. system updates the database with the link to the repository 10. calls email API and sends a notification email to the participants
Exit condition	The system shows a message to the organizer that tells battle creation was successful
Exception	The details of the battle don't fill completely, system shows a message to correct the form.

Table 12: Grand Organizer

Name	Grand organizer
Actors	User
Entry condition	The organizer press add organizer button
Event flow	1. system asks for an email or a username and generates a shareable link 2. organizer Fill the form or copy the link 3. organizer click on the Add organizer button 4. system adds a new organizer to the tournament or copies the link in the user clipboard and saves it in the database
Exit condition	New organizers have been added to the tournament and the system shows a message Educator added
Exception	The inserted user does not exist in the platform and the system shows a message to the user

Table 13: Subscribe to a tournament

Name	Subscribe to a tournament
Actors	User
Entry condition	The participant presses the Subscribe the tournament button or clicks on the tournament link.
Event flow	1. system registers the details of the user in the tournament 2. system shows a message to the user 3. user redirects to the page of the tournaments and can see the list of the battles
Exit condition	The participant subscribed to the tournament and the system shows a message
Exception	The deadline of subscribing the tournament is finished

Table 14: Join a battle

Name	Start battle
Actors	User, GitHub
Entry condition	The organizer presses the start the battle button or sets a time for the start of the battle
Event flow	1. system calls GitHub API 2. GithHub makes the repository public 3. system calls the Email API 4. Email provider sends an email to subscribers of the tournament with the link of the repository
Exit condition	The battle started and the system showes a message to the organizer

Table 15: Join a battle

Name	Join a battle
Actors	User
Entry condition	The participant presses Join the battle button or clicks on the invitation link
Event flow	1. system registers the details of the user in the battle 2. system updates the rank if the battle was started 3. system calls the GitHub API and triggers the repository of the participant
Exit condition	The participant joins the battle and the system shows a message
Exception	The capacity of the battle be full or the deadline is finished system shows a message

Table 16: Play a battle

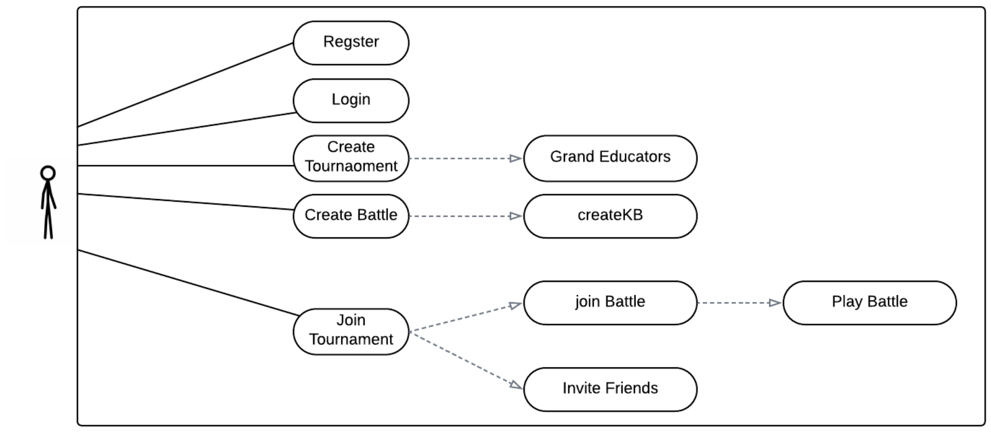
Name	Play a battle
Actors	User, GitHub, GitHub API
Entry condition	The participant joined a battle and the battle is started
Event flow	1. The participant pushes the code/answer of the battle to the repository 2. The participant can repeats this, until the deadline of the battle

Table 17: Close a tournament

Name	Close a tournament
Actors	User, Email API
Entry condition	The Organizer presses Close the Tournament button
Event flow	1. system asks the organizer to confirm the closing of the tournament 2. organizer click on confirm 3. system closes all the battles of the tournament 4. system closes the tournament 5. system notifies the participants of the tournament about the closing of the tournament with an email
Exit condition	The system shows a message to the organizer

3.3 Use case diagrams

1. **User**



3.3.1 Sequence diagrams

1. **User Registration**

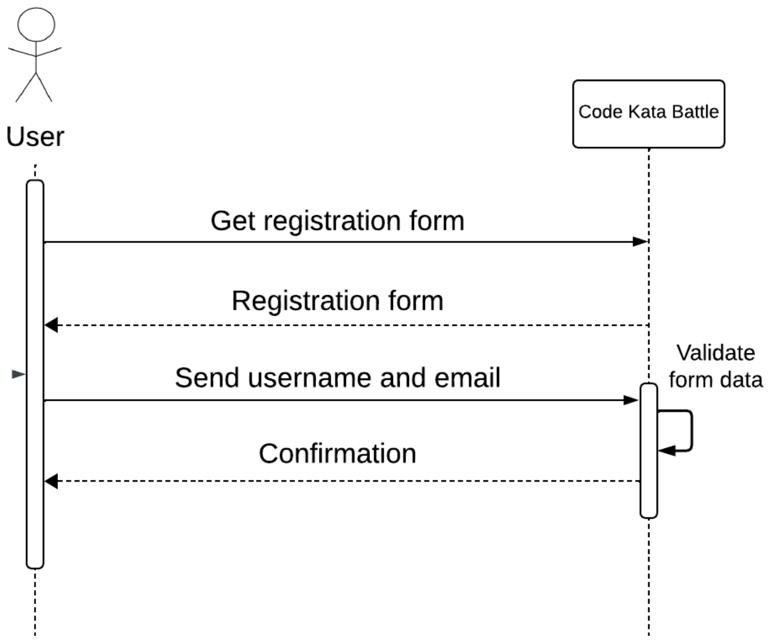


Figure 6: User registration sequence diagram

2. **User Login**

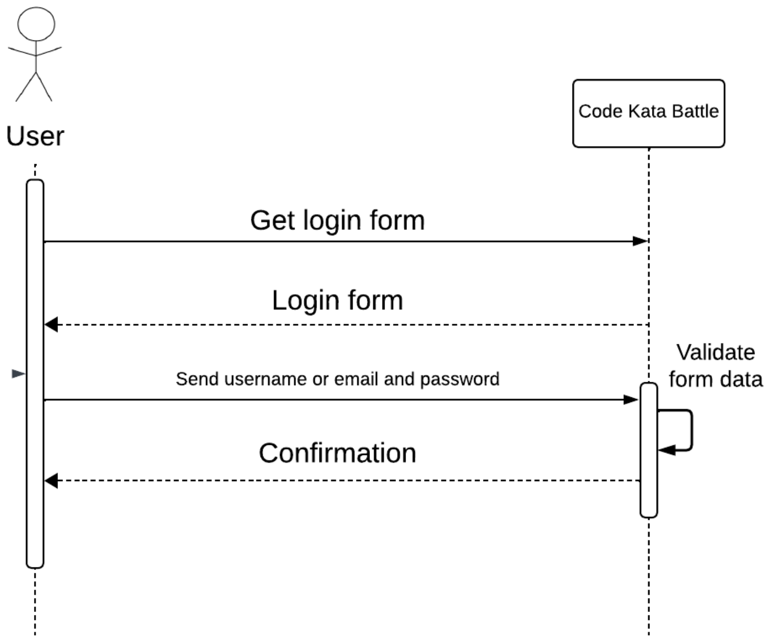


Figure 7: User login sequence diagram

3. **Create Tournament**

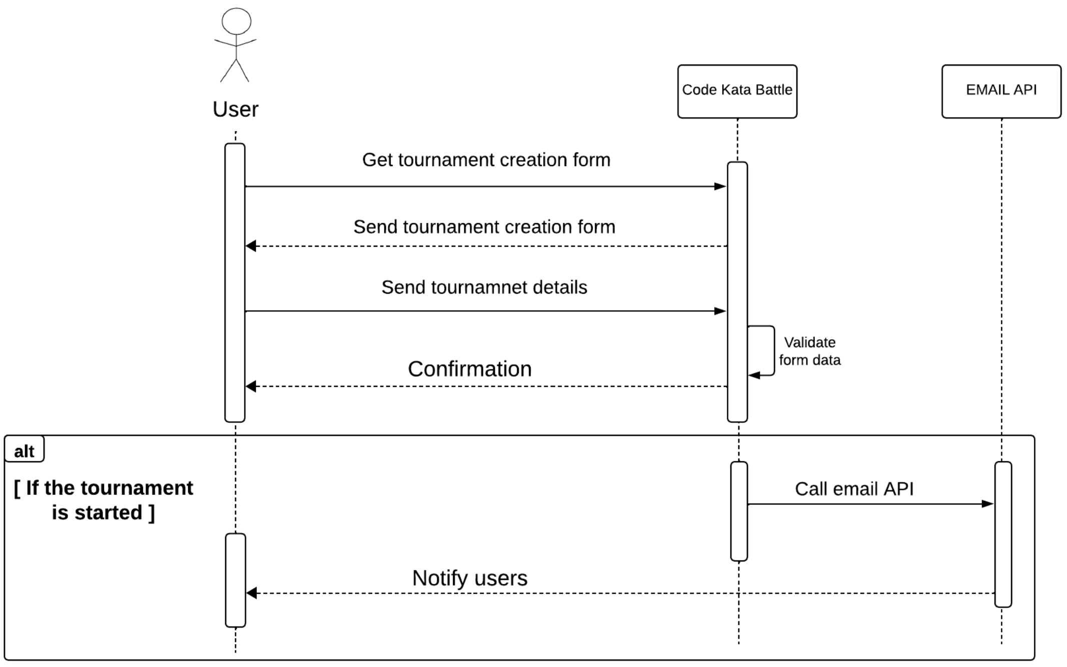


Figure 8: Create tournament sequence diagram

4. **Create Battle**

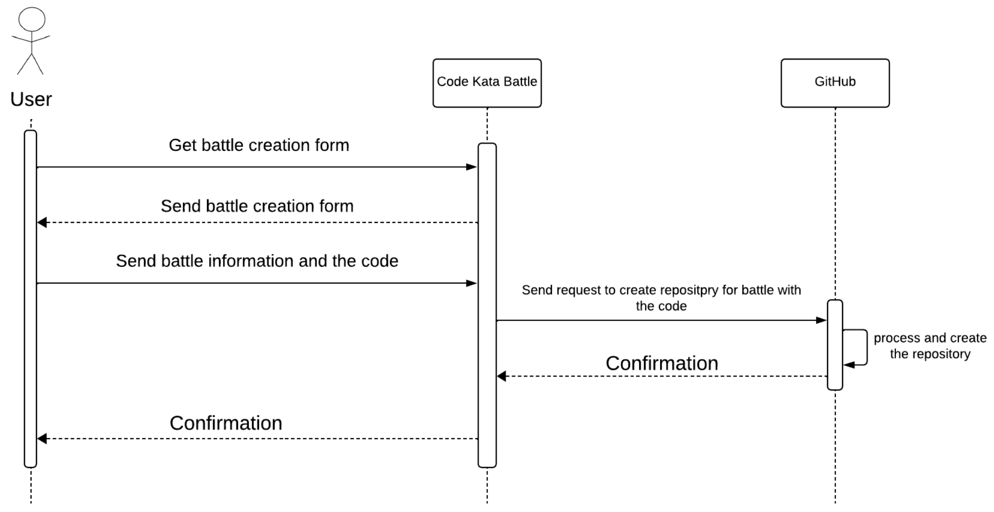

User
Code Kata Battle
GitHub
Get battle creation form
Send battle creation form
Send battle information and the code
Send request to create repositpry for battle with the code
process and create the repository
Confirmation
Confirmation

Figure 9: Create battle sequence diagram

5. **Grand organizer**

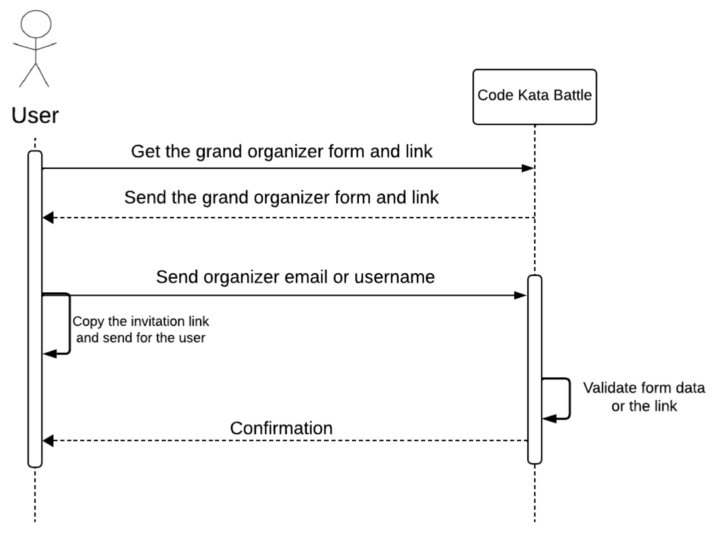

User
Code Kata Battle
Get the grand organizer form and link
Send the grand organizer form and link
Send organizer email or username
Copy the invitation link and send for the user
Validate form data or the link
Confirmation

Figure 10: Grand organizer sequence diagram

6. **Subscribe to a tournament**

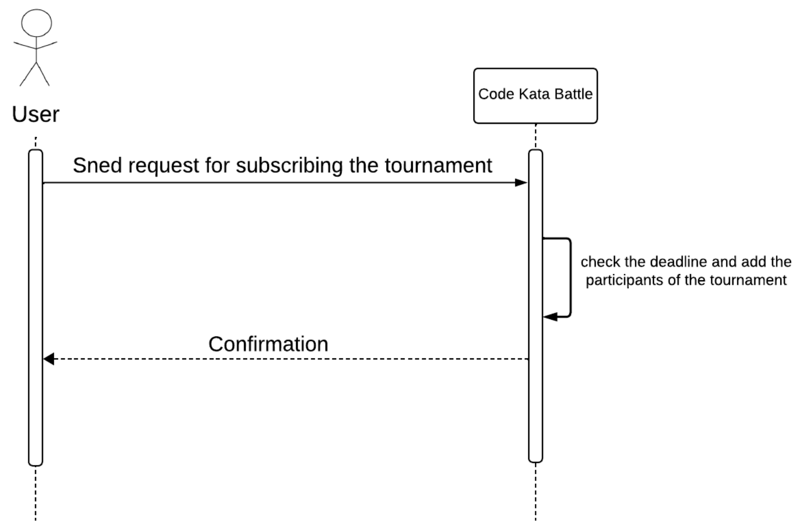

User
Code Kata Battle
Sned request for subscribing the tournament
check the deadline and add the participants of the tournament
Confirmation

Figure 11: Subscribe to a tournament sequence diagram

7. **Start battle**

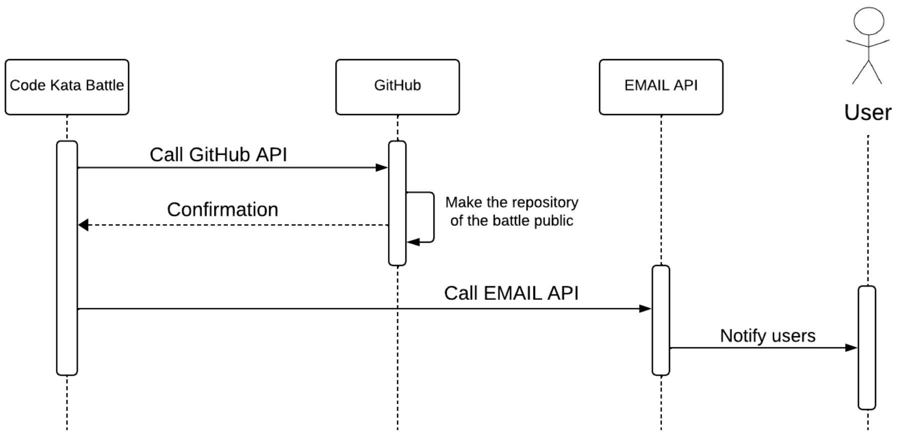


Figure 12: Start battle

8. Join Battle

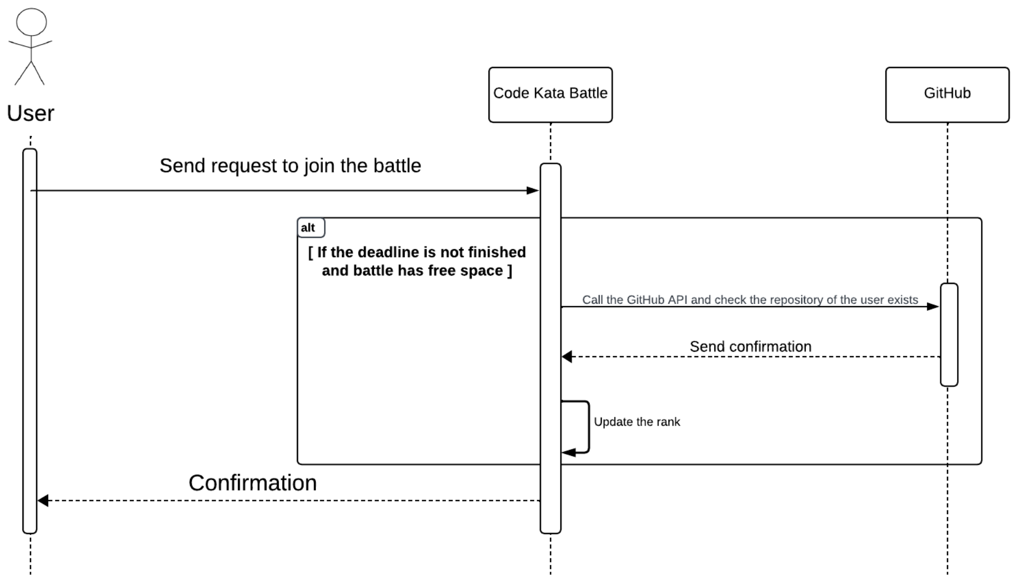

User
Code Kata Battle
GitHub
Send request to join the battle
alt
[If the deadline is not finished and battle has free space]
Call the GitHub API and check the repository of the user exists
Send confirmation
Update the rank
Confirmation

Figure 13: Join battle sequence diagram

9. Play a Battle

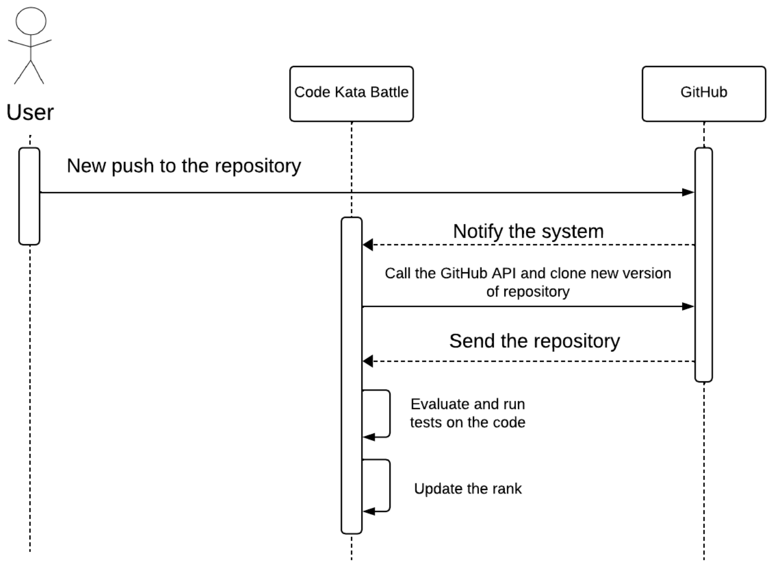

User
Code Kata Battle
GitHub
New push to the repository
Notify the system
Call the GitHub API and clone new version of repository
Send the repository
Evaluate and run tests on the code
Update the rank

Figure 14: Play a battle sequence diagram

10. **Close a tournament**

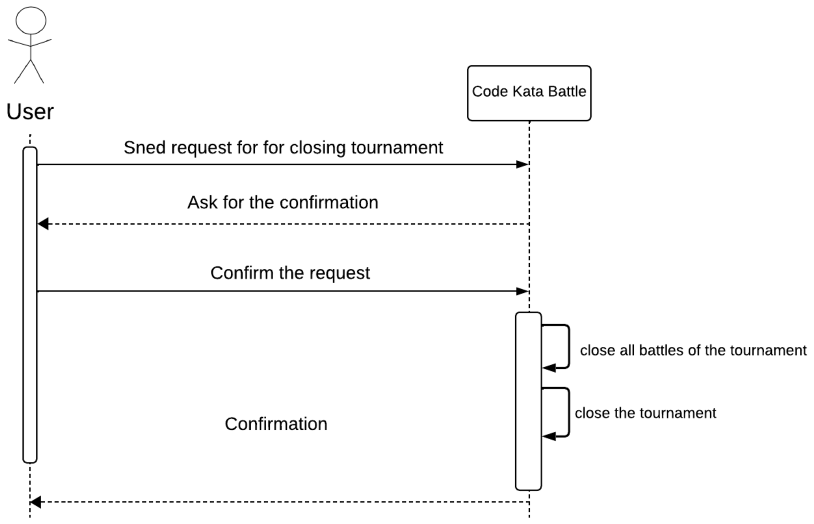


Figure 15: Close a tournament sequence diagram

3.3.2 Mapping on requirements

Table 18: Mapping on requirements

Use Case	Requirements
Register user	R1
Login user	R2
Create tournament	R4, R7
Create battle	R5
Grand organizer	R6
Subscribe to a tournament	R8
Start battle	R10, R14, R15
Join a battle	R9, R12
Play a battle	R16, R17, R18, R20, R22, R23
Close a tournament	R26, R27

3.4 Performance Requirements

The system needs to guarantee availability and good performance. It means the system could be able to handle many users simultaneously and respond to all requests. The response to the users should be at most couple of seconds. Another necessary requirement for the system is to evaluate users' codes with great accuracy in a short time. It should take at most a couple of minutes to analyze and run tests on codes and update the rank. Hence, connection to the GitHub by the API should be always stable.

3.5 Design Constraints

3.5.1 Standards compliance

About the privacy of data, the eMall project is subject to the General Data Protection Regulation (GDPR), a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA). The system has to adopt international standards about date and time use and representation.

3.5.2 Hardware limitations

- Regarding the user: Internet connection (2G/3G/4G/5G/Wi-Fi), iOS/Android/Mac/Linux/Windows Desktop, Laptop, Tablet, Smartphone, modern web browser

3.6 Software System Attributes

3.6.1 Reliability & Availability

The web application needs to load in at most 15 seconds so users won't wait a long to access the platform and with the Lazy-load feature and caching, the availability of the system can be better. Due to the fact, that the system is providing a service that is not very crucial, the system must provide availability of 99.9%. This means that the average time between the occurrence of a fault and service recovery (MTTR) has to be contained at around 0.365 days per year.

3.6.2 Security

The system stores some data from users, these data are not very sensitive but there should be encryption for data which is stored in the database to protect users' data and their accounts. Also, the API of GitHub is very important to protect and should be encrypted and changed every month for better security. Furthermore, the system needs a backup server to backup all data of users, tournaments, and battles to protect from any important data loosing of the platform. The backup should be daily to avoid of any disaster for the platform.

3.7 Maintainability

3.7.1 Portability

The platform is just accessible through the web application, via the browser. Due to the fact that it's a platform for learning and the variety range of students and their hardware and software could be completely different from each other, the system should be runable even on the old and weak systems and the focus should be on Google chrome and chromium, Firefox, Safari and Microsoft Edge. Also being responsive is important and users with tablets and smartphones should be able to use the platform.

4 Formal Analysis

4.1 Alloy code

```
//Signatures

sig user {}
sig organizer extends user {}
sig participant extends user {}


abstract sig code {} //sig for code(answer)of each battle
abstract sig rank {} //sig for rank of each tournament

sig tournament {
        //tournament start and end time
        tournamentStartTime: disj one Int,
        tournamentDeadLine: disj one Int,
        //organizers of tournamnet
        tournamentOrganizers: some organizer,
        //participants of tournament
        tournamentSubscribers: some participant,
        //each tournament has a relation with just one rank
        tournamentRank: disj one rank,
}{
        //conditions for time fields in tournament
        tournamentDeadLine > tournamentStartTime and tournamentStartTime > 0
            ↪ and tournamentDeadLine > 0
}

sig battle {
        //battle start and end time
        battleStartTime: disj one Int,
        battleDeadLine: disj one Int,
        //each battle has a relation with just one code
        battleCode: one code,
        //each battle has an integer field, as capacity
        battleCapacity: disj one Int,
        //each battle has a realation with just one tournament
        battleTournament: one tournament,
        //battle organizers and participants
        battleOrganizers: some organizer,
        battleParticipants: some participant
}{
        //conditions for time fields in battle
        battleDeadLine > battleStartTime and battleStartTime > 0 and
            ↪ battleDeadLine > 0
}


//Facts
fact tournamentFact {
        #tournamentOrganizers ≥ 1 //each tournament should have at least one
            ↪ organizer
        #tournamentSubscribers ≥ 0 //each tournament can have zero or more
            ↪ subscribers
        #rank = #tournament //number of ranks should be equal to tournaments
}
fact battleFact {
        #battleOrganizers ≥ 1 //each battle should have at least one organizer
        #battleParticipants ≥ 0 //each batlle can have zero or more
            ↪ participants
        #code = #battle //number of codes should be equal to battle
}

//a battle organizer should be the battle's tournament organizer too
fact battleOrganizersAreTournamentOrganizer {
  all b: battle | all o: b.battleOrganizers | o in b.battleTournament.
      ↪ tournamentOrganizers
}

//a battle participant should be the battle's tournament subscriber too
fact battleParticipantsAreTournamentSubscribers {
  all b: battle | all o: b.battleParticipants | o in b.battleTournament.
      ↪ tournamentSubscribers
}

//each battle has a capacity and number of participants should be control
fact battleCapacityLimitation {
     all b: battle | all p: b.battleParticipants | all c: b.battleCapacity | #
         ↪ p ≤ c} // limitation of a battle participants


//Pred & Commands
run {} //for 7//but 3 organizer
```

4.1.1 Meta model

The figure 4 shows the model for the alloy code of the project:

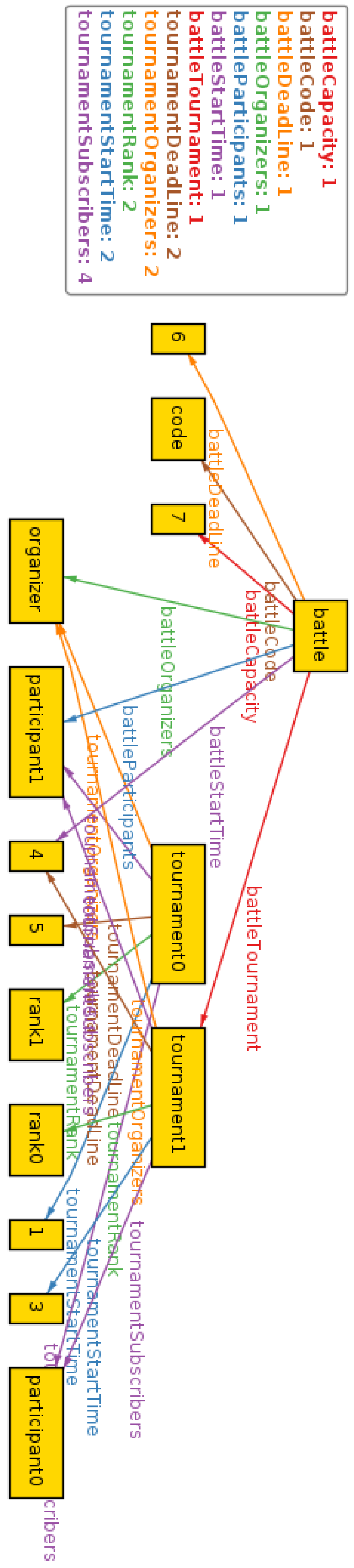
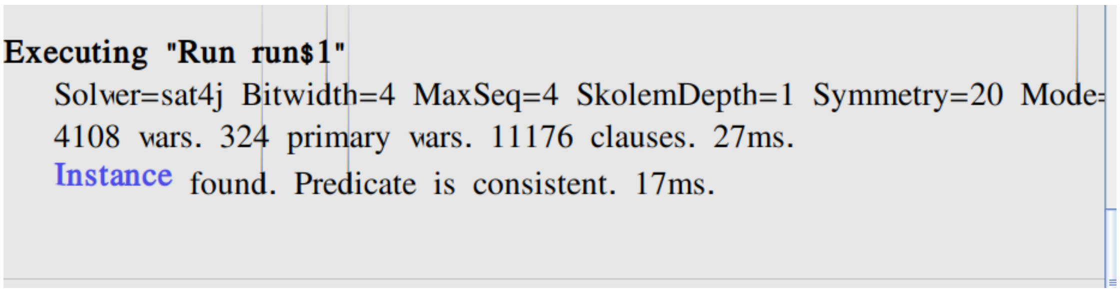


Figure 16: The model for the alloy code

4.1.2 Result of predicates

As you can see in the code, there is just one predicate in the Alloy code, and the mode is shown in figure 4. Also here is the execution and number of Instances:



```
Executing "Run run$1"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=
   4108 vars. 324 primary vars. 11176 clauses. 27ms.
   Instance found. Predicate is consistent. 17ms.
```

Figure 17: predicate results

5 Effort spent

5.1 Léandre

5.2 Reza

6 References