



POLITECNICO
MILANO 1863

Design Document

Software Engineering 2
CodeKataBattle

Authors

Christian Rossi - 10736464
Kirolos Sharoubim - 10719510

Contents

1	Introduction	2
1.1	Purpose	2
1.2	Scope	2
1.3	Definitions, acronyms and abbreviations	3
1.3.1	Definitions	3
1.3.2	Acronyms	3
1.4	Revision history	3
1.5	Reference documents	3
1.6	Document structure	3
2	Architectural design	5
2.1	Overview	5
2.2	Component view	6
2.3	Deployment view	6
2.4	Runtime view	6
2.5	Component interfaces	6
2.6	Selected architectural styles and patterns	6
2.7	Other design decisions	6
3	User interface design	7
3.1	Flowchart for students UI	7
3.2	Flowchart for educators UI	8
4	Requirements traceability	9
5	Implementation, integration and test plan	10
6	Effort spent	11
7	References	12

Chapter 1

Introduction

1.1 Purpose

This document aims to provide a comprehensive insight into the CodeKataBattle system outlined in the RASD. It delves into the system's architecture, elucidating on its components, their interactions, processes, and algorithms designed to meet RASD requirements. Furthermore, it offers explicit instructions pertaining to the implementation, integration, and testing plan. Geared towards developers, testers, and project managers, this document serves as a valuable reference for the system's implementation phase.

1.2 Scope

CodeKataBattle serves as a platform utilized by educators to engage students in coding Katas - challenges designed to be addressed using a programming language specified by the challenge organizers. Educators possess the capability to establish a code Kata within a particular tournament, defining essential parameters such as the problem statement (inclusive of test cases), the minimum and maximum number of students allowed per group, and the deadlines for both code Kata registration and solution submission. Following the tournament creation, students are empowered to create groups and commence their collaborative efforts on the solution, following a test-first approach. As the ultimate deadline approaches, the system autonomously computes the final rankings, ultimately revealing the victorious participants. For a more comprehensive overview of the features accessible to end users, please consult the RASD. The architecture of the S2B is structured into three physically separated layers, each installed on distinct tiers. These layers are:

1. *Presentation Layer*: Responsible for overseeing the presentation logic and handling all interactions with end users.
2. *Business Logic Layer*: Manages the application functions provided by the S2B, ensuring seamless operation and functionality.
3. *Data Layer*: Handles secure storage and facilitates access to data, ensuring the integrity and reliability of information.

1.3 Definitions, acronyms and abbreviations

1.3.1 Definitions

Code Kata: adaptation of the concept of karate katas, where you repetitively refine a form, to the realm of software development, fostering iterative practice and improvement.

Test-first approach: software development process relies on the transformation of software requirements into test cases before the software is completely developed, and it involves monitoring the entire software development by iteratively testing the software against all these test cases.

1.3.2 Acronyms

CKB: CodeKataBattle

CK: Code Kata

1.4 Revision history

Version 1.0 - Release - date 22/12/2023

1.5 Reference documents

Document 1 - Presentation about RASD structure

Website 1 - <http://codekata.com>

Website 2 - <https://en.wikipedia.org>

1.6 Document structure

This document comprises seven sections:

1. *Introduction*: This section furnishes an overview of the Design Document (DD), encompassing the project's scope, key term definitions, references to pertinent documents, and a brief outline of the design.
2. *User Interface Design*: Outlining the design of the user interface (UI) along with user experience (UX) flowcharts, this section provides a detailed perspective on how users will interact with the system.
3. *Architectural Design*: Describing the high-level components and interactions within the system, this section includes a component view, deployment view, runtime view, and insights into selected architectural styles and patterns.
4. *Requirements Traceability*: Establishing a clear connection between the requirements specified in the Requirements and Specification Document (RASD) and the components outlined in the DD, this section ensures traceability throughout the design process.
5. *Implementation, Integration, and Test Plan*: This section details the plan for implementing, integrating, and testing the system. It includes the sequence in which subsystems and components will be implemented, providing a roadmap for the development process.
6. *Effort Spent*: Offering insights into the effort invested in the design process, this section provides information on the resources and time dedicated to the various aspects of the design.
7. *References*: This section encompasses a list of references cited within the Design Document, providing a foundation for further exploration and understanding.

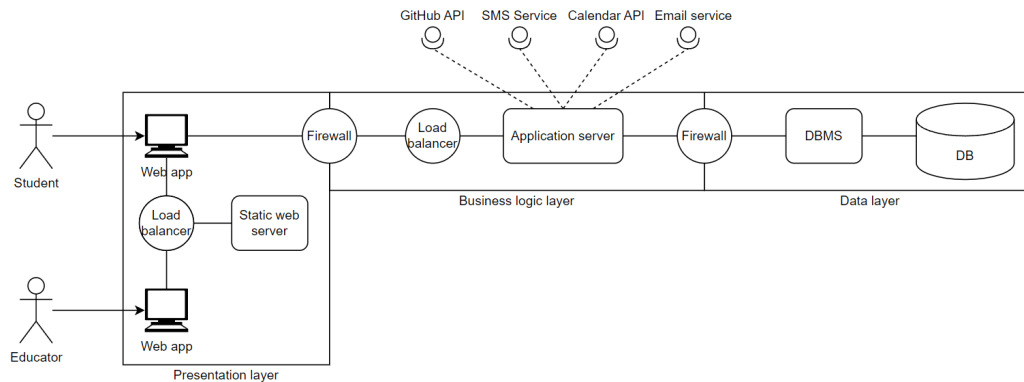
Chapter 2

Architectural design

The purpose of this section is to systematically present and assess the architecture of the S2B system in a top-down approach. We first introduce the comprehensive architecture and then provide a diagram illustrating the system's components, with a specific emphasis on the Educators and students subcomponents. Subsequently, we employ an Entity-Relationship (ER) diagram to articulate the system's logical data and present the system's deployment view, encompassing layers and tiers. We also use sequence diagrams to illustrate crucial runtime perspectives, while class diagrams are employed to scrutinize the interfaces of the components. Ultimately, we engage in a discussion on the architectural design choices and the rationale behind them.

2.1 Overview

The diagram below provides a high-level depiction of the components that form the System. In this document, we will use the term "Frontend" to encompass both the presentation layer and the Client (e.g., the Browser), while the term "Backend" will encompass both the Application Layer and the Data Layer.



2.2 Component view

The service will be accessed through a web interface, utilizing a single page application (SPA) that proves advantageous for this application type, enabling extensive interaction without frequent page reloads, thereby ensuring a faster and smoother user experience. The system's architecture is structured three into distinct layers, with application servers interacting with a database management system and utilizing APIs for data retrieval and storage. Adhering to REST standards, the application servers are designed to be stateless, and the system incorporates firewalls to bolster security.

2.3 Deployment view

2.4 Runtime view

2.5 Component interfaces

2.6 Selected architectural styles and patterns

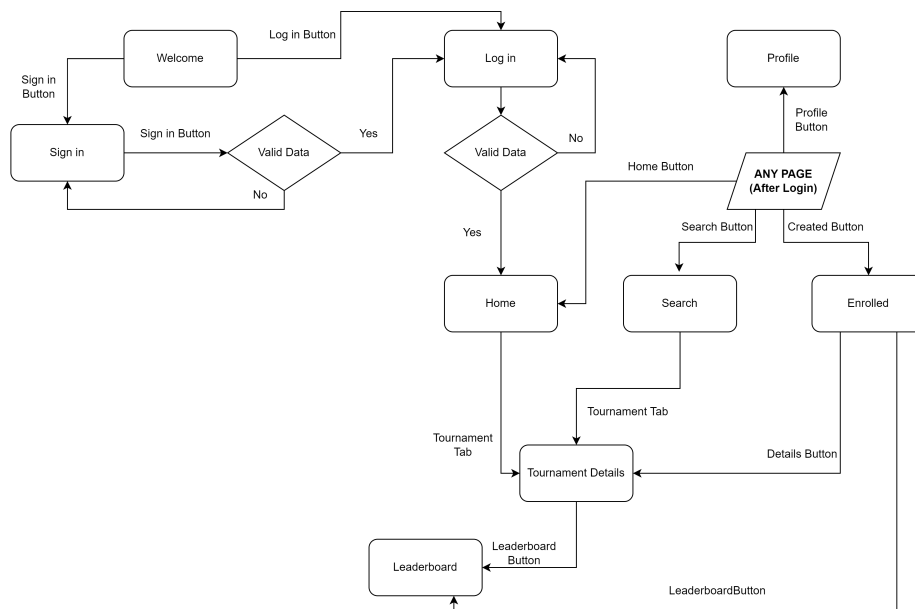
2.7 Other design decisions

Chapter 3

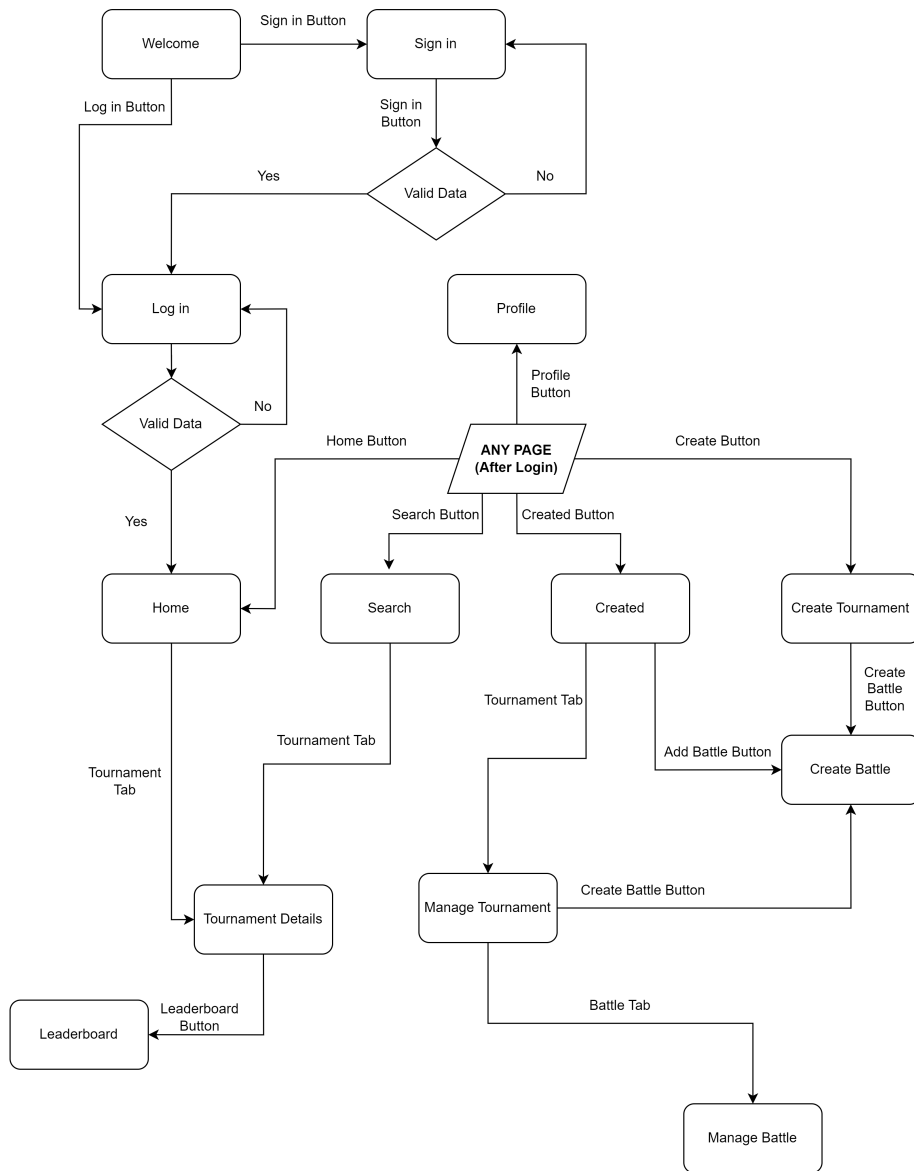
User interface design

The user interface is the visual representation of how the application appears to the end user. It should be intuitive and straightforward, ensuring easy access to all features. Mockups were introduced in the RASD document, and now flowcharts for both students and educators will be presented. Flowcharts offer a concise and clear overview of the interactions and steps involved in utilizing the web app.

3.1 Flowchart for students UI



3.2 Flowchart for educators UI



Chapter 4

Requirements traceability

Chapter 5

Implementation, integration and test plan

Chapter 6

Effort spent

Chapter 7

References