

New ValDB System Development

Chanchana Wicha

Supervisors: Justinas Rumsevicius, Chayanit

Asawatangtrakuldee.

CERN Summer Student Report.

July - August, 2021.

Contributing authors: wic.chanchana@gmail.com;

Abstract

ValDB is a validation database website. The previous version has a problem with the overall user experience. By developing a new version of ValDB, it can help improve user experience and introduce essential features to the system. Together with modern web development technologies such as Python, Flask, TypeScript and React, they ease the development process and make the codebase easily maintainable. A custom ORM takes advantage of native type hinting introduced in Python with simple syntax. CI/CD helps automate processes, ensure system quality, and quickly deliver new software changes to users.

1 Introduction

I got an opportunity to be selected as a CERN Summer Student Programme 2021 for 8 weeks. During the program, I was assigned as a part of the PmdV team with my supervisors: Justinas Rumsevicius and Chayanit Asawatangtrakuldee. My assignment was to work on developing a new ValDB system.

ValDB is a validation database website for tracking software packages' quality that are deployed on the CMS system. Currently, this system has some problems that affect the user experience, including a complicated user interface, missing information in report content, and a complicated user management system. The objectives of developing a new version of ValDB are to improve the overall user experience and implement essential features.

2 Acknowledgements

I would like to express my special thanks to my supervisors: Justinas Rumsevicius and Chayanit Asawatangtrakuldee, and the PdmV team members who gave me a golden opportunity to do this project as well as gave me valuable comments and suggestions which led to the completion of this project.

Furthermore, I would like to thank everyone who organized and coordinated CERN Summer Student Programme 2021. Although it was an online summer student program, it was an amazing and memorable experience for me, and everything was perfectly organized.

3 Technologies

Unlike the previous version of ValDB that used conventional technologies such as plain HTML, which made it hard to develop advanced features and maintain, the new version of ValDB uses modern technologies which are popular, powerful and maintainable. Technologies applied in this system can be divided into three main components: backend, frontend and database.

The backend is implemented with Python together with Flask to serve the requests from clients. The frontend is implemented with TypeScript and React. TypeScript is an enhanced version of JavaScript which supports static typing, so the frontend codebase is easy to maintain. For the database, MongoDB is used here. MongoDB is a document-based NoSQL database that supports dynamic schemas. It is a powerful database that can handle large amounts of various types of data.

4 Methodology

4.1 System Requirements

- Administrators can manage campaigns in the system.
- Administrators can manage users' permission and role.
- Administrators and validators can view campaigns and reports.
- Validators can only write reports based on their permission group.
- Validators can see their assigned reports based on their permission group.
- Anyone can comment on reports.
- A report editor needs to support various types of information, including tables, hyperlinks, and attachments.
- Emails need to be sent on certain events in the system, such as campaign or report modification, users' permission change, and new comments on reports.
- Data from the previous version needs to be migrated to the new version.

4.2 Use-case Diagram

Figure 1 shows the overall use-case of the new system. Users in the system can be divided into two roles: administrators and validators. Administrators

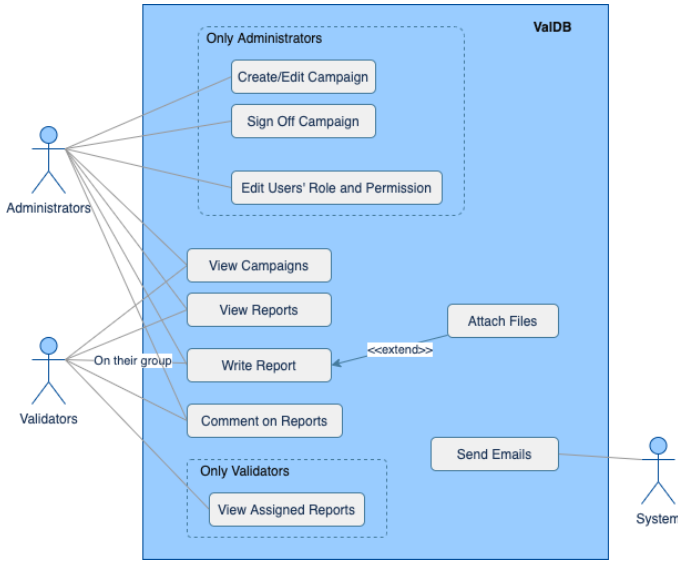


Fig. 1 Use-case Diagram

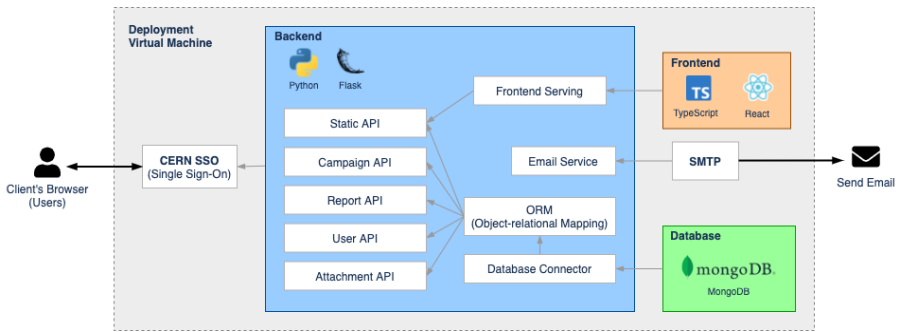
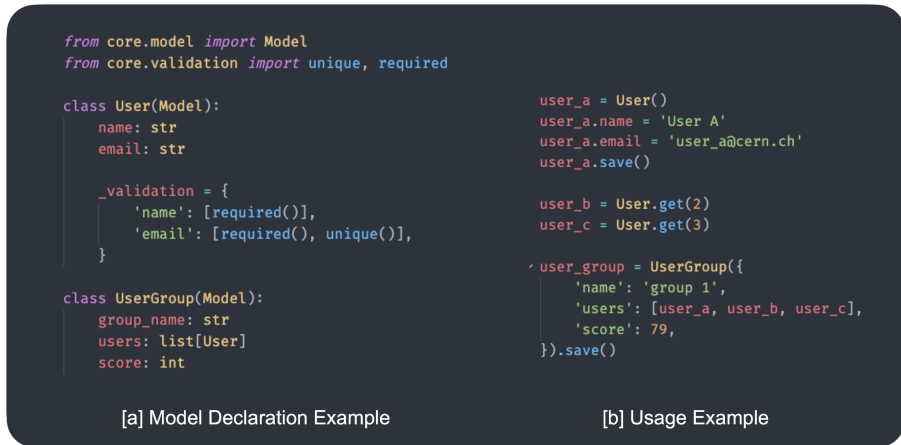


Fig. 2 System Architecture

are responsible for campaign and user management. They can create, edit and sign off campaigns and edit users' role and permission. On the other side, validators are responsible for writing reports based on their permission groups. They can add attachments to the reports. However, they can view campaigns and reports from another group, but they do not have permission to modify them. Any user can comment on any report in the system. Validators can also view their assigned report which they can have quick access to write reports. Lastly, the system can automatically send emails to users on certain events.

4.3 System Architecture

Figure 2 shows the architecture of the system. Every request sent from the client's browsers will be authenticated by the CERN SSO (single sign-on) Service before passing to the backend of the system. The backend is the main

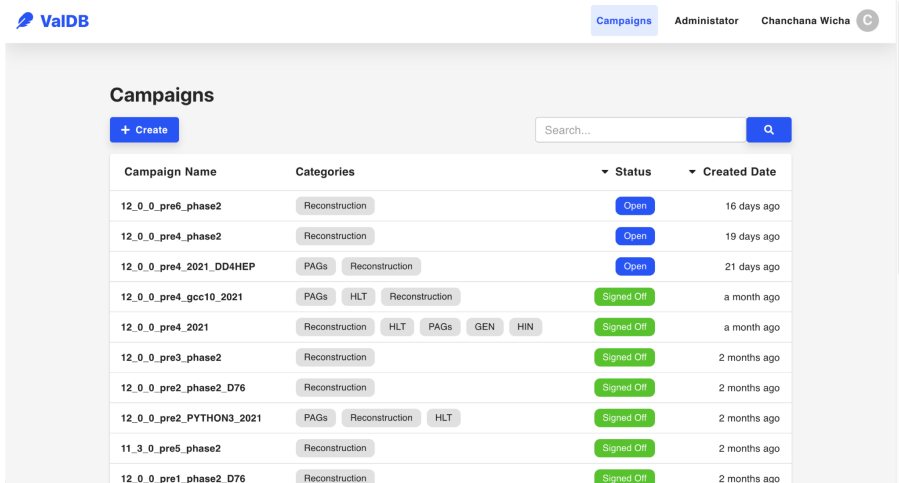
**Fig. 3** ORM Usage Example

component in the system that handles all the logic. Data will be mapped into native Python objects using ORM (Object-relational Mapping) so that the backend can manipulate data in the database using its objects with a simple interface. ORM handles database operations via the database connector which connects to the MongoDB database. For the frontend component, it is built into a static site that can serve clients via a backend static API. For email service, it uses pre-configured SMTP on their deployment virtual machine to send email to users.

4.4 Custom ORM

Because Python is a dynamic type programming language, dealing with data that does not declare specific types or schemas can be frustrating and hard to maintain. However, Python 3.9 introduces native generic type hinting. A custom ORM implemented in this system aims to take advantage of native type hinting to make the codebase more maintainable but still keep the simple syntax style of Python.

Figure 3 shows the example usage of custom ORM in the system. For data model declaration, it supports native type annotation such as string, integer, floating point, boolean and enumeration. It supports data relationships such as one-to-many and many-to-one relationships. It provides a simple interface which resembles the native Python coding style without strict schema or complex declarations. For example, on Figure 3.a, the user group contains many users which can be declared by a list of user objects using native type hinting, and the ORM will automatically create a data relationship. Moreover, it supports field validation that validates each field by specified rules before committing a transaction in the database. The ORM provides all essential database operation interfaces such as create, update, query, and delete with simple usage.



Campaign Name	Categories	Status	Created Date
12_0_0_pre6_phase2	Reconstruction	Open	16 days ago
12_0_0_pre4_phase2	Reconstruction	Open	19 days ago
12_0_0_pre4_2021_DD4HEP	PAGs Reconstruction	Open	21 days ago
12_0_0_pre4_gcc10_2021	PAGs HLT Reconstruction	Signed Off	a month ago
12_0_0_pre4_2021	Reconstruction HLT PAGs GEN HIN	Signed Off	a month ago
12_0_0_pre3_phase2	Reconstruction	Signed Off	2 months ago
12_0_0_pre2_phase2_D76	Reconstruction	Signed Off	2 months ago
12_0_0_pre2_PYTHON3_2021	PAGs Reconstruction HLT	Signed Off	2 months ago
11_3_0_pre5_phase2	Reconstruction	Signed Off	2 months ago
12_0_0_pre1_phase2_D76	Reconstruction	Signed Off	2 months ago

Fig. 4 All Campaigns Page User Interface

4.5 CI/CD

CI/CD (continuous integration and continuous delivery) helps developers integrate their features and deliver them easily. Since Github is used to store the source code of this project, Github Actions takes care of CI/CD operations in this project. The operations are separated into two parts: pull requests and deployment pipelines.

For the pull requests pipeline, the pipeline runs on every pull request. The main action for this pipeline is linting, which includes pylint for Python and eslint and tslint for TypeScript. Linting helps keep code clean and conform to coding style standard rules. Unit tests are implemented in the project to check the correct functionality and behavior of the system. However, at the current state, unit tests need to be run manually because the pipeline is not configured to run them yet.

For the deployment pipeline, this pipeline runs every day at midnight. It keeps track of the main branch of the project's codebase for any new modifications. If there are changes, it runs these actions: build and release. For build action, it builds the frontend source into a static site and removes the frontend source and unit test folders since they are not needed for production. For release action, it makes an artifact file that can be executed on a production virtual machine.

5 Result

All of the components are developed and deployed in a virtual machine. The following figures: Figure 4, Figure 5, Figure 6 and Figure 7, show the examples of the new system's user interfaces.

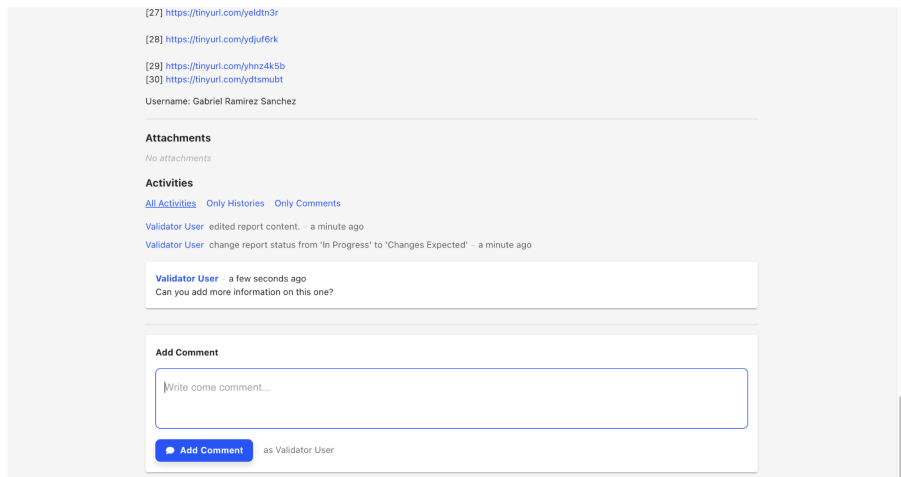
6 *New ValDB System Development***Fig. 5** Report Editor User Interface**Fig. 6** Report Comment Section User Interface

Figure 4 shows the campaign list page. Users can see the list of all campaigns, search by keyword, and sort by columns. Administrators can create new campaigns by clicking on the “Create” button.

Figure 5 shows the new report editor. This editor supports the markdown format, allowing users to add different types of content to the report, such as tables, hyperlinks, checkboxes, and item lists. Attachments can be added by dropping the file into an editing area or by clicking on the “Add Attachment” button.

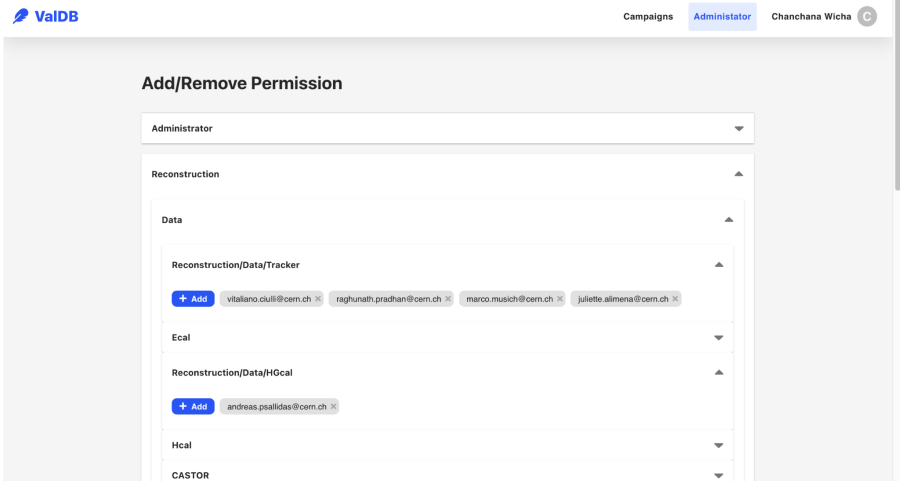


Fig. 7 User Permission Management Page User Interface

Figure 6 shows the comment section in the report page. When a new comment is added to a report, an email will be automatically sent to authors and users commented before. This feature solves the problem that users need to contact authors by emails frequently when some information is missing, and they need to find authors' email by themselves.

Figure 7 shows the user permission management page. This page is used by administrators to manage validators' access rights. Administrators can add or remove each person from certain groups in the system. Unlike the previous version of ValDB that requires administrators to input both username and email when adding a new user, the new version requires only email and the system will automatically get users' full name from their account.

6 Conclusion

The new version of ValDB improves the overall user experience from the previous version. It also solves the problems of the previous version by introducing new essential features. The overall user interface looks nicer and easier to understand. Modern technologies applied to the system make it easy to develop and maintain. The custom ORM supports all use-cases in the system. However, the ORM has a performance issue when querying a very large amount of data compared to directly querying with a database. CI/CD helps speed up deployment time, ensure the correctness of the system and reduce manual human work. Lastly, the data from the previous version is migrated to the new one, so that users can continue using the new version and see historical data of the system.