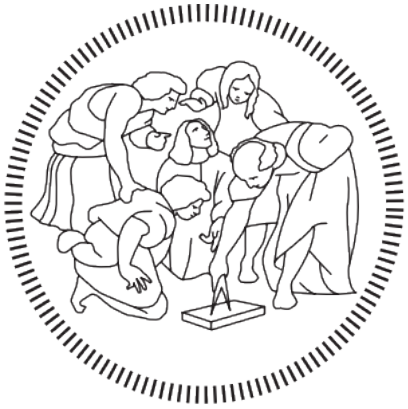

POLITECNICO
MILANO 1863

CodeKataBattle

Acceptance test
deliverable documentation

Software Engineering 2 project
Academic year 2023 - 2024

6 january 2024
Version 1.0

Authors:
Tommaso Pasini
Elia Pontiggia
Michelangelo Stasi

Professor:
Matteo Camilli

Contents

1. Introduction

This document has the purpose of showing the results of the acceptance tests performed on the CodeKataBattle platform developed by another team. The tests are performed on the system to verify that the system meets the requirements and the expectations of the customer.

In the following pages it is reported a complete and detailed description of the tests performed, the results obtained and the bugs found during the tests.

1.1 Scope

The project analyzed was developed by the team composed by:

- Maria Lucia Porfido
- Rosanna Iannaccone
- Federica Persico

Team's repository: `https://github.com/marialucia54/IannacconePersicoPorfido`

2. Installation and setup

Although the installation instruction reported on the ITD document are complete and exhaustive, the code presents a README file that briefly explains how to install and run the system (if the machine is already equipped with python), and the major dependencies are automatically installed by simply running the `pip install -r requirements.txt` command.

All the installation went smoothly, and the system was up and running in a few minutes.
The only thing that should be added is a default location for the application, because if the user directly connects to `localhost:8000` as suggested by the server log, no page is shown, and the user has to manually navigate to `localhost:8000/login` to access the login page.

3. RASD document

3.1 Alloy

This part is well reported, all the facts are described accurately with the comments and the result of the predicate **show** is explained in all its details.
However, in the facts is missing the unicity of a player; we would suggest to add some properties to signatures like **User**, such as the **email, username and password**.
Another thing that is not specified, is the feature about the boundaries of the teams subscribed to a certain battle : in fact, these properties are missing in the **Battle** signature, therefore we can not say nothing about the teams' size and we can not manage the number of students in a team with respect to the battle boundaries.

4. Acceptance test cases

The test cases are taken entirely from the use cases reported in the RASD document. The test cases are reported in the following tables:

4.1 Test cases

Name	Registration on CKB platform
Input	The user inserts its credentials and clicks on the registration button
Expected	The user is registered on the platform and can access the main page
Pass/Fail	Pass

Name	Discovering and Participating in Tournaments
Input	The STU navigates on the platform and selects and joins a tournament
Expected	The STU is now part of the tournament
Pass/Fail	Incomplete
Notes	Once the student selects the tournament on the main page, the system will automatically add the student to the tournament; it would be better to add a confirmation message to the user (as it is expected to happen according to what is written on the RASD use case). Furthermore, nowhere it is specified the submission deadline to the tournament

Name	Creating a Tournament
Input	The EDU creates a tournament and fills in the required fields, then clicks on the create button
Expected	The tournament is created
Pass / Fail	Pass
Notes	A tournament can be created selecting dates in the past, moreover it is not required to define an end deadline because the educator manually close the tournament.

Name	Granting permissions
Input	In a tournament, created by edu A, A can search for another EDU B and grant him the permission to manage the tournament
Expected	EDU B is now a manager of the tournament
Pass / Fail	Pass
Notes	When the educator sign up could omit to complete the email input field because there are no controls if the field is completed or not

Name	Create a Battle
Input	The EDU compile the for to create a battle and then submit it
Expected	The battle is created and shown inside the tournament
Pass / Fail	Pass
Notes	The educator, in the upload file input field, can put any type of file because there are no controls. This is a problem because when creating the repository the code expect a zip folder

Name	Joining a battle
Input	The STU selects a tournament and joins a battle within It
Expected	The STU is now part of the battle
Pass / Fail	Pass

Name	Viewing tournament rankings
Input	The User selects a tournament and selects the rankings button
Expected	The User can see the rankings of the tournament
Pass / Fail	Pass

Name	Pushing a commit
Input	The STU push a commit to the github forked repository
Expected	The platform runs tests, analyzes the code for quality, timeliness, and other functional aspects and then update the score
Pass / Fail	We could not figured out how to set up properly the workflow action in the forked repository so we could not test it by doing the push, besides that we looked at the code and it seems that it should get the data from the workflow action
Notes	The number of test passed are checked by the workflow action, but we did not find where the other analysis are done such as the timeliness or the static analysis

Name	Manually evaluating
Input	The EDU clicks on evaluate button to go through the sources produced by STUs
Expected	The team code is evaluated and the additional score is assigned
Pass / Fail	We can not express because we did not manage to reach the evaluation scenario

Name	Closing a tournament
Input	The EDU selects a tournament and closes it
Expected	The tournament is closed
Pass / Fail	Pass

More about the use cases

GitHub Repository
The creation of a GitHub repository through its APIs is functioning correctly and thus this request has been implemented. However, we have a note regarding compliance with the specification.
It is required that the repository be created immediately after the deadline for registration ends and that all students registered on the platform be notified. In the case of the reviewed project, the repository is generated only when a Student or an Educator decides to view the details of the battle (after checking the deadlines). This also implies that the first user who performs this action may wait indefinitely before seeing the details of the battle because they have to wait for the repository to be generated. Our recommendation is, in the short term, to execute the creation of the repository in a separate thread from the main one, while in the long term, to adapt the code to be more consistent with the client's request.
Despite what was said, the repository is correctly generated and available to proceed with the battle.

View Student Profile
We have observed that when entering Name, First Name and Last Name, Email, or GitHub Account Name, the student search function appears not to be working. We suspect that a decision was made not to make this functionality available because, primarily in the user profile, it is used to view the badges earned. However, since badge implementation was not required, it may have been decided not to proceed with the implementation.

4.2 Bug reports

There are quite a few recurring bugs in the system, which are listed below:

- The system does not show the message to confirm that a general action (e.g. subscription to a tournament, creation / closure of a tournament, etc.) has been completed: this is as simple as important to the user experience, as it is expected to happen according to what is written on the RASD use case.
- Whenever an error occurs, the system does not show any message to the user, but it simply dumps the python error page on the screen: this is not only bad for the user experience, but it is also a security issue, as it could reveal sensitive information to the user
  The stack dump is very useful for the developers, but it should be removed after the development phase.
- The students are not always notified about relevant events occur (the *Notification* section doesn't show the new entries)

- Some form fields do not have controls in the frontend and even in the backend, this brings the platform to unexpected error that are not handled and therefore they show the python stack trace

4.3 Final Comments

In general, we found the application usable and quite user friendly, with a well-structured code and sufficiently commented in large files

We think that the only problems we found out are due to the short time the project was developed in and we are quite sure that, in a real world scenario, after our revision the developers would immediately fix the bugs and the inconsistencies we found out. Even if we are not familiar with python, we still think that it is just a matter of a bunch of lines of code. The main functionalities are done and the app works fine.

5. Effort Spent

Team

Topic	Time

Table 5.1: Effort Spent during team meetings

Tommaso Pasini

Topic	Time
Second part of the use cases	2h

Table 5.2: Effort Spent by Tommaso Pasini

Elia Pontiggia

Topic	Time
First part of the use cases	3h

Table 5.3: Effort Spent by Elia Pontiggia

Michelangelo Stasi

Topic	Time
RASD and ITD	2h

Table 5.4: Effort Spent by Michelangelo Stasi