Design Document

Francesco Spangaro - Tosetti Luca - Francesco Riccardi

07 January 2024

POLITECNICO
MILANO 1863

Prof.
Matteo Camilli

Version 0.1
Academic Year 2023 - 2024

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide an exhausting and implementative description of the platform that will be implemented (CKB platform). In particular the document is focused on the description of the architectural styles and decisions that will be adopted, the modules that compose the platform and their interfaces. The document will contains also several details regarding the deployment choices, the runtime view of the core functionalites of the platform that will be used in it. The document contains some mockups of the user interface design. The document also covers the implementation, integration and testing processes required to implement correctly the CKB platform.

1.2 Scope

CodeKataBattle (CKB) is a platform which aims to give to Educators an easy-to-use experience, and let them propose homework and/or lessons in a new and fresh way. The main goal of the platform is to give the Students the possibility to improve and acquire new software developing skills by particpating to several battles in as many tournaments. The platform let Educators of the Students to create such tournaments and battles within them in order to challenge the Students to upload the best possible solution to the battle's problem. That solution will be then automatically evaluated by the platform which will give it a score, and eventually even by the Educator who created the battle, and will be associated to it a proper score. The platform also allow Educators to add several recognition badges for the work done by the students. This badges can be personalized by the Educators themselves.

From the architectural point of view we have decided to adopt a 4-Tier Client-Server architecture combined with a MSA server side, in addition to a MVC software architectural choice.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

Term	Definition
4-Tier Architecture	→ A 4-Tier architecture in the field of informatic systems, very simply a software and hardware architecture in which a running application/platform is divided in four different modules or also called "layers" which usually are: Presentation Layer, WebServer layer, Logic Layer, Data Layer.
Presentation Layer	→ The top layer of the 4-Tier architecture. It takes care of the interaction between the user and the application.
WebServer Layer	→ The second layer of the 4-Tier architecture. It takes care of handling the requests sent by the users throught a browser application.
Logic Layer	→ The third layer of the 4-Tier architecture. It takes care of implementing the real application logic that allows to the application/platform to actually work.
Data Layer	→ The bottom layer of the 4-Tier architecture. It takes care of all the data generated by the users or the application, and with which the application itself has to interact. The interactions can include queries, updates, deletions, ...
Microservice architecture	→ An architectural style that consist in the creation of an application/platform as a suite of small services, each one handling a part of the business logic of the application and that comunicates with each other throught lightweight protocols (such as HTTP).

Term	Definition
Model-View-Controller	→ An architectural pattern used to develop the software logic of an application. This pattern consist in dividing the application in three different parts: View, Model and Controller.
View	→ Part of the MVC pattern which takes care of the visualization of the data contained in the model and the interaction with the user.
Controller	→ Part of the MVC pattern that receives commands from the user, and execute them by modifying the View and/or Model parts.
Model	→ Part of the MVC pattern that gives to the Controller part, the methods to access the application's data and to modify them.

1.3.2 Acronyms

Acronym	Meaning
MSA	→ MicroServices Architecture
MVC	→ Model-View-Controller
RASD	→ Requirement Analysis and Specification Document
DD	→ Design Document
CKB	→ CodeKataBattle
	→

1.3.3 Abbreviations

Abbreviation	Meaning
e.g.	→ Exempli gratia, latin phrase meaning "for example".
	→
	→
	→
	→

1.4 Revision history

- **Placeholder data**: version 1.0

1.5 Reference documents

UML official specification → `https://www.omg.org/spec/UML`

Sequence diagrams specification → `https://www.uml-diagrams.org/sequence-diagrams.html`

Component diagrams specification → `https://creately.com/blog/software-teams/component-diagram-tutorial/`

Deployment diagrams specification → `https://pubs.opengroup.org/architecture/archimate32-doc.singlepage/`

1.6 Document structure

- ***Section 1: Introduction***
  This section offers a brief description of the problem and the platform/application that will be developed in order to resolve it. It describes the major purpose of this document, a very brief recap of the domain which is described in detail in the RASD document. In addition, in this section are inserted definitions, acronyms and abbreviations used in the document, its revision history and refereced documents or web pages.
- ***Section 2: Architectural Design***
  This section is the main part of the document. It describes the architectures used to realize the platform, the CKB platform's components, its interfaces, its deployment structure and finally its runtime behaviour.

All these aspects are described through several diagrams such as: component diagrams, class diagrams, deployment diagrams and other generic diagrams which are used to give a representation of main and most important features of the platform.

- ***Section 3: User interface design***
  This section describes the user interface design of the platform. It contains several mockups of the interface that the Educators and Students will find when they access to the platform. The presented mockups refers to the client-side experience throught an appropriate browser application.
- ***Section 4: Requirements traceability***
  This section describe the connection between the RASD and DD document, by providing a complete map of the requirements and goals expressed in the RASD to the modules presented in this document.
- ***Section 5: Implementation, Integration & Test plan***
  This section describes the plan followed for implementing, testing and integrating the platform's components, the order in which these operations are performed and what they generate.
- ***Section 6: Effort spent***
  This section contains all the information about the time spent by each group member in order to complete this document and its division by each section of the document.

2 Architectural Design

2.1 Overview: High-level components and interactions

To ensure high maintainability, security and reliability, the service is structured by following the four-tier architecture model. Figure 1. shows how the tiers are divided, and what are the relations between each tier of the system.

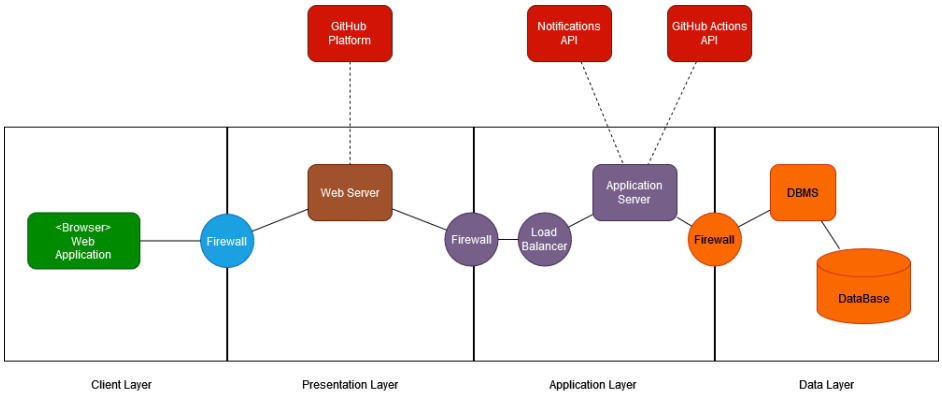

Figure 1: Four-tier architecture

The main components are:

Web Application: The web application allows users to connect to the platform's services. The web application can be accessed by any device that is connected to the internet and can browse the web. Students and Educators will have different views of the web application, because they have different parameters that they can view and manage.

Web Server: The web server is what manages the web application. It is connected directly to the GitHub Platform to give Educators the possibility of seeing their Students' uploaded solutions. It is connected to the GitHub platform to also give Students the possibility of seeing their solutions uploaded to the platform in real time. It is the main container for the JavaScript and general backend code for the platform. The web server is connected to the Application server because it needs to be automatically updated when new grades for Students' solutions are generated by the platform.

Application Server: The application server is the main backend of the CKB Platform. It contains all the code needed for th eplatform to run smoothly and without interruptions. It contains the logic needed to answer the API requests made by the users to the platform. It automatically evaluates Students' solutions proposed via the GitHub platform and uploads the new grade to the web server.

DBMS: The DBMS is the main interpreter between the CKB platform and the data stored onto the database.

DataBase: The database stores all the information needed by the application.

External Services: These services provide informations and funtionalities that the CKB platform alone could not provide. These funtionalities include a *GitHub actions API* that notifies the platform of the upload of a new Student's solution, a *notification API* that notify Students when a new tournament or a new battle in a tournament they are subscribed to is created and a connection to the *GitHub platform* since the code needs to be uploaded from the GitHub platform to the CKB platform automatically.

2.2 Component view

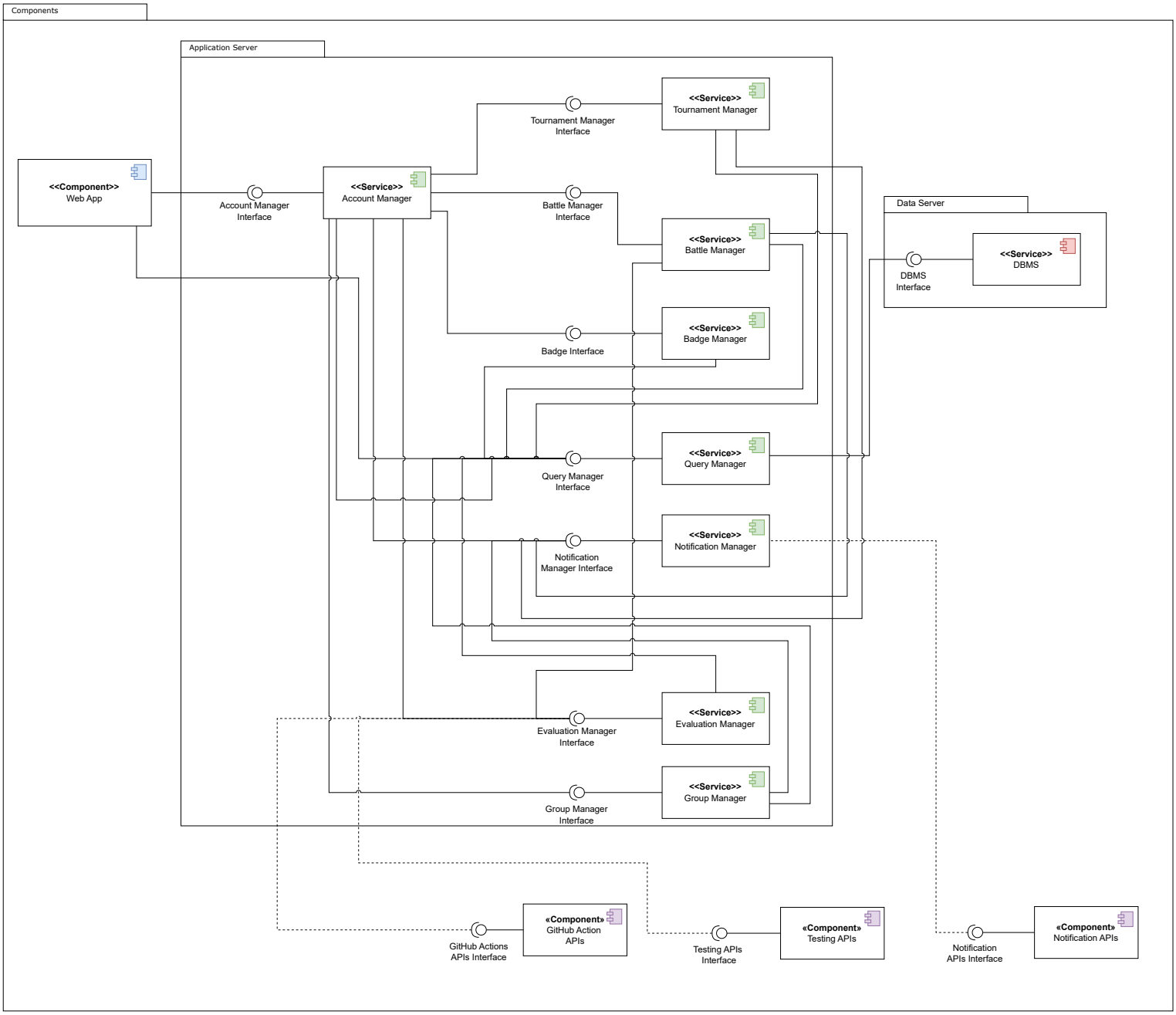

Figure 2: Component diagram

In Figure 2 we can see a more detailed diagram, representing all the components previously described. The Web App is what Students and Educator connect to via their web browsers. It collects all of the Students' and Educators' requests, forwards them to the right service or component and sends back different responses.

Account Manager: The Account Manager is the module that handles the client's logins and creates the session. It checks the client's data and permissions and creates a session with the data he collects. This data session is what is used by the other services and interfaces to grant the user different permission levels.

E.G. A Student can use the Tournament Manager differently from an Educator, since the Student can subscribe to a tournament and see the battles it contains, while an Educator can use the Interface to manage the tournament.

Tournament Manager: The Tournament Manager is the module that lets Educators create and manage tournaments and it lets them grant permission to create battles in their tournaments to other Educators.

It lets Students visualise all open tournaments and subscribe to each tournament they want.

Battle Manager: The Battle Manager is the module that lets Educators create and manage battles within their tournaments, or within tournaments for which they have been granted permission to create and manage battles from the tournament's creator.

It lets Students visualise the battles in tournaments they are subscribed into and lets them subscribe to each battle they want, if they are within the subscription deadline for the battle they want to participate in and if their group respects the battle's requirements.

Badges Manager: The Badges Manager lets Educators create and manage badges for their tournaments. It lets Educators define new rules for obtaining badges in tournaments. It also checks if Students have obtained any badges and updates the Students' pages on rule completion.

Query Manager: The Query Manager is the module that connects the Application Server to the Data Server. It functions as the mediator between the clients' requests and the data stored in the DataBase.

Notification Manager: The Notification Manager is the module in charge of notifying Students of new tournaments' creation, new battles' creation in a tournament they are subscribed into and when they receive an invite to join a group.

The Notification Manager achieves this by interfacing itself with some external Notification APIs, in charge of sending each notification.

Evaluation Manager: The Evaluation Manager is the module responsible for grading each Student's entry as solution for a battle. The Evaluation Manager uses external Testing APIs to test the Students' code for correction checking and the GitHub Actions APIs for downloading the Students' solution they upload on the GitHub platform.

The Evaluation Manager interacts with the Query Manager to update and upload the Students' grades for each battle, after the Evaluation Process ends.

The Evaluation Manager also lets Educators manually evaluate each Student's solution, and upload the grade to the platform via the Query Manager.
Group Manager: The Group Manager lets Students' create new groups to participate in battles with. It lets Students' send invites to other Students.
DBMS: The DBMS is the module in charge of interacting directly with the DataBase, translating each query written by the Query Manager so that the DataBase understands the request, and responding with the correct data requested.
GitHub Actions APIs: The GitHub Actions APIs are external APIs used to have the CKB platform interact with the GitHub platform, since the Sutdents will upload their solutions on the latter.
Testing APIs: The Testing APIs are external APIs used to run tests on the code written by the students and submitted as a solution for each battle.
Notification APIs: The Notification APIs are external APIs used to notify Students of some events happening on the CKB platform.
E.G. A new tournament is created, a new battle is created in a tournament they are subscribed into and they received an invite for joining a group.

2.3 Deployment view

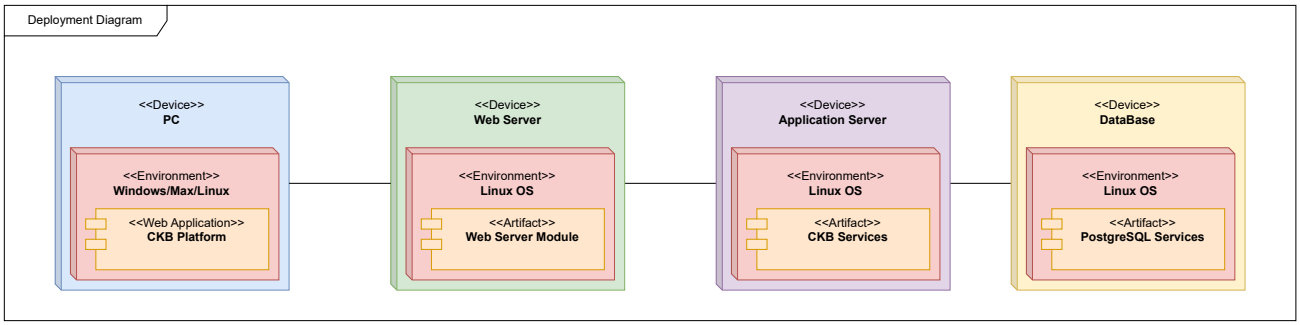

Figure 3: Component diagram

The Deployment Diagram in figure shows the components needed for the correct system behaviour. The External APIs are not included, since they are already implemented and found online.
Each device has its own operating system downloaded, the different tiers in the figure are:
Tier 1: The First tier is the client tier. On the client's machine can run any OS (Windows 11 32 bit, for example) and any web browser. The web application needs to be able to correctly run on any web browser downloaded on any OS.
Tier 2: The Second tier is the Web Application tier. Here no logic is found, is where the CKB platform can be accessed. The Web Server is stored here, is

contains no logic besides the bases needed for a correct execution of the website (CSS, JavaScript and Bootsrap 5.0).
This tier simply receives request from the client and forwards them to the Third tier, it then receives and shows the answer he received from the Third tier to the requesting client.
Tier 3: The Third tier stores the Application Servers. The servers include all the main logic needed by the CKB platform to correctly perform. The Third tier receives all the forwarded requests from the clients, interacts with the Fourth tier to get data, and evaluates a response. The response is then sent up to the Second tier, which will show the answer to the client. Each module is mapped onto this tier. This tier is the one that mainly interacts with all external APIs.
Tier 4: The Fourth tier is composed by the DBMSs needed by the CKB platform for correctly interacting with the DataBase. The DBMSs function as gateways between the Third tier and the actual DataBase. They perform the actions requested by the Third tier and respond with the data requested.

2.4 Component interfaces

2.5 Runtime view

2.5.1 Student sign-up to the platform

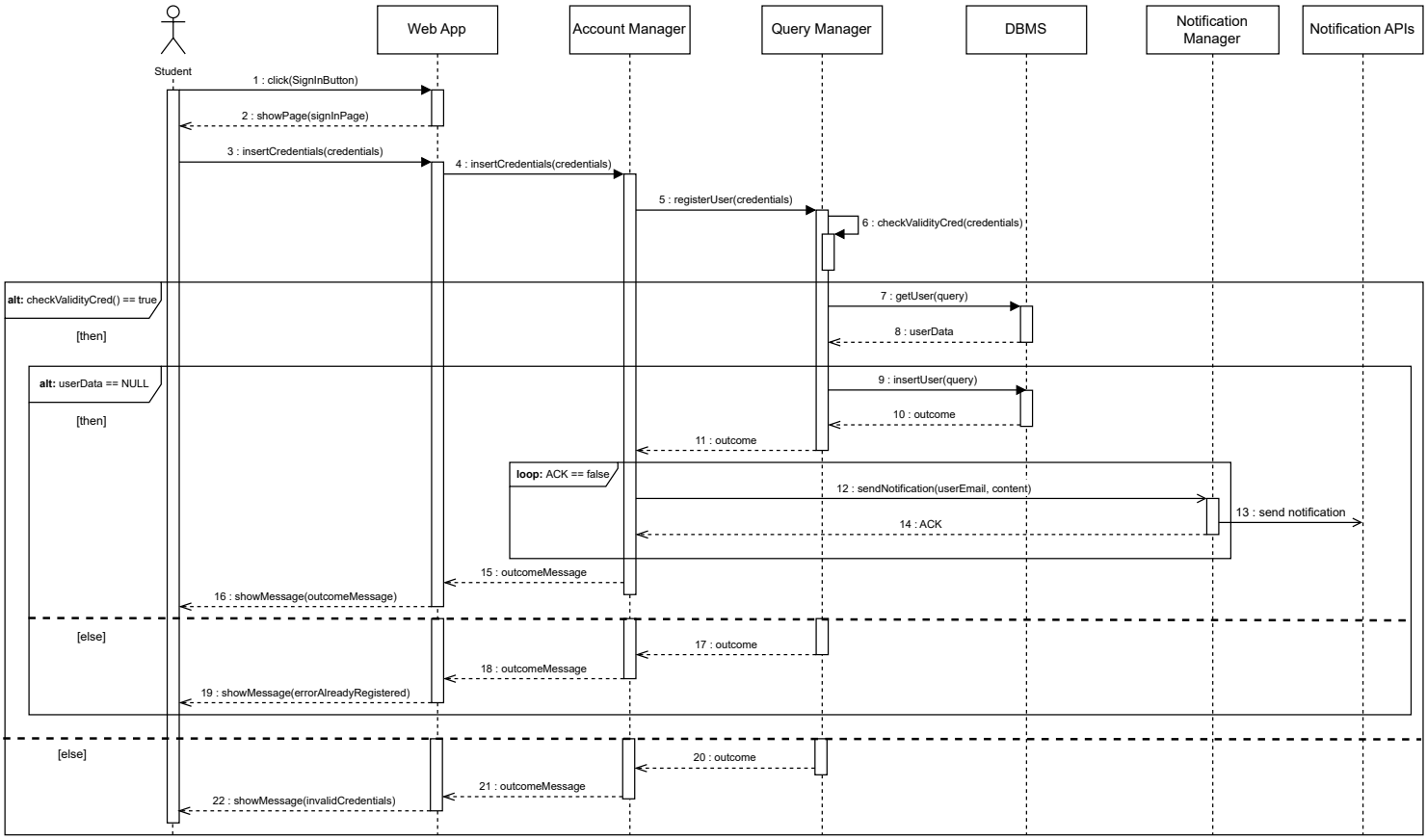

Figure 4: Sign-up sequence diagram

The above diagram represents the process of signing up a Student. The Student accesses to the CKB platform WebApp Homepage through their browser (omitted for simplicity). They then clicks on the "SIGN IN" button as shown in figure **??**. The WebApp will show a form which the Student has to fill with their informations such as name, surname, username that want to use in the platform, email, password, attended school, ...

Once the Student confirms the registration the WebApp contacts the Account Manager which redirects the request to the Query Manager. The Query Manager is in charge of checking the validity of the Student's credentials, i.e. checking whether the name, surname or username inserted by the Student contains some not acceptable characters for some reason, the password don't respects security standards, ...
Other than that the Query Manager takes care to verify if the Student is already registered, i.e. if the used email already appears in the DB. This is done by interfacing with the DBMS.

At this point we can have three possible situations:

1. **Credentials validation went wrong:** In this case the WebApp will show to the Student an error message of invalid credentials
2. **Student has already registered with the same email:** In this case the WebApp will show to the Student an error message stating that they have already registered.
3. **Valid credentials and email used for the first time:** In this case the Query Query Manager proceeds to communicate to the DBMS to insert the Student's data in the DB. Proceeds to inform the Notification Manager to send an email of confirmation. Finally returns the positive outcome of the operation to the Account Manager, which will do the same to the WebApp, which in turn will show a confirmation message to the Student.

2.5.2 Educator sign-up to the platform

Figure 5:

2.5.3 Educator creates a new tournament

Figure 6:

2.5.4 Educator creates new badges

Figure 7:

2.5.5 Educator deletes and/or updates a badge

Figure 8:

2.5.6 Educator creates a new battle

Figure 9:

2.5.7 Educator closes a tournament

Figure 10:

2.5.8 Educator evaluates a battle's results

Figure 11:

2.5.9 Student forms a group

Figure 12:

2.5.10 Student joins a battle

Figure 13:

2.5.11 Student uploads a new solution

Figure 14:

2.5.12 Student uploads a solution after submission deadline

Figure 15:

2.5.13 Student visualizes his tournament's results and badges

Figure 16:

2.6 Selected architectural styles and patterns

2.7 Other design decisions

3 User Interface Design

4 Requirements traceability

- **R.1:** The CKB platform should allow an unregistered Student to create a new account.
  - **Query Manager:** Used to store Students' informations after registration.
- **R.2:** The CKB platform should allow an unregistered Educator to create a new account.
  - **Query Manager:** Used to store Educators' informations after registration.
- **R.3:** The CKB platform must allow access to its pages only if the used credentials are correct.
  - **Account Manager:** Used to check whether the credentials used are correct.
  - **Query Manager:** Used to retrieve users' data.

- **R.4:** The CKB platform must not allow a Student to register more than once in the system.
  - **Account Manager:** Used to check if the Student has already registered to the platform.
  - **Query Manager:** Used to retrieve Student's data.
- **R.5:** The CKB platform must not allow an Educator to register more than once in the system.
  - **Account Manager:** Used to check if the Educator has already registered to the platform.
  - **Query Manager:** Used to retrieve Educator's data.
- **R.6:** Educators can access the platform's services only if they are registered to it.
  - **Account Manager:** Used to check if the Educator is registered to the platform.
  - **Query Manager:** Used to retrieve Educator's data.
- **R.7:** Students can access the platform's services only if they are registered to it.
  - **Account Manager:** Used to check if the Student is registered to the platform.
  - **Query Manager:** Used to retrieve Student's data.
- **R.8:** The CKB platform should not allow Students to create tournaments and/or battles.
  - **Account Manager:** Used to check if the user is a Student and prevent him to access Tournament Manager APIs that should not have access to.
  - **Query Manager:** Used to retrieve Student's data.
- **R.9:** The CKB platform should allow Educators to create battles within a tournament only to the tournament creator and to any other Educator that has been granted permission to do so by the tournament creator.
  - **Account Manager:** Used to check whether the logged Educator can create a battle within a certain tournament.
  - **Battle Manager:** Used to create the new battle within the tournament.
  - **Query Manager:** Used to retrieve Educator's data and eventually store battle's data just created.

- **R.10:** The CKB platform must allow Educators to personalise the tournaments they create.
  - **Tournament Manager:** Used to create and personalize new tournaments.
  - **Query Manager:** Used to store tournament's data.
- **R.11:** The CKB platform must allow Educators to personalise the battles they create.
  - **Battle Manager:** Used to create and personalize new battles.
  - **Query Manager:** Used to store battle's data.
- **R.12:** The CKB platform must allow Educators to define new obtainable badges for each tournament they create.
  - **Badges Manager:** Used to create new badges for a new tournament.
  - **Query Manager:** Used to store badges data.
- **R.13:** The CKB platform must allow Educators to manually evaluate the solutions uploaded by the Students for the battles that the Educators created.
  - **Battle Manager:** Used to check whether the battle is enabled to be manually evaluated.
  - **Evaluation Manager:** Used to allow the Educator to manually evaluate Students' work.
  - **Query Manager:** Used to retrieve Educator and battle's data. Used to store scores given by the Educator through the manual evaluation.
- **R.14:** The CKB platform must allow Educators to delete or update badges before finalizing a tournament's creation.
  - **Badges Manager:** Used to update or delete some badges during tournament creation.
  - **Query Manager:** Used to retrieve old badges's data and store the new ones.
- **R.15:** The CKB platform must allow Educators to define rules to obtain badges in tournaments created by them.
  - **Badges Manager:** Used to define the rules to obtain the badges.
  - **Query Manager:** Used to store badges's obtaining rules.
- **R.16:** The CKB platform must ensure that badges' characteristics respect guidelines regarding their name, icon format and rules to obtain them.

  - **Badges Manager:** Used to check if the characteristics of the new badges that the Educator want to create, or the modifications made to the ones already created respect some constraints.

- **R.17:** The CKB platform must allow Educators to create new tournaments.
  - **Tournament Manager:** Used to create new tournaments.
  - **Query Manager:** Used to store tournament's data.
- **R.18:** The CKB platform must ensure that tournaments' characteristics respect guidelines regarding their name, deadline, access method, programming language.
  - **Tournament Manager:** Used to check if the characteristics of the tournaments that the Educator want to create, respect some constraints.
- **R.19:** The CKB platform must allow Educators to close tournaments they have created.
  - **Tournament Manager:** Used to allow Educators to close the tournaments that they have created.
  - **Query Manager:** Used to retrieve old tournament's data and store the new ones.
- **R.20:** The CKB platform must ensure that when a tournament is closed, Educators cannot create new battles within it.
  - **Tournament Manager:** Used to prevent Educators to create new battles within closed tournaments.
  - **Battle Manager:** Used to try to create a new battle.
  - **Query Manager:** Used to retrieve the data of the tournament in which the Educator want to create a new battle.
- **R.21:** The CKB platform must ensure that if a group uploads a solution to a battle after the submission's deadline, that solution will not be considered in the score computation by preventing its upload.
  - **Battle Manager:** Used to check if the submission phase of the battle has ended.
  - **Query Manager:** Used to retrieve battle's data.
- **R.22:** The CKB platform must ensure that the score given to a group in a tournament is coherent with scores given to the same group in the battles they have partecipated in.

  - **Evaluation Manager:** Used to automatically evaluate Students' uploads, give them a score and subsequently update Students' tournament score.
  - **Query Manager:** Used to store Students's scores.

- **R.23:** The CKB platform must ensure fair competition between group scores. In the tournament's evaluation, the final group score should be the average score of all the battles in the tournament for each group. Any battle with no solution submitted will count as 0 points.
  - **Evaluation Manager:** Used to automatically evaluate Students' uploads. In case of no upload by a group assigns to them 0 points.
  - **Query Manager:** Used to store Students's scores.

- **R.24:** The CKB platform must allow Students to subscribe to a tournament.
  - **Tournament Manager:** Used to allow Students to register to a tournament.
  - **Query Manager:** Used to store registrations' data.

- **R.25:** The CKB platform must allow Students to subscribe to a tournament's battle within the registration deadline.
  - **Battle Manager:** Used to allow Students to register to a battle if the registration deadline has not expired yet.
  - **Query Manager:** Used to retrieve battle's data and store Student's registration to the battle.

- **R.26:** The CKB platform must allow Students to submit solutions to a tournament's battle within the battle's deadline relying on the external GitHub service.
  - **Evaluation Manager:** Used to retrieve group's solution from GitHub.
  - **Query Manager:** Used to store solution's data.

- **R.27:** The CKB platform must allow Students to send and receive group invitations to and from other Students in order to form groups.
  - **Group Manager:** Used to search Students and send them invitations to groups.
  - **Notification Manager:** Takes care of sending emails to the invited Students.
  - **Query Manager:** Used to retrieve Students' data and store invitation and group's data.

- **R.28:** The CKB platform should allow Students to join a battle only if the group composition rules for that battle are complied with.

  - **Battle Manager:** Used to check whether the group that is trying to access the battle is violating some battle's rules, if not allows it to register to the battle.
  - **Query Manager:** Used to retrieve battle's data and store registration's data.

- **R.29:** The CKB platform must ensure that solutions uploaded by a Student for a battle are evaluated.
  - **Evaluation Manager:** Used to evaluate solutions uploaded on GitHub by Students.
  - **Query Manager:** Used to store Students' scores.

- **R.30:** The CKB platform must ensure that only the latest solution uploaded by a Student for a battle he is subscribed to will be taken into consideration for the final score.
  - **Evaluation Manager:** Used to evaluate the last solution uploaded and overwrite Students' score according to it.
  - **Query Manager:** Used to store Students' scores and new solutions' data.

- **R.31:** The CKB platform must allow groups partecipating in a battle to change their solution, if the battle's submission deadline hasn't expired yet.
  - **Evaluation Manager:** Used to retrieve latest group's solution from GitHub and evaluate it.
  - **Query Manager:** Used to store Students' scores and new solutions' data.

- **R.32:** The CKB platform must allow an Educator to modify the score for a Student's solution.
  - **Evaluation Manager:** Used to allow an Educator to manually evaluate Students' work and change their score.
  - **Query Manager:** Used to retrieve data of Students' work, score and store the new score.

- **R.33:** The CKB platform must ensure that when a new tournament is created, all Students subscribed to the platform are going to receive a notification.
  - **Notification Manager:** Used to send a notification to all the Students registered to the platform.
  - **Query Manager:** Used to retrieve Students' data.

- **R.34:** The CKB platform must ensure that when a new battle is created in a tournament, all Students subscribed to that tournament are going to receive a notification.
  - **Notification Manager:** Used to send a notification to all the Students registered to the tournament in which the battle will take place.
  - **Query Manager:** Used to retrieve Students' data.
- **R.35:** The CKB platform must allow Students to visualise the score they obtained in a battle they partecipated in.
  - **Battle Manager:** Used to allow Students access the score obtained in a battle they partecipated to.
  - **Query Manager:** Used to retrieve battle's data and specifically Student's results in it.
- **R.36:** The CKB platform must allow Students to visualise the score they obtained in a tournament they partecipated in.
  - **Tournament Manager:** Used to allow Students access the score obtained in a tournament they partecipated to.
  - **Query Manager:** Used to retrieve tournament's data and specifically Student's results in it.
- **R.37:** The CKB platform must allow Students to visualise the badges they obtained.
  - **Account Manager:** Used to allow Students to access their, or others' profile pages and visualize the badges in it.
  - **Query Manager:** Used to retrieve Student's data.
- **R.38:** The CKB platform must ensures that battles' characteristics respect guidelines regarding their name, deadlines, programming language, number of member per group.
  - **Tournament Manager:** Used to check if the characteristics of the battles that the Educator want to create, respect some constraints.

5 Implementation, Integration & Test plan

6 Effort Spent

Group member		Effort spent
Francesco Spangaro	Introduction	*Xh*
	Architectural Design	*Xh*
	User Interface Design	*Xh*
	Requirement Traceability	*Xh*
	Implementation, Integration and Test plan	*Xh*
Luca Tosetti	Introduction	*Xh*
	Architectural Design	*Xh*
	User Interface Design	*Xh*
	Requirement Traceability	*Xh*
	Implementation, Integration and Test plan	*Xh*
Francesco Riccardi	Introduction	*Xh*
	Architectural Design	*Xh*
	User Interface Design	*Xh*
	Requirement Traceability	*Xh*
	Implementation, Integration and Test plan	*Xh*

7 References