POLITECNICO DI MILANO

SOFTWARE ENGINEERING II

CodeKataBattle

ATD Document

Version 1.0

Daniel Maurio Ruiz Suarez
Sebastian Enrique Perea Lopez

February 10, 2024

Contents

1. PROJECT TO REVIEW

The purpose of this review is to analyze the project titled CodeKataBattle, authored by Mattia Piccinato, Gabriele Puglisi, and Jacopo Piazzalunga. This project is hosted on GitHub at the following repository: CodeKataBattle Repository.

In this review, we will examine key documents and artifacts associated with the project, including the Implementation and Test deliverable (ITD), Design Document (DD), and Requirement Analysis and Specification Document (RASD). These documents provide essential insights into the project's objectives, technical specifications, and requirements.

The analysis will focus on assessing the project's adherence to established standards, its alignment with stakeholder requirements as outlined in the RASD, and the effectiveness of its design and testing strategies as documented in the ITD and DD.

By conducting this review, we aim to identify strengths, weaknesses, and areas for improvement within the project, ultimately contributing to its overall quality and success.

2. INSTALLATION SETUP

In this section, we delve into the installation setup process for the CodeKataBattle project. The successful installation of any software prototype is foundational to its evaluation and eventual adoption. This chapter outlines the steps taken to install the prototype, along with the encountered challenges and any inconsistencies uncovered while following the accompanying documentation.

By detailing the installation process and highlighting areas of improvement or clarification within the documentation, this section aims to provide valuable insights into the project's accessibility, usability, and user experience. Through this analysis, we seek to contribute to the enhancement of the project's overall accessibility and ease of adoption.

2.1. Installation Process

The Implementation and Test deliverable (ITD) provided minimal instructions for installing the prototype, as it was primarily deployed on the website CodeKataBattle. However, accessing the backend via the provided link https://codekatabattle.it:8443/ proved challenging due to token requirements.

Acknowledging the necessity of a local installation for comprehensive testing, communication with the developers was initiated to obtain step-by-step instructions. Subsequently, detailed instructions were acquired and are presented below.

- **Frontend Installation:**
    - Navigate to a clean directory and download the repository files afresh, either by downloading the zip file or using git clone.
    - Change directory to ITD/Frontend/codekatabattle-app and execute the following commands:
        - npm install
        - npm run dev
    - This will initiate the React web application, accessible on port 3000.

- **Backend Installation:**
    - Open IntelliJ and load the project located at ITD/Backend/CodeKataBattle.
    - Allow IntelliJ to index the project and, once completed, add a new configuration by selecting "springboot" configuration.
    - Choose OpenJDK21 (or equivalent) for Java21, ensuring automatic download if required.

- Select the CodeKataBattleApplication path: com.polimi.PPP.CodeKataBattle.CodeKataBattleApplication.
- If the path doesn't appear automatically, wait for IntelliJ to finish indexing the project.
- Upon reaching this stage, contact the appropriate personnel to obtain the necessary environment variables.
- Set the environment variables as instructed to enable successful execution.
- Upon completion of these steps, the backend will run on HTTPS port 8443.

2.2. Challenges Faced

- **Frontend Installation:**

  No issues faced during the frontend installation process.

- **Backend Installation:**

  For users setting up the backend, configuring the project, setting up the JDK, specifying the main class, and configuring environmental variables posed initial challenges. Due to these complexities, reaching out to the developers for guidance and troubleshooting assistance was necessary to ensure the smooth execution of the application.

2.3. Incoherences in Documentation

The original documentation's deficiency in providing instructions for running the system in a local environment emphasized the project's reliance on web deployment. While this approach may suit many users, those necessitating a local installation encountered obstacles due to the lack of comprehensive installation guidelines. This omission underscored the importance of tailoring documentation to accommodate diverse user needs. Comprehensive instructions encompassing local setup procedures, including backend deployment and configuration, would have facilitated a smoother onboarding process for users seeking to test, develop, or customize the application in their local environments.

Moreover, while the documentation furnished a link to an already deployed backend, it overlooked the critical aspect of providing guidance on acquiring a proper authentication token for direct access. This gap in information left users attempting to interface with the backend independently grappling with ambiguity and uncertainty. As authentication is paramount for securing access to backend resources, the absence of clear instructions hindered users' ability to seamlessly integrate backend services into their applications. A well-documented authentication process would have empowered users to confidently interact with the backend, enhancing the overall usability and accessibility of the project.

3. ACCEPTANCE TEST CASES

In this section, we'll delve into the Acceptance Test Cases (ATCs) for the CodeKataBattle project. Acceptance testing is a crucial phase in software development, ensuring that the product aligns with stakeholder requirements. By examining the ATCs, we aim to assess the project's adherence to specified criteria and its capability to deliver the intended functionality.

This chapter will explore the various test cases defined to validate the system's functionality, user interactions, and overall performance. We will utilize the Requirements expressed in their Requirement Analysis and Specification Document (RASD) as a benchmark for our evaluation.

It's worth noting that any requirement referencing the "Badges" feature will be omitted from our testing process, as it was not needed to be implemented.

Through analysis and evaluation, we'll identify strengths, weaknesses, and areas for improvement in the project's acceptance testing framework. Our goal is to provide insights into the project's readiness for deployment and its ability to meet user needs effectively. Through this exploration, we aim to contribute to the enhancement of the project's quality and reliability.

3.1. Review of requirements

3.1.1. Recapitulation of key requirements outlined in the RASD document.

In this section, we'll review the original requirements outlined in the Requirement Analysis and Specification Document (RASD) for the CodeKataBattle project. These requirements serve as the foundation for the Acceptance Test Cases (ATCs) and play a crucial role in ensuring the system meets stakeholder expectations.

R1: User Registration

- The system must allow an unregistered Educator to sign up.
- The system must allow an unregistered Student to sign up.
- The system must allow a registered User to log in.

R2: Tournament Creation

- The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
- The system must provide registered Educators with Tournament-related statistics for Badges definition during the Tournament creation process.
- The system must provide registered Educators with a specific language to define Badges during the Tournament creation process.
- The system must allow registered Educators to grant other registered Educators permission to manage the Tournament.

- The system must allow registered Educators to end the creation process of a Tournament that they started themselves.

R3: Notifications

- The system must be able to send notifications to every registered User.
- The system must notify every registered Student about the creation of a new Tournament.
- The system must notify every registered Student about the creation of a new Battle within a Tournament they are enrolled in.
- The system must notify every registered Student about the end of a Battle they are participating in.
- The system must notify every registered Student about the end of a Tournament they are enrolled in.

R4: Battle & Tournament Management

- The system must allow registered Educators to create a Battle in a Tournament if they are the creator of the Tournament or if they were granted permission to by the latter.
- The system must allow registered Students to create a group for a Battle in a Tournament.
- The system must allow registered Students to accept an invitation to a group for a Battle in a Tournament.
- The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline has not expired yet.
- The system must allow registered Students enrolled in a Battle to perform code submissions.

R5: GitHub Integration

- The system must be provided with proper APIs to let registered Students perform code submissions through GitHub Actions.
- The system must update the Battle ranking when a valid code submission is performed.
- The system must set the Consolidation Stage of a Battle when its submission deadline expires.
- The system must allow Educators to end the Consolidation Stage of a Battle if they are the creator of the Tournament or if they were granted permission to by the latter.
- The system must update Tournament ranking when a Battle exits the Consolidation Stage.

R6: Tournament Closure and Achievements

- The system must allow Educators to close a Tournament if there are no Battles with unexpired subscription or submission deadlines or if they are still in the Consolidation Stage.

- The system must assign an achievements' Badge for a given Tournament to any Student who satisfies the conditions defined by the creator of the Tournament.
- The system must allow every User to see the Badges obtained by a given Student.

By reviewing these requirements, we gain a comprehensive understanding of the functionality expected from the CodeKataBattle system. These requirements will serve as the basis for defining and executing acceptance test cases to ensure the system meets stakeholder expectations.

3.1.2. Explanation of How Each Requirement Will Be Tested:

Each requirement identified as relevant to the ATCs will undergo testing to ensure its compliance with specified criteria. The testing process will involve the following steps:

Requirement Analysis

- Review the specific functionalities and user interactions outlined in each requirement to understand the expected behavior of the system.

Test Case Design

- Develop test cases based on the identified requirements, specifying the input data, expected outcomes, and steps to be executed.

Test Execution:

- Execute the designed test cases in a controlled environment, simulating real-world user interactions to validate the system's behavior.

Result Evaluation:

- Analyze the test results to determine whether the system's behavior aligns with the specified requirements.

Documentation:

- Document the test results, including any discrepancies or deviations from expected outcomes, for further analysis and resolution.

Through systematic testing of the identified requirements, we aim to ensure the robustness, reliability, and usability of the CodeKataBattle system, thereby enhancing its overall quality and user satisfaction.

3.2. Test Case Design

3.2.1. Approach to Designing ATCs

The design of acceptance test cases for the CodeKataBattle project involves a systematic approach aimed at validating the system's functionality, user interactions, and performance against specified requirements. The following steps outline our approach

Requirement Analysis:

- Thoroughly review the requirements outlined in the RASD document to understand the expected behavior of the system.
- Identify key functionalities, user actions, and system responses specified in each requirement.

Test Case Identification:

- Based on the requirements analysis, identify specific scenarios and user interactions that need to be tested.
- Determine the input data, expected outcomes, and steps to be executed for each test case.

Test Case Prioritization:

- Prioritize test cases based on their criticality and impact on system functionality.
- Focus on high-priority scenarios that are essential for system operation and user satisfaction.

Test Case Design:

- Develop clear and concise test cases that cover a wide range of scenarios, including positive and negative test cases.
- Ensure that each test case is independent, meaning that the outcome of one test case does not affect the execution of another.

3.2.2. Criteria for Scope and Coverage of Test Cases

The scope and coverage of the test cases are determined based on several criteria to ensure comprehensive testing of the CodeKataBattle system

Functional Coverage: Test cases are designed to cover all functional requirements outlined in the RASD document, ensuring that each requirement is thoroughly tested.

User Interaction Scenarios: Test cases include scenarios that simulate various user interactions, such as user registration, tournament creation, battle management, and notifications.

Boundary Cases: Boundary test cases are included to validate the system's behavior at the edges of input ranges, ensuring robustness and error handling.

Negative Testing: Negative test cases are designed to verify how the system handles invalid inputs, error conditions, and unexpected user actions.

Integration Testing: Test cases include scenarios that validate integration points with external systems, such as GitHub for code submissions.

3.3. Test Case Execution

3.3.1. R1 User registration and Login Tests Cases

Test T1 - Baseline Functional Coverage

- Enter the platform using the link “https://codekatabattle.it” provided by the team
- We first can appreciate a Log-in section, were we are required to provide our credentials “Email/Username and Password” but as we don't have an user, we first click on the “Sign up” hyperlink to create a new account
- On the user registration form containing the fields “Name, Surname, Username, LinkBio, Email, Password” and a radio button to select our type of user “Student or Educator”, we filed them in an appropriate way
- We were then sent to the previous page to try and Log-in again
- After correctly validate our credentials, we can now access their platform with the permissions corresponding our selected role

Test T2 - Boundary and Negative Testing

We will try to register an user with the following cases:

- Try to reach the character limit for the fields
- Leave some fields without information
- Try to input an incorrect format on the email field
- Not choose any role
- Try to test the minimum and maximum characters required for the password
- Test the standard safety password required by the platform

3.3.2. R2 Tournament Creation

For these test cases we assume we are already logged in as an Educator.

Test T1 - Baseline Functional Coverage

- On the homescreen the user has the option to directly enter the details of a new tournament.
- We enter the Subscription deadline and the tournament name
- Optionally we can Invite other educators by using the “Invite an Educator” button, in this button we add the username of the user and click on “add”
- Then we press the create button and get a confirmation that it was created

Test T2 - Boundary and Negative Testing

We will try to:

- Test the limit of characters for the name of the tournament
- Register the tournament with a subscription deadline already passed
- Invite another educator to collaborate in the tournament, and see how many educators we are allowed to invite
- Leaving the name field empty

3.3.3. R3 Notifications

For these requirements we did not apply any direct test case as they should appear while doing other test cases, like creating battles or tournaments.

3.3.4. R4 Battle & Tournament Management

From the homescreen the user can go to his tournaments in the right side of the screen, or in the list with all the tournaments on the left, then by clicking on “Info” we can get the details of the tournament

Test T1 - Baseline Functional Coverage

- The user enters the name for a battle he wants to create in the tournament.
- The user selects the start and end deadlines in accordance with the time interval set by the tournament.
- The user needs to upload two zip files corresponding to the CodeKata project and the test that the platform will need to apply to that project.
- Pick the programming language in which the battle will be managed.
- Select what are the type of scoring systems that the user wants this specific battle to have.
- Then click on “Create”

Test T1.5 - Baseline Functional Coverage- As invited educator

- From the homescreen logged in as an educator that was invited, we can go to invited tournaments on the right side of the screen
- Then by clicking on “Info” we can get the details of the tournament
- Once in the details we have all of the fields to add a battle
- Now the name, Subscription deadline, submission deadline, group size, codekata project, codekata tests, coding language and the option to activate manual scoring can be properly filled out.
- Once completed we can click on “Create”
- After a while an alert will appear showing that the battle was created successfully

Test T2 - Boundary and Negative Testing

We will try to:

- Leave the input for the name ant the group sizes empty
- Define an invalid interval for the battle to start, both set the start date of the battle to a date previous to the tournament start, and the end date after the tournament ends
- Test the minimum and maximum sizes for the teams allowed

- Don’t upload zip files for the battle and upload zip fields with an incorrect internal structure as the one specified by the developers
- Don’t select a programming language for the battle
- Leave the scoring methodologies for the battle in blank

Test T3 - Baseline inviting students to form groups

- To join a battle the user goes to the tournament, then sees the available battles
- If a battle has the option to join then we can join the battle
- During the popup of enrollment we can add the username of other students we want to invite
- We add the usernames and then click on “Enroll”
- The system prompts that the user has enrolled properly

3.3.5. R5 GitHub Integration

For these cases we assume we are logged in as a student that is part of a tournament and a battle.

Test T1 - Baseline Functional Coverage

- After a battle is ongoing the system will show a “Github Token” button
- Once the button is clicked the system prompts with a guide on how to add the token and the action to github
- We then can go to github and add the information to the repo, with this we can make the proper pushes and calls to the API
- If we go back to the info screens then we need to refresh the page to see our score updating
- Once the battle goes to consolidation and then ends we will be able to see the ranking

Test T2 - Boundary and Negative Testing

- We will do the same process as T1 but the battle that was created this time had a project created by ourselves (instead of the ones provided by the developers)

3.3.6. R6 Tournament Closure and Achievements

Test T1 - Baseline Functional Coverage

- To end the tournament the user just need to click in the “End Tournament” button that is located in the center-top of the section, besides the current status of the tournament
- After that, the user can go back to the main screen to create a new tournament

Test T2 - Boundary and Negative Testing

Tried to end the tournament when multiple cases:

- The tournament was under the subscription and end status
- There were battles on the Subscription, ongoing and consolidation Stage

3.4. Test Results and Analysis

3.4.1. R1 User registration and Login Tests Cases

Test T1 - Baseline Functional Coverage

- There was no distinction between required and no required fields
- The input field “Link Bio” is unclear when it comes to what it’s content must be
- If an error occurs when we try to submit, there is just a generic 400 error

Test T2 - Boundary and Negative Testing

- We noticed that there was not specific limit set for the user in the frontend, there is an error that arises when the user reach a certain limit of characters, but is not set or handled by the fronted, it’s an error that is created by the server and the max limit of characters defined in the database for that data type.
- All the fields are validated against leaving them empty.
- The email field is not validated against incorrect email formats, it just checks whether there is an “@” with at least a character in front and behind, but it doesn't validate whether that email address is at least one of the most common ones.
- The form lets the user register without choosing a role, but is later set by default as an educator role. It shouldn’t allow a user to register if the role is not selected.
- When attempting to register with a short password, the system generates an unclear error. By examining the console log, it becomes evident that a minimum of 9 characters is required for the password field. Moreover, there is no specific error message to guide users about the password criteria. The only feedback provided is a generic 400 error when attempting to create an account. The password requirements include the necessity for at least one capital letter, one lowercase letter, one number, and a special symbol (excluding #, ^, or ()). This information is not explicitly communicated to the user, resulting in a less user-friendly experience.

3.4.2. R2 Tournament Creation

Test T1 - Baseline Functional Coverage

- There was no feedback of the button to see if the system was doing something
- No Error Messaging
- When trying to add an Educator it gave an error (Checked via console) that the name of the educator was not valid as it was a String and the system was expecting an integer. After validating it requires the educator ID, the only way to get this ID is via the database directly.
- Adding the User ID properly adds the user to the tournament, although there is no feedback on it.
- The “Add a badge” button was created even though it was not implemented, generating possible confusion on the user when it tries to use the button but it does nothing.

Test T2 - Boundary and Negative Testing

- When the user try to register a tournament with an invalid date, although the user will not be allowed to proceed, there is also no feedback as to why, because there is not any kind of feedback, neither as to know if the tournament is in process of being created or if there was an error with the parameters selected by the user, not even the generic “error 400” seen before, in this case, the minimum subscription deadline is set to not be before the current moment, but this is not known by the user
- There was no specific limit set for the user in the frontend, there is an error that arises when the user reaches a certain limit of characters, as before, it’s an error that is created by the server and the max limit of characters defined in the database for that data type.
- The name field is validated against leaving it empty, but this time there is no feedback for the user to let it know it's mandatory to fill it, there is just a message in the console log.
- We tried to invite educators to the mock tournament we were creating, but neither the username or the email of the invited educator allowed to do so, it seem that the only way to invite an educator is by using the private identification that said educator is assigned when added to the database, identification that a normal user should not have access to

3.4.3. R3 Notifications

Test T1 - Baseline Functional Coverage

- The system properly sends notifications via email to the email that was used to register the account
- The system properly notify of a battle only if the user is subscribed to that tournament
- The system properly notify when a tournament ends
- There is no notification to when the battle goes to the consolidation or end state

3.4.4. R4 Battle & Tournament Management

Test T1 & T1.5 - Baseline Functional Coverage as invited Educator

- No issues when doing the baseline process with correct information.
- In general, the way the invited educator is managed in a tournament is by giving it the same authority that the owner of the tournament has, so the invited educator can do everything that the owner can do.

Test T2 - Boundary and Negative Testing

- After try and leaving the fields as they are, the user may notice that all the fields are validated against leaving them empty, but in case of the “Select language” field, the error message that arises its from the server, and not directed in a clear way to the user
- The subscription deadline and submission deadline are validated against trying to initiate a battle before the start of the tournament, and how there is no specific date for the end of the tournament, the only restrain for the submission battle deadline is that it can’t be before the subscription deadline
- Although the error message or constraints are not managed in a friendly way to the user, because the user can input negative values, or massive ones, the field is validated against the max users being lower than the min users, and that both of them need to be positive integers, but the fields are not properly constrained, so the min and max limit of students is equal to the max value an int variable can have (“2147483647”)
- The user is required to always provide a zip file for the project and the tests, and even though the user can upload any other type of file, the creation does not proceed till the files are zip.
- Special annotation in the fact that when we tried uploading a zip project that followed their specified instructions, it wasn't recognized because some internal distinction between the zip files compiled on mac and windows, we were forced to use an external compiler to try our projects, and even with that, it is not properly recognized or evaluated by their scoring system
- The user is required to always select a language for their battle, although the only option available is “Java”, and the error message that pops up when left unselected it's not very user friendly, because it's not a message handled by the developers, but most likely for the backend when trying to receive the format with that section empty
- The user is not forcefully required to select this option, but even if they do, it's not something that is taken into account in a further scenario

Test T3 - Baseline inviting students to form groups

- When the user press the button “Join”, they are presented the option to add another student users to the battle to form teams, but this function we are not sure of its behavior, because, in the first place the popup is misleading, given that it has an incorrect title
- We add the usernames and then click on “Enroll” button, adding it to a list containing all the usernames that the user want to invite to join its team
- Although it has an “Invite” specific button, it does nothing, only closes the popup.
- If the usernames provided are valid, an alert announcing that the enrollment was made successfully popups, but the invited user would not get a notification or even be allowed to participate in the battle, we couldn’t figure out why, but it was rejected in the backend (checking the database directly)

3.4.5. R5 GitHub Integration

Test T1 - Baseline Functional Coverage

- The system provides the information via an alert, this does not allow to copy the content of it on all of the systems and browser, therefore we couldn't copy the token and the URL to the action example, we had to check the console output to copy the proper token
- The system does not provide any way to find the original repo created by it.
- There is a button that is called “Show CodeKata” but when clicked it just shows the “This feature is not available yet” error.
- To find the repo we had to check the console to get the proper URL
- Once the information is extracted we can go to github and fork the project
- Then add the token and action workflow
- Then this will properly call the backend system to create the testing and scores
- To update the scores we need to refresh the website until the new score appears, this does not update dynamically and needs to be refreshed each time.
- The system properly sets the battle to consolidation once the submission deadline is reached
- The system does not allow to end the battle, when clicking the “manage consolidation” button as an educator it just shows “This feature is not available yet” therefore the system does not update the Tournament scores.

Test T2 - Boundary and Negative Testing

- Same issues as before but this time the scores are always 0, it appears that, as mentioned before, the backend can't handle the project created by ourselves, the project was validated before uploading and it was uploaded as the website requested.

3.4.6. R6 Tournament Closure and Achievements

Test T1 - Baseline Functional Coverage

- When cursing the end tournament there is no feedback that it is working on ending it
- After completion the system shows an alert that the tournament has ended successfully

Test T2 - Boundary and Negative Testing

- If we try to end the tournament when it's in the registration state then it gives a proper error mentioning that “Tournament is not in ongoing phase”
- If the tournament is ongoing but there are active battles we get a proper error “There are still Battles ongoing or in consolidation stage.”

4. RECOMMENDATIONS AND CONCLUSION

4.1. Requirement Achievement Analysis

4.1.1. Achieved Requirements

A. The system must allow an unregistered Educator to sign up.
B. The system must allow an unregistered Student to sign up.
C. The system must allow a registered User to log in.
D. The system must allow registered Educators to start the creation process of a Tournament of Code Kata Battles.
E. The system must be able to send notifications to every registered User.
F. The system must notify every registered Student about the creation of a new Tournament.
G. The system must notify every registered Student about the creation of a new Battle within a Tournament they are enrolled in.
H. The system must notify every registered Student about the end of a Tournament they are enrolled in.
I. The system must allow registered Educators to create a Battle in a Tournament if they are the creator of the Tournament or if they were granted permission to by the latter.
J. The system must allow registered Students to see the list of ongoing Tournaments and join any of those if its subscription deadline has not expired yet.
K. The system must be provided with proper APIs to let registered Students perform code submissions through GitHub Actions.
L. The system must set the Consolidation Stage of a Battle when its submission deadline expires.
M. The system must allow Educators to close a Tournament if there are no Battles with unexpired subscription or submission deadlines or if they are still in the Consolidation Stage.

4.1.2. Unachieved Requirements:

A. The system must notify every registered Student about the end of a Battle they are participating in.
B. The system must allow registered Students to create a group for a Battle in a Tournament.
C. The system must allow registered Students to accept an invitation to a group for a Battle in a Tournament.
D. The system must allow Educators to end the Consolidation Stage of a Battle if they are the creator of the Tournament or if they were granted permission to by the latter.
E. The system must update Tournament ranking when a Battle exits the Consolidation Stage.

4.1.3. Partially Achieved Requirements:

A. The system must allow registered Educators to grant other registered Educators permission to manage the Tournament.
B. The system must allow registered Educators to end the creation process of a Tournament that they started themselves.
C. The system must allow registered Students enrolled in a Battle to perform code submissions.
D. The system must update the Battle ranking when a valid code submission is performed.

4.2. Recommendations for Improvements or Enhancements

4.2.1. Error Messaging Enhancement:

Implement clear and informative error messages throughout the system to provide users with actionable feedback when issues occur during registration, tournament creation, and other interactions. Improve error handling to ensure that users are guided effectively through error resolution processes.

4.2.2. User Interface Feedback:

Provide visual feedback, such as loading indicators or progress bars, to inform users about ongoing processes, such as account registration or tournament creation. Enhance user interface elements to communicate system actions and status changes more effectively.

4.2.3. Input Validation Improvements:

Strengthen input validation mechanisms to prevent invalid data submissions and improve user experience. Validate email formats, enforce role selection during registration, and provide clear password requirements to users to reduce errors and enhance data integrity.

4.2.4. Integration and Compatibility Testing:

Conduct thorough integration and compatibility testing, particularly with external systems like GitHub, to ensure seamless interaction and functionality. Address issues related to project recognition, scoring discrepancies, and dynamic updates to enhance system reliability and performance.

4.3. Additional Suggestions and Identified Errors

4.3.1. Suggestions

- **Log Management**: Consider implementing a log management system to remove unnecessary logs and improve code cleanliness.
- **Non-Functional UI Elements:** Address non-functional buttons in the navbar and UI, such as the profile button or search bar, to prevent user confusion and frustration.
- **Feedback for Loading Processes:** Provide user feedback for actions that may require time to load or process, such as creating a tournament or battle, to improve user awareness and experience.
- **Navigation Improvement:** Implement back buttons in all views to enhance navigation usability and convenience for users.
- **Post-Creation Tournament Management:** Add functionality to invite educators to a tournament after its creation to facilitate collaboration and management.
- **Edit Tournament Parameters:** Provide users with the ability to edit previously defined parameters of tournaments or battles as needed to accommodate changes or adjustments.

4.3.2. Identified Errors

- **Incorrect Login Information:** Clarify login instructions to only allow email input, as the current instruction suggests the use of username, which is not supported.
- **Tournament Enrollment Validation:** Implement validation to prevent a user from joining a tournament they are already part of, as it may cause confusion and inaccuracies in tournament participation.
- **Access Control for Battles:** Ensure that users can only join battles of tournaments they are part of to maintain proper access control and integrity.
- **Missing Battle Descriptions:** Add descriptions for battles to provide users with necessary context and information before participating.
- **Static Analysis Evaluation Setting Confusion:** Address confusion caused by the presence of settings for static analysis evaluation that are not implemented, possibly by removing or clearly indicating their status.
- **Ineffective Manual Evaluation Option:** Improve the functionality of the manual evaluation option to ensure it performs as expected and provides value to users.
- **Score Identification Issue:** Enhance score indexing in submissions to enable users to properly identify their scores and track their progress accurately.

4.4. Conclusion:

The results of the acceptance testing process have provided valuable insights into the functionality, usability, and performance of the CodeKataBattle system. Through rigorous testing of various scenarios and user interactions, we have identified both strengths and areas for improvement.

While the system demonstrates alignment with specified requirements in several areas, there are notable deficiencies in error handling, user feedback, and integration functionality that require attention. Addressing these issues will be crucial to enhancing the overall user experience and system effectiveness.

While the CodeKataBattle system shows promise in meeting user needs for code kata battles and collaboration, further refinement and iteration are necessary to ensure deployment readiness. By implementing the recommended improvements and addressing identified deficiencies, the system can enhance its usability, reliability, and user satisfaction.

5. EFFORT SPENT

Daniel Mauricio Ruiz Suarez

Chapter	Effort(in hours)
1	2
2	10
3	35
4	1

Sebastian Enrique Perea Lopez

Chapter	Effort(in hours)
1	5
2	20
3	30
4	2

6. REFERENCE

IEEE Standards:

IEEE. IEEE Standards. Retrieved from https://grouper.ieee.org/groups/802/22/Documentation/format-rules.html#Document

Software Engineering 2 course material:

Politecnico di Milano 2023-2024

RASD Document

RASD document for group PiazzalungaPiccinatoPuglisi. Retrieved from https://github.com/GabP404/PiazzalungaPiccinatoPuglisi-CodeKataBattle/blob/main/DeliveryFolder/RASDv1.pdf

DD Document

DD document for group PiazzalungaPiccinatoPuglisi. Retrieved from https://github.com/GabP404/PiazzalungaPiccinatoPuglisi-CodeKataBattle/blob/main/DeliveryFolder/DDv2.pdf

ITD Document

ITD document for group PiazzalungaPiccinatoPuglisi. Retrieved from https://github.com/GabP404/PiazzalungaPiccinatoPuglisi-CodeKataBattle/blob/main/DeliveryFolder/ITDv1.pdf