POLITECNICO
MILANO 1863

Travlendar+ project YOUR NAMES

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	YOUR NAMES
Version:	1.0
Date:	31-January-2016
Download page:	LINK TO YOUR REPOSITORY
Copyright:	Copyright © 2017, YOUR NAMES – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

This document has been prepared to help you approaching Latex as a formatting tool for your Travlendar+ deliverables. This document suggests you a possible style and format for your deliverables and contains information about basic formatting commands in Latex. A good guide to Latex is available here https://tobi.oetiker.ch/lshort/lshort.pdf, but you can find many other good references on the web.

Writing in Latex means writing textual files having a `.tex` extension and exploiting the Latex markup commands for formatting purposes. Your files then need to be compiled using the Latex compiler. Similarly to programming languages, you can find many editors that help you writing and compiling your latex code. Here https://beebom.com/best-latex-editors/ you have a short oviewview of some of them. Feel free to choose the one you like.

Include a subsection for each of the following items[1]:

- Purpose: here we include the goals of the project
- Scope: here we include an analysis of the world and of the shared phenomena
- Definitions, Acronyms, Abbreviations
- Revision history
- Reference Documents
- Document Structure

Below you see how to define the header for a subsection.

1.1 Purpose

In today's digital age, education has evolved significantly, with online learning becoming more and more important, especially in subjects like computer science. However, only theoretical learning often can be insufficient in practical fields like coding. This is where the CodeKataBattle (CKB) project steps in, addressing the need for a hands-on, collaborative, and engaging approach to learning programming and software development.

Although theoretical knowledge is essential in computer science, it is the practical application and coding practice that strengthen students' understanding and skills. Hence, CKB is designed to support students with a platform where they can actively practice coding, experiment, learn from their mistakes, and improve their skills. Moreover, it encourages students to work together in teams, promoting teamwork and communication, essential qualities in the software development field. For educators, CKB offers a valuable teaching tool, enabling them to create coding challenges in accordance with their curricula, making learning more engaging, and helping assess students' progress.

In summary, the CodeKataBattle project serves as a bridge between theoretical learning and practical application in computer science education. It offers students a platform to practice coding, collaborate, and improve their skills, and provides educators with effective teaching tools and assessment methods. The aim of CKB is to make computer science education more engaging, practical, and rewarding for both students and educators.

1.1.1 Goals

1. Students are able to find, join, or create teams for specific code kata battles within a tournament.
2. Educators can create and manage code kata battles, including the provision of detailed descriptions, project templates with test cases, and build automation scripts.

[1] By the way, what follows is the structure of an itemized list in Latex.

3. Educators have the capability to set up and customize various battle parameters such as team size limits, registration deadlines, submission deadlines, and scoring configurations.
4. Students commit their code implementations to a specific GitHub repository for each battle, triggering automated testing and scoring through the platform.
5. The platform automatically calculates and updates battle scores based on predefined criteria including the number of passed tests, commit timeliness, and code quality.
6. Educators can manually score submissions, considering aspects outside of automated evaluations.
7. Students and educators can view the evolving ranks and scores of teams during a battle.
8. The final battle ranks are determined after considering both automated and manual evaluations, and made visible to all participants upon completion of each battle.
9. Cumulative personal tournament scores are updated and displayed for each student, contributing to an overall tournament ranking.
10. The platform notifies students of new tournaments, battle opportunities, and updates on their tournament standings.
11. Students can view their rankings on their profiles.

1.2 Scope

1.2.1 Product Identification

- The **target audience** of the platform is mainly **students and educators** in computer science and related fields.
- CKB is an **interactive web-based coding platform** specifically designed to improve the coding skills of students by participating in battles designed by educators. It is mainly an **educational tool** where practice can be applied.
- CKB is also a **competitive learning environment**. By organizing the coding exercises in battle format, the students are expected to be more motivated to perform their best in challenges.
- The platform **promotes teamwork and collaboration** by allowing students to form groups to tackle battles together. This reflects real-world software development scenarios where collaborative team dynamics are key.
- The platform has an **educator-centric control**. The educators play a crucial role in the platform because they are the ones creating, managing, and scoring the battles and tournaments.
- CKB enables students to **use professional tools and practices** such as version control, and fork-pull-push mechanisms so that they can enhance their real-world software development skills with the help of Github.
- The platform makes use of **automated testing** to assess student submissions, ensuring objective evaluation of functional correctness and code quality. Additionally, it allows educators to perform manual evaluations for aspects that require subjective judgment.
- CKB is designed to provide **instant feedback** on submissions and **real-time updates** of team scores and rankings.
- The platform offers **flexibility** to educators in terms of the choice of programming languages, difficulty levels of challenges, and the scope of coding tasks, making it adaptable to various learning curves and educational needs.

1.2.2 Domain Analysis

The CKB platform operates within the domain of educational technology, specifically tailored for coding and software development education. From this perspective, we have the following users:

Educators: Educators use CKB to **create and manage coding exercises**, known as code katas, in a battle format. They have the capability to set parameters for these exercises, including difficulty levels, deadlines, and specific technical requirements. Educators can **evaluate student submissions** both automatically (using the platform's tools) and manually, **providing feedback and scores**.

Students: Students exploit the platform to enhance their coding skills by **participating** in these code battles. They work individually or in teams to solve the challenges set by educators, fostering both individual and collaborative learning experiences. Students use the platform **to submit their code, receive real-time feedback, and track their progress in coding proficiency**.

In summary, the domain of the CKB platform is focused on interactive, competitive coding education, enabling students to have an engaging learning experience while offering educators powerful tools for managing and evaluating coding exercises.

1.2.3 World Phenomena

1. Educators prepare the necessary documents for code battles including descriptions, test cases and build automation scripts.
2. Students fork the GitHub repository.
3. Students set up an automated workflow through GitHub Actions.
4. Students write code on their devices.

1.2.4 Shared Phenomena

World Controlled

1. Educators upload coding challenges, including descriptions and test cases, to the platform.
2. Educators create tournaments and give permission to their colleagues to create battles for that tournament.
3. Educators set specific rules and criteria such as deadlines, number of team members, and additional configurations for scoring for code kata battles.
4. Educators decide on the inclusion of manual scoring components for battles.
5. Students invite peers to form teams within the platform or join individually.
6. Students submit their code solutions by pushing the code via GitHub.

Machine Controlled

7. The CKB platform sends notifications to students about new battles and tournaments, final battle ranks and final tournament ranks.
8. The platform generates and manages GitHub repositories for each battle and sends links to students that are participating.
9. The CKB platform automatically evaluates code submissions against test cases.
10. The platform updates battle scores and battle rankings in real-time.
11. The platform displays leaderboards (i.e. the rank of the sum of all battle scores received in that tournament).

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Student**: An individual enrolled in an educational program or course who uses the platform to participate in coding exercises and improve software development skills.
- **Educator**: A person, such as a teacher or an instructor, responsible for creating coding challenges and managing learning activities on the platform.
- **Automated Testing**: A process where the CKB platform automatically executes predefined tests on student code submissions to assess their functionality and correctness without manual intervention.
- **Manual Scoring**: The process where educators evaluate student code submissions subjectively, complementing the automated testing system.
- **Battle**: A competitive coding challenge on the platform where students or teams of students solve specific programming problems within set parameters and time frames.
- **Tournament**: A series of code kata battles organized and managed by educators on the CKB platform, that ranks students or teams based on cumulative scores from individual battles.
- **Ranking**: A system within the CKB platform that orders participating students or teams based on their performance in individual code kata battles, determined by scores from automated and manual evaluations.
- **Leaderboard**: A feature on CKB that displays the standings of students or teams based on their performance overall in a tournament.

1.3.2 Acronyms

- **CKB**: CodeKataBattle

1.3.3 Abbreviations

- G_x: x-th Goal
- WP_x: x-th World Phenomena
- SP_x: x-th Shared Phenomena

2 Overall Description

2.1 Product Perspective

2.1.1 Scenarios

1. **Educator creates a tournament**
   Professor Emanuelle, a computer science educator giving an "Introduction to Programming" course in Politecnico di Milano, decides to improve her student's understanding of programming through coding exercises. She is already registered in CodeKataBattle as an educator, so she logs in to the platform with his credentials, which are email and password. Then, she clicks the "Create Tournament" and a form screen is prompted. She fills out the form respectively:
   - entering tournament title, "Winter Semester Coding Exercise 1",
   - adding a description of the tournament,
   - choosing a deadline for the subscription. After clicking the "Next" button
   Professor Emanuelle is asked to select colleagues for permission to create battles in this tournament. She selects 2 of her colleagues because they all together gives the same lecture. Finally, she clicks on the "Create" button, and tournament is created. She is directed to the "My tournaments" section after creation.

2. **Student Registers for tournament**
   Emre is a Bachelor's student in Computer Science at Politecnico di Milano. At the start of the winter semester, he registered CodeKataBattle as a requirement of his "Introduction to Programming" course. In the middle of the semester, he gets an email notification about a tournament created by Professor Emanuelle on the platform. Then, He clicks the link in the email. Because he has already logged in to the platform with his email and password, he is redirected to the tournament's page, where he finds detailed information about Emanuelle's tournament. After reading the description, he thinks that this tournament will be very helpful for him to understand the topics covered in the lecture. So he clicks to the "Register" button to enroll tournament. Some registration details consisting of descriptions, educators, and tournament creator are shown on the screen. Also, he is informed that he will get email notifications about upcoming battles, battle rankings and tournament rankings. Finally, Emre clicks on the "Accept and Register" button and registers the tournament.

3. **Educator Sets Up a Battle**
   A week ago Professor Mottola, was invited to a tournament created by his colleague in the CodeKataBattle platform. He accepted the invitation and learned about the tournament from its description. He also gives the "Introduction to Programming" course and wants to create a new exercise about the topic he showed in the last lecture, which is "Recursion". After the subscription deadline passed, he logs back into CKB, he selects the "Create Battle" option within the "Winter Semester Coding Exercise 1" tournament. He is presented with a detailed form where he inputs the battle's title, "Check if String is Palindrome" and provides a thorough description that includes the battle's focus on recursion, expected coding languages (Java and Python, both accepted), and software project including necessary scripts for build automation (Gradle for Java, no need for Python) and test cases. To encourage collaboration, he sets the minimum and maximum group size to three and five students, respectively. he then specifies the registration deadline, two weeks from the current date, and a final submission deadline, giving students a month to work on their solutions. Finally, he configures additional scoring parameters as **efficiency**, choosing from a list of aspects including reliability, maintainability, etc. Then he clicks on the "Create" button and he is redirected to battle main page illustrating information about the battle. After a couple of minutes, he got an email saying that an email about this battle was sent to all students in the tournament.

4. **Student Joins for Battle**
   Samet got a notification from the tournament he enrolled in saying that Professor Mottola has created a battle with the name "Check if String is Palindrome". He clicks on the link and reads the description carefully. He clicks on the "Register" button then a screen is shown asking Samet whether he wants to invite other students to form a team or not. Samet selects "Yes", then, he names her team 'Code Warriors' as a first step. Then he chooses students from a list of students registered for the tournament, considering minimum and maximum group size. Samet knows Emre, Jack, and Luca also registered for the tournament, so he sends them invitations. After clicking "Complete", he is redirected to the battle information page. On this page, there is a section showing group status, which is pending until all invitations are answered. After a while, Samet sees that Emre and Luca accepted but Jack rejected it. Because the minimum group size is fulfilled, Samet clicks on the "Finalize" button indicating the final decision. Thanks to the help of instructions during the process, Samet understands that if he wants to decline registration, he should have clicked on the "Decline" button. After every member accepts the invitation to finalize registration, the process ends and the status becomes "Registered". They are given a competitor id.

5. **Students Sets Up Environment for CodeKataBattle**
   With the 'Code Warriors' team formed and the registration deadline for the "Check if String is Palindrome" battle passed, the CKB platform takes its next automated step. It creates a unique GitHub repository for the battle, containing the provided code kata with its test cases and build scripts. The system then sends an email to Emre, Samet, and Luca with the repository link and instructions. The team members collaboratively decide to schedule a virtual meeting to set up their working environment. During the meeting, they fork the repository to their group account and set up GitHub Actions in order to make proper API calls with the competitor id they have. This setup is crucial for automating their workflow and ensuring that every code push not only updates their repository but also notifies the CKB platform. They test the setup by pushing a minor change, and upon seeing that the CKB platform acknowledges their commit, they know their system is correctly configured. This marks the start of their coding journey in the battle.

6. **Students Solve Battle Challenge**
   After forking the project and setting environment, they started to think about the solution of the project. As they understand from the description they should return "True" or "False" regarding whether the given string is a palindrome. They implement the algorithm and commit/push the code to the repository. Each push prompts the platform to pull the latest code, run tests, and analyze the quality of their solutions using static analysis tools. After a while, they revisit the ranking on the battle's page to see their score of the last push. They see their scores in 3 different categories:
   - functional aspects: 32/80 (4/10 test cases passed)
   - timeliness: 5/5
   - quality level of the sources: 8/10 (Aspects: Efficiency)
   In the rankings, they are informed via tooltips about the scaling of their scores and the calculation methods of them. So they decided to focus more on code and try to find why some test cases are not passed. The team keeps an eye on the CKB dashboard, which updates their battle score after each commit. They note improvements in their score as they refine their solutions, ensuring more test cases pass and optimizing their code for better quality. This iterative process of coding, committing, and refining continues, with the team members frequently discussing strategies and sharing insights to improve their solutions. After looking at some exercises related to recursions they finally find the wrong part in the algorithm they implemented. After changes they commit and push the code. They see in the rankings that they got 93 points from battle. This iterative process of coding, committing, and refining continues, with the team members frequently discussing strategies and sharing insights to improve their solutions. They think that this is the most efficient

algorithm they can implement. So they decide not to do anything else until code kata battle deadline.

7. **Educator Evaluates Submission Manually**
   After the submission deadline expires, there is a consolidation stage enabling educators to assign optional points for teams. Professor Mottola begins her manual evaluation of the submissions for the "Check if String is Palindrome" battle. He logs into CKB and accesses the educator's dashboard from the battle's page, where he can review each team's submission. On this page code of the submission and other materials from the repository are shown to the educator. He starts from the latest submission and gives extra points if he finds the solution by analyzing 'Code Warriors', impressed by their timely submission and high score, viewable in the rankings table. Professor Mottola examines their code, focusing on the efficiency of their solution, which got 8/10 from the analysis tool. Considering these factors, she awards them a high personal score. This score reflects her assessment of their problem-solving skills and coding proficiency, adding a crucial human element to the automated evaluation.

8. **Educators Analyze Tournament Leaderbord**
   Professor Emanuelle have created a tournament called "Winter Semester Coding Exercise 1" at the beginning of the semester. Some other colleagues of hers registered for the tournament as educators and created koda battles during the semester. During the semester, she has viewed the leaderboard and at the end of every week, she has exported the result. Eventually, at the end of the semester, Professor Emanuelle closes the tournament. She then goes to the leaderboard screen and starts to see the general success of her students by comparing leaderboard exports from the very first battle. In this way, she notices that the leaderboard becomes more competitive as we get closer to the end of the year. She interprets this situation as students improving their programming skills battle by battle. After analyzing the leaderboard she searched other ongoing tournaments of other professors from the university to see the success of their students in different tournaments created by teachers of different courses.

2.1.2 Domain Class Diagram

In Figure(1.1), the Domain Class Diagram for the CodeKataBattle (CKB) platform is illustrated. This diagram is a crucial element in understanding the structure and relationships within the CKB system. It visually represents the key classes, their attributes, and the interactions among them in the context of the platform. This diagram is designed to provide a clear and concise overview of the domain model, offering insights into how the different components of the CKB platform interact to facilitate a competitive and educational coding environment. Below are descriptions of the most relevant parts of the diagram:

- **Student and Educator Classes**: The diagram differentiates between two primary types of Users: students and educators. Both share common attributes such as name, surname, email, and password, but they interact with the system differently. Educators, for instance, have the ability to create tournaments and battles, while students can participate in these competitions.

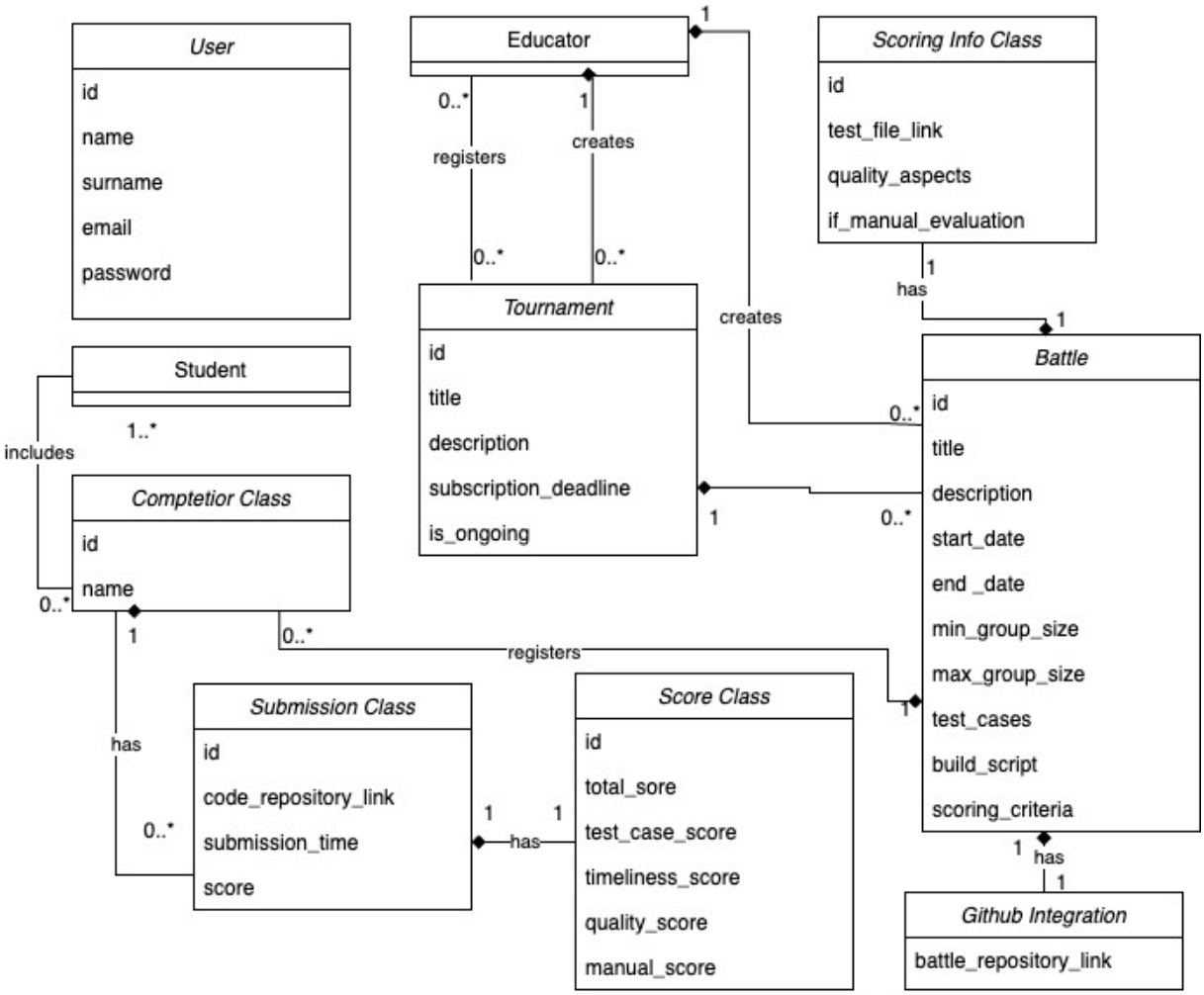
- **Tournament and Battle Classes**: These classes represent the core competitive elements of CKB. A Tournament, created by an Educator, can contain multiple Battles. Each Battle, associated with a specific Tournament, includes attributes like description, start_date, end_date, min-max of group size, and scoring_criteria.

- **Competitor Class**: Reflecting the collaborative aspect of CKB, the Competitor class includes attributes like name and members, the latter being a collection of Students in case of team formation. If a student registers individually, the member's field consists of just one member and the name is the student's name.

- **Submission Class**: This class represents the solutions submitted by the Competitor for a Battle. Key attributes include code_repository_link, time_of_submission, and score.

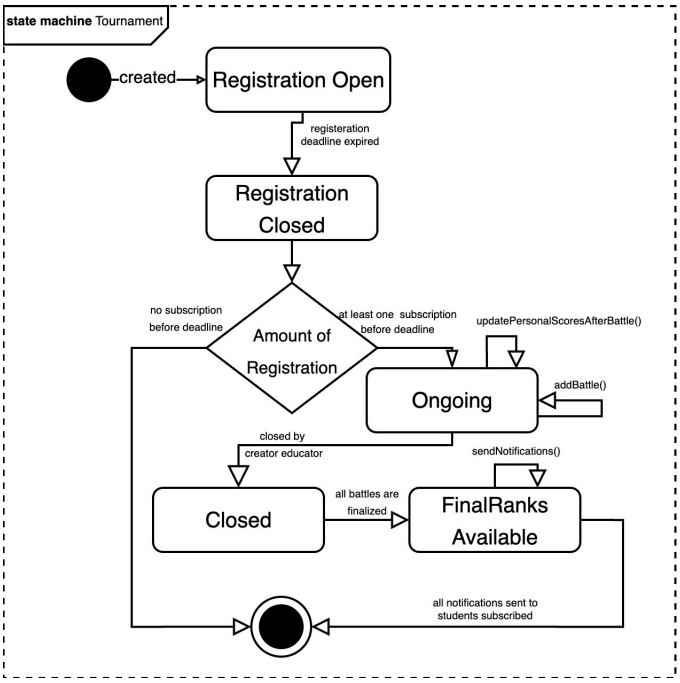
- **GitHub Integration**: A separate class to represent the integration with GitHub, showing how students' code repositories are connected to the CKB platform for automatic evaluation. For every battle, an instance of Github Integration is created and responsible for the API calls from Github Actions and pulling source code from the forked repository.

- **Scoring Class**: This class illustrates the scoring mechanism of CKB, including both automated and manual scoring components as detailed in the project description.



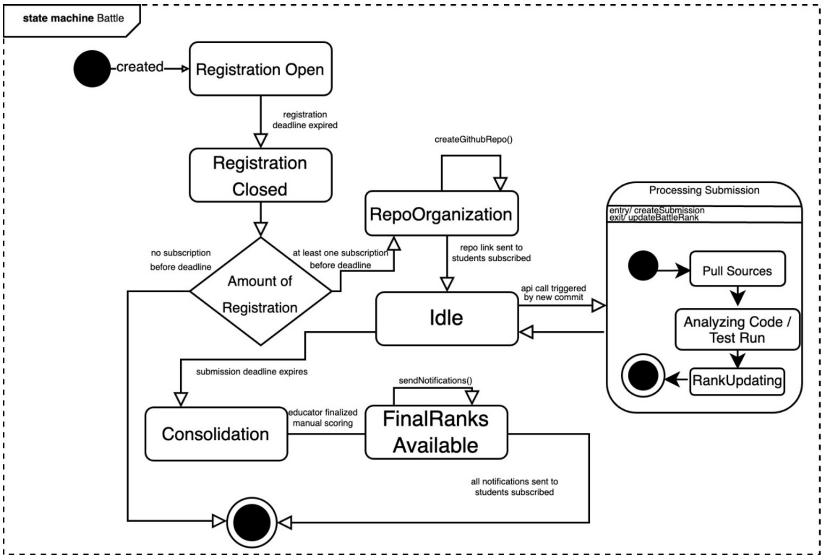
2.1.3 Statecharts

The state charts presented in this section elucidate the operation of the CodeKataBattle (CKB) project, illustrating the state transitions for a code kata battle and a tournament within the platform. Especially for tournaments, a state chart is very crucial to gain a better understanding of the tournament.
Tournament



Firstly, the tournament is in the very first state called "Registration Open", which indicates that Tournament is created and registration is open for students and invited educators. Until the registration deadline, they can register. When the deadline expires registration is closed, so the tournament goes into the "Registration Closed" state. In this state, the process controls the amount of registration for the tournament. If there is no registration it goes to "Final" state. Otherwise, the tournament starts with the "Ongoing" state. In this state, throughout the tournament, educators can create battles, and personal scores are updated after every battle. If the creator educator closes the tournament, it goes to the "Closed" state. However, there can be some battles in the consolidation stages and it is mandatory to finalize them to calculate final ranks. After every battle is finalized, the tournament transitions to the "FinalRankAvailable" state in which notifications about final rankings are sent to the students. After finishing this notification process tournament reaches the "Final" state. There won't be any rank changes or notifications hereafter.

Battle



Firstly tournament is created with the "Registration Open" state. In this state, students can register to battle with teams or individually. When the deadline expires, Battle goes into the "Registration Closed" state. At this time, it is calculated that if there is any registration for battle. If no, it goes to the "Final" state with no registration and any other action. If there are registrations, it goes to a new state called as RepoOrganization in which GitHub repository for the battle is created with proper instructions. Sending the link and instructions triggers the process to go into the "Idle" state. This means that Battle has started and waits for API calls from forked repositories, triggered by a new commit. An API call from repositories of competitor leads to go into "Processing Submission. Here with the "createSubmision" input 3 sequential states follow each other, respectively, "Pull Sources", "Analyzing Code / Test Run" and "RankUpdating". In this 3 state code is pulled and analyzed. Then, the score and rankings are updated. Eventually, it returns to the "Idle" state. Until the submission deadline, students' commit can change their scores. When the deadline expires, Battle goes into the "Consolidation Stage" state in which educators can change rankings via manual scorings. When they are finalized, Battle is closed, indicating as state "FinalRanksAvailable". Immediately, the notification sending process starts. After all notifications are sent, Battle goes into the Final state.

2.2 Product Functions

2.2.1 User Authentication

Login and Sign up functions are available for all kinds of users. There is two types of user in the platform, which should be indicated when signing up for the system. In signup, all users are requested to provide the following information respectively: name, surname, email, password, and type. A verification email is sent to the provided email in order to verify the user. After confirmation, the user is redirected to the platform login page. Here, an email and password are required to enter the platform. Also "stay logged in" option is available to avoid the need to enter credentials every time entering the platform.

2.2.2 Tournament

In the platform, educators can create a tournament with a title, description, and deadline for registration. The title and description should briefly explain tournament content. Until the given subscription deadline, students can subscribe to the tournament. Also creator should be able to invite some of her/his colleagues to a tournament in order to create battles in it. While the tournament continues, educators create battles in which students can participate. Scores students gained from battles are summed up for the leaderboard of the tournament. After the creator educator closes the tournament, if all battles are finalized, all scores from battles are summed up to build the final leaderboard including rankings. In the period of the tournament, students are notified with emails about new battles. Also, at the end of the tournament, final rankings are sent to students.

2.2.3 Battle

An educator in a tournament can be able to create battles in it. These battles are created with titles, descriptions, start-end dates, min-max group sizes, test cases, build scripts, and scoring criteria. Scoring criteria are defined by the educator. It should provide the system with a test file including test cases, whether there is manual evaluation, and quality aspects chosen from a list in the system such as efficiency, clean code, etc. With this information system can evaluate the submissions.

3 Specific Requirements

Organize this section according to the rules defined in the project description.

4 Formal Analysis Using Alloy

Organize this section according to the rules defined in the project description.

5 Effort Spent

Provide here information about how much effort each group member spent in working at this document. We would appreciate details here.

References

