



8 October 2021
rayane.belaid@cern.ch

Development of a Monitoring Tool for Software Licenses

CERN Summer Student Programme 2021

Author: Rayane Belaid ¹

Supervisor: Mario Rey Regulez ²

CERN ³ - European Organization for Nuclear Research, CH-1211, Geneva, Switzerland

Contents

1	Introduction	3
2	Background	3
2.1	About CERN and AD-Section	3
2.2	Client-Server Model	4
3	Conception	5
3.1	Software Architecture	5
3.1.1	Unified Modeling Language (UML)	5
3.1.2	Use Case Diagram	5
3.1.3	Sequence Diagrams	7
4	Implementation	9
4.1	Tools	9
4.2	Explanation	9
4.3	Results	10
4.4	Deployment	12
5	Conclusion	13

¹Student at the Faculty of Science, Department of Mathematics and Computer Science, (University of Algiers, Algeria)

²Information and Technology Department, Collaboration, Devices and Application (CDA) group, Application and Devices (AD) Section

³Conseil Européen pour la Recherche Nucléaire

1 Introduction

The purpose of this report is to present the content of my project as a summer student at CERN during the summer of 2021. I participated as a Non-Member State and was assigned to the IT ¹ department for this program.

Software license management gives the transparency into the enterprise's software assets, usage, licenses, and contracts so that administrators and users can understand what software is being used, how much, where and by whom. However, without managing licenses for big sized organizations can cost time and money. It can also hinder productivity and efficiency.

IT administrators or software asset managers may find it challenging to track products installed on numerous devices across a network. They run the risk of purchasing too many licenses then needed. Also, The Application and Devices (AD) section provides over 100 engineering applications [1] used by the technicians, engineers, and physicists at CERN to design and implement the different components that are used to carry out the accelerators and experiments work. Such a large portfolio of applications also entails a large set of licenses that need to be maintained and monitored to give information to users about their current status. In that stage our solution attempt to answer this following problem, How can we manage a complex system efficiently of a large number of engineering tools, users holding licenses and users request licenses? In order to answer to this problem will need a tool that monitors and report on engineering tool licenses and usage across the organization. Therefore, my project consists of developing a monitoring tool for software licenses, It is a web application called Licmon built for tracking and facilitate the resolution of license shortage incidents. The main objectives of this application are. First, to maintain an organized software license with detailed information on all engineering tools, including, contracts, and license types. This helps users stay informed of all available software. Second, to Stay updated on license usage and expiration to prevent unnecessary spending on unused licenses and maintain compliance. Next, to provide admins with statistics about licenses and their status. Also, to allow users to request licenses by sending an automatic e-mail that describes their requests.

2 Background

This section covers a brief introduction to CERN company, in particular the service where I carried my internship. As well as an overview of the client-server model.

2.1 About CERN and AD-Section

CERN (derived from the name Conseil Européen pour la Recherche Nucléaire), is a European research organization that operates the largest particle physics laboratory in the world. Established in 1954, the organization is based in Geneva on the Franco-Swiss border. Its main research area is particle physics, providing particle accelerators and other infrastructures needed for high-energy-physics research. CERN gather a large number of scientists, physics,

¹Information and Technology

engineers that come from different countries to consolidate their knowledge and contribute to different experiments. Currently it has 23 Member States, 3 Associate Member States in the pre-stage to Membership and 7 Associate Member States [3]. CERN is the site of the Large Hadron Collider (LHC), the world’s largest and highest-energy particle collider. It consists of a 27-kilometre ring of superconducting magnets with a number of accelerating structures that boost the energy of the particles along the way [5]. In Figure 1, is illustrated the fourth particle detectors where collision of accelerator particles happen.

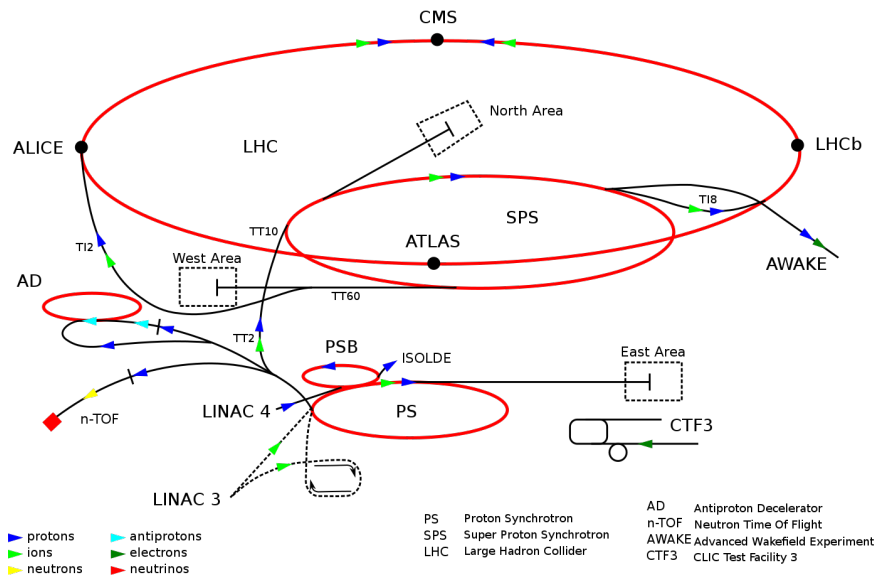


Figure 1: CERN's Accelerator Complex

The IT department at CERN composed of multiple groups. The group Collaboration, Devices and Applications (CDA) provides information services such as video conferencing, webcast recording, indico, the CERN Document Server (CDS), software development tools and more [1]. My project takes place within the software development tools specifically in the Applications and Devices (AD) section. The AD section of the IT department currently supports Windows and Mac computers, public screens and printing and provides over 100 applications including mechanical engineering tools, mathematics tools, electronic design automation tools, general productivity applications and antivirus [1]. Such a large number of applications also require a large number of licenses as well as users that need to be managed, monitored and tracked.

2.2 Client-Server Model

The communication between the application and the server is based on the client-server model. The backend takes the role of an intermediate between the frontend and the server. The client-server model, or client-server architecture, is a distributed application framework dividing tasks between servers and clients, which either reside in the same system or communicate through a computer network or the Internet. The client relies on sending a request to another program in order to access a service made available by a server. The server runs one or more programs that share resources with and distribute work among clients. Client

and server exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All protocols operate in the application layer. The application layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an application programming interface (API). The API is an abstraction layer for accessing a service. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange [4].

3 Conception

3.1 Software Architecture

Design is one of the important and creative phases in engineering projects. it consist of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. An IT project indeed requires a design phase, during which we model our solution in order to clarify it and provide more details.

The aim is to define the system sufficiently so that satisfies the user, be manufactured, built, installed, and be functional. In order to achieve this design the use of a modeling language is required. It is an artificial language that expresses information in a defined structure by a coherent set of rules. These rules are used to interpret the meaning of components in the structure. In this section, we will use the UML modeling language to represent the different aspects of our system.

3.1.1 Unified Modeling Language (UML)

The UML language also known as (Unified modeling language) provides diagrams to graphically represent the software to be developed. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. It is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The UML diagrams we used to model our system are the following, Use Case Diagrams and Sequence Diagrams. They were modeled using the Star Uml tool.

3.1.2 Use Case Diagram

The use case diagram [9] is used to give an overview of the functional behavior of a software system. See Figure 2. It is a list of actions or event steps typically defining the interactions

between a role (known in UML as an actor) and a system to achieve a goal. The actor can be a human or other external system.

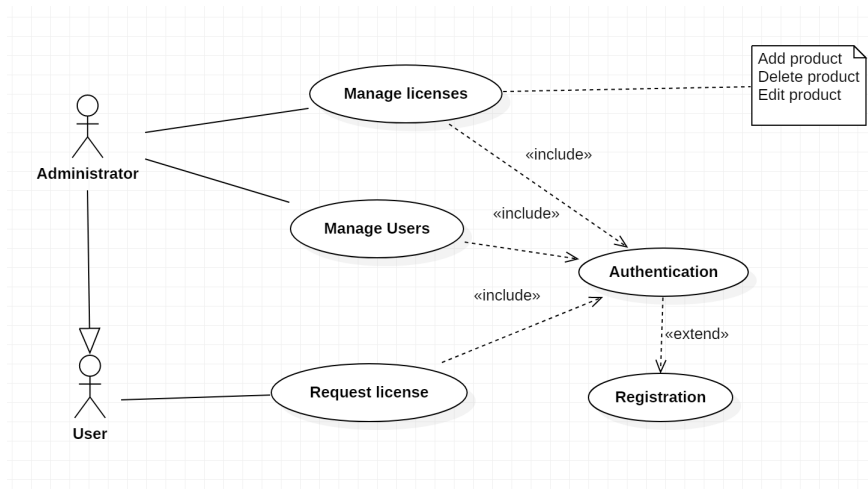


Figure 2: Use case Diagram

Use case description: Case: "Manage Licenses"

1. **Objective:** Allows the administrator to process, manage and access all information relating to the products and licenses.
2. **Main actor:** Administrator
3. **Preconditions:** Access and/or modify information relating to the products, licenses in the system.
4. **Nominal scenario:**
 - Inclusion point: Execute the "Authenticate" use case.
 - Once the administrator accesses his personal account, he clicks on the Product page on the left side bar.
 - The administrator can add, edit or delete any information related to products.

Use case description: Case: "Manage Users"

1. **Objective:** Allows the administrator to process, manage and access all information relating to the users.
2. **Main actor:** Administrator
3. **Preconditions:** Access and/or modify information relating to the users.
4. **Nominal scenario:**
 - Inclusion point: Execute the "Authenticate" use case.

- Once the administrator accesses his personal account, he clicks on the Users page on the left side bar.
- The administrator can add, edit or delete any information related to users.
- The administrator can deactivate user from the list.

Use case description: Case: "Request license"

1. **Objective:** Allows administrators and users request licenses.
2. **Main actor:** Administrator, User
3. **Preconditions:** Requesting license from the system.
4. **Nominal scenario:**
 - Inclusion point: Execute the "Authenticate" use case.
 - Once the administrator/user accesses his personal account, he clicks on the Product page on the left side bar.
 - Request license from the list by clicking on the button.

3.1.3 Sequence Diagrams

Sequence Diagrams [9] are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when. I have modeled two cases, case Authentication and case Request license to sequence diagrams as illustrated in Figure 3 and 4, respectively.

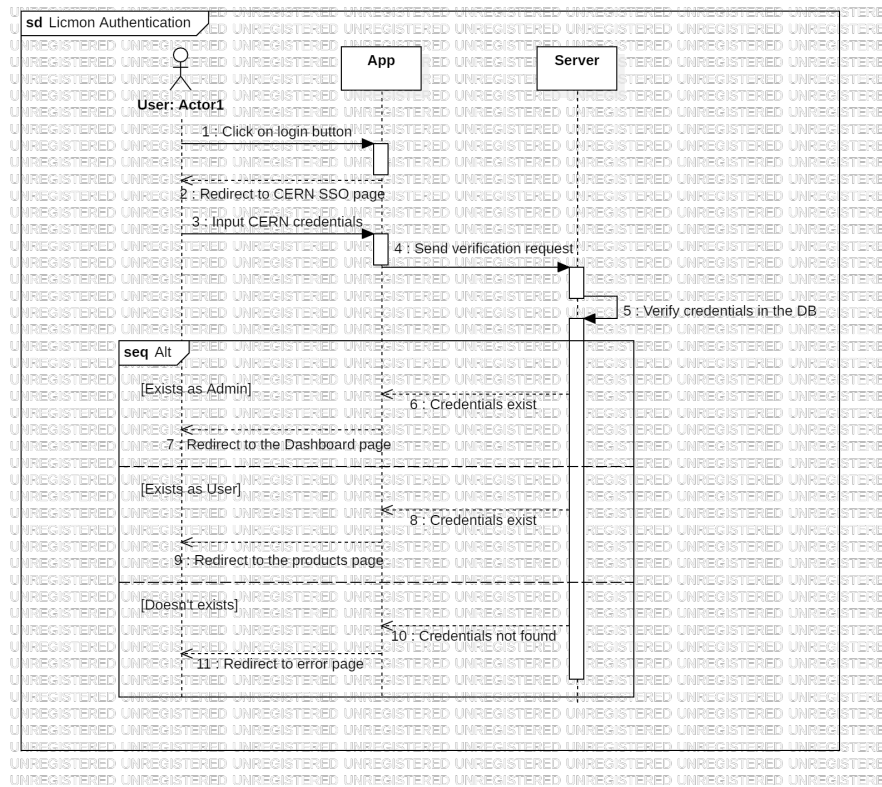


Figure 3: Sequence Diagram: Case "Authentication"

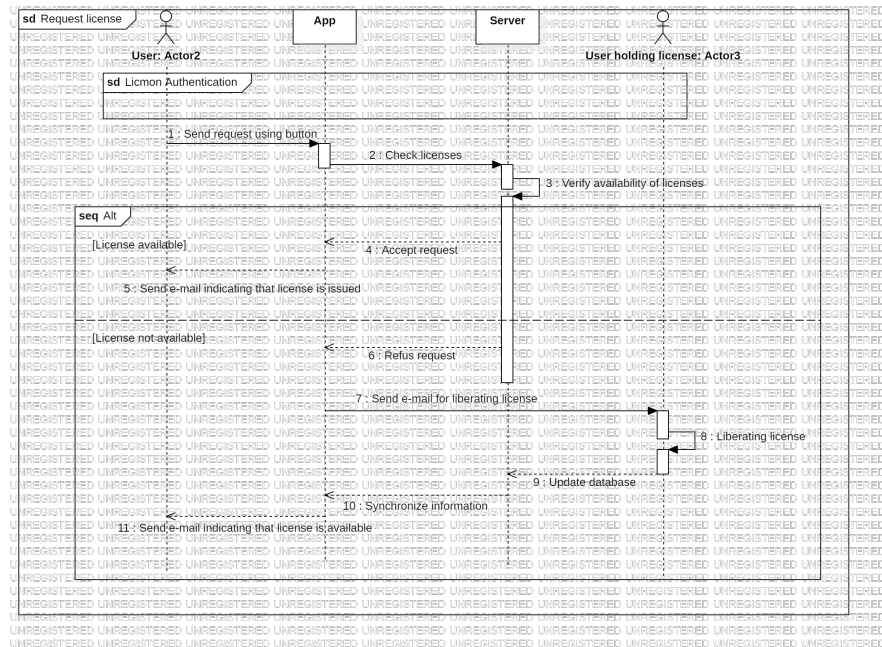


Figure 4: Sequence Diagram: Case "Request license"

4 Implementation

4.1 Tools

During this summer internship I gained insights in valuable tools and sharpen my skills such as working remotely on a CERN OpenShift OKD, using lxplus service and SSH, tunneling for remote access to the computing facility, as well as used the git and Gitlab in order to organize the workflow. But most of all I started learning a new JavaScript framework: React [8]. I followed a tutorial in React in Udemy platform thanks to CERN Learning as well as a tutorial in Docker [10] and Flask [7]. I learned how to use React its components and libraries such as Material UI and Redux. Also how to deploy the applications using docker containers. My current experience is on distributed computing, Algorithms and AI so learning React, Flask and Docker was interesting and gives me a hand to discover the web development technologies.

4.2 Explanation

The project is composed of two applications. First, **licmon-frontend** application built with React js. React is a JavaScript framework. It was originally created by engineers at Facebook to solve the challenges involved when developing complex user interfaces with datasets that change over time [8]. In order to make the interface more resilient an interactive React was the best framework for that. I used Material-UI [6] which allows to import and use different components to create a user interface in my application. Second, **licmon-backend**, is a backend application built with Flask. It is a lightweight web application micro-framework written in Python. It makes use of the flexibility of Python to provide relatively simple template for web application development [7]. Flask has built-in blueprints capabilities which allow us to register modular components for our backend application. A blueprint defines a collection of views, templates and static assets. See Figure 5. This way we can easily reference a view for each component. The main reason why we should use blueprints is to de-couple our application into smaller re-usable components. Thus, the code gets more maintainable and easier to debug.

The frontend and the backend applications interact with each other using two **API**, one for the authentication, and another for server requests. For the authentication, CERN provides services responsible for a centralized user authentication and authorisation infrastructure [2]. So, all CERN's websites are based on the Single Sign On (SSO)[2], which requires the unique CERN credentials. This later providing central, federated and social authentication for applications.

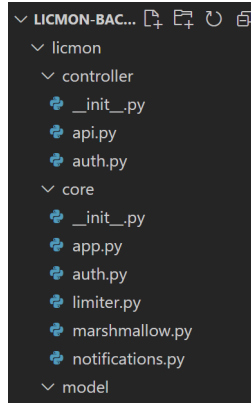


Figure 5: Flask Blueprints Structure

4.3 Results

The Results section corresponds to the last part of our report. It has for objective to represent the implementation phase of the monitoring licence system that we have achieved. The frontend application is represented in different interfaces that will be used by Administrators and users at CERN in order to manage, monitor, track, and request licenses. We basically have three important sections in the frontend application as illustrated in Figure 6, the Dashboard page, Product page, and the Users page.

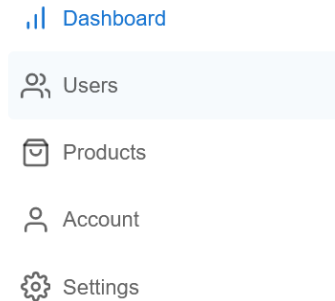


Figure 6: Admin Dashboard Side Bar

1. **Dashboard:** This page, see Figure 7, is used to display statistics about licenses, users, how much licenses are issued, total number of licenses, and available licenses. The dashboard page will be visible and accessible only for admins.

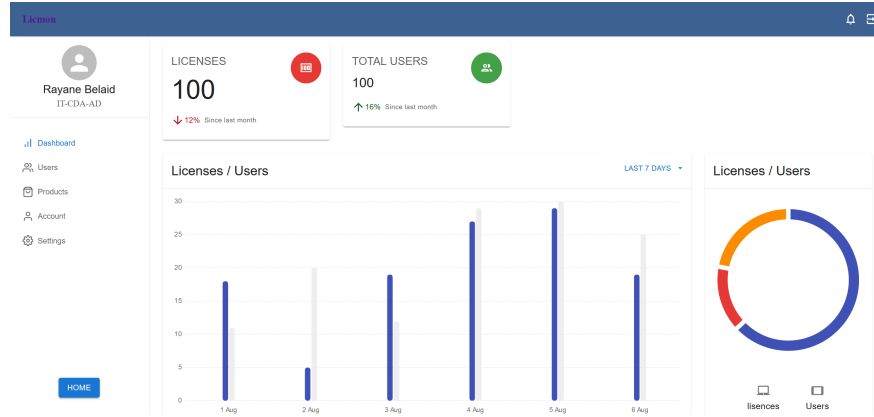


Figure 7: Admin Dashboard interface

2. **Product:** This page as illustrated in figure 8, contain all products including their names, total licenses, and number of licenses issued by users for each product. Users can automatically request a license through a button, this later sends an email to the license holder in order to ask for liberating one if possible. This page will be visible for both users and admins.

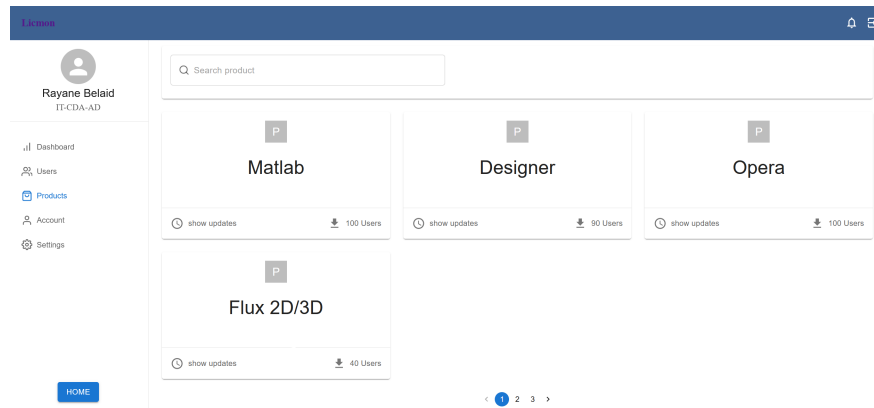


Figure 8: Products list interface

3. **Users:** A list of users holding licenses including their names, their contacts, and the product issued. This page will be visible only for admins. Once the user obtained an engineering tool, it will subsequently issued him a license, then this user will be added to that list as a license holder. See Figure 9.

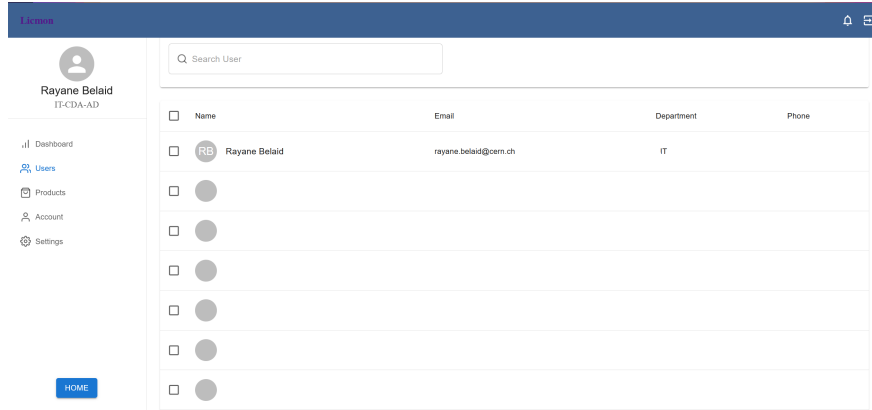


Figure 9: Users list interface

4.4 Deployment

Deploying a Web Application enables to serve the components of a Web Application to clients. Our applications are deployed in OpenShift, as illustrated in Figure 10, using Docker containers. Docker is an open source platform and a virtualization software that runs applications and makes the process easier to develop and distribute [10]. The applications that are built in the docker are packaged with all the supporting dependencies into a standard form called a container. These containers keep running in an isolated way on top of the operating system's kernel. Using operating-system-level virtualization, it offers a set of platform as a service (PaaS) tools, that enable the developer to receive and deliver software in well defined packages, which are containers. The communication between containers is also very clean and simple. Docker makes it easy to build a container system that allows the developer to run lightweight, independent software tasks in any machine. It is easy to configure how docker will be run for someone's project by adding a Dockerfile in it.

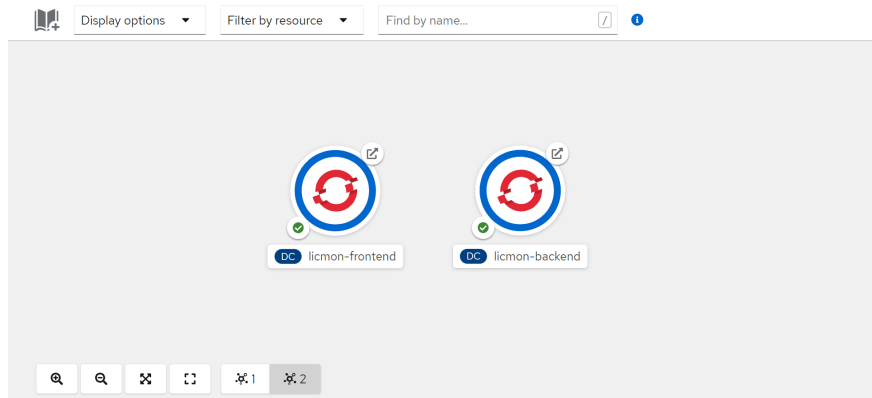


Figure 10: OpenShift deployment

5 Conclusion

Managing software licenses in large and complex companies requires the consolidation of several areas of expertise, in order to make it work efficiently. The objective of our project is the realization of a license management system to maintain an organized software license with detailed information on all engineering tools. Next, allow users to stay updated on license usage and expiration. Also, to provide admins with statistics about licenses and their status. And enable users to request licenses.

The results obtained as part of our project are intended to improve things as regards the problem of the management of software licenses. So far, it will still under building concerning the backend side, and to add more features depends on the need. As well as more functionalities regarding the Admin Dashboard page to showcase the statistics needed which simplifies the tasks and gives more visibility.

This project allowed me to take an insight in web development frameworks, and especially in React, and its libraries, and Flask. I learned about deployment using OpenShift and Docker containers. I also improved my software engineering skills, by revising JavaScript and by using Git. This experience will be valuable for my future career despite all the challenges I have had, especially at the beginning of the internship where my exams coincide with the program. Also, the teleworking was not easy due to unstable connectivity that affected my productivity throughout the internship. Last, I had a great time, I enjoyed the social experience, e-meeting people from all over the world, and great atmosphere which can be sensed even virtually.

6 Acknowledgment

I would like to express my gratitude to CERN for giving me this great opportunity to be a part of this year's Summer Student Programme and to have an insight in all of the CERN resources. As well as working with students from all over the world on an interesting and challenging project, attending the virtual lectures and workshops from several scientists from around the world sharing their knowledge, and Visiting virtually to the accelerators and experimental areas. I am especially thankful to my supervisor Mario Rey Regulez for providing me the tools and resources and guiding me throughout the project, as I am thankful to all the AD section team.

References

- [1] CERN - IT Department — Collaboration, Devices and Applications, URL: <https://information-technology.web.cern.ch/about/organisation/collaboration-devices-application>, 2021.
- [2] CERN Service Portal — Single Sign On and Account Management Services, URL: <https://cern.service-now.com/service-portal>, 2021.
- [3] CERN – our member states, URL: <https://home.cern/about/who-we-are/our-governance/member-states>, 2021.
- [4] Client-server model from Wikipedia, the free encyclopedia, URL: https://en.wikipedia.org/wiki/client-server_model, 2021.
- [5] Facts and figures about the LHC, URL: <https://home.cern/resources/faqs/facts-and-figures-about-lhc>, 2021.
- [6] Materil UI — React and Material UI, URL: <https://material-ui.com/getting-started/usage>, 2021.
- [7] Shalabh Aggarwal. *Flask framework cookbook*. Packt Publishing Ltd, 2014.
- [8] Bonnie Eisenman. *Learning react native: Building native mobile apps with JavaScript*. ” O’Reilly Media, Inc.”, 2015.
- [9] Harpreet Kaur and Pardeep Singh. Uml (unified modeling language): standard language for software architecture development. In *International Symposium on Computing, Communication, and Control*, pages 313–320. Citeseer, 2011.
- [10] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.