Accademic Year: 2023-2024

POLITECNICO
MILANO 1863

Requirements Analysis and Specification Document

CodeKataBattle

SOFTWARE ENGINEERING 2

PROFESSOR DI NITTO ELISABETTA

CONTI ALESSANDRO CODICE PERSONA: 10710583 MATRICOLA: 252665

DI PAOLA ANTONIO CODICE PERSONA: 10717589 MATRICOLA: 956038

Table of contents

1. INTRODUCTION

1.1. Purpose

The purpose of our project, named CodeKataBattle (abbreviated as CKB), is to provide a digital platform that enables students to develop and refine their skills in software development through a collaborative experience. The CKB platform allows students to practice in a programming language of their choice. The exercises offered on this platform are created by educators and require students to complete the code or parts of it, adding the missing components they deem appropriate. The code is then executed and subjected to specific tests created by educators to verify its correctness.

Exercises are organized into tournaments by educators, within which 'battles' are held to evaluate the performance of the students. This process culminates in the creation of performance rankings among all students involved in the battles and tournaments. In this way, CKB encourages friendly competition and the growth of students' skills in the field of software development.

1.1.1. Goal

[G1] Educator creates a tournament and within it, he creates one or more battle.

[G2] Student registers for a tournament and will receive all notifications related to it.

[G3] Student joins a battle within a tournament they are register for and invite some friends to partecipate with them, thus creates a team for the battle.

[G4] Student receives an evaluation for the code he pushed to their GitHub repository.

[G5] Educator reviews all the evaluated code, and if they deem it necessary, he can modify the evaluation they have received.

[G6] Educator closes a tournament and the platform publishes the ranking.

[G7] Both students and educators check the update ranking of the battle and the tournament.

1.2. Scope

1.2.1. World phenomena

Phenomena events that take place in the real world and that the machine cannot observe.

[WP1]. Educator wants to create a tournament

[WP2]. Educator wants to create a battle

[WP3]. Student wants to join a tournament

[WP4]. Student contacts peers participating in the tournament to form a team

[WP5]. Students fork the GitHub repository

[WP6]. Students and Educator wants to subscribe into the platform

1.2.2. Shared phenomena

1.2.2.1. Controlled by the machine and observable by the world

[SP1]. The platform assigns scores to groups to create a competition rank

[SP2]. Students are notified when a new tournament are creates

[SP3]. All student that are subscribed on a tournament are notified of all upcoming battles creation

[SP4]. The platform creates a GitHub repository containing the code Kata and then sends the link to all students who are members of subscribed teams

[SP5]. When the battle ends the system evaluates the student automatically

1.2.2.2. Controlled by the world and observable by the machine

[SP6]. Students join into a tournament

[SP7]. Educator creates a tournament

[SP8]. Educator creates a battle

[SP9]. Educator gives permission to other ones to create battle in the tournament

[SP10]. Group of student join into a battle

[SP11]. Groups deliver their solution to the platform

[SP12]. Students set up the automated workflow in GitHub so that informs the CKB as soon as students push a new commit.

[SP13]. Each push before the deadline triggers the CKB platform, which pulls the latest sources, analyzes them, and runs the tests on the corresponding executable to calculate and update the battle score of the team

[SP14]. At the end of the battle, optionally the educator evaluates the student by overwriting the result that the platform returned

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

- Commit: Commits are the core building block units of a Git project timeline. Commits can be thought of as snapshots or milestones along the timeline of a Git project. Commits are created with the `git commit` command to capture the state of a project at that point in time.
- Fork: A fork is a new repository that shares code and visibility settings with the original “upstream” repository.
- Code kata: Code kata contains the description of a battle and the software project on which the student will have to work, including also test cases and build automation scripts
- Test Case: A test case is a singular set of actions or instructions that the Educator wants to perform to verify a specific aspect of the project pushed by the students. If the test fails, the result might be a software defect that students might haven’t found.
- Upload: Upload in which the educator sends to the platform database the code kata for a specific battle.
- Repository: A Git repository is the .git/folder inside a project. This repository tracks all changes made to files in your project, building a history over time. Meaning, if you delete the .git/folder, then you delete your project’s history.
- Branch: In Git, a branch is a new/separate version of the main repository. Branches allows users to work on different parts of a project without impacting the main branch. That main branch is the one seen as the default one.
- Push: The `git push` command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo.
- Pull: The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.

1.3.2. Acronyms

- CKB: CodeKataBattle
- CK: CodeKata
- GH: GitHub
- GHA: GitHub Action

1.3.3. Abbreviations

- [Gn]: the n-th goal of the system
- [WPn]: the n-th world phenomena
- [SPn]: the n-th shared phenomena
- [UCn]: the n-th use case
- [Rn]: the n-th functional requirement
- Repo: Git repository

1.4. Revision history

1.5. Reference documents

This document is strictly based on:

- The specification of the RASD and DD assignment of the Software Engineering 2 course, held by professor Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli at the Politecnico di Milano, A.Y 2022/2023.
- Slides of Software Engineering 2 course on WeBeep.
- Official link of CodeKata: http://codekata.com/.

1.6. Documents Structure

2. OVERALL DESCRIPTION

2.1. Product perspective

2.1.1. Scenarios

2.1.2. Domain class diagram

2.1.3. Statecharts

2.2. Product functions

2.3. Assumption, Dependencies and Contraints

2.3.1. Domain Assumption

3. SPECIFIC REQUIREMENTS

3.1. External interface

3.1.1. User interface

3.1.2. Hardware interface

3.1.3. Software interface

3.1.4. Communication interface