Summer Student Report

gadea.lucas.perez@cern.ch

# DTH DAQ Phase 2 TestBench

**Gadea Lucas Pérez**
**Supervisor: Dominique Gigi**
*CERN, Geneva, Switzerland. CMS Experiment.*

**Abstract**

This report outlines a Summer Student project conducted at CERN, focusing on optimizing the testing and data management process for newly designed Data Acquisition (DAQ) boards in the Compact Muon Solenoid (CMS) experiment. To store the results of the tests carried out on these new boards, the project involved designing a database system and creating Python scripts to automate database creation and data insertion. Additionally, a user-friendly web interface was developed using PHP to display and facilitate easy access to the test results. This initiative streamlines the testing workflow, enhances data organization, and provides researchers with an efficient tool for assessing the performance of CMS DAQ boards.

Geneva, Switzerland
August 24, 2023

# Contents

## List of Figures



## List of Tables

# 1 Introduction

The CMS experiment at CERN is at the forefront of pushing the boundaries of particle physics research, demanding the development and rigorous testing of advanced DAQ boards. This report by a summer student provides a comprehensive overview of the undertaken project, which aims to enhance the efficiency of the testing process and the management of results for newly designed DAQ boards.

The primary focus of the project was to streamline the testing procedure and result storage. To accomplish this, a robust database system was designed to house the test results. Accompanying Python scripts were developed to automate the database creation process and simplify the insertion of new records. This automation significantly expedites the integration of test data into the database, ensuring accuracy and consistency.

Moreover, the project incorporated the visualization of these test results through a user-friendly web interface. By using PHP, a dynamic web page was crafted to present the stored data. This web-based presentation offers easy access to the test results, empowering researchers and engineers to efficiently analyze the performance of the DAQ boards.

To sum up, this summer student project successfully tackled the challenges posed by storing the testing results of the DAQ boards. The implemented database system and automated scripts contribute to a more efficient and organized process of collecting and storing test results. The web interface enhances accessibility and usability, enabling users to seamlessly interpret the performance of the tested DAQ boards.

# 2 Tools

The successful execution of the project involved the utilization of several tools and technologies. These tools played a pivotal role in enhancing the efficiency and effectiveness of the testing and management processes. The following sections provide an overview of the key tools deployed during the course of the project, showcasing their contributions to achieving project objectives.

## 2.1 Programming Languages

The following programming languages were essential for the successful completion of the project:

- **Python:** Used to create automation scripts for database management and data insertion, contributing to the efficiency of the testing process.

- **SQL:** Employed to design and manage the database system, ensuring structured storage and retrieval of test results.
- **HTML:** Utilized to structure the content and layout of the web interface for displaying test results.
- **PHP:** Integrated to create dynamic web pages that present the stored data in an accessible and user-friendly format.
- **CSS:** Applied for styling and visual enhancement of the web interface, providing a polished and consistent appearance.
- **JavaScript:** Used to implement interactive elements on the web interface, enhancing user experience and interactivity. It is worth noting that no special frameworks or libraries were employed to ensure ease of maintenance. However, there are two exceptions: the *Charts.js* library for graphical data representation and libraries for exporting tables in Excel and CSV formats.

### 2.2 Database

The project relied on the following database-related tools:

- **MySQL:** Chosen as the relational database management system to store and manage the test results. Its robustness and compatibility with the project's requirements made it a suitable choice.
- **Python MySQL Connector Driver:** Integrated within Python scripts to establish a connection between the Python environment and the MySQL database. This connector facilitated seamless data insertion and retrieval operations.

### 2.3 Testing Tools

Several testing tools played a crucial role in ensuring the functionality and quality of the project components:

- **Unittest for Python Scripts:** Employed to perform unit testing on the Python scripts responsible for automation and database interaction. This testing framework ensured the reliability and correctness of the implemented functionalities.
- **Selenium for Web Pages:** Utilized to conduct automated testing of the web interface's interactive elements and user flows. This tool validated the seamless functioning of the user-facing components.
- **W3C Web Validator [4]:** Implemented to validate the HTML and CSS code of the web interface against industry standards set by the World Wide Web

Consortium (W3C). This validation ensured compliance with best practices and enhanced cross-browser compatibility.

### 2.4 Documentation

The documentation process, vital for maintaining transparency and clarity, employed a combination of tools and formats. In addition to the comprehensive in-code comments detailing the implementation, the project's documentation extensively utilized Markdown documents. Markdown, a lightweight markup language, facilitated the creation of README files, user manuals, and programmer manuals. These documents serve as accessible resources, guiding users and developers through the functionalities and technical aspects of the project.

For documentation related to the test plan, PDF files were generated to provide detailed information. These PDF files offer structured insights into the testing procedures, helping to ensure comprehensive testing coverage and robustness.

Moreover, the creation of this report was carried out using LaTeX, a typesetting system renowned for its capabilities in producing professional and structured documents.

## 3 Database

The results obtained from performing the tests on the boards will be stored in this database. This section delves into the database-related aspects of the project.

### 3.1 Database Design

The design of the database was a critical aspect of the project, with a focus on achieving modularity and extensibility.

- **Entity-Relationship Diagram (ERD):** A visual representation of the database structure was created using an ERD (see figure 1). This diagram illustrated the relationships between various entities and helped in conceptualizing the database's organization.
- **Modularity and Extensibility:** The database design aimed at modularity and extensibility, allowing for easy incorporation of new data attributes or entities without major disruptions to the existing structure. This flexibility is essential for accommodating potential future requirements.

# Database Entity Diagram

This diagram follows UML (Unified Modeling Language) notation (https://www.uml.org/).

**NOTE:** Relationships between tables (entities) follows the UML convention where **'0'** is related to *no entity*, **'1'** is related to just *one entity* and **'*'** means related to *many entities*.

**EXAMPLE**: A **1...*** relationship means that one entity (from left table) can be related to many entities (from right table). For example, a country can have multiple cities, but each city belongs to only one country.

**NOTE:** Arrows also have their own meaning. To have more information (specially for composition ◆), check this link: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/

**NOTE:** Note that the "NOT NULL" annotation has been removed from PKs since, by their nature, they are assumed to be non-nullable (the "UNIQUE" annotation is also assumed).

**About TIME and DATE**

It is important to consult the format for Date and Time fields, because data types may differ when changing from SQL to other programming languages, especially if it is O.O. (Object Oriented) where the Impedance Mismatch problem arises.

As we are using MySQL, we need to keep this issues in mind:

- **DATE:** MySQL retrieves and displays DATE values in 'YYYY-MM-DD' format.
- **TIME:** MySQL uses the `'HH:MM:SS'` format for querying and displaying a time value. It also allows you to use the `'HHMMSS'` format without delimiter ( : ).

(https://dev.mysql.com/doc/refman/8.0/en/date-and-time-types.html)

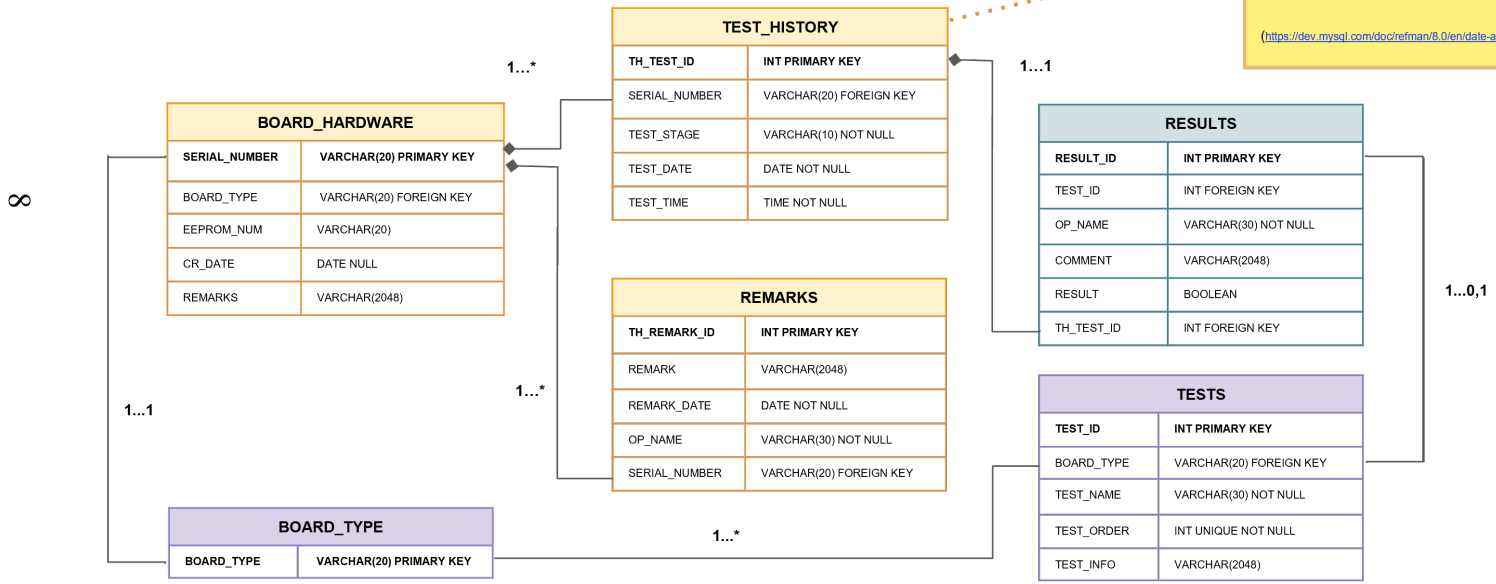


Figure 1: Entity-Relationship Diagram (ERD) illustrating the database structure.

## 3.2 Database Management

This subsection focuses on the management of the database, detailing the creation of Python files responsible for automating both the database creation and data insertion processes. A Packages Diagram is included to provide a visual representation of the content discussed in this section (see figure 2).

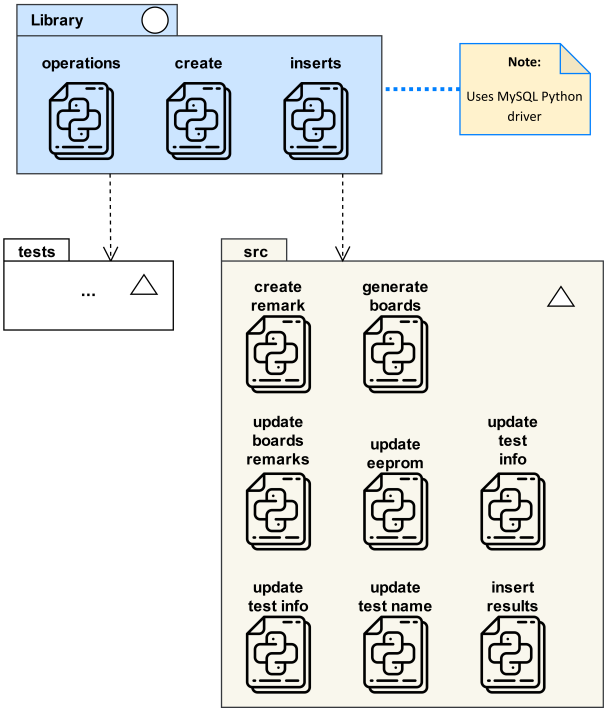


Figure 2: Packages Diagram (using icons from [3]).

### 3.2.1 LIBRARY

A Python library has been meticulously developed to encompass all aspects of database management. This comprehensive library facilitates operations such as data insertion, updates, and more, ensuring a high degree of security through error validation. By utilizing this library, developers can execute database operations with confidence, as it minimizes the need for repetitive checks and it also ensures a hight level of security. Given the paramount importance of security in the database realm, this initiative significantly enhances the integrity and reliability of the data management process.

#### 3.2.2 SRC

Within the `src` directory, Python scripts have been crafted to automate the data insertion process. These scripts are designed to be invoked either from the command line or other Python files. They leverage the aforementioned library to carry out efficient and secure data insertion tasks.

#### 3.2.3 TEST

For each Python script in the `src` directory, corresponding test scripts have been developed in the `test` directory. These tests rigorously assess the functionality and reliability of the database management scripts. The testing framework ensures that any potential issues are identified and rectified early in the development process, enhancing the overall quality of the solution.

The structure outlined above underscores the project's commitment to efficient, secure, and reliable database management, facilitating streamlined testing and maintenance procedures.

### 3.3 Documentation

The project places a strong emphasis on comprehensive documentation, both within the codebase and in supplementary files. A meticulous approach has been taken to ensure that every aspect of the project is well-documented and easily understandable.

Within the code, every module, function, and method is thoroughly documented using detailed header comments. These comments encompass various aspects such as input arguments, expected output, functionality, exceptions, and even include illustrative usage examples. This level of documentation within the codebase aims to make the understanding and utilization of the implemented components seamless for both current developers and future maintainers.

In addition, each package, without exception, is accompanied by a dedicated README file in Markdown format. These README files provide a comprehensive explanation of the package's content, functionalities, and guidelines for utilization. These documents serve as essential reference points for anyone engaging with the project's components, ensuring they can easily comprehend their purpose and integration.

Furthermore, the testing aspect of the project has been extensively documented. Detailed documentation regarding the testing strategies, methodologies, and test cases can be found in a dedicated PDF file within the project repository. This document offers insights into the rigor and thoroughness of the testing process, enhancing the project's reliability and overall quality.

At the conclusion of that PDF document, a glossary is included, providing explanations for key concepts mentioned throughout. Additionally, an annex is appended, outlining best practices in secure programming. These practices have been implemented to ensure the utmost security of the database and its associated functionalities.

# 4 Webpage

The development of a dynamic and user-friendly webpage stands as a pivotal aspect of the project, enhancing visualization of relevant data. This section delves into the construction and features of the project's dedicated webpage, detailing the technologies, design principles, and functionalities that contribute to its effectiveness.

## 4.1 Requirements

The following requirements outline the functionalities (see table 1) and characteristics (see table 2) of the webpage.

| Id | Requirement | Description |
|---|---|---|
| RNF01 | Maintenance Standards | The webpage's codebase should be consistently updated and well-documented to ensure efficient maintenance and future enhancements. |
| RNF02 | Usability Guidelines | The webpage's user interfaces should follow established usability principles to provide an intuitive and satisfying user experience. |
| RNF03 | Security Protocols | The webpage must implement robust security measures, including data encryption and user authentication, to safeguard user information and prevent unauthorized access. |
| RNF04 | Responsive Design | The webpage should be designed to adapt seamlessly to various screen sizes and devices, ensuring optimal user experience across platforms. |

Table 1: Webpage Non-Functional Requirements

| Id | Requirement | Description |
|---|---|---|
| RF01 | Authentication | Users with CERN e-group access should be able to log in to the webpage. Unauthorized users should be restricted from access. |
| RF02 | Navigation | Users should navigate using breadcrumbs. Clicking on the DTH-DAQ icon should lead back to the homepage. |
| RF03 | Help | A small help section accessible via a blue question mark button on the homepage should guide users on form usage and navigation. |
| RF04 | Form | A form allowing data entry via serial number should query other interfaces and display relevant information. Input validation and error handling must be implemented, including border highlighting for incorrect inputs. |
| RF05 | Board Listing | The interface should display a table with board information (SERIAL NUMBER, BOARD TYPE, EEPROM Number, Date, Remarks). Table sorting (ascending and descending, numeric and alphabetic), search by serial number, pagination, and export to CSV and Excel formats should be available. |
| RF06 | Additional Functions | After entering a serial number, users should access a main menu with options:<br>• Help: Access an online guide explaining basic functionalities.<br>• Remarks: View board remarks in a table format (Author, Comment, Date).<br>• Testbench: Access a historical test record interface.<br>• Statistics: Display general and specific (per board) graphs for test status assessment. |
| RF07 | Test Bench | Access a historical test record interface with entries (Serial Number, Stage, Date, Time). Entries should be expandable for detailed test results (Order, Action, Result, Comment, Operator), sortable, and exportable to CSV and Excel formats. |

Table 2: Webpage Functional Requirements

## 4.2 Procedural design

In this section, we provide an overview of the use case scenarios and the primary potential interactions between the user and the webpage.

### 4.2.1 Actors

Before continuing, it is necessary to clarify who the actors involved in the use case scenarios are. They are categorized as follows:

- **CERN User**: This actor represents the parent user who possesses CERN credentials. The CERN User has the highest level of access and inherits the characteristics of both *User* and *Non User* actors.
- **User**: This actor represents authenticated users with access permissions to the web application. They have the privilege to interact with various features of the system.
- **Non User**: This actor represents users without access permissions. Non Users are restricted from interacting with the system's functionalities.

Access privileges are strictly enforced, with Users being granted comprehensive access, while Non Users have no interaction rights within the system. To simplify the representation of interactions, in the subsequent sequence diagrams, we will focus exclusively on the authenticated User actor with access permissions.

Please, note that while we refer to the authenticated User with access permissions as "User" it should be understood that this refers specifically to the privileged user type within our system.

### 4.2.2 Use Cases

The use cases capture the interactions between different actors and the system, providing a comprehensive view of the system's functionality (for Use Cases Diagram, see figure 3).

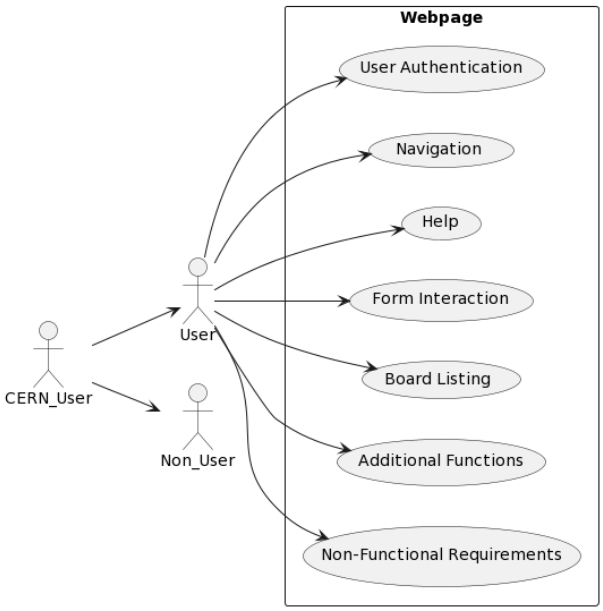


Figure 3: Use Cases Diagram.

The following use cases outline the main scenarios depicting the interactions between users and the webpage's functionalities.

#### 4.2.3 Use Case 1: User Authentication

**Actor:** Authenticated CERN e-group user
**Scenario:** User logs in using CERN e-group credentials.
**Outcome:** User gains access to the webpage's functionalities.
See figure 4.

#### 4.2.4 Use Case 2: Navigation

**Actor:** Authenticated user
**Scenario 1:** User navigates the webpage using breadcrumbs. User clicks on the breadcrumb for the homepage.
**Scenario 2:** User clicks on the DTH-DAQ icon.
**Outcome:** User returns to the homepage.
See figure 5.

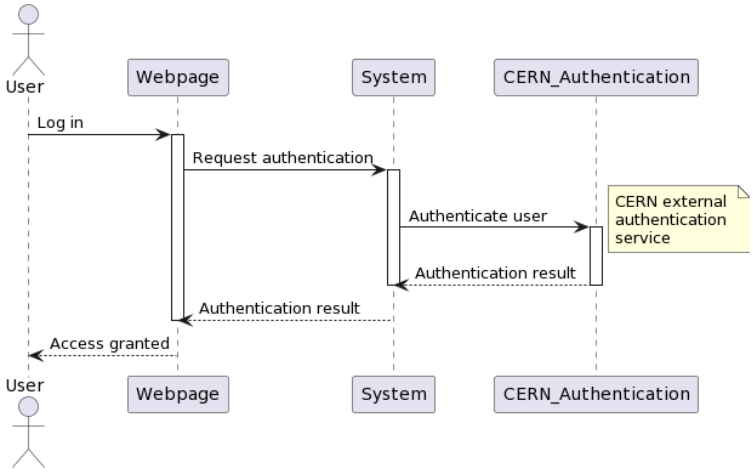


Figure 4: Sequence Diagram for Use Case 1.

**Note:** The *Webpage* participant will symbolize the web interfaces, and the *System* participant will denote the logical processing (implemented using JavaScript or PHP) taking place on either the client or server side, respectively.

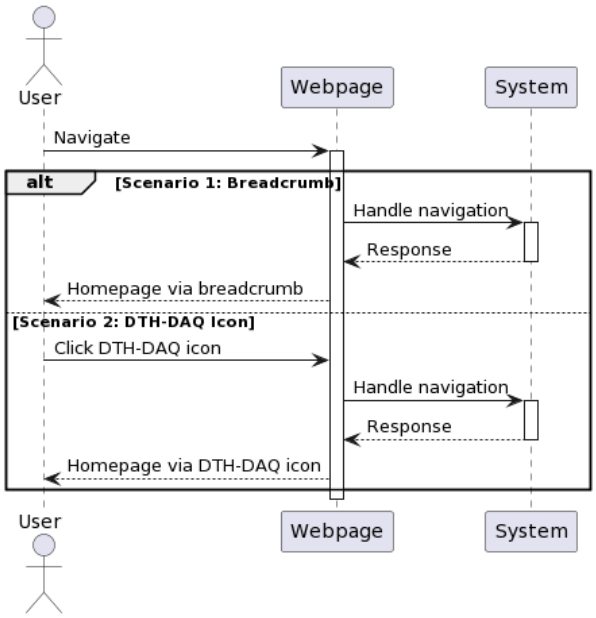


Figure 5: Sequence Diagram for Use Case 2.

#### 4.2.5 Use Case 3: Help

**Actor:** Authenticated user
**Scenario:** User clicks on the blue question mark button on the homepage.
**Outcome:** User accesses a help section providing guidance on form usage and navigation.
See figure 6.

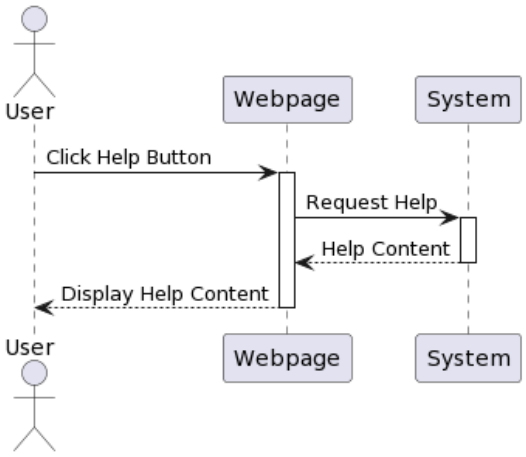


Figure 6: Sequence Diagram for Use Case 3.

#### 4.2.6 Use Case 4: Board Listing

**Actor:** Authenticated user
**Scenario:** User accesses the board listing interface from the homepage.
**Outcome:** User views a table displaying board information. Users can sort columns, search by serial number, paginate, and export the table.
See figure 7.

#### 4.2.7 Use Case 5: Form Interaction

**Actor:** Authenticated user
**Scenario:** User enters a serial number in the form and submits the request.
**Outcome:** System queries related interfaces (main menu) and displays relevant information. Incorrect inputs trigger error messages with border highlighting.
See figure 8.

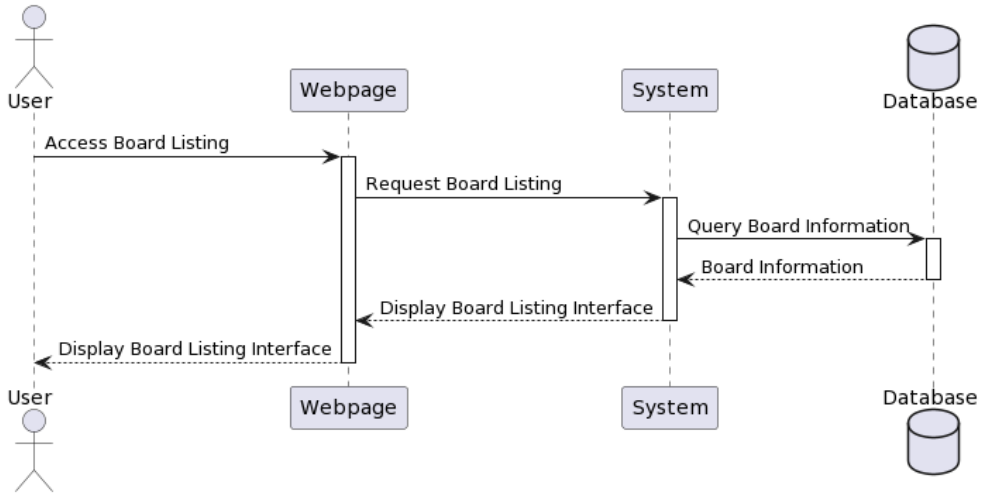


Figure 7: Sequence Diagram for Use Case 4.

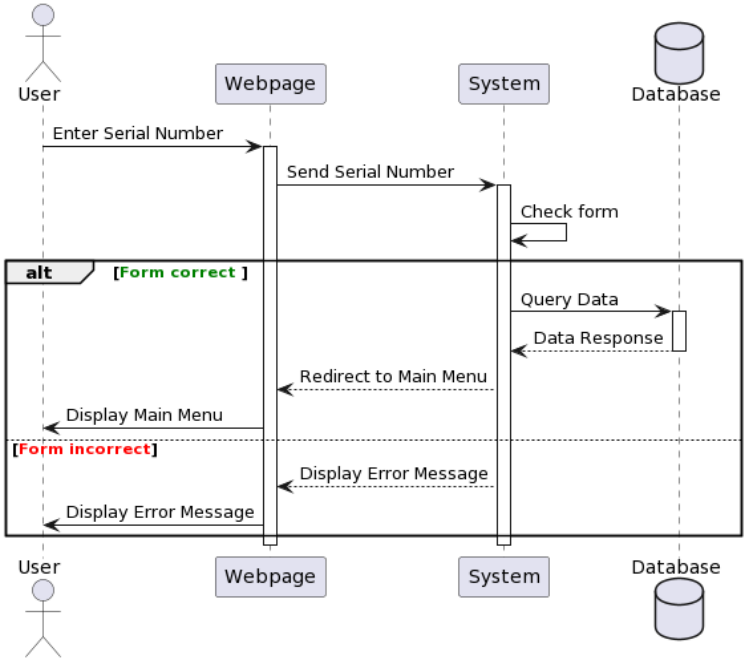


Figure 8: Sequence Diagram for Use Case 5.

### 4.2.8 Use Case 6: Main Functions

**Actor:** Authenticated user
**Scenario:** After the user accesses the main menu, 4 main options are desplayed to be selected. The user selcts one.

**Outcome:** User is presented with options to access:

- Help: Online guide explaining basic functionalities.
- Remarks: View board remarks in a table format.
- Testbench: Historical test record interface.
- Statistics: Display graphs for test status assessment.

See figure 9.

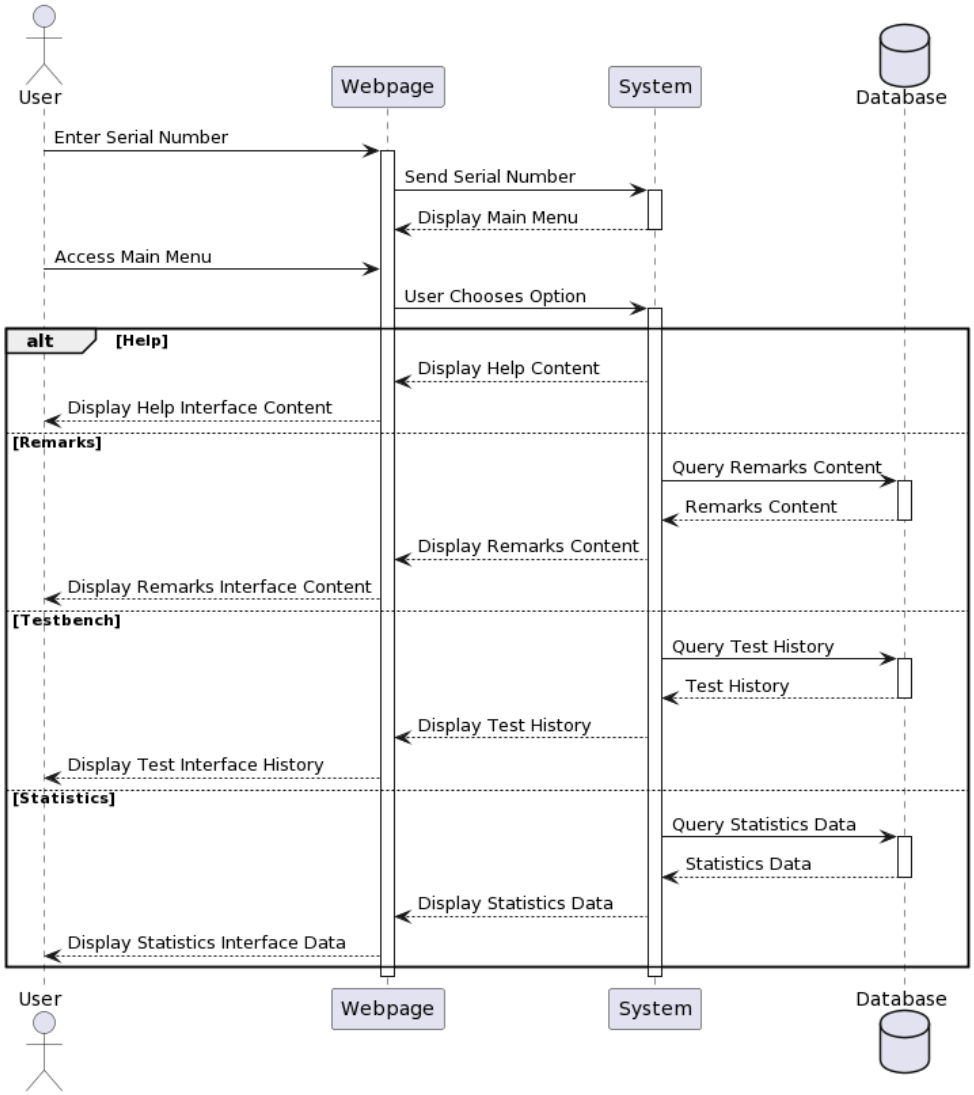


Figure 9: Sequence Diagram for Use Case 6.

### 4.2.9 USE CASE 7: TESTS RESULTS

**Actor:** Authenticated user
**Scenario:** User has selected option "Test Bench" in the main menu and now selects one test.
**Outcome:** The user will be redirected to that test result and will be able to interact with them.
See figure 10.

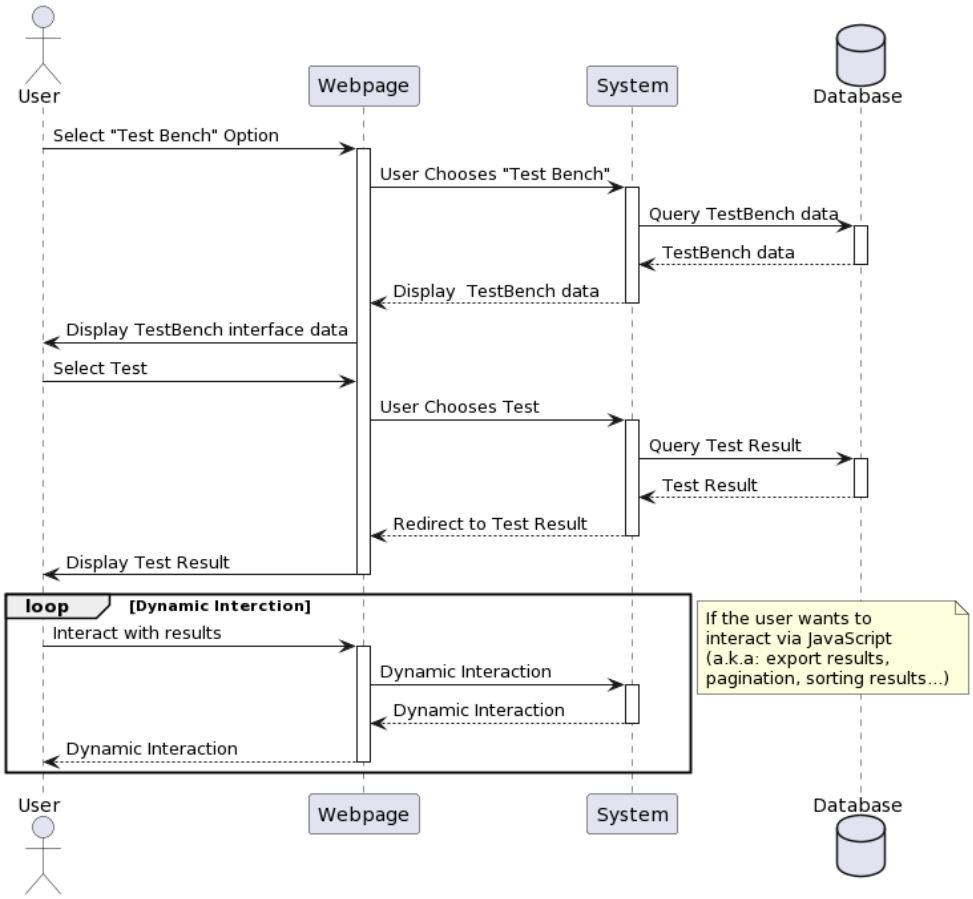


Figure 10: Sequence Diagram for Use Case 7.

## 4.3 Webpage Design

The design of the webpages is a critical aspect of the user experience. The designs are created to ensure ease of use, clear navigation, and effective presentation of information to the users. In this section, we present the interfaces that have been developed.

### 4.3.1 Interfaces

The user interfaces have been developed to match the mock-ups and deliver the intended user experience. The interfaces provide a user-friendly environment for users to interact with the system and access various features.

Due to space limitations within this section, the detailed images of the interfaces are not directly included here. Instead, for a comprehensive view of the interface designs, some of them (not all of them due to space limitations) they have been appended as pictures to the end of this document for easy reference (see Appendix B).

### 4.3.2 Appearance

To achieve a unique visual style and maintain consistency across the webpages, a custom CSS has been developed. This file ensures that the webpages adhere to the design guidelines and provide a cohesive user experience.

The webpage structure has been rigorously validated using the W3C (World Wide Web Consortium) [4] web validator to ensure compliance with web standards and best practices. This validation process guarantees that the webpages are accessible, well-formed, and compatible across various browsers and devices.

The combination of custom CSS and W3C validation ensures that the user interfaces are not only aesthetically appealing but also functional and accessible to a wide range of users.

On the other hand, all the icons utilized in the website's design have been sourced from Flaticon [3]. It is a comprehensive platform housing a vast collection of professionally crafted icons that enhance the visual appeal and functionality of the site. This resourceful repository offers a wide array of icon choices, enabling us to seamlessly integrate visually pleasing elements that resonate with the overall design aesthetic.

**Note:** For a more comprehensive understanding of the icons utilized and the intricacies of the website's design, additional information can be found within the project's documentation.

### 4.4 Webpage Architecture

In the context of modern web development, webpages have evolved to become dynamic entities that provide enhanced user experiences. This dynamism is often facilitated by a two-tier client-server architecture [2], where the client (web browser) interacts with a remote server to retrieve and display content. This architecture enables the separation of concerns and allows for more efficient resource utilization.

The interaction between the browser and the server involves a series of steps that ensure the seamless delivery of content. When a user enters a web address or clicks on a link, the browser initiates an HTTP request to the server [5]. This request contains various parameters, including the URL and the HTTP method.

Upon receiving the request, the server processes it and determines the appropriate action to take. Depending on the nature of the request, the server may respond with static resources (such as HTML, CSS, and JavaScript files) or dynamically generated content. For dynamic content, the server may employ technologies like PHP to generate HTML templates that are tailored to the user's request.

If the webpage requires data from a database, the server formulates a database query based on the user's request and retrieves the relevant information. The retrieved data is then integrated into the HTML template to create a fully customized response. Once the server completes these steps, it sends back an HTTP response to the browser.

The browser, upon receiving the response, processes the HTML, CSS, and JavaScript resources. It renders the content and presents it to the user. The user can then interact with the displayed content, initiating further HTTP requests as they navigate through the webpage.

This dynamic interaction between the browser and the server is a fundamental aspect of modern webpage architecture. It allows for real-time updates, personalized experiences, and efficient data retrieval. The client-server model plays a crucial role in delivering a seamless user experience while also enabling the development of sophisticated web applications.

In summary, the architecture of modern webpages follows a two-tier client-server model, as discussed in the article "A Review on Client-Server Based Applications and Research Opportunitie" [2]. This model ensures efficient communication between web browsers and remote servers, facilitating dynamic content delivery and user interaction.

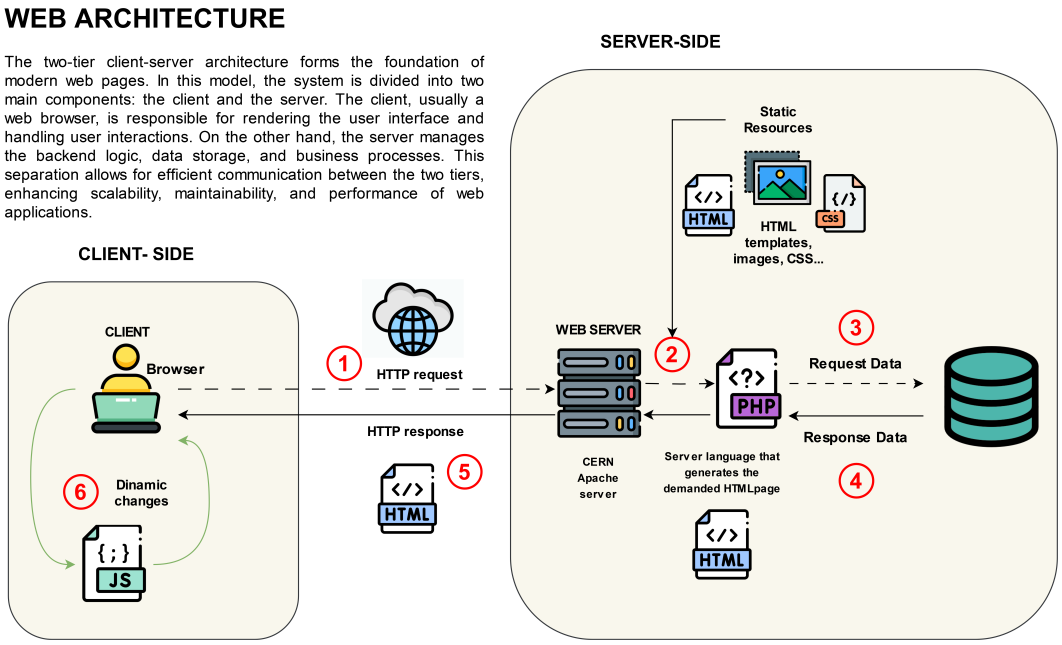


Figure 11: Web architecture: client-server (using icons from [3]).

## 4.5 Features

This section delves into some other features that contribute to the robustness and usability of the webpage.

### 4.5.1 Security:

The CERN environment adds an extra layer of security to the webpage. Access to the website requires CERN login credentials, and specific permissions for reading or writing are granted by my supervisor to specific CERN accounts. This ensures controlled access and data protection.

Moreover, building upon the client-server architecture discussed earlier, the separation of JavaScript and PHP functionalities contributes to enhanced security. JavaScript is utilized solely for user interactions, while PHP handles interactions with the database. This approach prevents direct client access to database-related functions, minimizing the risk of vulnerabilities such as SQL injections.

The combination of the CERN security framework and the well-structured client-server architecture enhances both the security and functionality of the webpage, creating a user-friendly and protected environment.

#### 4.5.2 Responsive Design:

The website has been designed to be highly responsive and cross-platform compatible, ensuring a consistent user experience across various devices and web browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Opera.

Additionally, to showcase the responsiveness of the website, the Appendix B includes images of the interfaces, including the version optimized for smaller screens. This provides a visual demonstration of how the webpage adapts to different screen sizes, further highlighting its user-centric design.

### 4.6 Documentation

Comprehensive documentation has been developed to ensure the clarity and ease of understanding of the webpage's functionality and structure. The documentation includes:

#### 4.6.1 Diagrams:

Detailed diagrams have been provided to illustrate the architectural layout and flow of the webpage. These diagrams aid in understanding how different components interact and contribute to the overall user experience.

#### 4.6.2 User Manual:

A user manual has been prepared to guide users through the various features and functionalities of the webpage. This manual provides step-by-step instructions on navigating the webpage, interpreting results, and utilizing its interactive features effectively.

#### 4.6.3 Programmer Manual:

For those involved in the technical aspects of the webpage's development and maintenance, a programmer manual has been created. This manual delves into the intricacies of the codebase, providing insights into the technologies used, coding conventions, and guidelines for future development and modifications.

# 5 Test Project

The significance of conducting thorough testing on any software product cannot be overstated. Testing ensures that the software functions as intended, meets the required specifications, and delivers a reliable user experience. Therefore, as emphasized throughout this report, the establishment of a robust test project has been a pivotal aspect of this endeavor. In parallel with the main development efforts, a comprehensive test project has been initiated to validate the functionality, performance, and reliability of the webpage.

## 5.1 Test Plan

To this end, a well-structured test plan has been devised, encompassing unit tests, integration tests, and system tests. The test plan's layout and coverage can be observed in Figures 12 and 13. Each type of test serves a specific purpose in assessing different aspects of the webpage's capabilities.

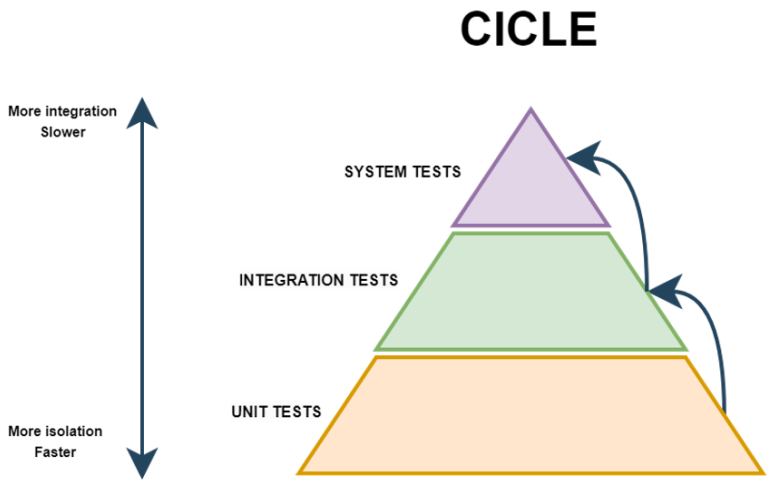


Figure 12: Test plan overview.

Notably, the meticulous documentation of every test is a priority. Each test, its procedure, methodology, and techniques employed have been carefully documented. This documentation is readily accessible in the project repository, providing transparency and insight into the testing process for future reference.

The tests have been meticulously automated, allowing for reproducibility at any given point. The automated nature of the tests ensures consistent and accurate execution, minimizing the potential for human error.

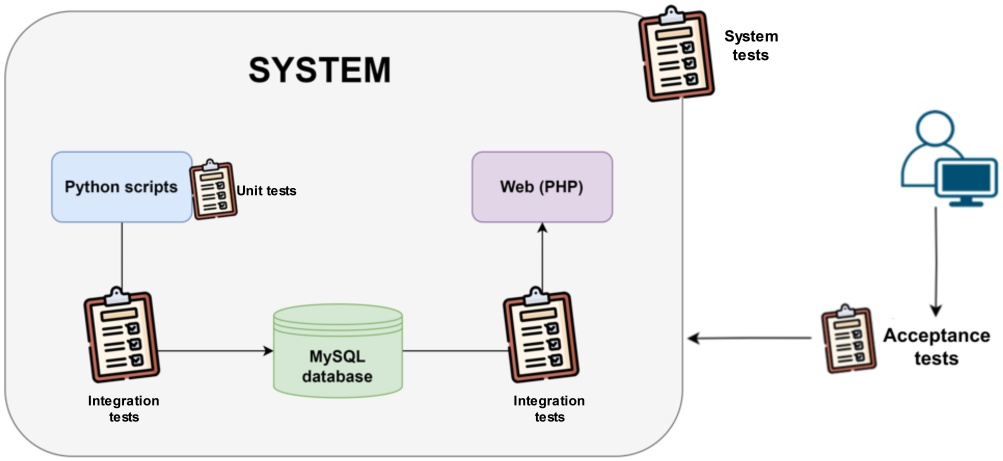


Figure 13: Tests on our project (using icons from [3]).

In summary, the establishment of a parallel test project alongside the main development process attests to the commitment to delivering a dependable and high-quality product. The integration of unit tests, integration tests, and system tests, along with their careful documentation, forms a robust foundation for the validation of the webpage's functionality and performance. The automated nature of the tests further guarantees their reliability and repeatability, fostering confidence in the webpage's capabilities.

## 5.2 Methodology

The development of the webpage and its associated scripts was guided by well-defined methodologies to ensure efficiency, reliability, and adherence to quality standards.

### 5.2.1 Test-Driven Development for Python Scripts:

The development of the Python scripts, which manage the database and handle backend functionalities, adhered to the Test-Driven Development (TDD) methodology. TDD is an iterative development process that emphasizes writing tests before writing the actual code [1]. This approach has several advantages:

- **Clear Requirements:** TDD begins with writing tests based on the requirements. This ensures that developers have a clear understanding of the desired functionality before starting the implementation.

- **Robust Code:** Tests written in TDD serve as a safety net, allowing developers to make changes to the codebase with confidence. If a change causes unintended issues, the tests will catch them.
- **Focused Development:** TDD encourages developers to focus on solving one problem at a time. This results in smaller, manageable portions of code that are easier to debug and maintain.
- **Documentation:** Tests act as living documentation, providing insights into the expected behavior of the code. This is particularly helpful when multiple developers are working on a project.

TDD follows a structured approach with three distinct stages: the Red stage, the Green stage, and the Refactoring stage.

1. **Red Stage:** In this initial phase, a failing test is written to represent the desired functionality. This test, known as the "red" test, serves as a clear indicator that the intended functionality is not yet implemented.
2. **Green Stage:** Following the Red stage, the developer proceeds to write the minimal code necessary to make the failing test pass. This phase is called the "green" stage, as the test transitions from failing to passing, confirming that the desired functionality is now implemented.
3. **Refactoring Stage:** After successfully passing the test, the code is refactored to improve its structure, maintainability, and efficiency. This phase, the "refactoring" stage, ensures that the code remains clean and readable without altering its behavior.

### 5.2.2 Automated Web Testing with Selenium:

For the frontend of the webpage, an automated testing approach was employed using the Selenium framework. Selenium is a widely used tool for automating web browsers, allowing developers to simulate user interactions and validate that the webpage functions correctly.

Automated scripts interact with the webpage's user interfaces, ensuring that all functionalities are functioning as expected. By simulating user interactions, such as clicking buttons and filling out forms, the scripts validate the accuracy and responsiveness of the webpage.

Additionally, the W3C validator has been employed to ensure the webpage's HTML and CSS adhere to web standards. Although a detailed explanation has been provided earlier in this report, it's worth reiterating that the validator assesses the structural integrity of the webpage, contributing to its overall quality.

# 6 Conclusion and Future Lines

In conclusion, this project has successfully addressed the challenges associated with testing and managing newly designed DAQ boards for the CMS experiment at CERN. By streamlining the testing process, enhancing result storage, and creating a user-friendly web interface, the project contributes to more efficient data collection, storage, and analysis within the CMS experiment. The implemented database system, automated scripts, and responsive web interface have collectively improved accessibility, usability, and security.

Looking ahead, several exciting avenues for future development are anticipated:

- **Internationalization:** Expanding the reach of the application to a wider audience is a primary future enhancement. The addition of language support beyond English will enhance accessibility and usability, making the application more inclusive and user-friendly.
- **CSV Import:** Continuing to enhance the application's capabilities, a planned feature involves introducing the ability to import test results through CSV (Comma-Separated Values) files. This feature streamlines the process of inputting test data, reducing manual effort and facilitating data entry.
- **Insertions from Webpage:** An interesting feature that was explored involved the direct insertion of new data through the webpage. Icons and forms were integrated to allow users to input new data entries conveniently. However, due to security considerations, modifications to the webpage were restricted to Python files. Users are now able to view results through the webpage, while a separate version with modification capabilities remains accessible for potential future integration.

The development process undertaken in this project adhered to rigorous methodologies, ensuring the quality, reliability, and security of the resulting application. The successful implementation of a test project with automated testing, adherence to web standards, and the utilization of a client-server architecture further contribute to the robustness of the project's outcomes. As the field of particle physics continues to evolve, the methodologies and enhancements developed in this project will undoubtedly play a crucial role in expediting experimentation, analysis, and collaboration within the scientific community.

## Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dominique, for his invaluable support throughout the course of this project. Furthermore, I extend my appreciation to the CERN and the CMS project for their generous support, making this project possible. I am truly fortunate to have had the opportunity to work during this 8 weeks under Dominique's mentorship and alongside the CMS project, and I am deeply appreciative of their contributions to this endeavor.

# A Appendix I: Glossary

- **DAQ:** Data Acquisition. It refers to the process of collecting, measuring, and storing data from various sources, often used in scientific and engineering applications.
- **CMS:** Compact Muon Solenoid. It is one of the major particle detectors at CERN's Large Hadron Collider (LHC), designed to study a wide range of physics phenomena.
- **ERD:** Entity-Relationship Diagram. It is a visual representation used in database design to illustrate the relationships between entities (objects or concepts) in a database system. It shows how data entities are connected to each other through relationships.
- **Refactoring:** Refactoring is the process of improving the internal structure of code without changing its external behavior. It helps enhance code quality, readability, and maintainability while reducing technical debt.
- **Test Phase - Red, Green, and Refactor:** This term refers to the three steps in Test-Driven Development (TDD). The "Red" phase involves writing a failing test, the "Green" phase involves writing code to pass the test, and the "Refactor" phase involves improving the code's design without changing its behavior.
- **Package Diagram:** A package diagram is a UML (Unified Modeling Language) diagram that depicts the organization of the system's components or packages and their dependencies. It helps visualize the high-level structure of the software system.
- **Use Case Diagram:** A use case diagram is a UML diagram that represents the interactions between users (actors) and the system, showing how users interact with the system's functionalities.
- **Sequence Diagram:** A sequence diagram is a UML diagram that illustrates the interactions between various components or objects in a system over time. It visualizes the order of messages exchanged between components to accomplish a specific task.
- **Secure Programming:** It refers to the practice of writing code with a focus on eliminating vulnerabilities and potential security risks. It involves implementing coding techniques and strategies that prevent common security threats, such as SQL injection, cross-site scripting (XSS), and unauthorized access. Secure programming helps safeguard software applications and their underlying systems from exploitation and unauthorized manipulation.

# B Appendix II: Interface Illustrations

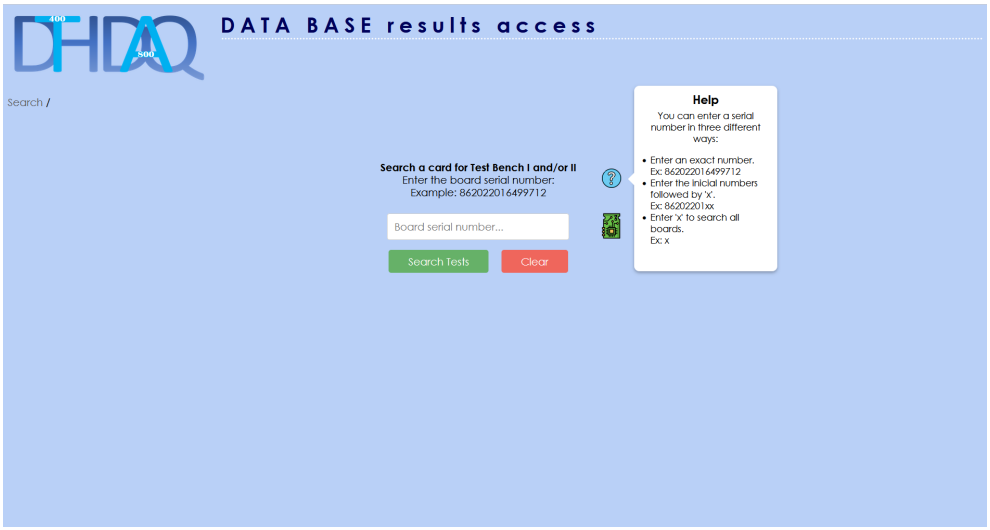


Figure 14: Main Interface.

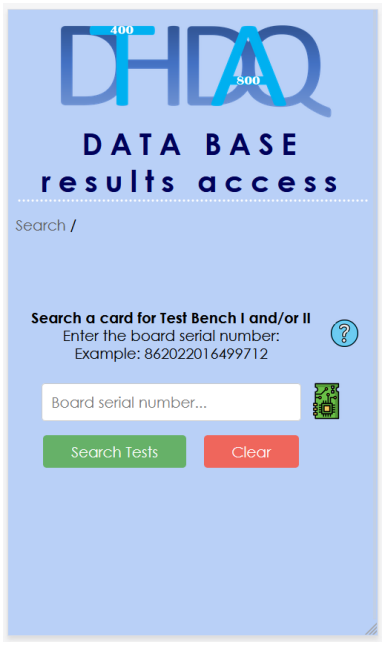


Figure 15: Main Interface small screen.

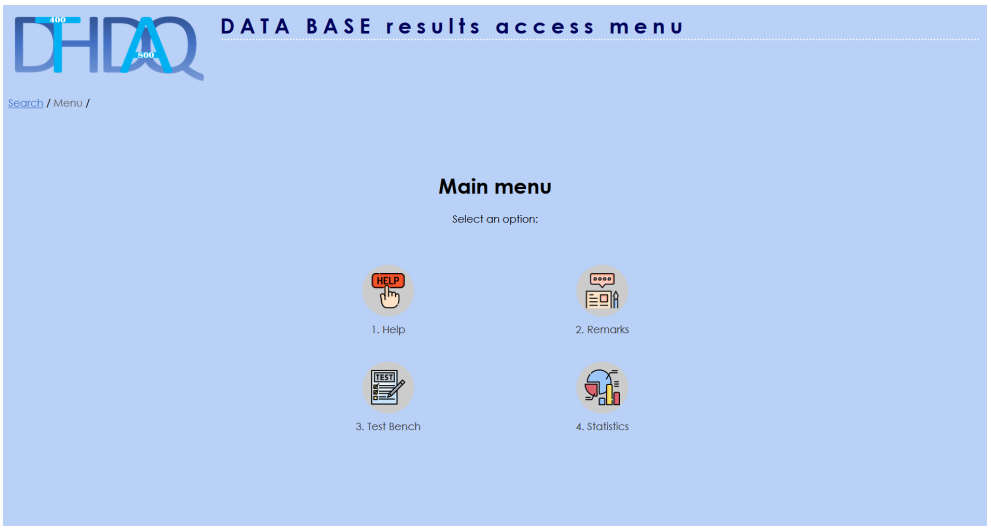


Figure 16: Menu Interface.

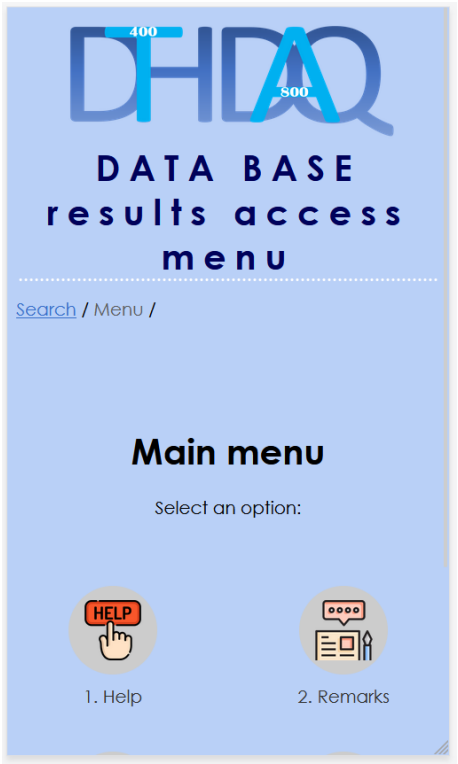


Figure 17: Menu Interface small screen.

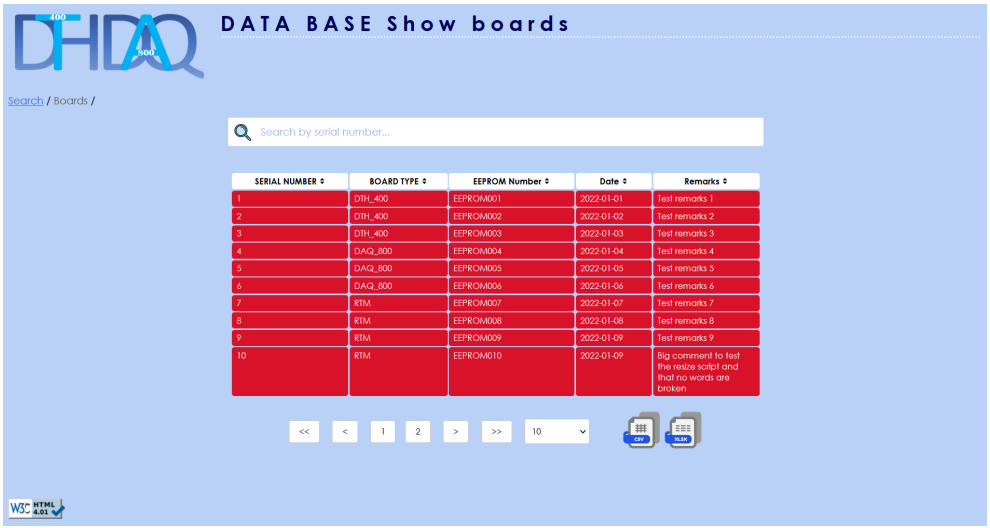


Figure 18: Boards list Interface.

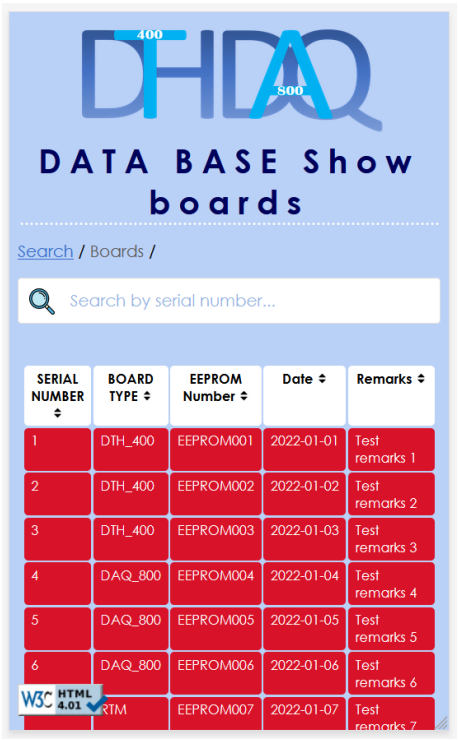


Figure 19: Boards list Interface small screen.

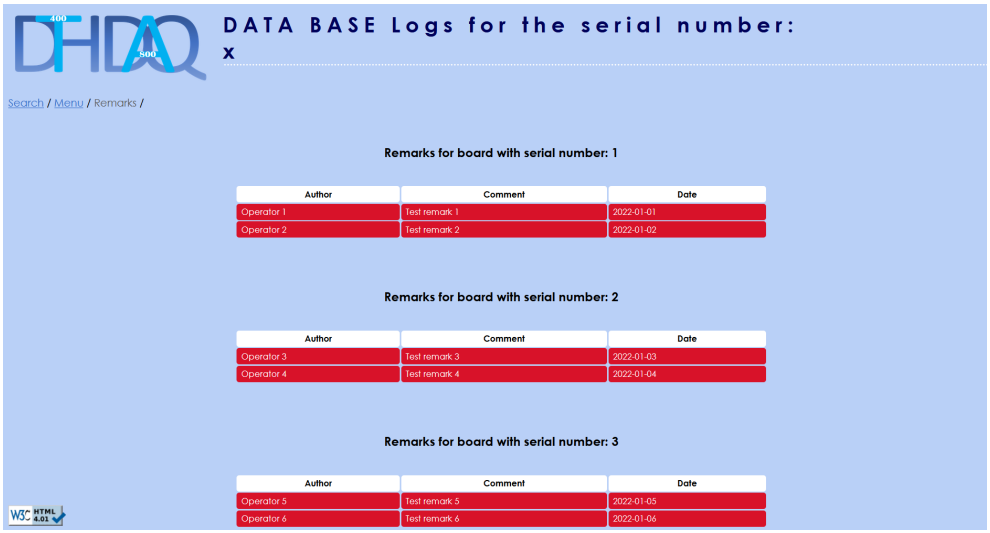


Figure 20: Remarks Interface.

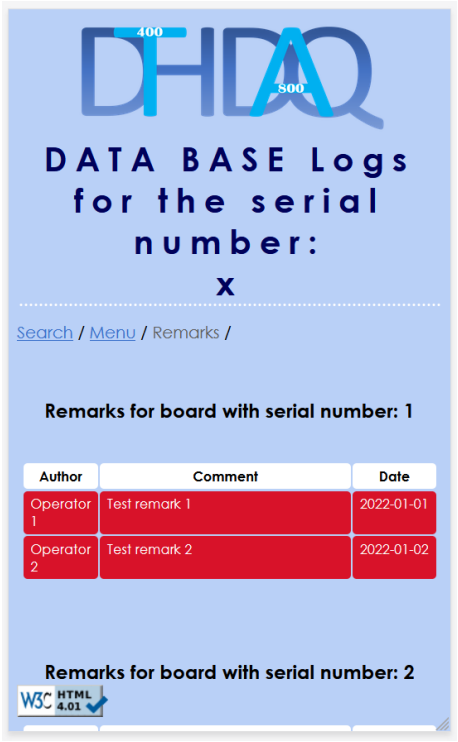


Figure 21: Remarks Interface small screen.

## References


[1] Beck, K., *Test-driven development by example*, (Addison-Wesley, Boston, 2015).

[2] Santosh Kumar, et al., *A Review on Client-Server Based Applications and Research Opportunities*, (2019) Vol. 10, Issue, 07(H), pp. 33857-33862, ISSN 0976-3031.

[3] Flaticon, *Free icons and stickers - millions of images to download*, https://www.flaticon.com/ (accessed Aug. 21, 2023).

[4] The W3C Markup Validation Service, *Markup validation service*, https://validator.w3.org/ (accessed Aug. 21, 2023).

[5] MozDevNet, *Client-server overview - learn web development: MDN, Learn web development | MDN*, https://developer.mozilla.org/en-US/docs/Learn/Server-side/First$_s$*teps*/*Client* − *Server*$_o$*verview* (accessed: 23 August 2023).