

POLITECNICO

MILANO 1863

Acceptance Testing Document

CodeKataBattle

Authors

Name	Surname	ID
Alessandro	Saccone	11013852
Matteo	Sissa	10972783
Sara	Zappia	11016799

Version 1.0 - 06/02/2024

A.Y. 2023/2024

Contents

Contents	1
1 Introduction and Group Reference	3
2 Acronyms	4
3 Installation Set Up	5
4 Acceptance Testing	6
4.1 UC 1 & UC 2 - Registration and Log in	6
4.1.1 Setup of the environment	6
4.1.2 Test outcomes	6
4.2 UC3 - Create a Tournament	8
4.2.1 Setup Environment	8
4.2.2 Test Outcomes	8
4.3 UC4 - Create a Battle	10
4.3.1 Setup of the environment	10
4.3.2 Test outcomes	11
4.4 UC5 - Share permissions	12
4.4.1 Setup of the environment	12
4.4.2 Test Outcomes	12
4.5 UC6 - Tournament join	13
4.5.1 Setup of the environment	13
4.5.2 Test outcomes	13
4.6 UC7 - Join a Battle	14
4.6.1 Setup of the environment	14
4.6.2 Test Outcomes	14
4.7 UC8 - Join a team	15
4.7.1 Setup of the environment	15
4.7.2 Test outcomes	15
4.8 UC9 - Manual Evaluation	15
4.8.1 Setup of the environment	15
4.8.2 Test Outcomes	17
4.9 UC11 - Close tournament	18
4.9.1 Setup of the environment	18
4.9.2 Test Outcomes	18
4.10 UC12 - View battle rank	18

4.10.1	Setup of the environment	18
4.10.2	Test Outcomes	19
4.11	UC13 - View tournament rank	19
4.11.1	Setup of the environment	20
4.11.2	Test Outcomes	20
4.12	UC14 - View STU's profile	20
4.12.1	Setup of the environment	20
4.12.2	Test Outcomes	20
4.13	UC15 - View tournaments	20
4.13.1	Setup of the environment	20
4.13.2	Test outcomes	20
4.14	UC16 - View battles in tournament	21
4.15	Setup of the environment	21
4.15.1	Test Outcomes	21
4.16	UC 17 - Push commit and score updating	21
4.16.1	Setup of the environment	21
4.16.2	Test outcomes	24
4.17	UC 18 - Repository creation	25
4.17.1	Setup of the environment	25
4.17.2	Test outcomes	26
5	Effort spent	28
6	References	29

1 Introduction and Group Reference

In the following sections of this document, acceptance testing is performed on the artifacts for the CodeKataBattle application developed by

- Tommaso Pasini
- Elia Pontiggia
- Michelangelo Stasi

The following documents have been taken as a reference in order to design and tailor the test cases of this document to the specific characteristics of the software artifact:

- Requirements Analysis and Specification document of the CodeKataBattle project.
- Design Document of the CodeKataBattle project.
- Original assignment of the stakeholders.
- Implementation and Testing document (especially for installation purposes).

All the document can be found in the project repository at <https://github.com/pontig/sw-eng-2-PasiniPontiggiaStasi>

2 Acronyms

The following acronyms can be used throughout this document and their meaning is here exemplified:

- RASD stands for Requirements Analysis and Specification Document.
- CKB stands for CodeKataBattle.
- ED stands for EDucator.
- ST stands for STudent.
- DD stands for Design Document.
- ITD stands for Implementation & Testing Document.

3 Installation Set Up

The installation of the software final product is sufficiently easy and convenient. Nevertheless, there are some things that it's important to point out in order to complement the information provided by the authors in the ITD document.

First off, the CKB application performs static analysis thanks to an external service called SonarQube. In order to make the CKB platform and the SonarQube server interact seamlessly, it is fundamental to have installed the SonarQube server locally and to start it before the CKB application. The SonarQube server has to listen for incoming requests on the default port 9000. When initiating the CKB application, there is a prompt asking for the absolute location of the StartSonar file, which is the script provided by SonarQube to start the server. The problem is that independently on the path that is passed to the CKB instance in this circumstance, the StartSonar script is always searched in a default location "C:\sonarqube\bin\windows-x86-64\StartSonar.bat". Therefore, the SonarQube instance has to be started externally.

Then, there is no specification on the ITD document of which is the local port the CKB instance is bound to. It is the default 8080 for Spring applications.

4 Acceptance Testing

To make the Acceptance Testing structured and organized, it has been decided to try out how the CKB application works in the various Use Cases described in the RASD document that comes with the application. For each of the following tests, an initial description of the setup for the test environment is provided, as well as the process and outcomes.

4.1 UC 1 & UC 2 - Registration and Log in

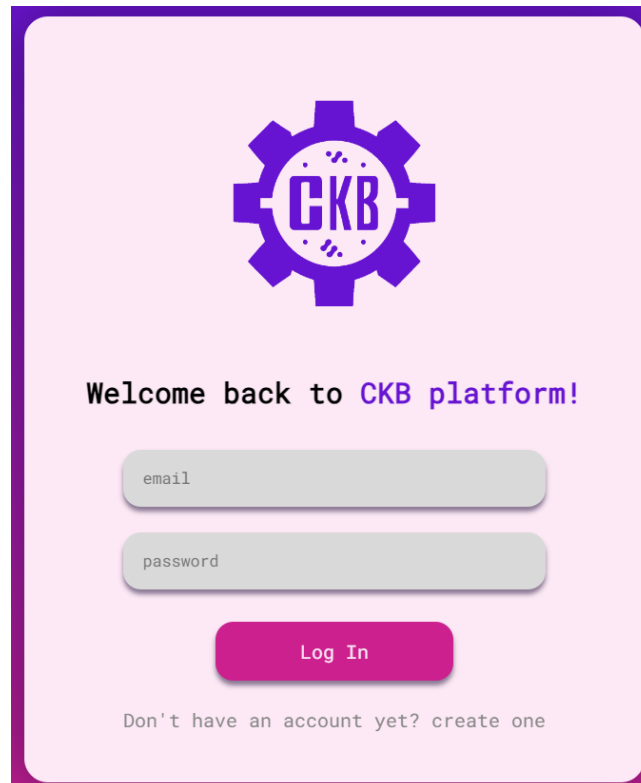


Figure 1: Login view of the web application

4.1.1 Setup of the environment

The login and registration functions were tested through the User Interface, connecting to the url: http://localhost:8080/ckb_platform and interacting with the login/register interface.

4.1.2 Test outcomes

Among the positive aspects, we can highlight:

- The system correctly prevents the insertion of an email without '@' or incomplete form submissions.
- The system effectively prevents the creation of duplicate accounts with the same email, thereby avoiding situations where a user might attempt to simultaneously register as both an educator and a student.

- Accounts with the same name and surname but different emails can be created successfully.
- Passwords must correspond in the two password fields.
- The system appropriately redirects users to the login page if they attempt path traversal with a URI of a signed-in user.
- Error handling for incorrect login parameters is properly managed.
- The cascade deletion policy prevents the deletion of a user with associated records in other tables, such as created tournaments.

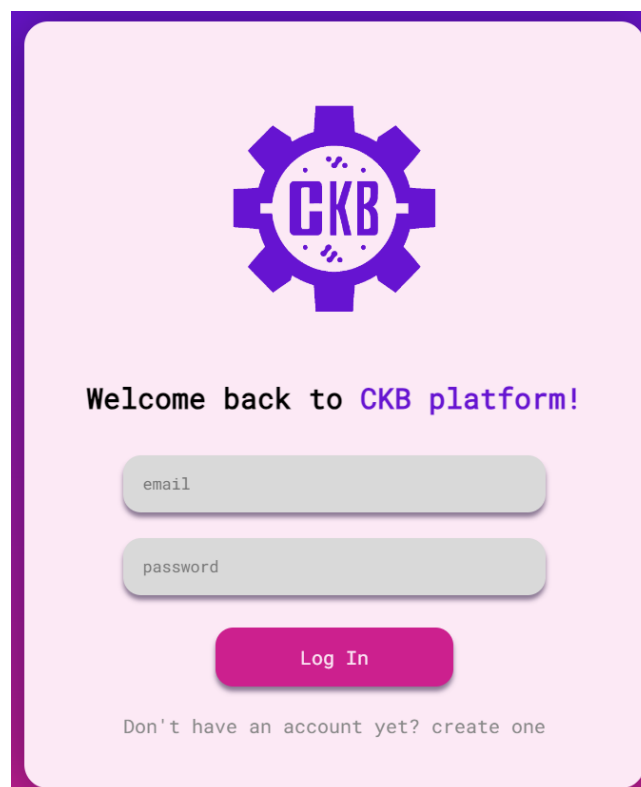


Figure 2: Sign up view of the web application

However, there are areas where improvements could have been made, including:

- It would have been beneficial to confirm the existence of an email by requiring confirmation during the registration process (as demonstrated further in the following description).
- Password strength is not verified, and it can be as short as one character.
- Clicking on the "Don't have an account yet" link and then attempting to return back results in a 404 error.

4.2 UC3 - Create a Tournament

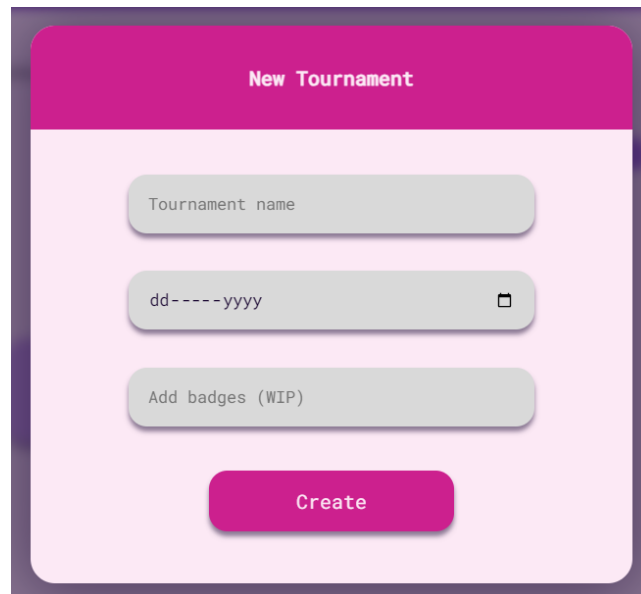
The image shows a mobile app interface for creating a new tournament. It features a pink header with the text "New Tournament". Below the header, there are three input fields: "Tournament name", "dd-----yyyy" (with a calendar icon), and "Add badges (WIP)". At the bottom, there is a pink "Create" button.

Figure 3: Form for the creation of a new tournament

4.2.1 Setup Environment

This feature was tested through the UI logging in as an Educator. The system displays a button for creating a new tournament in the educator's dashboard. We created a new tournament filling out the form in figure 3.

4.2.2 Test Outcomes

Among its positive aspects are:

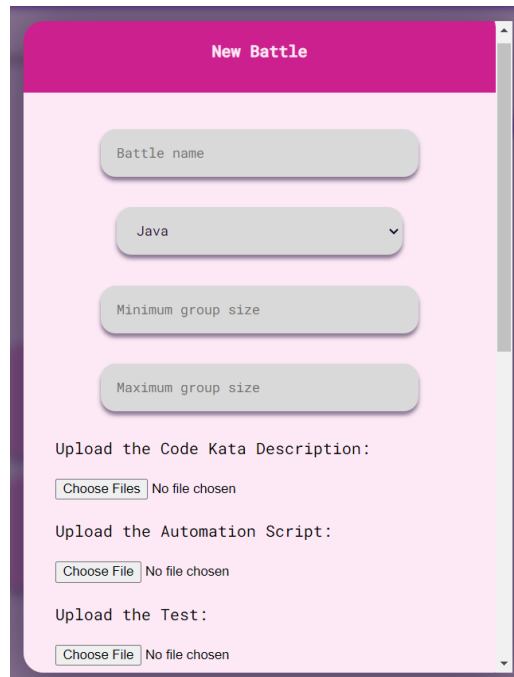
- The system effectively manages situations where users attempt to insert a date prior to today or with a year exceeding four digits.
- After creation, emails are correctly sent to the provided email address during registration, if it exists.
- In case of providing an incorrect email during registration, an error is displayed in the console, although it does not disrupt the application's functionality.

However, there are also some minor issues that could be improved in future versions of this prototype:

- After clicking on "View All Tournaments", to only see one's own tournaments requires navigating to the "Dashboard" which is not clearly visible or understandable.
- When creating a tournament, it is not indicated for what purpose the date requested in the form is required (as can be seen in Figure 3).

- The system does not allow the creation of a tournament with the same name, although the specifications did not require this restriction.
- The remaining time is displayed in a format that is not easily recognizable.
- Due to the lack of validation for email existence during registration, an error is displayed in the console when an incorrect email has been provided while trying to send a message to it.

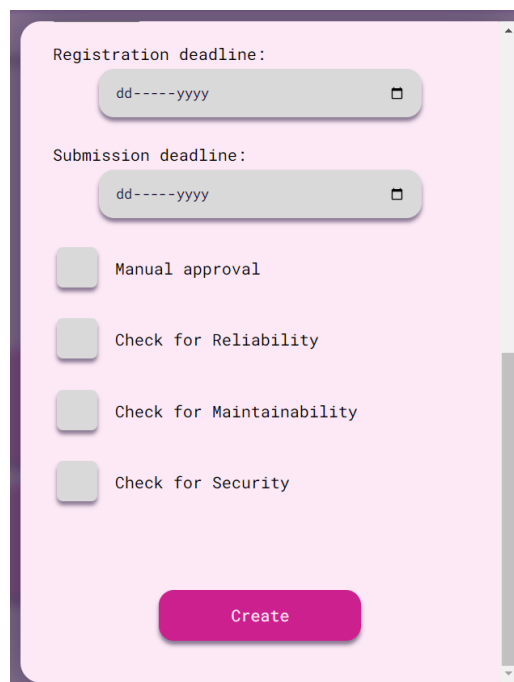
4.3 UC4 - Create a Battle



The screenshot shows the 'New Battle' form with a pink header. The form contains the following fields and sections:

- Battle name:** A text input field.
- Language:** A dropdown menu currently showing 'Java'.
- Minimum group size:** A text input field.
- Maximum group size:** A text input field.
- Upload the Code Kata Description:** A section with a 'Choose Files' button and the text 'No file chosen'.
- Upload the Automation Script:** A section with a 'Choose File' button and the text 'No file chosen'.
- Upload the Test:** A section with a 'Choose File' button and the text 'No file chosen'.

Figure 4: Form for the creation of a battle - first section



The screenshot shows the second section of the 'New Battle' form. It contains the following fields and options:

- Registration deadline:** A date input field with the placeholder 'dd-----yyyy' and a calendar icon.
- Submission deadline:** A date input field with the placeholder 'dd-----yyyy' and a calendar icon.
- Manual approval:** A checkbox.
- Check for Reliability:** A checkbox.
- Check for Maintainability:** A checkbox.
- Check for Security:** A checkbox.
- Create:** A pink button at the bottom of the form.

Figure 5: Form for the creation of a battle - second section

4.3.1 Setup of the environment

To test this use case we logged in as an Educator, created a tournament, and clicked on the **New Battle** button inside the tournament view page. The system displayed a form to allow the educator to upload

a description file, a build automation script file and a Test file; and also to be filled there are the battle's name, the code language of the challenge, minimum and maximum of participants, registration and submission dates, manual evaluation and the aspects to be considered for static analysis, as shown in figures 4 and 5.

4.3.2 Test outcomes

The create battle use case, like others involving a creation form, is well-built and executed. Regarding correctness, we can highlight the following aspects:

- The "create battle" button is correctly displayed only for educators authorized to create battles in the considered tournament.
- The system verifies if all fields are filled correctly before creating the battle.

Despite these two basic features, some minor improvements can be made, such as:

- Allowing the addition of different types of files for the Description, as currently, it only accepts PDF files, limiting the options for updating.
- After creating the battle, the form remains on the screen, requiring manual closure.

4.4 UC5 - Share permissions

4.4.1 Setup of the environment

To test this use case we logged in as an educator, opened the tournament view page and clicke on the **Share permission** button, that authorizes other educators through their email.

4.4.2 Test Outcomes

The share permission use case is very intuitive and well thought out. Its basic functions work correctly, so when requesting permission to someone, the email is correctly sent (if it exists), and the other educator now has access to the tournaments accordingly.

Among the things that can be improved, we include:

- When searching for a colleague and there are namesakes, it is not clear the difference between them. It could be better to associate the email (and display it) next to the name.
- According to the Requirements and Specifications Document (RASD) only the creator can close the tournament, while here also the educator with permissions can do so.

4.5 UC6 - Tournament join

4.5.1 Setup of the environment

The **UC6** is tested manually through the UI. When logged in as a student is possible to select a tournament and view the tournament home page. From the tournament home page is possible to subscribe through a subscribe button.

4.5.2 Test outcomes

The system successfully :

- process the request and confirms the enrollment
- redirects to the tournament view page when clicking on a tournament link sent by email from the system
- displays a human-readable error message when a student tries to enroll when the deadline is expired
- displays a human-readable error message when a student tries to enroll in a tournament he is already enrolled in

4.6 UC7 - Join a Battle

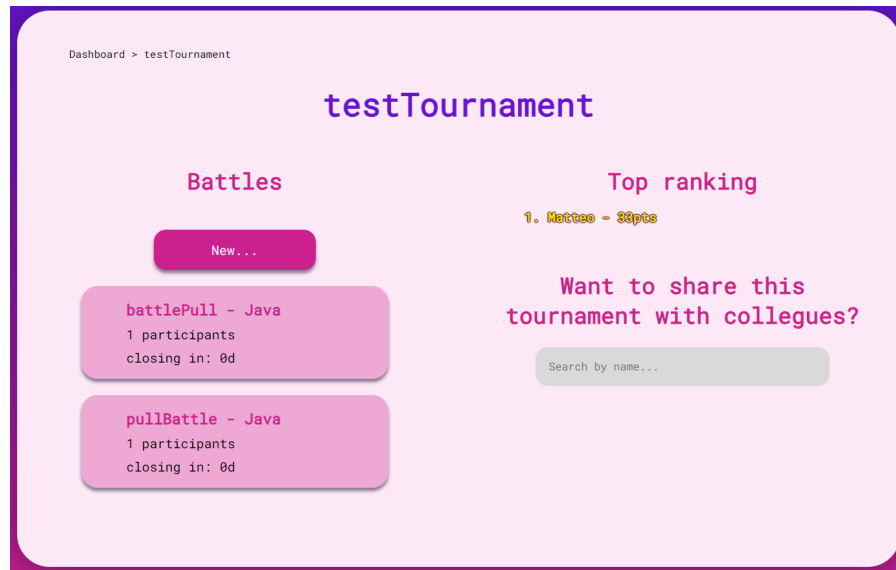


Figure 6: Tournament Homepage

4.6.1 Setup of the environment

The UC7 use case was tested through the UI login-in as a student, subscribing to a tournament and then clicking on the button **Join Battle** in the tournament home page.

4.6.2 Test Outcomes

While subscribing to a battle, it is important to ensure that features involving the inviting of others work properly, as the subscription function itself is correctly implemented. There are few errors that could be found among the invitation systems, including:

- When attempting to invite others with a name, the system asks for an email, which may not exist (in such cases, the invitation will simply expire).
- The system does not allow the creation of a team with the same name, even in different tournaments.

Additionally, there are well-designed features, such as:

- When providing an email that belongs to an educator instead of a student, an error is correctly displayed, and the situation is avoided.
- If the same email as the user who is creating the team is entered, an error message is displayed, stating that you can't invite yourself.
- When attempting to create a team with a name that is already taken, the system correctly displays an error message and does not subscribe the user to the battle.

4.7 UC8 - Join a team

4.7.1 Setup of the environment

The UC8 use case has been tested manually through the UI. We logged-in as a student, joined a battle, and the User Interface displayed a button for inviting other students by filling out their email. After pushing the button an email is was to all the selected students.

4.7.2 Test outcomes

The application successfully allows to :

- invite students not subscribed to the battle yet
- invite students whoa already joined the battle alone

The only discordance found with the RASD document we thought should have been explained is that:

- the student is added to the team when clicking on the link, even if this is not explained in the email text
- a window is not displayed in the invited student dashboard page where he can accept or decline the invitation, as written in the RASD

4.8 UC9 - Manual Evaluation

4.8.1 Setup of the environment

In order to test the Manual Evaluation of an Educator, the following actions have been taken:

- An educator with an account on the system has created a new tournament and a battle (with language Java) inside the tournament. During the battle creation, the checkbox for manual evaluation is selected. The tournament will be referred to as testTournament, while the battle as testBattle. The battle build script is the one that can be found at the end of this bullet list, as well as the test class.
- A student with an account on the system has subscribed to the tournament and the battle.
- Once the registration deadline for the battle passes (and the CKB platform correctly creates the GitHub repository), the student:
 - Forks the repository on his personal account.
 - Clones the repository locally.
 - Codes a very simple solution to the kata battle (a simple Main class that passes the test from the test class).
 - Adds the ".github/workflows" directory and the wokflow to make GitHub notify the CKB instance. The workflow can be found at the end of this bullet list.
 - Pushes the result.

Build Script

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
4         org/xsd/maven-4.0.0.xsd">
5
6     <modelVersion>4.0.0</modelVersion>
7
8     <groupId>org.problem</groupId>
9     <artifactId>Main</artifactId>
10    <version>1.0-SNAPSHOT</version>
11
12    <properties>
13        <maven.compiler.source>1.8</maven.compiler.source>
14        <maven.compiler.target>1.8</maven.compiler.target>
15    </properties>
16
17    <dependencies>
18        <dependency>
19            <groupId>junit</groupId>
20            <artifactId>junit</artifactId>
21            <version>4.12</version>
22            <scope>test</scope>
23        </dependency>
24    </dependencies>
25
26    <build>
27        <plugins>
28            <plugin>
29                <groupId>org.apache.maven.plugins</groupId>
30                <artifactId>maven-surefire-plugin</artifactId>
31                <version>3.0.0-M5</version>
32                <configuration>
33                    <includes>
34                        <include>/**/*.Test.java</include>
35                    </includes>
36                </configuration>
37            </plugin>
38        </plugins>
39    </build>
40</project>
```

4AcceptanceTesting/buildScript.xml

Test Class

```
1 import org.junit.Test;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class MainTest
6 {
7     @Test
8     public void testApp(){
9         Main myApp = new Main(2);
10        myApp.addOne();
11        assertEquals("Error", 3, myApp.score);
12    }
13 }
```

4AcceptanceTesting/MainTest.java

Github Workflow

```
name: Pull Request Notification
on:
  push:
    branches:
      - main
jobs:
  notify:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v2
      - name: Notify Endpoint
        run: |
          team_name="myTeam"
          payload="{\"pusher\": \"${{ github.actor }}\", \"team\": \"
            $team_name\", \"repository\": \"${{ github.repository }}\" }"
          curl -X POST -H "Content-Type: application/json" -d "$payload"
            https://8618-176-206-223-189.ngrok-free.app/ckb_platform/battle/
            pulls
```

4AcceptanceTesting/work.yaml

4.8.2 Test Outcomes

Once the student pushes the solution of the code kata battle on GitHub, the workflow correctly fires the notification from the remote GitHub machine to the CKB instance. Unfortunately, the score computation performed by CKB is partially achieved, due to the errors encountered and illustrated in the use case 17 (below). In any case, it was still open the possibility of verifying whether the

manual evaluation of the educator was correctly working or not. When the submission deadline for the testBattle comes (at midnight of the day set as the submission deadline), the educator logs in his account and navigates to the testBattle page. It is possible to see from the following image that the battle is marked with the writing "waiting for evaluation", but there is no section or action that the educator can take in order to start the procedure and then close the battle. Very likely, the error that CKB encounters during the score computation phase also prevents the system from storing the source code of the students somewhere (either in the DB or in a local repository) and allowing the educator to visualize and assess the results.

4.9 UC11 - Close tournament

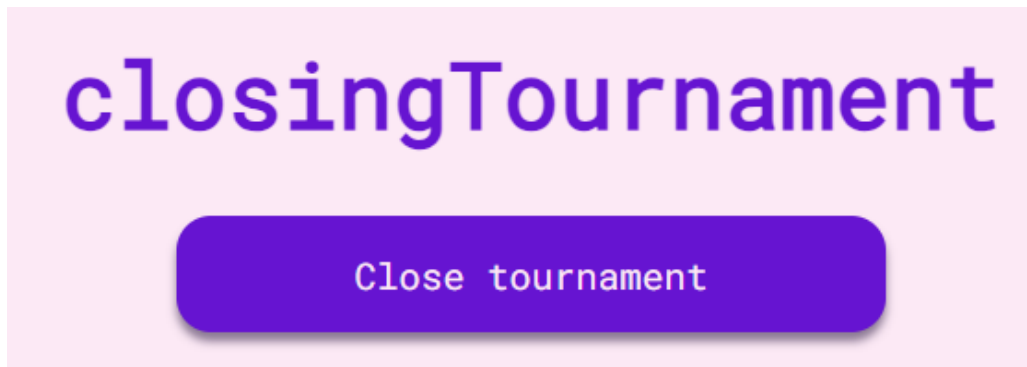


Figure 7: Battle Home Page

4.9.1 Setup of the environment

The **UC11** was tested through the User Interface. We logged in as an Educator and selected a managed tournament from the dashboard page. In the tournament view page a **Close tournament** button should be displayed, allowing for the closure of the tournament.

4.9.2 Test Outcomes

The acceptance test was successfully passed as the system :

- closes the tournament when clicking on the button
- displays a human-readable message when closing a tournament that is already closed
- sends an email to all the students subscribed to the tournament when it is closed

4.10 UC12 - View battle rank

4.10.1 Setup of the environment

The **UC12** was tested manually through the UI. We logged-in as a student and subscribed to a tournament and a battle relative to that tournament. We also tested the feature logging-in as an educator, and clicking on a created tournament from the dashboard and then on a specific battle.



Figure 8: Battle Home Page

4.10.2 Test Outcomes

- The system displays the battle rank on the right, both for students and educators
- if the battle hasn't started yet the rank is empty

4.11 UC13 - View tournament rank

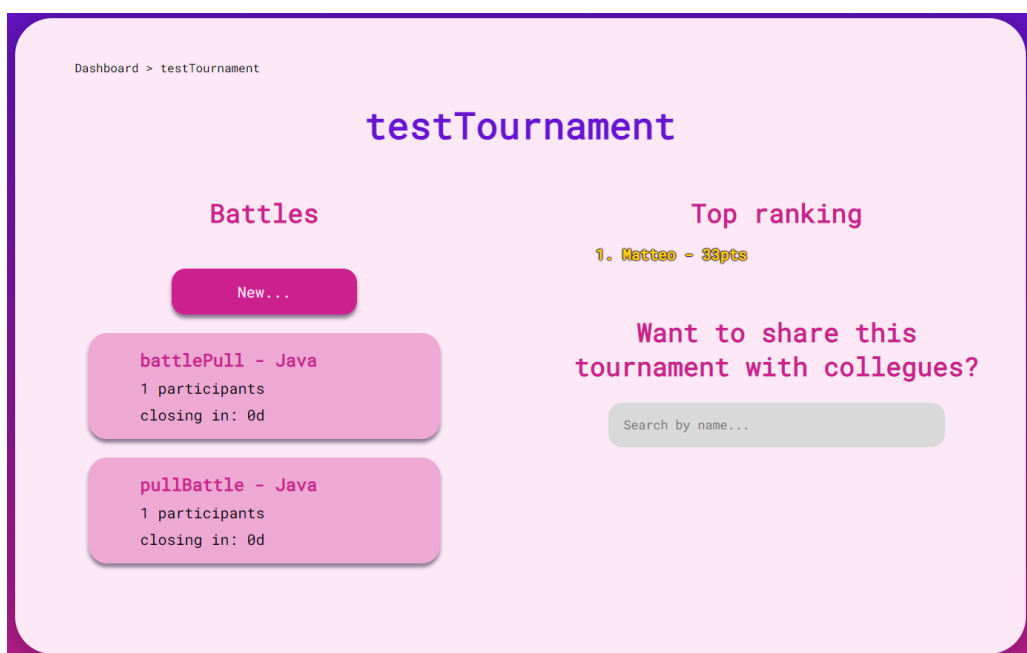


Figure 9: Tournament Home Page

4.11.1 Setup of the environment

The use case was tested through the User Interface, logging in first as a student and then as an educator, as the rank should be displayed to both users, and clicking on a tournament from the dashboard. When clicking on a tournament the tournament view page should be displayed, showing also the tournament rankings.

4.11.2 Test Outcomes

The test is passed successfully, the rank is displayed on the right both for students and educators, as shown in figure 8.

4.12 UC14 - View STU's profile

4.12.1 Setup of the environment

The use case was tested through the User Interface. We logged in as a student and searched another user username through the search bar. When the username was found we clicked on it. The system should redirect to the student profile view, where the list of tournament he is enrolled in should be displayed

4.12.2 Test Outcomes

The system successfully passed the test, the student view page is displayed correctly as well as the tournaments he is subscribed in. Also is possible to access a student view page not only by searching is username through the search bar but also clicking on a student username present in a tournament rank.

4.13 UC15 - View tournaments

4.13.1 Setup of the environment

The use case was tested through the User Interface. We logged in first as a student and then as an educator as the tournament view page should be displayed both to students and educators. From the dashboard we searched for a tournament, that can be closed or not, and we clicked on it.

4.13.2 Test outcomes

The system successfully passed the test, as it was possible to:

- access the tournament view page as an educator clicking on one of the managed tournaments in the dashboard
- access the tournament view page as an educator clicking the **show all** button and searching a tournament by name
- access the tournament view page as a student clicking on a tournament I am subscribed to from the dashboard

- access the tournamentview page as a student clicking the **show all** button and searching a tournament by name

4.14 UC16 - View battles in tournament

4.15 Setup of the environment

The use case was tested through the UI logging in as a student and then as an educator, as the the list of battles within a tournament should be displayed to both users.

4.15.1 Test Outcomes

The test was successfully passed, the list of battles was correctly displayed in the tournament view page. This use case doesn't have any exception to handle so the test is passed.

4.16 UC 17 - Push commit and score updating

4.16.1 Setup of the environment

This is probably the most troublesome test to run, as it requires many components to collaborate together. First off, it is assumed in this test that an educator has created a battle, which has been called in this testing environment as "pullBattle", and a student with an account on CKB has already subscribed to the battle. The name of the user on the platform is "Matteo Sisa", registered on CKB with the email "matteo.sissa@mail.polimi.it". Once the GitHub repository for pullBattle is successfully created by CKB, the user Matteo Sissa logs in his personal GitHub account and forks the repository from the official CKB account "CodeKataBattlePlatform". Then, the forked repository is cloned locally, in order to work on it. As required by the instructions, a workflow is added to the repository under the directory ".github/workflows". This is its content:

```

name: Pull Request Notification
on:
  push:
    branches:
      - main
jobs:
  notify:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v2
      - name: Notify Endpoint
        run: |
          team_name="myTeam"
          payload="{\"pusher\": \"${{ github.actor }}\", \"team\": \"
            $team_name\", \"repository\": \"${{ github.repository }}\" }"
          curl -X POST -H "Content-Type: application/json" -d "$payload"
            https://8618-176-206-223-189.ngrok-free.app/ckb_platform/battle/
            pulls

```

4AcceptanceTesting/work.yaml

As recommended by instructions, the name of the team has been substituted with the name of the team "myTeam". Moreover, an instance of ngrok has been initiated on the laptop in order to provide a public URL to the remote GitHub machine where the GitHub Actions workflow is going to be executed every time a push occurs on the repository. This public URL is necessary as long as the CKB instance only runs locally (on "localhost:8080").

A simple java class is added to the folder under "src/main/java", coding a solution to the CKB problem. More specifically, this is the simple java test class that has been inserted by the educator at battle creation time:

```

1 import org.junit.Test;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class MainTest
6 {
7     @Test
8     public void testApp(){
9         Main myApp = new Main(2);
10        myApp.addOne();
11        assertEquals("Error", 3, myApp.score);
12    }
13 }

```

4AcceptanceTesting/MainTest.java

The java class coded by the user and added to the repository is a simple class that "complies" to the test class (to pass all tests).

```
1 public class Main {  
2  
3     int score;  
4  
5     public Main(int score){  
6         this.score = score;  
7     }  
8  
9     public int addOne(){  
10        score += 1;  
11        return score;  
12    }}
```

4AcceptanceTesting/Main.java

The directory structure described in the README.md file of the GitHub repository of the educator is also respected, as there are:

- A ".github" folder for the GitHub workflows (as explained above).
- A "CKBProblem" folder in which only the .pdf file with the battle description is present.
- The "JavaProject" folder, in which it is possible to find a "buildScript.xml" file which is the build script, and the usual directory structure of a Maven project with the source code under "src/main/java"

The build script is the following file (a simple Maven pom.xml file):


```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
   org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5
6   <groupId>org.problem</groupId>
7   <artifactId>Main</artifactId>
8   <version>1.0-SNAPSHOT</version>
9
10  <properties>
11    <maven.compiler.source>1.8</maven.compiler.source>
12    <maven.compiler.target>1.8</maven.compiler.target>
13  </properties>
14
15  <dependencies>
16    <dependency>
17      <groupId>junit</groupId>
18      <artifactId>junit</artifactId>
19      <version>4.12</version>
20      <scope>test</scope>
21    </dependency>
22  </dependencies>
23
24  <build>
25    <plugins>
26      <plugin>
27        <groupId>org.apache.maven.plugins</groupId>
28        <artifactId>maven-surefire-plugin</artifactId>
29        <version>3.0.0-M5</version>
30        <configuration>
31          <includes>
32            <include>/**/*.Test.java</include>
33          </includes>
34        </configuration>
35      </plugin>
36    </plugins>
37  </build>
38 </project>

```

4AcceptanceTesting/buildScript.xml

4.16.2 Test outcomes

This is what happens when the modified folder is pushed on the GitHub repository of the user Matteo Sissa:

- The CKB platform receives the notification from GitHub.
- The score is not updated.

Unfortunately, there is something going wrong in the process. By inspecting the source code and the messages from the terminal, the following error has been encountered: The application seems to be looking at the file "MainTest.txt" under the path "C:\Users\matte\Desktop\sw-eng-2-PasiniPontiggiaStasi\ITD\CodeKataBattle\fileStorage\PullFiles\12\27\sissamatteo29-pullBattle-31da964\Java\target\surefire-reports\MainTest.txt". But an error reports that the file cannot be found. Maybe some compilation errors throughout the process do not produce the expected result.

Other folder structures have been tried out in order to make the CKB platform work correctly, for instance:

- A root package "org.problem" has been added to the under the "java" source folder in both the "main" and "test" repositories. But the code reported the same error.
- The "buildScript.xml" has been renamed "pom.xml" to make sure Maven was able to find it, but the code reported the same error.

From a user perspective (so without delving deeper into the source code) it is not possible to jump to any conclusion regarding this part of the project, the errors encountered have been reported.

Besides, as a consequence of this, the interaction with the SonarQube server couldn't be fully tested from a user perspective, as it was a direct consequence of the push action of the user on his GitHub repository.

4.17 UC 18 - Repository creation

4.17.1 Setup of the environment

Testing the Repository creation is not an easy task for this CKB application, for the simple reason that CKB creates the repository at midnight of the day of the registration deadline for a specific battle. Therefore, the CKB instance was initiated locally, and a new battle created with a registration deadline coming on the day following the one of the creation. In order to make the creation of the repository successful, it was mandatory to supply to the CKB application a valid set of files for the battle, more specifically the creation of the battle has been tested in the case of Java applications. On the GUI for the creation of the battle, the following data has been provided to the CKB instance:

tryGit

Java

1

3

Upload the Code Kata Description:

Choose Files desc.pdf

Upload the Automation Script:

Choose File pom.xml

Upload the Test:

Choose File AppTest.java

Registration deadline:

This image shows the fact that:

- The Code Kata Description has been provided as a pdf file (otherwise the insertion generates an error)
- The build automation script is an XML file for a java application. In this case, it is a Maven file specifying the parameters of the Surefire plugin for testing.
- The test cases are a simple .java file in which the tests have been designed with the help of JUnit.

As previously said, the registration deadline has been set to the midnight of the following day.

4.17.2 Test outcomes

Once the midnight is passed, the CKB application correctly generates the GitHub repository and sends the link to all the subscribed students. CKB also takes care of structuring the GitHub repository in a specific way (with a specific directory structure). This is a screenshot taken from GitHub after the creation of the repository for the battle presented above:

CodeKataBattlePlatform Create JavaProject folder 8f97a47 · 1 hour ago 5 Commits		
CKBProblem	Create CKBProblem folder	1 hour ago
JavaProject	Create JavaProject folder	1 hour ago
README.md	Create folder	1 hour ago

Inside "CKBProblem" the original description of the educator can be found, and in the folder "JavaProject" there is a standard Maven project in which the tests supplied by the educator are added in the correct location (under src/test folder).

5 Effort spent

Studente	Hours spent
Alessandro	8
Matteo	8
Sara	8

Table 1: Time spent on Acceptance Testing

6 References