POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

ATD Document

Author: **Mattia Piccinato, Gabriele Puglisi, Jacopo Piazzalunga**

Academic Year: 2023-24
Link to the repository: Click here

Contents

Contents **i**

1 Tested Project **1**

2 Installation **3**

3 Test cases **5**

3.1 Premise 5

3.2 Testing 5

3.3 Further notes 16

4 Effort spent **17**

1 | Tested Project

- **Authors:**

  Polito Attilio
  Rigione Pisone Raimondo
  Soricelli Francesco

- **Github Repository:**

  Click here to see the repository

- **Reference Documents:**
  RASD: Click here
  DD: Click here
  ITD: Click here

2 | Installation

- **1.** We utilized a Windows Sandbox as our testing environment.
- **2.** We cloned the repository.
- **3.** We moved the folder "IT/Implementazione" to the desktop.
- **4.** We installed Python version 3.10.0 on the machine.
- **5.** We added the path to python.exe to the PATH environment variable.
- **6.** We opened the Command Prompt.
- **7.** We navigated to the "Implementazione" folder on the Desktop.
- **8.** We executed the following commands:

```
python -m venv env
env\Scripts\activate
pip install django
pip install django-crispy-forms
pip install github
pip install PyGithub
pip install flake8
cd codekatabattle
python manage.py runserver
```

- **9.** We accessed the login interface at http://127.0.0.1:8000/ckbapp/login.
- **10.** We registered on ngrok.com.
- **11.** We downloaded the Windows x64 zip folder.
- **12.** We extracted the contents of the zip folder.

- **13.** We opened the ngrok.exe terminal located within the zip folder.
- **14.** Using the token provided immediately after clicking the download button, we executed the following commands:

```
ngrok config add-authtoken <token>
ngrok http http://localhost:8000
```

- **15.** We copied the public endpoint, navigated to "IT/Implementazione/codekatabattle/-codekatabattle", and replaced the endpoint URL in the settings.py file as specified in the installation guide.

The installation process proceeded smoothly without any issues.

---

3 | Test cases

3.1. Premise

It is our responsibility to specify that, upon starting the server locally, some issues arose with the display of the contents, resulting in a subsequent loss in terms of usability.

Specifically, the pages would repeatedly redirect to themselves until manually interrupted. Upon clicking, the browser would display the page's HTML without CSS.

Although, during testing, we observed that after extensive navigation, the server began to display the pages correctly.

However, when we requested the developers to grant us online access to a functional version of the application, we were provided with a version that did not exhibit this problem in any way. In conclusion, we advise the development team to double-check the versioning indicated in the installation guide.

3.2. Testing

This section will outline the testing procedures we conducted to evaluate both functionality and adherence to the project's product functions as outlined in the provided RASD.

Each test case was meticulously designed to encompass the most relevant usage scenarios from the user's perspective.

Below is a comprehensive list of the test cases executed to validate the product's performance and compliance. For each, we have stated the goal, the steps performed, the cases considered, and the results obtained.

- **F1 - Registration and Login**

  1. Goal:

     Verify that the registration and login processes work as expected, adequately addressing security measures while ensuring a seamless user experience in terms of usability and accessibility.

  2. Steps:

     1. Create a Student according to the Domain Assumption D10
     2. Login
     3. Logout
     4. Create an Educator
     5. Login
     6. Logout

  3. Test cases:

     (a) Correct flow

     (b) Already existing username

     (c) Username case-sensitivity

     (d) Invalid email field

     (e) Weak trivial password

  4. Results:

     The test results indicate that the registration and login processes of the app generally proceed smoothly, exhibiting user-friendly and secure functionalities.

     However, there is a lack of enforcement regarding password strength, as weak passwords such as "a" were accepted during registration.

- **F2 - Creation, Management and Closure of Tournaments**

  1. Goal:

     Verify that the Educators are indeed able to create and close a Tournament, and creating Battles in the context of a Tournament.

2. Steps:

    1. Login as an Educator
    2. Create a Tournament
    3. Create a Battle for the Tournament
    4. Log out
    4. Access the database with administration privilege
    5. Move up the subscription deadline of the Tournament
    6. Move up thesubscription deadline of the Battle
    7. Go back to the login interface
    8. Log in as a Student
    9. Visualize the Battle repository link in the Battle's status page
    10. Logout
    11. Login as the Educator who created the Tournament
    12. Close the Tournament

3. Test cases:

    (a) Correct flow

    (b) Invalid maximum number of students

    (c) Any deadline in the past

    (d) Subscription deadline after the submission deadline

    (e) Submission deadline after the Tournament submission deadline

    (f) Wrong file format

4. Results:

    The test results suggest that the process of creating, managing, and closing tournaments generally runs smoothly, with secure functionalities in place.

    However, there's no validation on the file format, which is expected to be a Python file according to Chapter 5 in the ITD. This oversight could lead to problems when creating the GitHub repository for the Battle.

    Furthermore, when invalid parameters are provided in the request, the error visualization on the client-side isn't always clear. For instance, when a negative maximum number of students is specified.

- **F3 - Delegation of Tournament Management**

  1. Goal:

     Verify that an Educator who manages a Tournament can delegate the management of such Tournament to another Educator.

  2. Steps:

     1. Login as Educator A
     2. Create a Tournament
     3. Delegate Educator B
     4. Logout
     5. Login as Educator B
     6. Create a Battle in the Tournament
     7. Delegate Educator C
     8. Logout
     9. Login as Educator C
     10. Create a Battle in the Tournament
     11. Access the database with administration privilege
     12. Move up the subscription deadline of the Tournament
     13. Go back to the Educator interface
     14. Close the Tournament

  3. Test cases:

     (a) Correct workflow

     (b) Delegating the Educator itself

     (c) Delegating an Educator who already manages the Tournament

     (d) Delegating a Student

     (e) Delegating a User who does not exist

  4. Results:

     The test results confirm that Educators can indeed delegate the management of a Tournament they oversee to other Educators, but they cannot delegate it to Students.

Additionally, the User is appropriately notified if they attempt to delegate an Educator who already has management privileges, ensuring clarity and preventing inadvertent actions that may lead to confusion or errors in the system.

However, it also results that Educators who are allowed by the creator to manage a Tournament can only create Battles, as they cannot access the Tournament status page, making it impossible for them to close the Tournament.

- **F4 - Exploration of Coding Challenges**

  1. Goal:

     Verify that a Student can view all available Tournaments and Battles for subscription, ensuring they have access only to those that are relevant to their eligibility or criteria.

  2. Steps:

     1. Login as Educator
     2. Create a Tournament 1
     3. Create a Tournament 2
     4. Create a Battle in Tournament 2
     5. Logout
     6. Access the database with administration privilege
     7. Move up the subscription deadline of Tournament 1
     8. Go back to the login interface
     9. Login as Student
     10. Visualize available Tournaments
     11. Enroll in Tournament 2
     12. Visualize available Battles in Tournament 2
     13. Access the database with administration privilege
     14. Move up the submission deadline of Tournament 2
     15. Go back to the Student interface
     16. Visualize available Tournaments

  3. Test cases:

     (a) Correct flow

4. Results:

   The test results indicate that students can visualize all the information they should be able to.

   Additionally, they do not see expired Tournaments and Battles that they can no longer enroll in. Furthermore, they continue to see Battles that they have participated in, maintaining a consistent and relevant user experience.

- **F5 - Participation in Coding Battles**

1. Goal:

   Verify that Students can take part in Battles and Tournaments which are in their subscription phase.

2. Steps:

   1. Login as Educator
   2. Create a Tournament
   3. Create a Battle in the Tournament
   4. Logout
   5. Login as Student
   6. Enroll in the Tournament
   7. Enroll in the Battle

3. Test cases:

   (a) Correct flow

   (b) Invalid Tournament id

   (c) Invalid Battle id

4. Results:

   The test results indicate that users can effectively and exclusively enroll in tournaments and battles that they should be allowed to participate in.

   However, we advise the development team to always refresh the page and display an error message to the User in case of a bad request, in order to provide clear guidance and enhance the User experience.

- **F6 - Team Formation**

  1. Goal:

     Verify that Students who have enrolled in a Battle are allowed to form a team by inviting other Students, up to the maximum specified limit.

  2. Steps:

     1. Login as Educator
     2. Create a Tournament
     3. Create a Battle in the Tournament with maximum 3 Students per team
     4. Logout
     5. Login as Student A
     6. Enroll in the Tournament
     7. Enroll in the Battle
     8. Invite 2 Students B and C
     9. Logout
     10. Login as Student B
     11. Enroll in the Tournament
     12. Enroll in the Battle
     13. Accept the invite
     14. Logout
     15. Login as Student B
     16. Enroll in the Tournament
     17. Enroll in the Battle
     18. Accept the invite

  3. Test cases:

     (a) Correct flow

     (b) Invite more Students than allowed

     (c) Invite a Student as an invited Student

     (d) Invite a Student who is already in a team for the Battle

     (e) Invite a Student who is not enrolled in the Tournament

4. Results:

   The test results confirm that students who participate in a battle can successfully and exclusively form teams comprising up to the maximum number of students. These students must be enrolled in the tournament and not yet enrolled in the battle.

   However, the frontend appears to display a "useless" search bar to Students who have joined a Battle by invitation, despite the system's restriction on their ability to send invitations. It would be more beneficial if this search bar were not presented, thereby eliminating confusion and streamlining the User experience.

- **F7 - Real-time Competition and Scores**

  1. Goal:

     Verify that results of each commit for a Battle are updated in real-time.

  2. Steps:

     1. Login as an Educator
     2. Create a Tournament
     3. Create a Battle
     4. Logout
     5. Login as a Student
     6. Enroll in the Tournament
     7. Enroll in the Battle
     8. Access the database with administration privilege
     9. Move up the subscription deadline of the Tournament
     10. Move up the subscription deadline of the Battle
     11. Go back to the Student interface
     12. Click on the link of the repository
     13. Sign up to GitHub according to the Domain Assumption D10
     14. Fork the repository
     15. Create the yaml.yml file according to the installation guide and commit
     16. Edit the code_kata.py file in the folder code_katas folder of the repository and commit
     17. Go back to the Student interface 18. Refresh the page

3. Test cases:
   (a) Correct flow
   (b) Student who is enrolled in the Battle in a team
   (c) Student who is not enrolled in the Battle
4. Results:

   The test results indicate that the system successfully updates the rankings of both Battles and Tournaments when triggered by a push from a Student who is indeed participating in the Battle. Additionally, it is designed to ignore commits made by Students who do not have the appropriate permissions.

- **F8 - Manual Evaluation**

1. Goal:

   Verify that Educators can perform a Manual Evaluation at the end of the Battle.

2. Steps:

   1. Login as an Educator
   2. Create a Tournament
   3. Create a Battle
   4. Logout
   5. Login as Student A
   6. Enroll in the Tournament
   7. Enroll in the Battle
   8. Logout
   9. Login as Student B
   10. Enroll in the Tournament
   11. Enroll in the Battle
   12. Logout
   13. Access the database with administration privilege
   14. Move up the subscription deadline of the Tournament
   15. Move up the subscription deadline of the Battle
   16. Create GitHub accounts for the two Students according to D10
   17. Go back to the login interface

18. Login as Student A
19. Click on the link of the repository
20. Fork the repository
21. Create the yaml.yml file according to the installation guide and commit
22. Edit the code_kata.py file in the folder code_katas folder of the repository and commit
23. Go back to the Student interface
24. Logout
25. Repeat everything from step 18 to step 24 with Student B
26. Access the database with administration privilege
27. Move up the submission deadline of the Battle
28. Login as the Educator who created the Tournament
29. Perform Manual Evaluation

3. Test cases:

   (a) Correct flow

   (b) Close Tournament while Battle is ongoing

4. Results:

   The test results indicate that the Educators are indeed always able to perform Manual Evaluation, for every team who participated in the Battle, regardless how the Battle was stopped.

- **F9 - Results Display**

  1. Goal:

     Verify that results are promptly displayed upon the conclusion of a Tournament or Battle, and that they are coherent and comprehensive, considering all commits made before the deadline.

  2. Steps:

     1. Log in as an Educator
     2. Create a Tournament
     3. Create a Battle
     4. Log out

5. Log in as Student A
6. Enroll in the Tournament
7. Enroll in the Battle
8. Log out
9. Log in as Student B
10. Enroll in the Tournament
11. Enroll in the Battle
12. Log out
13. Access the database with administrative privileges
14. Move up the subscription deadline of the Tournament
15. Move up the subscription deadline of the Battle
16. Create GitHub accounts for the two Students according to D10
17. Return to the login interface
18. Log in as Student A
19. Click on the link of the repository
20. Fork the repository
21. Create the yaml.yml file according to the installation guide and commit
22. Edit the code_kata.py file in the code_katas folder of the repository and commit
23. Return to the Student interface
24. Log out
25. Repeat steps 18 to 24 with Student B
26. Access the database with administrative privileges
27. Move up the submission deadline of the Battle
28. Move up the submission deadline of the Tournament
29. Return to the login interface
30. Log in as Student A
31. Visualize the results
32. Log out
33. Log in as the Educator
34. Perform Manual Evaluation
35. Log out
36. Log in as Student A
37. Visualize the results

3. Test cases:

   (a) Correct flow

4. Results:

   The test results indicate that results are correctly displayed, both before and after Manual Evaluation by the Educators managing the Tournament.

- **F10 - Timely Notification**

  According to the ITD provided to us, the Timely Notification product function was not implemented.

- **F11 - Detailed Student Profile**

  Since gamification aspects were not among the required functions to be developed, we did not verify whether the student profiles were implemented, as their sole purpose was to display the Badges.

3.3. Further notes

At the conclusion of the entire testing process, it's important to note the system's behavior regarding HTTP status codes. We observed instances where the system returned inappropriate codes, such as frequently issuing a status code of 200 even when the request wasn't adequately fulfilled. We advise the development team to ensure adherence to the HTTP communication standard to improve interoperability, compatibility, and integration with third-party services. Aligning with this standard would indeed result in a more robust and user-friendly experience for all stakeholders involved.

4 | Effort spent

This section shows the amount of time that each member has spent to produce the document. Please notice that each unit is the result of coordinated work among all the members.

UNIT	MEMBERS	HOURS
Testing	Piccinato, Piazzalunga, Puglisi	20h
Report	Piccinato	10h