POLITECNICO
MILANO 1863

CodeKataBattle project by Russo Mario
and Picone Paolo

Acceptance Testing Document

Deliverable:	ATD
Title:	Acceptance Testing
Authors:	Picone Paolo and Russo Mario
Version:	1.0
Date:	8-January-2024
Download page:	https://github.com/piconepaolo/PiconeRusso
Copyright:	Copyright © 2024, Picone Paolo and Russo Mario – All rights reserved

The team that performed the *Acceptance Testing* of the prototype of the CKB platform is composed by two members:

- Russo Mario <mario1.russo@mail.polimi.it>
- Picone Paolo <paolo.picone@mail.polimi.it>

The project analyzed in this document is the *ITD* performed by the group Saccone, Sissa and Zappia. The project is available at the following link: SacconeSissaZappia
The main documents used for references are the RASD and the ITD of the project. The main usage of the *RASD* was to extract the features that the group identified as well as to extract some useful Test Cases on which to perform the *Acceptance Testing*. The *ITD* was used to understand the structure of the code, the technologies used and the configuration steps.

1 Installation Setup

The installation and the setup of the project was performed following the documentation written in the section *Installation Instruction* of the ITD document.

1.1 Installation Steps

The steps perfomed in order to configure the environment and to run the project are the following:

- Installation and setup of ***Apache Kafka***
  - Kafka was downloaded from the official website Apache Kafka following the installation instruction provided in the official documentation.
  - The configuration of the *server.properties* file was performed following the istruction provided by the other group.
- Installation and setup of ***SonarQube Server***
  - SonarQube was downloaded from the official website SonarQube following the installation instruction provided in the official documentation.
  - The configuration of the *sonar.properties* file was performed following the istruction provided by the other group.
  - After the installation was perfomed the SonarQube server was started and the web interface was accessed in order to create a *User Token* that was used to authenticate the *SonarScanner*.
- Installation and setup of ***SonarScanner***
  - SonarScanner was downloaded from the official website SonarScanner following the installation instruction provided in the official documentation.
  - The PATH environment variable was set in order to make the *sonar-scanner* command available by the CKB app at runtime as written in the group instructions.
- Installation and setup of ***PostgreSQL***
  - PostgreSQL was downloaded from the official website PostgreSQL following the installation instruction provided in the official documentation.
  - As the ITD document states, two databases were created: one for *Sonar* and another for the *CKB* app.
  - Moreover there were created two users: one for *Sonar* and another for the *CKB* app.
- Installation of ***Ngrok***: the installation was performed following the instruction provided in the official documentation.

2 Acceptance Testing

The tests performed are based on the functions implemented in the code. The following list shows the tests grouped by use case:

- **Educator and Student Login**: the test case was performed by using the provided front-end interface. The login was performed by logging in with a GitHub account. There were successfully created and logged in both an Educator and a Student account.
- **Educator and Student Registration**: the test was performed by using the provided front-end interface by signing up with a GitHub account and then selecting the role for the newly created user: Educator or Student. The registration was successful even if there is no control on whether the user is actually an educator or a student.
- **Tournament and Battle Creation**: the test was performed by using the provided front-end interface and by using an Educator account. The creation of both a Tournament and a Battle was successful.
- **Subscribe to a Tournament or a Battle**: After the creation of a Tournament and a Battle, the test was performed by using the provided front-end interface and by using a Student account. The subscription to both a Tournament and a Battle was successful.
- **Subscribe to a Tournament after the registration deadline**: The test was performed by creating a Tournament and setting an arbitrary registration deadline. Once done that, the entry related to the registration deadline for the Tournament was modified manually in the database. After permforming the update, the test was performed by using the provided front-end interface and by using a Student account. The subscription to the Tournament was successful even if the registration deadline was already passed. This is a bug that should be fixed since it does not respect the requirements of the project. The test was not successful.
- **Notification on newly Created Tournament or Battle**: The test was performed by creating a Tournament and a Battle and by using the provided front-end interface by using an Educator account. The notification was successfully received by the Student account in the corresponding notification page.
- **End of a Tournament or a Battle**: The test was performed by creating a Tournament and a Battle and by using the provided front-end interface by using an Educator account. Once in the Tournament page or in the Battle page, the end of the Tournament or the Battle was manually triggered using the corresponding button. The end of the Tournament or the Battle was successfully received by the Student account in the corresponding notification page.
- **Code Push**: The test was performed by first creating a Tournament and a Battle. Then by using a student account the subscription to the Tournament and the Battle was performed. Once set up the student repository with the appropriate workflow (provided in the ITD code), the student pushed the code to the repository. The push was successful and the request was correctly received by the platform.

3 Additional Considerations

The overall installation process was very tedious and cumbersome. The architecture is composed by several technologies that need a proper configuration in order to communicate with each other. The configuration process is very prone to human errors and not easily replicable. An easier and more user-friendly installation process would be using a *Docker* container.
Moreover, the documentation section (*Section 6.8*) related to the run of the application refers to an executable (*runCKB.bat*) compatible only with *Windows System*. This script was a limitation since the group that was redacting the ATD document was provided with only *Mac OS* machines. The problem was easily overcome by converting the script into a *.sh* file, compatible with *Unix Like Systems*.

4 Effort Spent

The effort spent for the redaction of this document as well as the installation process, the configuration and the execution of the tests is estimated to be about 10 hour for each member. The time was spent in the following way:

- 2 hours to read the relevant parts of the RASD and the ITD
- 5 hours for the installation and the configuration of the environment
- 2 hours for the execution of the tests
- 2 hours for the redaction of the document

References

- Document written using LaTeX and Visual Studio Code
- SacconeSissaZappia Repository: GitHub Repository
- SonarQube: SonarQube
- Apache Kafka: Apache Kafka
- PostgreSQL: PostgreSQL
- Ngrok: Ngrok