POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

CodeKataBattle - ATD

Software Engineering 2 Project
MSc Computer Science and Engineering

Authors: **Michele Bersani, Paolo Chiappini, Andrea Fraschini**
Reference Professor: **Prof. Matteo G. Rossi**

Student IDs: **10707192, 10743051, 10725610**
Academic Year: **2023-24**
Revision: **09/02/24**
GitHub repository: **https://github.com/paolo-chiappini/BersaniChiappiniFraschini**

Contents

1. Analyzed project

- **Authors**: Alessandro Feraboli, Marco Filippini, Simone Lucca.
- **Repository link**: https://github.com/SimoneLucca2/FeraboliFilippiniLucca.
- **Reference documents**:
    - Repository README/ITD for installation and instructions on how to run the prototype code;
    - RASD and ITD for requirements and acceptance tests;
    - DD and ITD for the description of the architecture and structure of the code.

2. Installation and setup

The installation and setup of the project were straightforward and clear and followed exactly the instructions presented both in the README in the repository and the ITD (also available in the repository).

In particular, we had to set the secret token for static analysis through the SonarQube docker and compose the docker image of the server by inserting the required tokens from SonarQube and GitHub.

Regarding the front end part, we followed the instructions with no problems. After downloading the requested libraries we were able to run the web application.

3. Acceptance test cases

Based on the implemented requirements described in the ITD, we tested the following features:

[R1]: The system allows users to sign-up and log-in.

User Registration:

- Creation of a Student account;
- Creation of an Educator account;

Description: This test verifies the correct behavior of the signup process. A user should be able to register for an account and information should be validated so that there can't be two identical users. At the end of the signup process the user should be presented with the correct information and available actions based on account type (Educators should have the option to create new tournaments).

Outcome: The platform correctly creates the account for the user with the corresponding information. The platform displays the correct information and actions. The platform doesn't allow the creation of repeat users.

User Login:

- Login as a Student;
- Login as an Educator;

Description: This test verifies that it is possible to log into an existing account. The test also ensures that it is not possible to log into an account that does not exist or to log into an existing account with the wrong credentials.

Outcome: The system allows you to log into an account when provided with the correct credentials. The system correctly displays an error when trying to log in to either a non-existing account or an existing account with the incorrect password.

[R3]: The system allows each educator to create a tournament at any moment.

Tournament Creation:

Description: This test verifies the correct behavior of the tournament creation. When creating the tournament, the platform should allow the user to enter the name of the tournament and the registration deadline. Moreover, the server should not create a tournament with the same name as an existing one or with a registration date before that at the time of the request.

Outcome: The platform successfully creates the tournament if the name and date are valid. If at least one of the two registration requirements is not followed, the platform shows a pop-up with the description of the error. Also, the server allows the creation of a tournament, even if other active ones exist.

[R4]: The system updates, upon the end of the battle, the tournament score of the students who have participated in it.

Consolidation of users' scores:

Description: This test aims to verify if the tournament leaderboard updates at the end of a battle with the scores awarded to students.

Outcome: The platform successfully displays the scores assigned to groups in the leaderboard. Note: the scores shown are just those coming from manual evaluation.

[R6]: The system allows the tournament leader to add other educators to the tournament at any point in time.

Invite an educator to an existing tournament:

Description: This test aims to verify that an educator, after creating a tournament, can add another educator to the same tournament.

Outcome: The platform allows an educator to add another educator into the tournament, only when the tournament is still active and not closed.

Additional features: A user may be interested in receiving a notification when being added to a tournament. An invite system similar to the one already implemented for the students may be useful to also allow educators to decline being added to the tournament.

[R7]: The system allows the tournament leader to close a hosted tournament.

The leader educator of a tournament can close the tournament:

Description: This test aims to verify the educator's ability to close the tournament when they are the host. The test is performed by using an educator account, and then a tournament is created and subsequently closed.

Outcome: The platform successfully allows only the tournament leader to close the tournament. Originally, reading the RASD, once the tournament is closed, the battles within it that were not yet closed can continue in a closing state. However, in this prototype, when the tournament is closed with battles still active, they are automatically closed in the closed state, the reason for the choice being to simplify the implementation.

[R8]: The system allows an educator to create a battle within the context of a tournament.

The leader educator and the invited educators can create a battle within a tournament if it is in the competition state:

Description: This test aims to verify the correctness of the battle creation. An educator account that has previously been invited to the tournament is used. Afterward, the form to create the battle is filled out with the registration deadline, submission deadline, and min/max group members. Furthermore, during the creation phase, the code should be uploaded, and additional configurations for scoring should be set.

Outcome: The developed prototype allows the user to create a battle just by entering the title, description, min/max members, and the two deadline dates. However, it does not allow the user to upload the code and to choose the evaluation parameters. Thus, the outcome of the test is not positive and contradicts what is stated in point 11 at *[ITD-v1.0, Implemented requirements].*

[R9]: The system allows students to join a battle only if it has not started yet.

[R18] The system allows group leaders to invite peers in the group if the battle has not started yet. Students can be invited even if they are participating in another group, but they need to choose only one.

[R19] The system stops students from entering groups that have reached the maximum number of participants.

A student can join a battle alone or with a group:

Description: The purpose of the test is to verify whether the platform allows a user who joined a tournament to invite other students to an open battle. To verify this, a student account that has just signed up for a battle is used. Afterward, the student sends an invite by clicking on the "Invite" button next to another student's email.

Outcome: The platform successfully allows the invitation of students. If the student is not yet registered in a group in that battle, they are automatically added to a new group. The system also allows to change group by entering the ID of the group to join. The system does not allow users to accept or decline invites and doesn't verify whether the users have been invited to the group upon changing it. From the error displayed by the UI, it looks like it is not possible to join groups that have reached the maximum amount of members.

[R10]: The system requires students to enter a group of an acceptable size before the battle starts, otherwise they will be expelled when the battle begins.

Acceptable group size and expulsion of invalid groups:

Description: This test aims to verify whether what is stated in R8 has been implemented. For this purpose, a battle with a minimum requirement > 1 is created and a student is added to the battle (hence, creating a new group with a single user).

Outcome: The system correctly expels the student from the battle upon its start.

[R11]: The battle starts when the timer set upon creation expires.

Battle starts upon reaching the registration deadline:

Description: The purpose of the test is to verify that when the current date and time exceed the registration deadline of the battle, the battle transitions to active status, allowing students to begin competing. To achieve this, a battle with a registration deadline close to the testing time is created, and we wait for the designated time interval before proceeding.

Outcome: The battle status correctly changes from "*Pre-battle*" to "*Battle*".

[R12]: The system creates a GitHub repository for the battle.

Creation of a GitHub repository for each battle:

Description: The purpose of the test is to verify whether after the creation of the battle the repository is created with the information entered by the educator during the creation of the battle.

Outcome: The platform successfully creates the repository with each battle and loads the information entered by the educator. We underline that it does not upload the battle code, as it does not allow the code to be uploaded during battle creation. If the system is unable to create the repositories, an error code appears, but the battle is still created.

[R13]: The system pushes in the appropriate GitHub repository the code KATA previously uploaded by the educator.

Pushing of the code that was previously uploaded by the educator:

Description: The purpose of the test is to verify that the educator can upload the code and that it is also uploaded to the GitHub repository created by the platform.

Outcome: The platform fails to meet this requirement because it lacks the functionality for educators to upload code to the corresponding GitHub repository during the battle. Only battle information such as the description and title are inserted, while the code is not uploaded.

[R13]: The system is notified about new pushes by the GitHub Action set up by the group leader.

The system receives notifications from GitHub:

Description: The purpose of this test is to verify that the system receives notifications from GitHub when the main branch of one of the groups registered in the battle changes. To verify this we created a group and forked the battle repository, making sure to set the link from the front-end. Afterward, we made a commit and pushed it to the main branch.

Outcome: The platform successfully receives the request sent from the GitHub action uploaded by the educators.

[R16]: The system evaluates the pushed solution giving a temporary score after each push.

[R24]: The system registers as the score obtained in a battle the last score calculated (which corresponds to the score of the last push on the repo).

After each push in the main branch of the group's repository, the code is automatically evaluated:

Description: The purpose of the test is to verify that the platform is able to evaluate the code at every push in the main repository of the group and therefore activate the GitHub action. To verify this, the battle repository is forked and a commit to the main branch is performed.

Outcome: The platform, as a prototype, lacks the ability to statically analyze code and automatically run tests. Additionally, it conducts the static analysis with the same evaluation parameters and does not allow educators to select them. Consequently, the final score for a group relies solely on manual evaluation.

[R17]: The ranking is shown at the end of the battle.

Visualization of the group's rank at the end of the battle:

Description: The purpose of the test is to verify whether you can see the ranking of the groups with their corresponding score acquired during the battle challenge. For this purpose, a battle is created and some students are added to the battle along with the links to their forked repositories. After the closing of the battle, the leaderboard is inspected.

Outcome: The platform displays the ranking of the battle groups with their corresponding scores and the links of their GitHub forks.

---

4. Code quality

The code is overall well structured and well written. The adoption of a microservice architecture inherently promotes strong separation between components, allowing for high maintainability and ease of updating the application.

The Spring framework's benefits and functionalities are leveraged very well. For example, interfaces were employed to decouple all services which introduces another layer of flexibility to the code.

Additionally, the extensive use of custom exceptions and annotations contributes to improving readability and maintainability by offering more specificity in error handling.

We ran a static analysis of the backend using IntelliJ to assess the quality of the code in many aspects and the results were very good. No errors or warnings were detected by IntelliJ apart from some unused properties defined in the application.properties files.

The main problem of the code is the lack of unit or integration tests. While there are some tests conducted on the Battle, Ranking, and Educator services, they don't fully cover all scenarios and aspects of the code to be considered comprehensive and sufficient.

5. Effort Spent

Bersani Michele	
Activity	**Effort spent**
Installation and setup	20min
Tests description	2h 40min
Total individual effort	**3h**

Chiappini Paolo	
Activity	**Effort spent**
Acceptance tests description and code quality	1h
Revisions and Acceptance test description	1h 40min
Total individual effort	**2h 40min**

Fraschini Andrea	
Activity	**Effort spent**
Code quality and static analysis	2h
Revisions	1h
Total individual effort	**3h**