

Politecnico di Torino

Master of Science in Engineering and Management



**Politecnico
di Torino**

Master of Science Thesis

Design of an Information System for Instrumentation Management:
the DIGEP Metrological Laboratory Case

Supervisor:

Prof. Gianfranco Genta

Candidate:

Michele Magni

Co-Supervisor:

Prof. Luca Mastrogiacomo
Dr. Giacomo Maculotti

ACADEMIC YEAR 2021/2022

"La pazienza e la perseveranza hanno un effetto magico davanti al quale le difficoltà scompaiono e gli ostacoli svaniscono"

John Quincy Adams (1735-1826)

ABSTRACT

This thesis, developed in collaboration with the “Department of Management and Production Engineering” (DIGEP) within Politecnico di Torino, is a feasibility study of an Information System for the DIGEP Metrological Laboratory in the context of its recognition as “Dipartimento di Eccellenza” by the MUR and potential development.

The new Lab lacks any structured system to store technical documentation and to book and manage utilization sessions (measurements, calibrations, surface characterizations) for its facilities, thus resulting in liability of overlaps and inefficiencies.

The present work is divided into two main sections: a detailed analysis of the problem, framed in managerial perspective, in the first, and a proposal and description of a technical solution in the second. Specifically, the solution consists of a web based application written in Python and rendered with HTML and CSS.

The first chapter briefly presents the context, the analytical tools and frameworks used in this project.

The second chapter analyzes the problem from a managerial perspective, considering its business implications, the stakeholders involved and their actual and potential requirements, and sketching an idea of solution. Such idea is then refined into a conceptual solution in the third chapter, which translates the high level requirements into concepts, and into functional elements in the fourth chapter.

The fifth chapter shows the website implementation, its relevant classes and pages while also describing the most relevant sections of code.

The sixth chapter deals with conclusions and potential future developments.

In addition, the complete code produced is included in the Appendix.

Table of Contents

ABSTRACT	1
Chapter 1	1
Introduction	1
1.1 Background of the Project.....	1
1.2 Purpose of the research	2
1.3 Main Contents and Framework Used	3
Chapter 2.....	4
Feasibility Study	4
2.1 Business Model Canvas	4
2.1.1 Problems and Customers Segments	5
2.1.2 Solution	5
2.1.3 Unique Value Proposition	6
2.1.4 Key Metrics.....	6
2.1.5 Channels	6
2.1.6 Revenue Streams and Costs	6
2.1.7 Unfair Advantage	6
2.2 Stakeholder Analysis	7
2.2.1 Stakeholder Matrix.....	7
2.2.2 SWOT Analysis.....	10
2.2.3 Spider Diagrams.....	11
2.2.4 Venn Diagrams	13
2.3 Problem Analysis	14
2.3.1 Problem Tree	14
2.4 Solution Tree	16
2.5 Strategy Selection.....	17
2.6 Objective Analysis	18
Chapter 3.....	19
System Analysis	19
3.2 IDEFØ.....	20
3.2.1 System Description	21
3.2.2 Main Functions.....	22
3.2.3 Sign UP & Log IN.....	23

3.2.4 Uploads	24
3.2.5 Sessions	25
3.2.6 Lab Facilities	26
Chapter 4	27
System Design	27
4.1 Use Case	27
4.2 ACTIVITY Diagrams	30
4.2.1 Registration & Login.....	30
4.2.2 WEB Structure	32
4.2.3 Booking a New Session.....	33
4.2.4 Lab Facilities	34
4.2.5 Editing or deleting a session.....	35
4.2.6 Uploading/Deleting Documentation	36
4.3 Entity Relationship Model	37
4.3.1 Cardinalities	39
Chapter 5	40
Implementation.....	40
5.1 Tools Used for Implementation	40
5.1.2 IDE -Virtual Environment: PyCharm	40
5.1.3 Programming Language: Python	40
5.1.4 Web Framework: Flask.....	41
5.1.5 Front-end.....	41
5.1.6 Programming Language: HTML and CSS	41
5.1.7 Library: Bootstrap.....	41
5.1.8 Templates.....	42
5.2 Database.....	43
5.3 Forms.....	46
5.3.1 Registration Form.....	46
5.3.2 Reservation Form	47
5.3.3 Upload Form	48
5.4 Routes.....	49
5.5 Pages	53
5.5.1 Registration & Home Page	53

5.5.2 Lab Facilities/Nanoindenter	56
5.5.3 Booking a new Session	57
5.5.4 Uploading new Documentation	59
5.5.5 Account Page.....	60
5.5.6 Editing an existing Session.....	60
Chapter 6.....	62
Conclusion and Future Developments	62
Appendix	63
Acknowledgements.....	81
Bibliography	82
Sitography	82

Chapter 1

Introduction

1.1 Background of the Project

The Department of Management and Production Engineering (DIGEP) inside Politecnico di Torino, is the University reference department for the study of the technological, economical and organizational characteristics of systems of production for goods and services. To that aim, it mixes “traditional” engineering competences with skills in business management, economics and law.

DIGEP promotes relations with national and international universities and research centers as well as with companies, and operates technology transfer. Its approach is interdisciplinary and transcultural, thus allowing the Department to act as a promoter, investigator and developer of research activities in the fields of:

- Managerial economics and business law, markets and new technologies
- Management of innovation and operations
- Engineering system and logistics
- Advanced manufacturing processes
- Design, management and quality of manufacturing processes and products

Most notably, in force of its know-how on manufacturing, production systems and processes, DIGEP constitutes a sort of “cross-joint” unit dealing with the “vertical” departments of the University specialized in technological aspects, and in 2017 has been recognized by MUR as “Dipartimento di Eccellenza“. By virtue of this recognition, DIGEP has been awarded funds to develop studies on the theme of Machine – Human interactions from different perspectives, such as Technological and Managerial/Economical.

Specifically, within the Technological Perspective, a worth mention goes to the Quality and Measurements Research Group. In this context, the group activity focuses on technologies aimed at measuring processes and products quality by working with innovative Sensors Fusions techniques, to improve and conceive innovative statistical quality control apt for zero defect manufacturing within a human-machine interaction framework. Within this context, a relevant research subject is the implementation of new technologies for dimensional measurement of mechanical components and surfaces. To that end, the department plans to enlarge and upgrade its Metrological Laboratory with state of the art instrumentation. This will bring about the need to implement, in parallel, an efficient information system to effectively manage the Lab in its everyday activities.

1.2 Purpose of the research

Based on the aforementioned framework, this thesis aims to study and design an Information System to be implemented in the Metrological Laboratory.

Activities carried out in the lab are:

- Calibration of instrumentation
- Precision dimensional measurements
- Precision mechanical measurements

All these take place in an environment with controlled temperature and humidity in order to avoid dimensional variations for the samples measured. The Lab is equipped with state of the art machines, they are :

- Nanoindenter Anton Paar NHT³
- CSI Microscope Zygo 9000
- FV Microscope
- Contact Stylus
- CMM (“Coordinate Measurement Machine”) GLOBAL
- CMM IOTA
- ATOS Scan Box

The present work focuses on the first two pieces of instrumentation, as they are those mostly involved in Mechanical and Dimensional measurements activities; specifically, it deals with how to manage their technical and safety documentation and how to book their utilization sessions¹.

The results of this work are a feasibility study that can be easily extended to other facilities in the lab and potentially to other labs, and a working prototype (Mock Up) of the Lab information system.

¹**Utilization Sessions** = Time Slots (Days of the week and hours) in which a specific machine is used for experiments, measurements and other activities.

1.3 Main Contents and Framework Used

The design process is structured following a “Top Down” approach, starting from empirical analysis enlightening all issues relevant to the problem and, in accordance to them, defining the most suitable strategy to follow.

In so far, the primary contents and frameworks of the thesis are the following:

- 1) Feasibility Study and System Analysis: in this initial stages, all the stakeholders are identified, and each one is analyzed to understand how he can be affected by the platform and its development. Besides, all known issues are grouped and presented hierarchically in a Problem Tree. This allows to understand which are the requirements, i.e. which functions the System needs to provide, as well as its feasibility.
- 2) System Design: where the functional frameworks of the system are designed, both functionally and technologically. Starting from functionals requirements enlightened in the previous chapters, the system is framed in terms of concepts, i.e. Logical Functions; these functions are then “translated” into functional specifications, adopting UML diagrams.
- 3) Implementation: in this final stage the functional specifications are developed into a working website via actual programming; the programming language chosen for this purpose is Python for its easiness of use, in conjunction with the micro framework Flask for web design.

Further specifications about these tools are detailed at the beginning of Chapter 5.

Chapter 2

Feasibility Study

In the first step of the development of a new project, few but essential questions need to be answered. It is of utmost importance to understand what is the current situation, in order to properly frame the problem, to be able to propose an effective solution. Furthermore, it is necessary to have a comprehensive view of the “actors” involved.

This chapter aims to provide a complete answer to these questions, first by looking at the whole scenario in terms of Business Model, then evaluating how the identified stakeholders are affected by the problems found and what are the relationships amongst them. This is done to select the appropriate strategy on which the system analysis in the next chapter is based on. The frameworks used are Business Model Canvas, Stakeholder Analysis, Problem and Solution Analysis, Venn Diagrams and Spider Diagrams.

2.1 Business Model Canvas

A Business Canvas is a simple and intuitive framework, and can be adopted in first instance to draft a business roadmap. It offers a visual chart with elements describing a firm or product, infrastructure, customers, and finances, assisting businesses aligning their activities by illustrating potential trade-offs. It has been proposed by Ash Maurya, an American entrepreneur and leader of the Lean Entrepreneurship movement which he described in his book “Scaling Management¹”. The canvas is the description and refinement of the business model, a concise business plan which drives the decision-makers in the direction of product or process development strategic planning.

As shown in Figure 1, it is composed of 9 main segments or “Business Grids”: (1) Problems; (2) Solutions; (3) Unique value proposition; (4) Key Metrics; (5) Channels; (6) Customer Segments; (7) Cost Structure; (8) Revenues Stream; (9) Unfair advantage.

¹<https://www.infoq.com/articles/scaling-lean-ash-maurya/>

BUSINESS MODEL CANVAS

PROBLEMS <ul style="list-style-type: none"> • Absence of a platform to properly manage documentation and sessions • Inefficiencies due to time schedule overlaps • Lack of a structured repository to store documentation 	SOLUTIONS <ul style="list-style-type: none"> • Web platform to enable booking and Sessions coordination • Creating a structured database to store scientific documentation 	VALUE PROPOSITION <ul style="list-style-type: none"> • Equipment Sessions Booking and Online Scientific Library • Chance of horizontal extension to other labs 	UNFAIR ADVANTAGE <ul style="list-style-type: none"> • User Friendly Interface • Updated Structured Library • Online Booking with editing options 	CUSTOMER SEGMENTS <ul style="list-style-type: none"> • University Personnel • Students and PHD Students • External Customers (Research Centers, Private Companies, Other Universities)
	KEY METRICS <ul style="list-style-type: none"> • Number of overlaps solved • Number of documents uploaded and downloaded 		CHANNELS <ul style="list-style-type: none"> • “Word of mouth” among University personnel • Advertisements and link on Polito Official Website. 	
COST STRUCTURE <ul style="list-style-type: none"> • Platform Development and Maintenance • Hosting Service Costs • Data Administration Costs 			REVENUE STREAMS <ul style="list-style-type: none"> • Financing coming from Politecnico • Research Award and Grants • Fees paid by external users/customers 	

Figure 1- Business Model Canvas

2.1.1 Problems and Customers Segments

The Canvas core is matching the problems with the customers segment affected. Currently the Lab lacks a structured repository to store its documentation and, although some documents about the Nanoindenter and the Microscope have been digitalized, a significant amount is still stored on paper or DVDs scattered around the Lab. Additionally, there are no formal procedures to book sessions, except for phone calls or voice requests, thus resulting in possible overlaps or misunderstandings. It’s clear how these issues affect firstly Students and Researchers, but also technicians and can, in turn “propagate “ to external entities as well. For instance, difficulties in finding results may lead to delays in fulfilling expectations from an external research center, damaging University reputation.

2.1.2 Solution

Providing a simple and efficient web platform, able to manage Sessions and scientific documentation of the Lab, to which users can easily access, in order to find documents or reserve sessions avoiding time wasting. This is also an advantage, as the Lab is expected to be used not only by internal personnel, but also from external entities who may not be aware of Politecnico internal procedures at first.

2.1.3 Unique Value Proposition

The unique value goal, is to attract the attention of the user, providing something unique and valuable, that constitute a great benefit for them in comparison to the current situation. In this case, an efficient and user-friendly platform with online database can really make the difference.

2.1.4 Key Metrics

The performance of a product after it is launched needs to be measured by indicators that helps identifying key points in the customer's life cycle. The simplest indicator is the numbers of overlaps solved, it also gives hints about website maturity; a very low score may signify the need for the website to be upgraded with new features, or signal possible errors in its code algorithm. Another indicator can be the number of documents uploaded and downloaded.

2.1.5 Channels

The channels used to promote a product or service can be grouped in inbound channels or outbound channels

The first employ "pull strategies" to let customers naturally find the product and include network media, SEO and social networking platforms.

On the other hand, outbound channels use "push strategies" to reach customers and include traditional media or television advertising, as well as direct telephone calls.

In the present case, an effective channel of promotion, aside from information exchanging between personnel (the so called "word of mouth"), can be the insertion of links on the web pages of Politecnico, or directly in the web page of the Quality Engineering and Management Group.

2.1.6 Revenue Streams and Costs

Cost Structure identified for the web platform is mainly related to its initial development within IT services and must also take into account the costs related to rented space in web databases. An efficient cybersecurity system is also needed to prevent infiltration from malignant hackers or dangerous visitors who can try to penetrate Lab Website to stole private data such as documents or patents; this entails additional expenditures.

Being the website intended for a public university, funds are expected to come from university itself, plus fees eventually paid by external users in order to work in the lab.

2.1.7 Unfair Advantage

Also called "Competitive Advantage" is, according to definition "the set of product's features by which the company enjoys higher profitability with respect to its competitors". In other words, which are the distinctive and inimitable qualities of the product. In the present case, the key features are the user friendly interface and the possibility for the user to edit each session.

2.2 Stakeholder Analysis

Stakeholder analysis is a well-established procedure used to analyze individuals, organizations and institutions that are likely to affect or be affected by a proposed action, and to identify the impact of significant stakeholders on the action when developing its strategy.

Stakeholders can influence the project, and their opinions must be taken into account by the decision-makers, most notably because it is impossible for all stakeholders to agree on all issues. Therefore some of them are more influential than others; how to balance the interests of all parties is the key issue in the strategic development.

A Stakeholder Analysis is typically a set of different tools; they are detailed in the following subsections and are:

- Stakeholder Matrix or Description Matrix
- Swot Analysis
- Spider Diagrams
- Venn Diagrams

2.2.1 Stakeholder Matrix

The Stakeholder Matrix is an efficient technique for stakeholder identification. Through this method it is possible to address the most relevant actors directly involved or that could be somehow be affected by the project, understand their interests, motivations and possible actions to address their needs.

The identified stakeholders are :

- **Web Developers:** software houses or IT department within Politecnico in charge of developing the website features, writing the code; they can bring professional development skills and need to be paid for their work.
- **Administrators:** Personnel in charge of managing the website, both in its technical features and its documentation; they have additional privileges which a normal user does not have. They are also responsible for user safety.
- **Politecnico di Torino:** the University itself, which can fund the development of the website
- **Customers:** are hereby named customers all the actors who can interact with the Lab and the platform. As such, everyone who is authorized to access the Lab is a customer. Since scientific activities carried inside the Lab are not limited to Politecnico personnel, two categories of customers have been identified; “Internal Customers” are Professors or Students/Researchers within DIGEP department (but not limited to), while “External Customers” are in general all others actors external to Politecnico who might be interested in conducting research in the Lab, paying appropriate fees.

Table 1- Stakeholder Description Matrix

STAKEHOLDERS	Interest and how affected by the problem(s)	Capacity and motivation to bring about the change	Possible Actions to address stakeholder interest
Developers : IT department or software house who developed the website	<ul style="list-style-type: none"> - Designing a reliable platform 	<ul style="list-style-type: none"> - Professional web development skills 	<ul style="list-style-type: none"> - Income - Potential future contracts
Administrators: Internal personnel, in charge of managing Documentation and access authorizations	<ul style="list-style-type: none"> - Having User Friendly Website - Having a clear view of how authorizations need to be addressed - Having well defined and clear Documentation to manage - Avoiding damages to either the equipment or the people 	<ul style="list-style-type: none"> - An efficient booking system and a structured scientific library means a better usage of the Lab and less risks of time wasting or hazards resulting in physical damages and money to spend 	<ul style="list-style-type: none"> - Fixed Income
Politecnico di Torino: university to which the Metrological Lab and its internal users belongs	<ul style="list-style-type: none"> - Having a better and more functional laboratory - More trained researcher who will in turn deliver better scientific results and publications 	<ul style="list-style-type: none"> - Integrating the website on the DIGEP official website - Funding development process 	<ul style="list-style-type: none"> - Possibility to extend horizontally the model if successful - Gaining higher reputation for the DIGEP department and the university as a whole

<p>Internal Customers:</p> <ul style="list-style-type: none"> - Professors: Belonging to DIGEP or other Departments - Students - PHD Students - Research Fellows - Technicians 	<ul style="list-style-type: none"> - “Making Order” in the session bookings - Structured Scientific Library promptly available - Speeding up learning inside the Lab - Improving quality of scientific research 	<ul style="list-style-type: none"> - Providing Complete Documentation and Results - Using the Lab up to its full capacity 	<ul style="list-style-type: none"> - Developing Thesis or experiments with highly technological equipment which is also more easy to understand
<p>External Customers:</p> <ul style="list-style-type: none"> - Other Universities - Research Centers - Private Companies 	<ul style="list-style-type: none"> - User friendly website and easy access to Lab facilities 	<ul style="list-style-type: none"> - Exchanging recommendations and advices 	<ul style="list-style-type: none"> - Advertisement on Politecnico Official Website - Through “Dipartimento di Eccellenza” recognition

2.2.2 SWOT Analysis

A SWOT analysis is one of the most common analytical theories used when carrying out market or product research. It is a method that summarizes all external and internal conditions of a company or a single product, focusing on four parameters, **S**trengths, **W**eaknesses, **O**pportunities and **T**hreats. The first two factors can be seen as “Internals” because they depend from the organization behavior. On the other end, Opportunities and Threats describe the impact of changes in the external environment on the organization or the project.

The following table is the SWOT analysis of the website.

Table 2 - SWOT Analysis

Metrological LAB SWOT ANALYSIS	
INTERNAL FACTORS	
STRENGTHS (+)	WEAKNESSES (-)
<ul style="list-style-type: none"> - User Friendly Interface - Mockup developed without expenses - Able to manage bookings, avoiding overlaps - Provides an online structured database to store documentation - Documentation classified in categories (Manuals, Calibrations ecc) for each facility 	<ul style="list-style-type: none"> - The website needs to be professionally developed - Administrators are not yet defined with a rigorous procedure - No function to ensure and record if a user has read and understood the general safety procedures in the Lab
EXTERNAL FACTORS	
OPPORTUNITIES (+)	THREATS (-)
<ul style="list-style-type: none"> - Chance of horizontal expansion to other Laboratories across Politecnico - Website modularity; it can be improved by adding multiple functions - Cross link with DIGEP or Quality Engineering Group websites 	<ul style="list-style-type: none"> - Professional development may require more time and additional expenses than expected - University can reallocate funds which are intended for the Lab development, thus blocking or delaying the project

2.2.3 Spider Diagrams

Spider Diagrams are used to show how much each requirement is important for each stakeholder, on a 1-5 scale as seen from the stakeholder's point of view. Stakeholders have been divided in two main categories, Users and Administrators, grouping Internal Customers and External Customers.

Requirements have been identified and classified as follows:

- **User Friendly Interface:** how the usage of the platform is intuitive.
- **Technical Support:** both Users and Administrators expect qualified and quick support to solve possible web related issues.
- **Website Efficiency:** the platform must perform according to specifications and required functions, without delays or bugs.
- **Training/Safety Procedures and Documents:** it's important that whoever works on Lab Facilities, have been properly trained to use such facilities and have read General Precautions and Safety Rules. Such documents must be present and easily accessible in the website.
- **Structured Documentation:** it's important to have organized repositories where each documents is stored and can be easily found, i.e. Manuals for the Nanoindenter and for the Microscope.
- **Documentation Constantly Updated:** to guarantee efficiency, old documentation such as expired versions of documents, i.e. Technical Sheets or out-of-date Lab software must be updated to most recent versions.
- **Easiness in Adding Features/Functions:** This last requirement is important for Administrators, but less for Users, who might eventually appreciate a more complex website "ex Post".

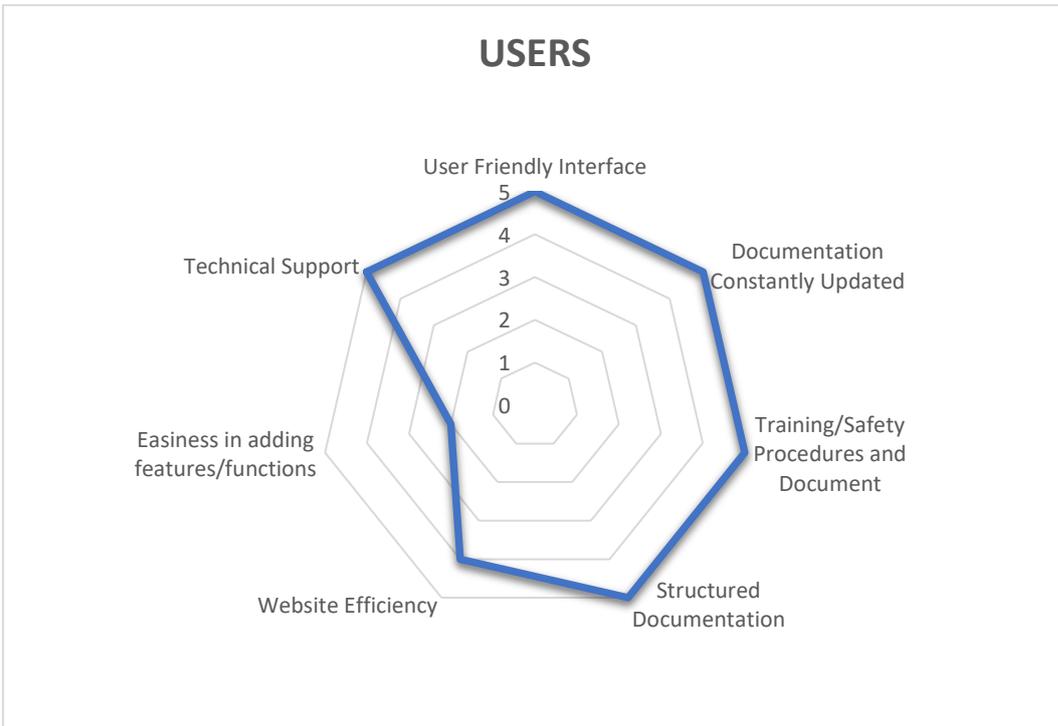


Figure 2 - Users Spider Diagram

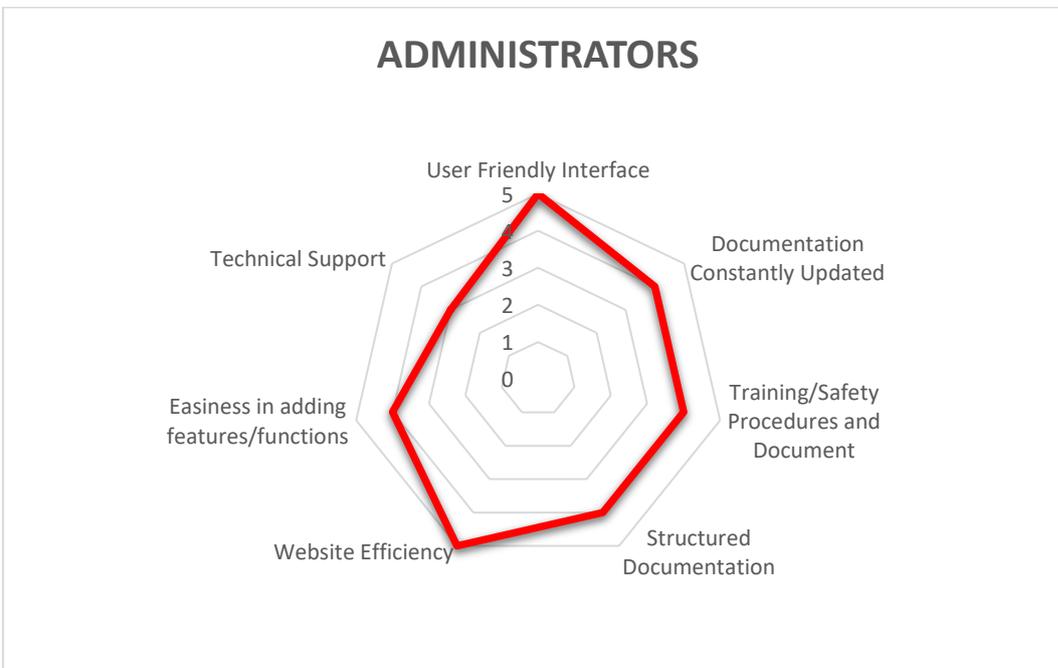


Figure 3 - Administrators Spider Diagram

2.2.4 Venn Diagrams

A Venn diagram is an effective way to graphically show the relationships amongst the stakeholders, providing with a clear visual display. Each stakeholder is identified by one circle, whose size indicates the relative importance of the stakeholder, while the overlaps represent their intersections. As can be seen below, Developers can be internal to Politecnico di Torino, being the IT department, or an external software house. In addition, as detailed in the Stakeholder Matrix section, Administrators and Internal Customers are employees or students of Politecnico and therefore are subsets of it. They also have an intersection since Administrators are chosen amongst internal personnel. On the other hand, External Customers, not belonging to Politecnico di Torino have no intersections with it.

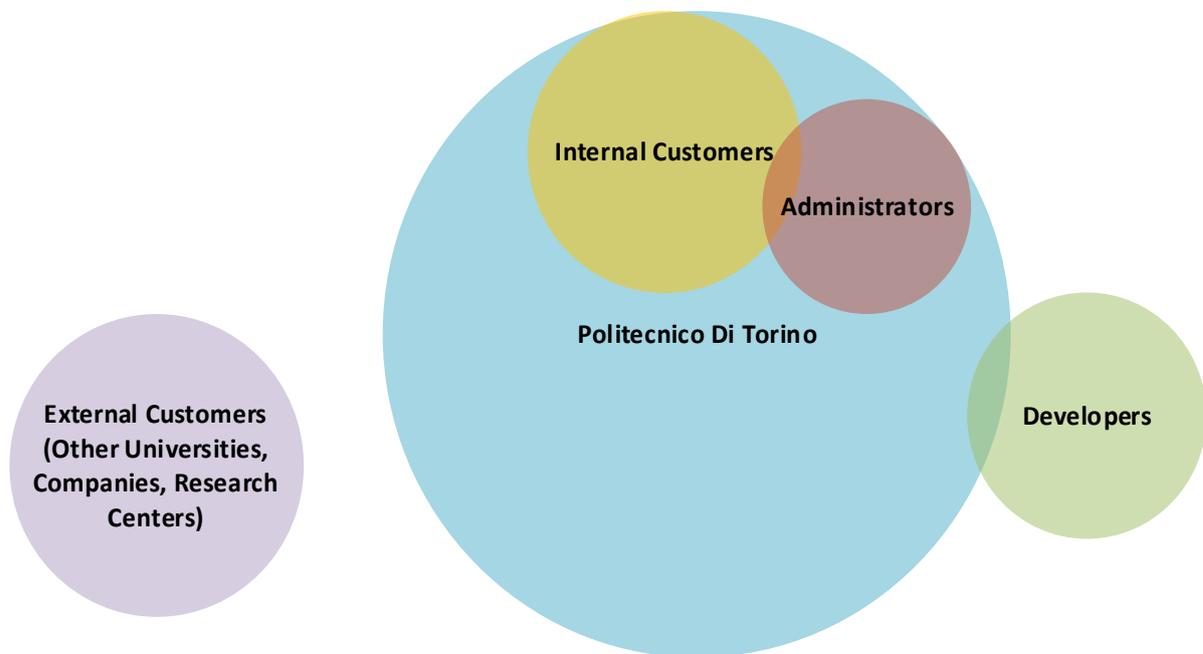


Figure 4 - Venn Diagrams

2.3 Problem Analysis

Problem Analysis is a very important step to successfully execute a project. Its goal is to frame and identify all the negative aspects of the current situation affecting the stakeholders, developing a cause and effect relationship between the various issues faced by them.

2.3.1 Problem Tree

The most efficient way to carry out this analysis is to represent the hierarchy of problems by using a Problem Tree. The “roots” of the tree represent the causes leading to the main problem, which is located in the ”trunk”, while the “branches”, in the upper side, show the negative effects. The main problem that the thesis aims to solve, as shown in the center of the graph, is the difficulty to manage the Lab without a proper system to classify scientific documentation and book work sessions; this is due to various causes linked to the recent renovation of the Lab, with equipment frequently sent to producers for upgrades or maintenance. As a result, documentation needs to be changed very often and stored in appropriate repositories, while currently most of it is written on scattered paper and DVDs.

As shown in Figure 5, the consequences can be quite dire, not only in terms of reputation damage for the department (and by extension for the University), but also for the personnel or who need to use the lab, since the absence of a clear documentation on safety can potentially lead to physical damage for the user and for the instrumentation, which is very expensive to fix and requires a long time.

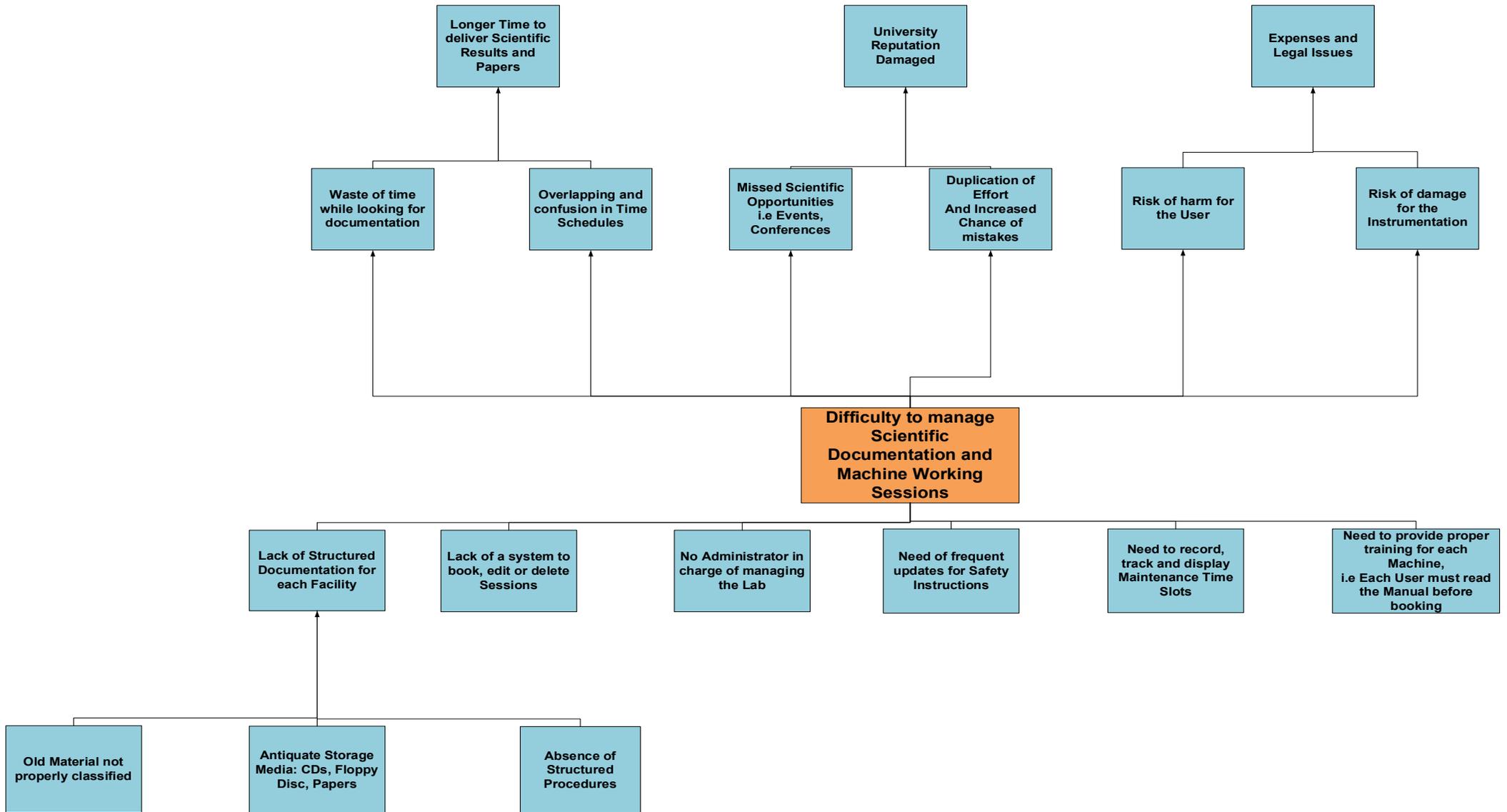


Figure 5 - Problem Tree

2.4 Solution Tree

The Solution Analysis is the logical step after the Problem Analysis. It is symmetrical to the problem tree and it follows the same hierarchical structure, but here the weaknesses and issues become positive and they are considered as strengths and opportunities.

Results are in the lower side: the main purpose for which the Platform will be designed, which is managing bookings and documentation efficiently, lies in the middle part, while the overall objectives are in the upper side.

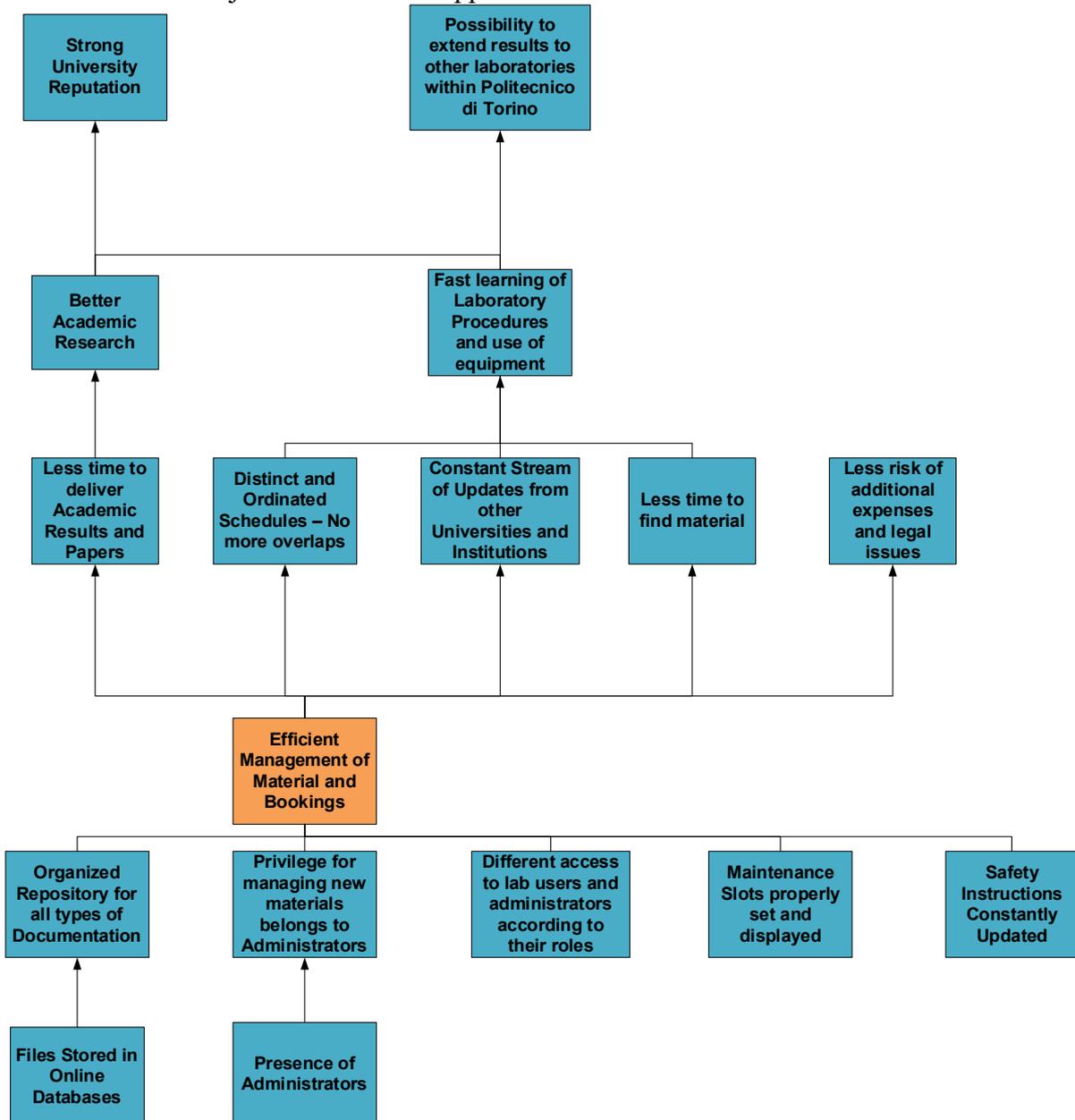


Figure 6 - Solution Tree

2.5 Strategy Selection

Once the problems and the solutions are identified, the next step is to decide which results are attainable through the project, by deleting all factors whose implementation is not feasible in early phases. In particular, the Mock-Up will not deal with maintenance schedules for Lab Facilities and different degrees of authorizations for the users, since neither are available nor have been defined yet.

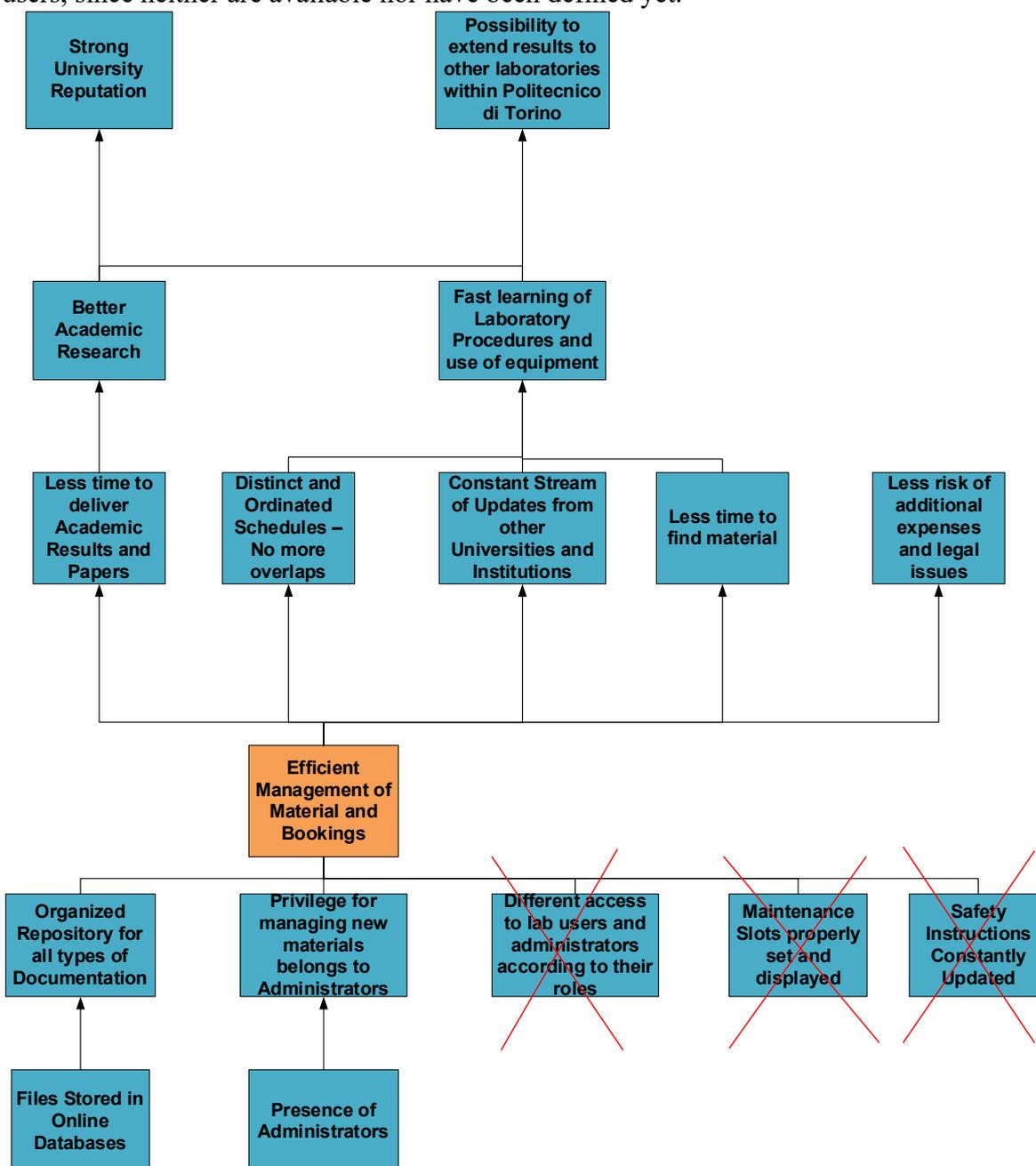


Figure 7 - Strategy Selection Tree

2.6 Objective Analysis

After figuring out the right approach to face problems, useful tools are chosen in order to achieve the selected results in the strategy analysis. As illustrated by the Objective Tree (Figure 8), the four main working areas recognized are Feasibility Study, System Analysis, Design and Implementation. Furthermore, several tools are identified and allocated to each working area with the aim of developing an effective work plan and laying down an activity schedule.

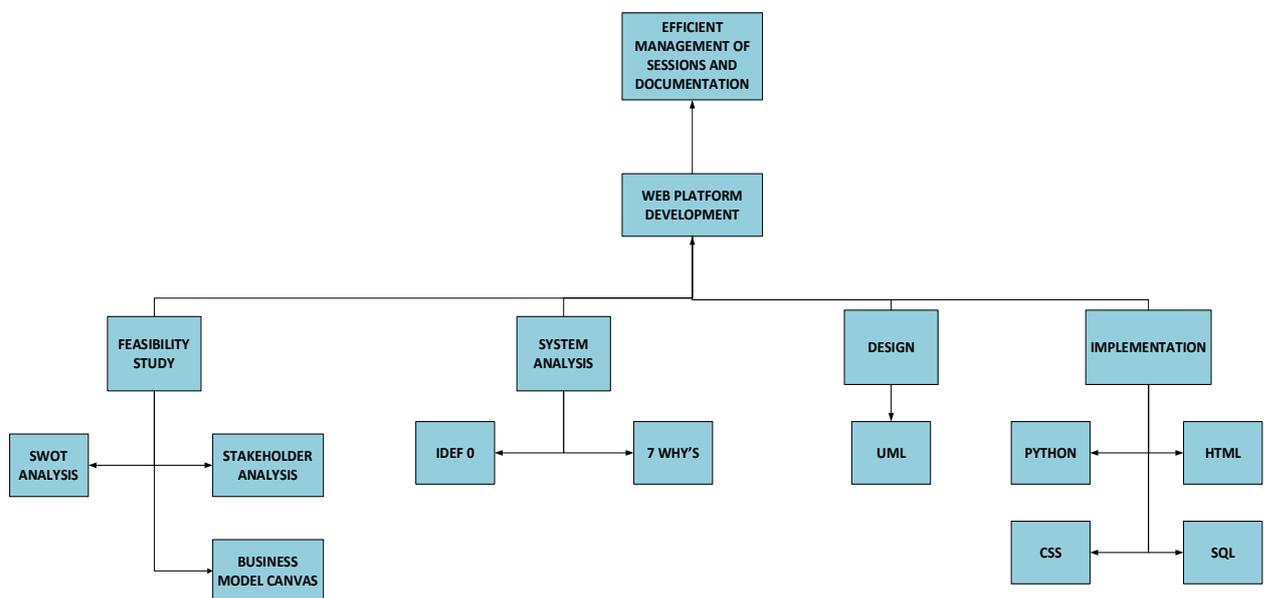


Figure 8 - Objective Tree

Chapter 3

System Analysis

This chapter focuses on the generation of concepts. Having studied the problem and defined a strategy, it is now time to identify the functions that the website needs to provide to satisfy customers' needs. This is done first at empirical level using 7WHY's and IDEF0 tools, and then going further representing actual website structure using UML diagrams in the next chapter.

3.1 7 WHY'S

Table 3 - 7 Why's

WHY? The goal that the website will achieve	Scientific Research is the main activity carried out in a University Department; it must be efficient.
WHAT? The functions of the website and its services	A web platform to manage Scientific Documentation, Sessions and Personnel who is involved with the Lab
WHO? The actors involved in the project	Students, PHD Students, Fellow Researchers Other Universities, Research Centers and Companies
WHERE? The place in which the website can be found	On the Internet
WHEN? The time by which the website is expected to be fully developed and being available	The Website is expected to be operational around April/May 2022
HOW? The activities performed to realize the website	<ul style="list-style-type: none"> - Feasibility Study: Stakeholder Analysis, Problem and Solution Trees, Strategy Selection - System Analysis: IDEF0 - System Design: UML - Implementation - Testing
HOW MUCH? The effort required to develop the project	Around three months for studying the problem and categorizing Documentation inside the Lab, while developing the mockup in parallel. This, plus the effort to write and change software sections, for a total of 6 months

3.2 IDEFØ

IDEFØ is a method designed to model the decisions, actions, and activities of an organization or system. It is derived from a well-established graphical language, the Structured Analysis and Design Technique (SADT), the latter having been commissioned by the US Air Force. IDEFØ is a useful tool for establishing the scope of an analysis, especially for a functional analysis. As such, IDEFØ assists the modeler in identifying which functions are performed, what is needed to perform those functions, what the current system does right, and what does wrong.

In December 1993, the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST) released IDEFØ as a standard for Function Modeling in FIPS Publication 183; more information is available at the following link².

That mentioned, an IDEFØ Analysis has been conducted to summarize and better explain the functions of the website, as well as the main Input/Output and Databases involved. Figure 9 depicts the whole process and is then “exploded” and detailed further in each single component.

Each analysis is represented by a “box”, or a collection of boxes connected and pointed by arrows on each side. Their meanings are respectively:

- Boxes: are the functions requested to the platform.
- Left Arrows : Input Functions, either Users or output from previous functions.
- Top Down Arrows: are all validations performed on the parameters inside functions, each time that one of them is called.
- Bottom Up Arrows: represent all databases involved in functions executions, the one named IS (“Information System”) is the website itself.

²https://www.edef.com/edefo-function_modeling_method/

3.2.1 System Description

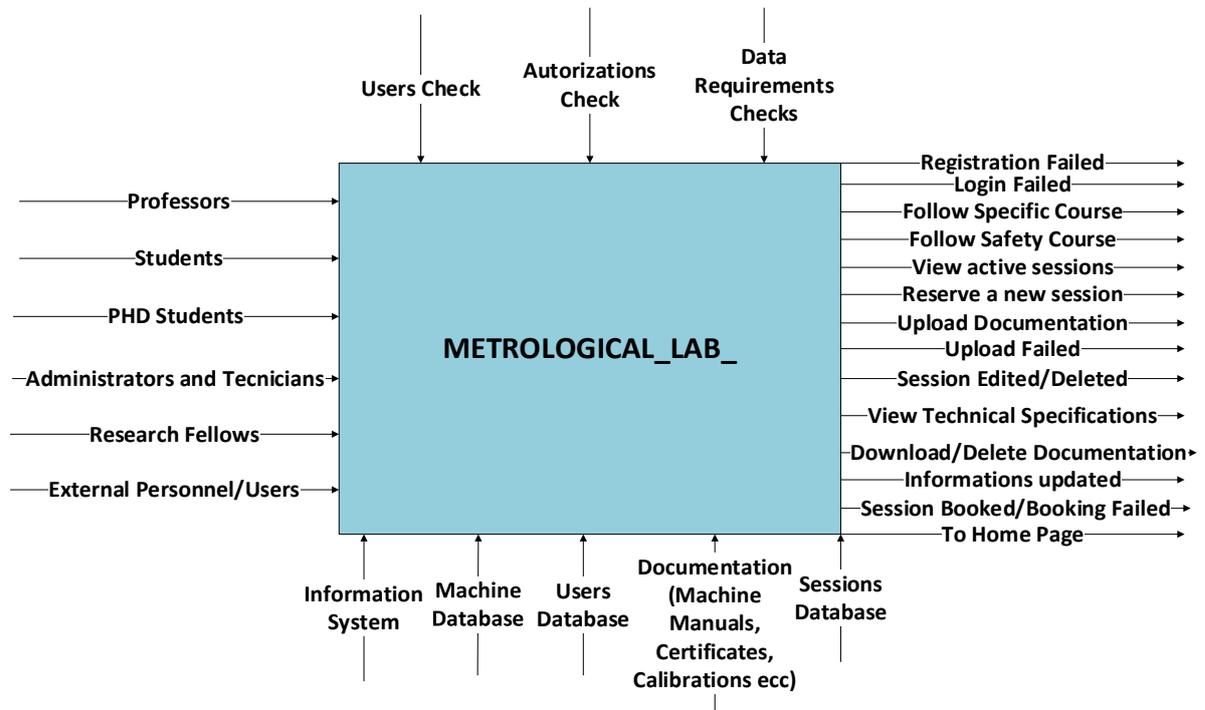


Figure 9 - IDEFO System

In the picture above, the system is depicted as a “Black Box”, only focusing on General Inputs and Outputs; usually in this phase the system is intended AS IS, but for the purpose of the present work, given the need to implement a website from scratch, the Platform is depicted as TO BE. As shown, System inputs are the Users, both Internals and Externals, while four major databases are needed: one for Users, Facilities (here named “Machine”), Sessions and lastly one for all Documentation needed, in file formats.

3.2.3 Sign UP & Log IN

Registration and Log In processes works sequentially, collecting both Structured and Unstructured users data. During both processes the system checks the inserted data to verify if they are in the appropriate type and format. After a successful Registration, all Users are asked to Log In and if no errors are displayed, they can proceed to the Home Page.

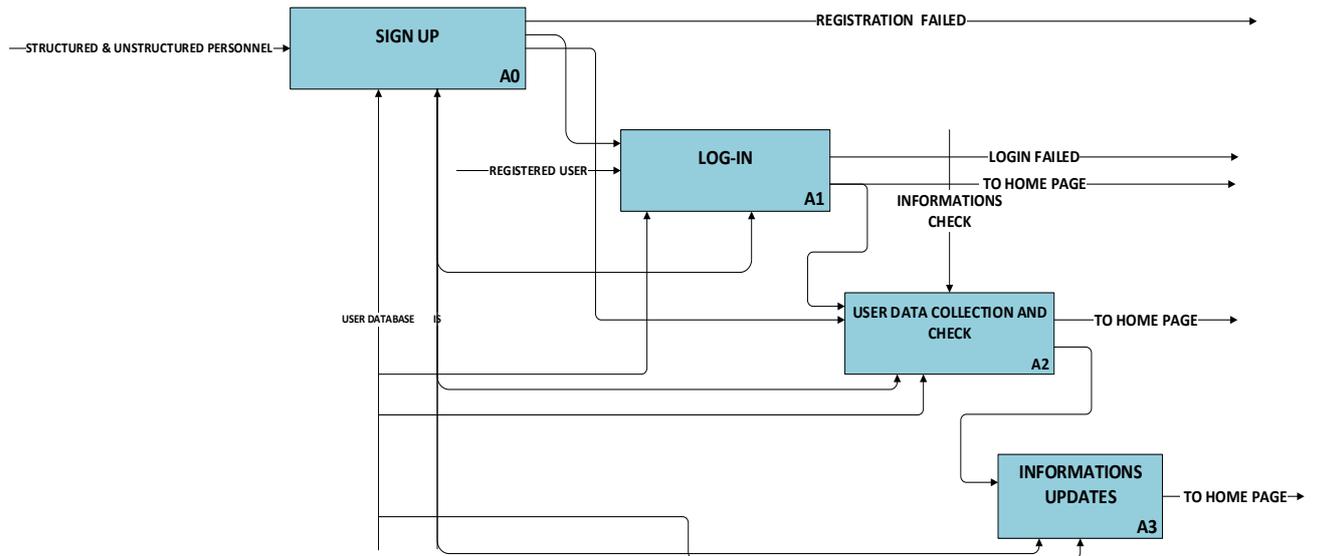


Figure 11 - IDEFO Registration Process

3.2.4 Uploads

The interface and the functions are very simple and intuitive: Administrators must first log in in the system, and after that, they can see all documentation currently stored on the website and, being in charge of its management, they can either upload new documents or delete existing ones.

In any case after each action the system will update the Material Database.

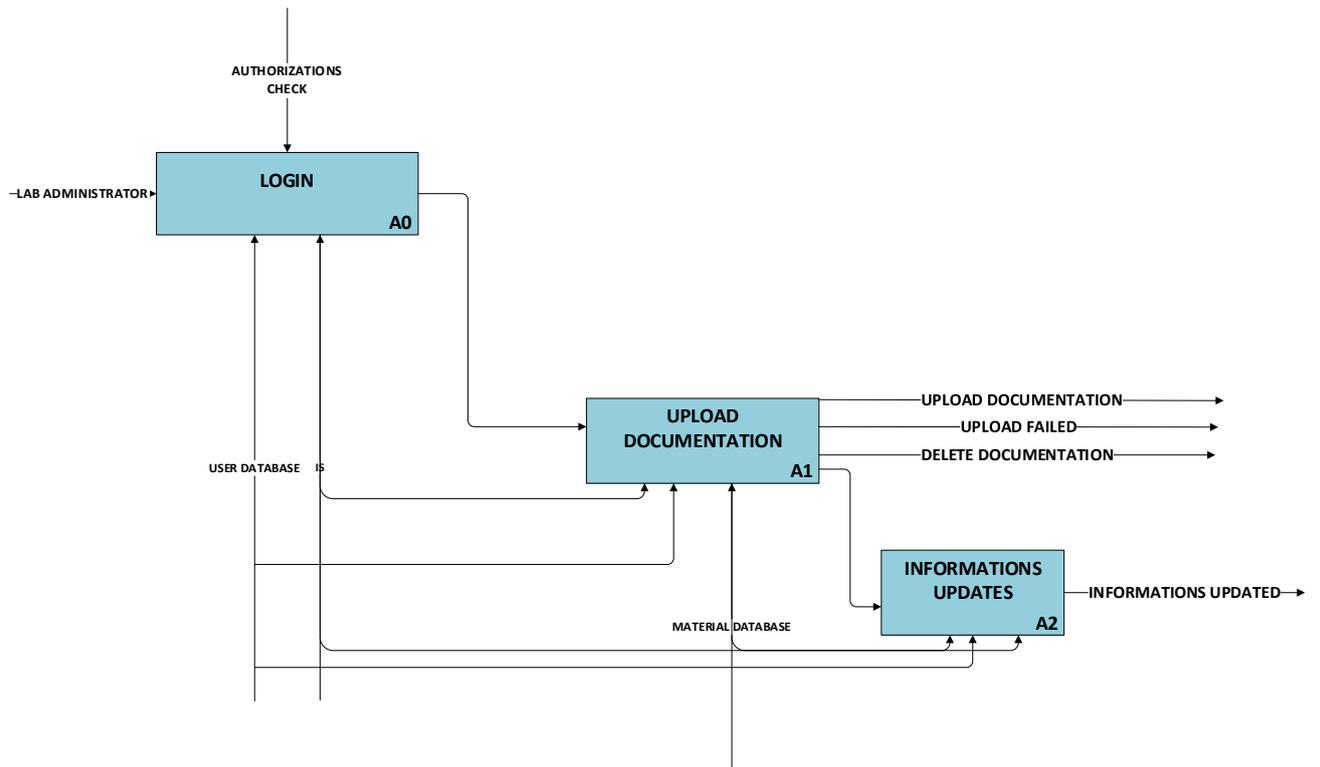


Figure 12 - IDEFO Uploads

3.2.5 Sessions

Only Registered Users have the possibility to book a new session and see which sessions are currently active. In addition, they may also need editing an existing session, i.e. selecting a different time slot or machine. Obviously an authorization control is effected in case of editing or deleting, since each user is allowed to change his sessions only, and not sessions booked by other users.

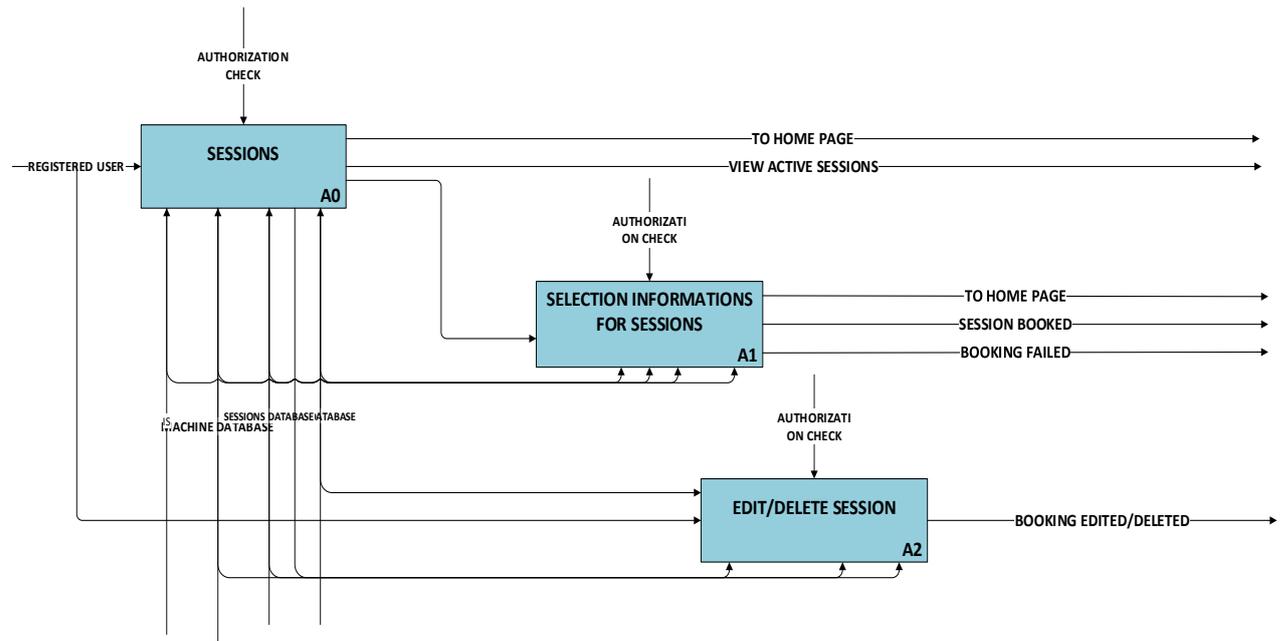


Figure 13 - IDEF0 Sessions

3.2.6 Lab Facilities

Each visitor must be able to retrieve lab documentation (in principle, while some types of documents can be restricted to selected users only) and therefore can access the list of Lab facilities, selecting the one of his interest.

As far as this thesis is concerned, as previous detailed, only Nanoindenter and CSI Microscope are available, each one giving access to its own documentation and training courses.

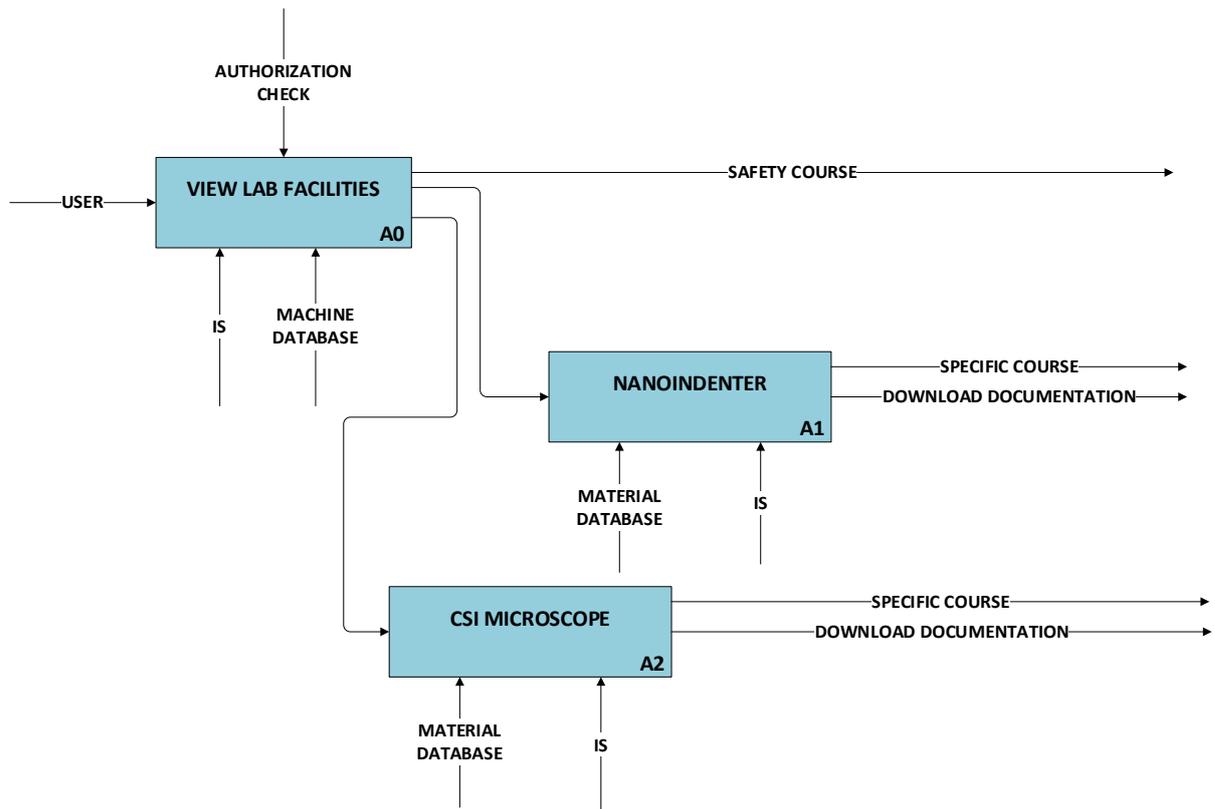


Figure 14 - Lab Facilities

Chapter 4

System Design

4.1 Use Case

In order to understand how the Platform needs to be developed, it is imperative first to have a clear understanding of HOW the main Actors involved will interact with the system; this has already been dealt with previously in raw terms, but for a precise and comprehensive view, another tool is needed.

The USE CASE is the most basic yet powerful diagram available under the UML paradigm and can be used to represent a variety of systems, not necessarily web platform or programs. In general terms it works by defining two distinct Categories:



- ACTORS: are all the users that can interact with the systems
- The System Boundaries, represented as a green bordered rectangle; all the blue boxes within are the functions characterizing the system, each of them in relationship with a user who can access it and for whom it is intended.

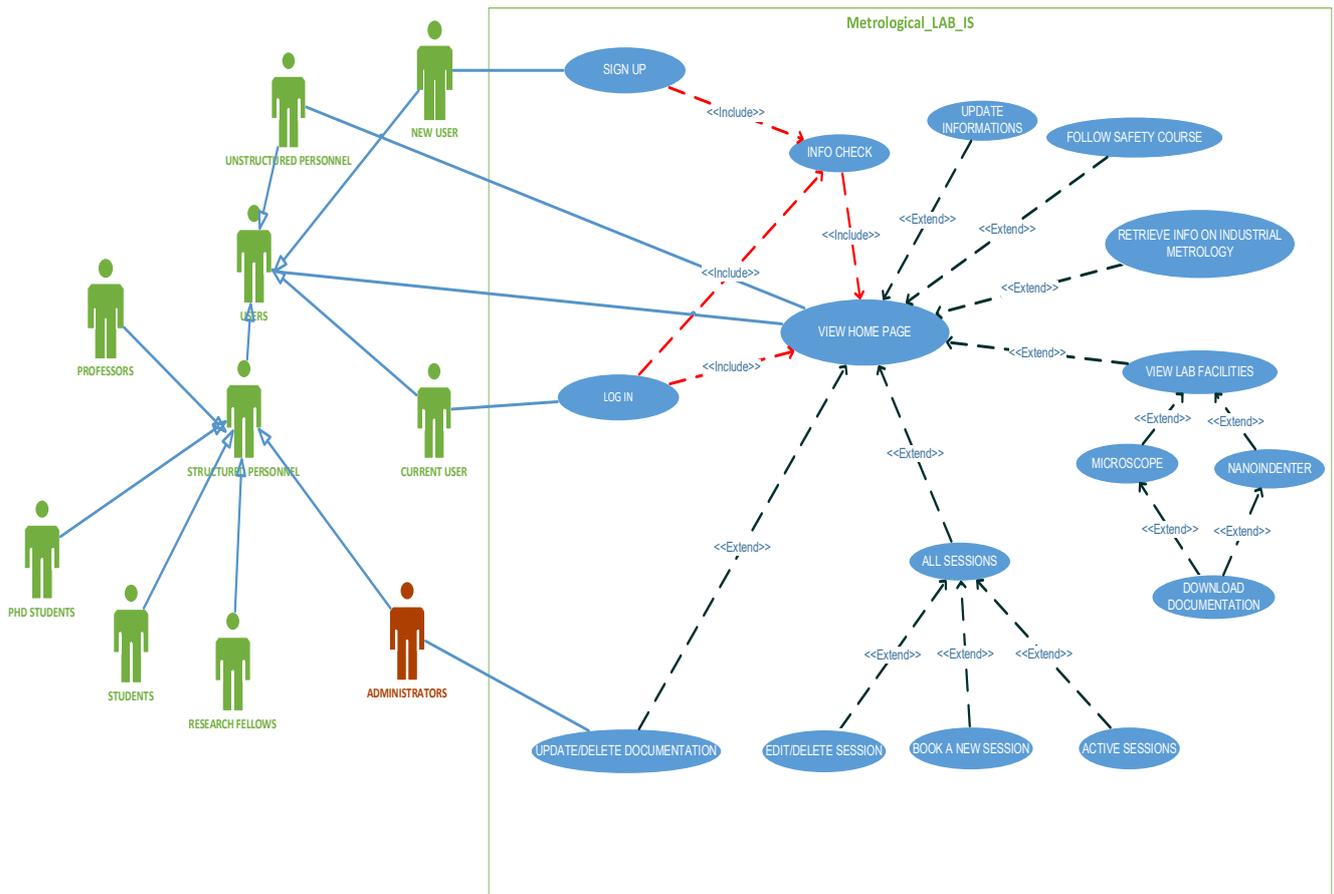


Figure 15 - Metrological Lab Use Case

As shown in Figure 15, several Actors have been identified; in fact the Laboratory could be used by Students, i.e. for thesis experiments, or by PHD students for research purposes, but also by users not belonging to Politecnico such as External Research Centers or Other University Groups.

All Users are classified as “Structured Personnel” if internal to Politecnico, or “Unstructured Personnel“ if not.

Moreover, a User can be Current or New, depending on whether such user is already registered to the website or not. System Administrators, shown in red, are Users with additional functions, since they are the only users allowed to upload Scientific Documentation and remove documents from the system, and are chosen amongst Structured Personnel.

Hence both Current and New Users are specialized actors who inherit from the general actor User. Figure 16 below shows the functions of the System, whereas the dotted lines connecting boxes are the relationship between functions, identified as “Inclusion”, with red arrows or “Extension” with blue ones.

Inclusion is used when any previous step is necessarily performed: for example, if Check is executed, it means that Registration has been executed before. Extension, conversely, describes optional behaviors which are subject to conditions of activation. For instance, a user could choose to access Facilities Page or Sessions Section or none.

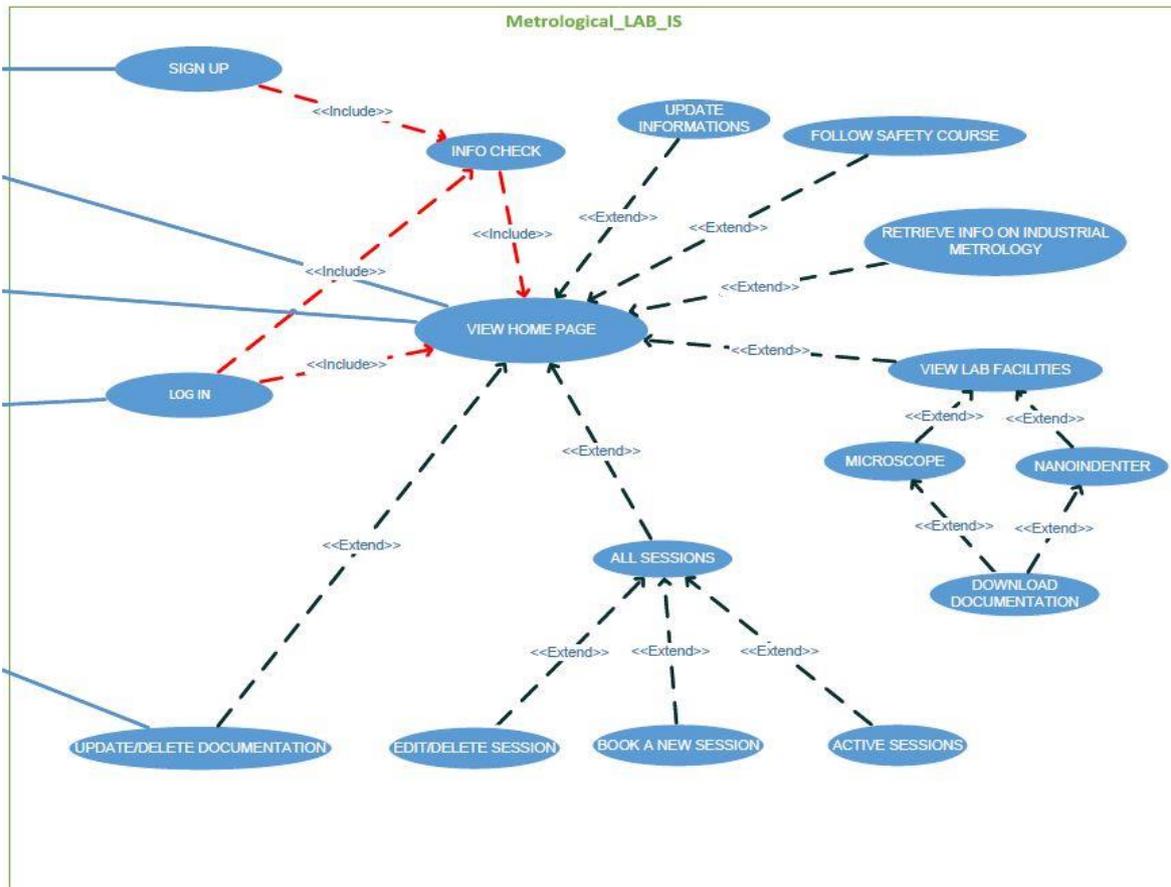


Figure 16 - Use Case Internal Functions

New Users, must register to the platform, which operates a check of the data inserted and then redirects the user to the Login Page which, in turn, can be accessed directly from Current Users. It is also possible for all Users, to access directly the Home Page, although they must be registered and logged to book utilization sessions.

After having logged in, Users can book new sessions or edit existing ones, and if they are Administrators, upload and manage documentation.

There is also a fundamental function called “Follow Safety Course”; no User must get access to the Lab without having read General Safety Procedure, but at the same time someone who has read such Procedure must not be obliged to read it again. Therefore the relation between Home Page and “Follow Safety Course” is an Extension.

4.2 ACTIVITY Diagrams

Activity Diagrams are important tools for the comprehension of the system behavior. They are flowcharts representing the stream from one activity to another, where each activity can be described as an operation of the System. While the Use Case is static, Activity Diagrams are Dynamic.

In this diagram, each symbol has a meaning:



Rectangles represent activities



Diamonds represent alternative paths which can converge/diverge or they can be applied as branching/merging tool;



Forks are used for parallel or synchronized activities and options.

The following subsections depict the different diagrams describing every single page of the Metrological Laboratory System.

4.2.1 Registration & Login

In the Home Page, the user can register if he is new or log in if he has already signed up. In case of registration, the new user is asked to fill in a form with his data (such as email, first name, last name, password, ecc...). The form is submitted and, if it passes the check exam, the new data are saved in the database and the user can proceed directly to the login page, otherwise the user has to fill the form again until the data are correct. In the log in case, the current user fills the form with his email and password and submits it. If the input data are present in the database and are correct, the Home Page opens and he can start his personal navigation; on the contrary, he has to insert his credentials again.

After a successful registration, an email is automatically sent to both new user and administrator. The former receive his credentials, the latter is warned that a new registration occurred; this is implemented to effectively know how many users are registered in the System.

Registration and log in are mandatory for booking sessions with machines. However, a generic user is allowed to access the Website without having to register or log in; this could be the case of a guest researcher or a student who doesn't have permission or interest in using instrumentation, but is just looking for scientific documentation or links to events or articles.

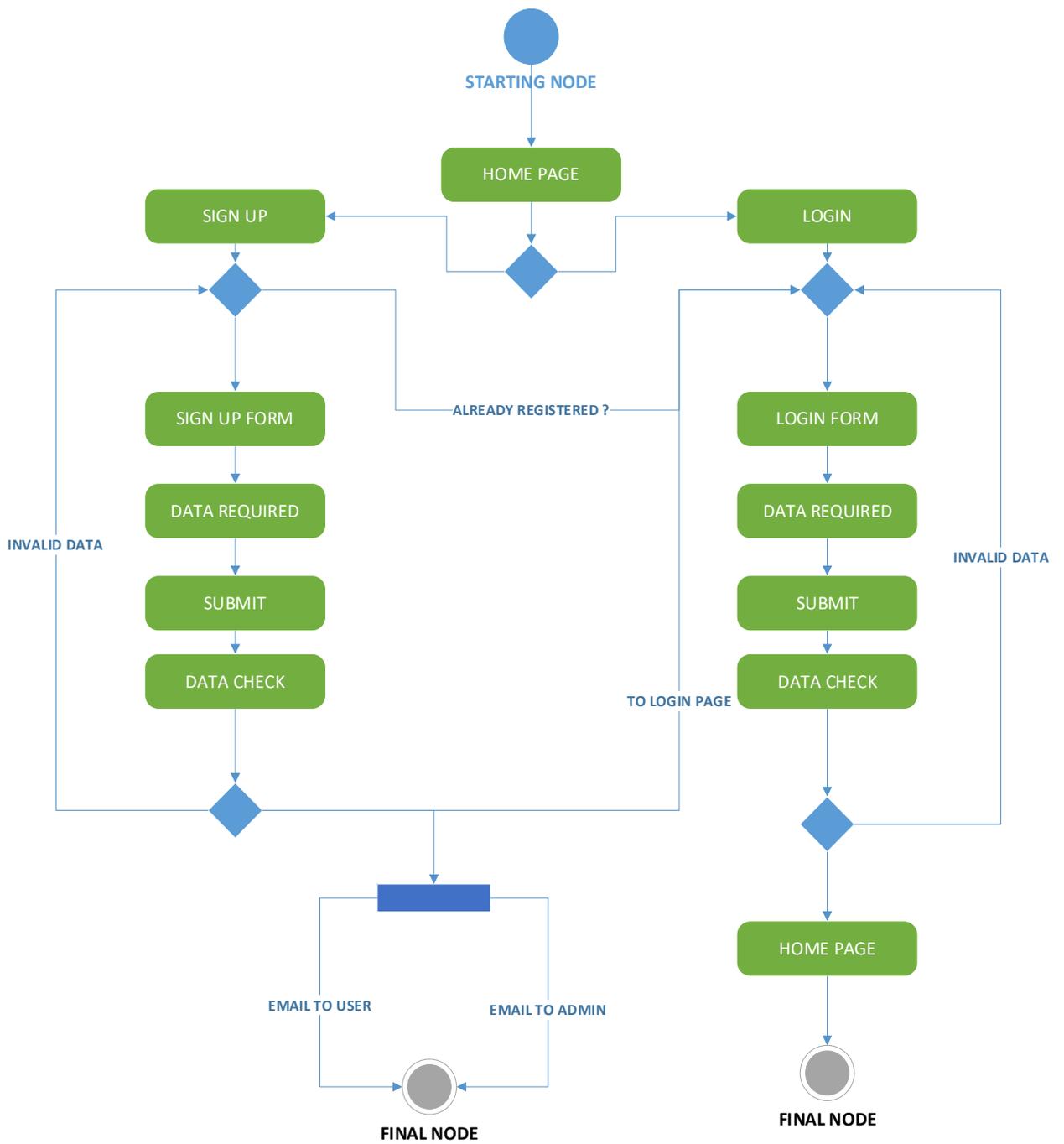


Figure 17 - Activity Diagram Sign-up/Login

4.2.2 WEB Structure

The HOME PAGE is the actual core of the website and has connections to all possible activities. From the home page it is possible, for any user, to access the other functional pages such as Lab Facilities in order to find information about instrumentations, or Contact Page, or simply searching information about recent events in the Industrial Metrology Field (conferences, publications ..) as well as review the available documentation.

An ever present navigation bar eases navigation between pages, and allows system administrators to access both the upload page from which they can upload new documentation, and the comprehensive documentation page; finally, a personal account page is available for each user in order to see and eventually update personal information like emails or usernames.

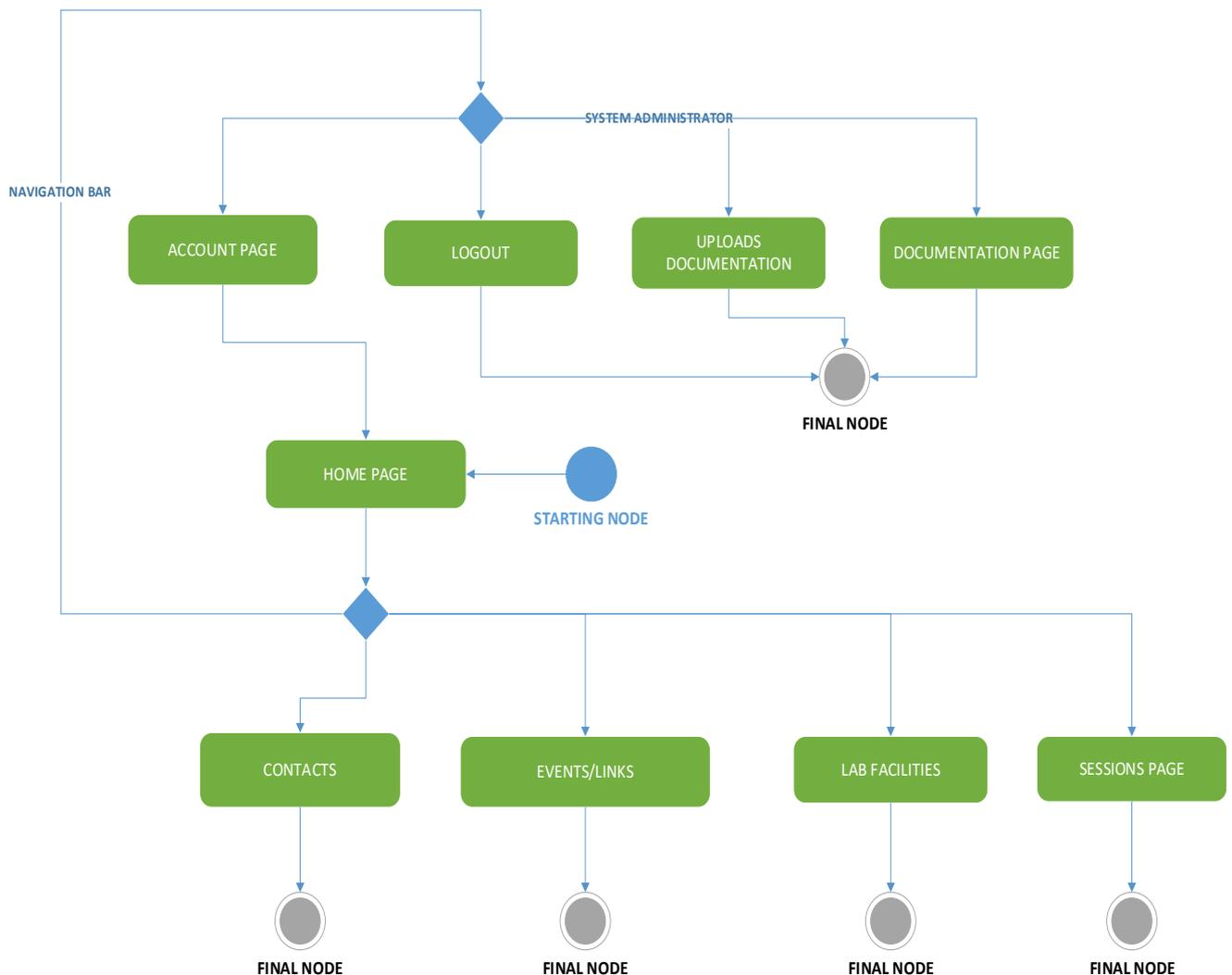


Figure 18 - Activity Diagram Web Structure

4.2.3 Booking a New Session

The second most important aim of the Website is, as previously stated, to allow Structured Personnel, like Students or Researcher, to reserve time slots (here called “Sessions”) to be able to work with Lab Instrumentation. To do so, the Session Page must be opened, and the User trying to reserve a session must be logged in first, since, for safety and technical precautions, no “uninformed“ personnel must be allowed to use the Lab equipment.

After having entered the Sessions Page, the logged user can see which sessions are currently active and reserve a new one. To do so, he must compile a form stating the desired day and hours as well as for which machine he is reserving; the system will then check if the selected timeslot is already booked, thus determining a collision. In such case an error will be displayed, and the user will be redirected back to the booking page; on the other hand, in case the timeslot is free, the session database will be updated with the new reservation, while the user is notified that the session has been correctly booked, and redirected back to the Session Page.

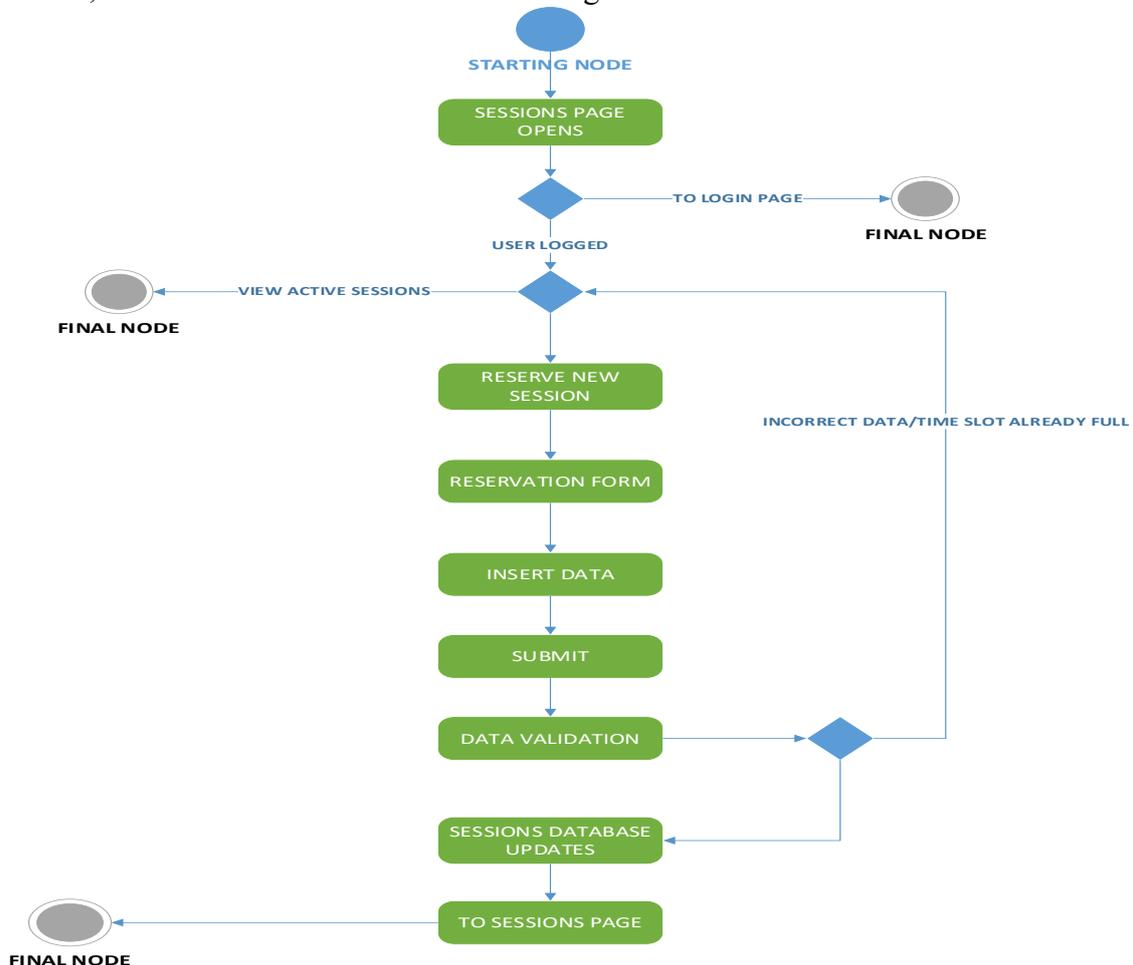


Figure 19 - Activity Diagram Session Booking

4.2.4 Lab Facilities

Figure 19 shows how a user can interact and navigate between the various facilities of the Lab, whether he is registered or not.

From the Home Page it is possible to access to the Lab Facilities Page, which redirects the user to a middle page, where he or she can select the personal page of the machine object of his/her interest. After that, the selected page will open, allowing to retrieve information about the machine, or giving the opportunity for documentation downloading.

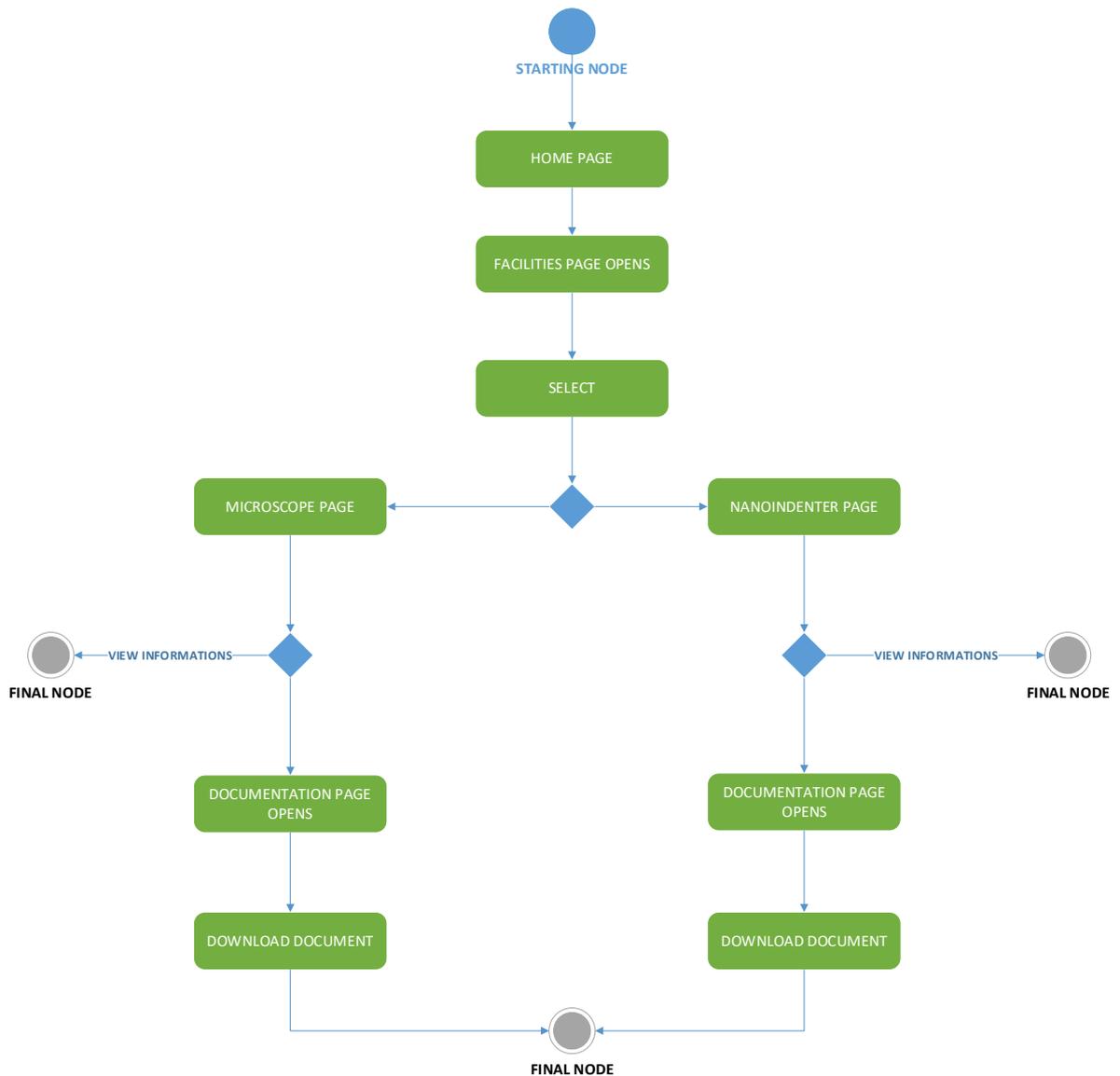


Figure 20 - Activity Diagram Lab Facilities

4.2.5 Editing or deleting a session

Editing or deleting a booked session is a very intuitive process; in fact this option is displayed from the same Session Page used before. After having selected his session, a user is presented with a form to change its parameters (machine, day, hour) or an option for deleting it. The latter requires a further confirmation to prevent unwanted deletions. After a successful deletion, both User and Session databases are updated, while only the latter is updated after a successful update.

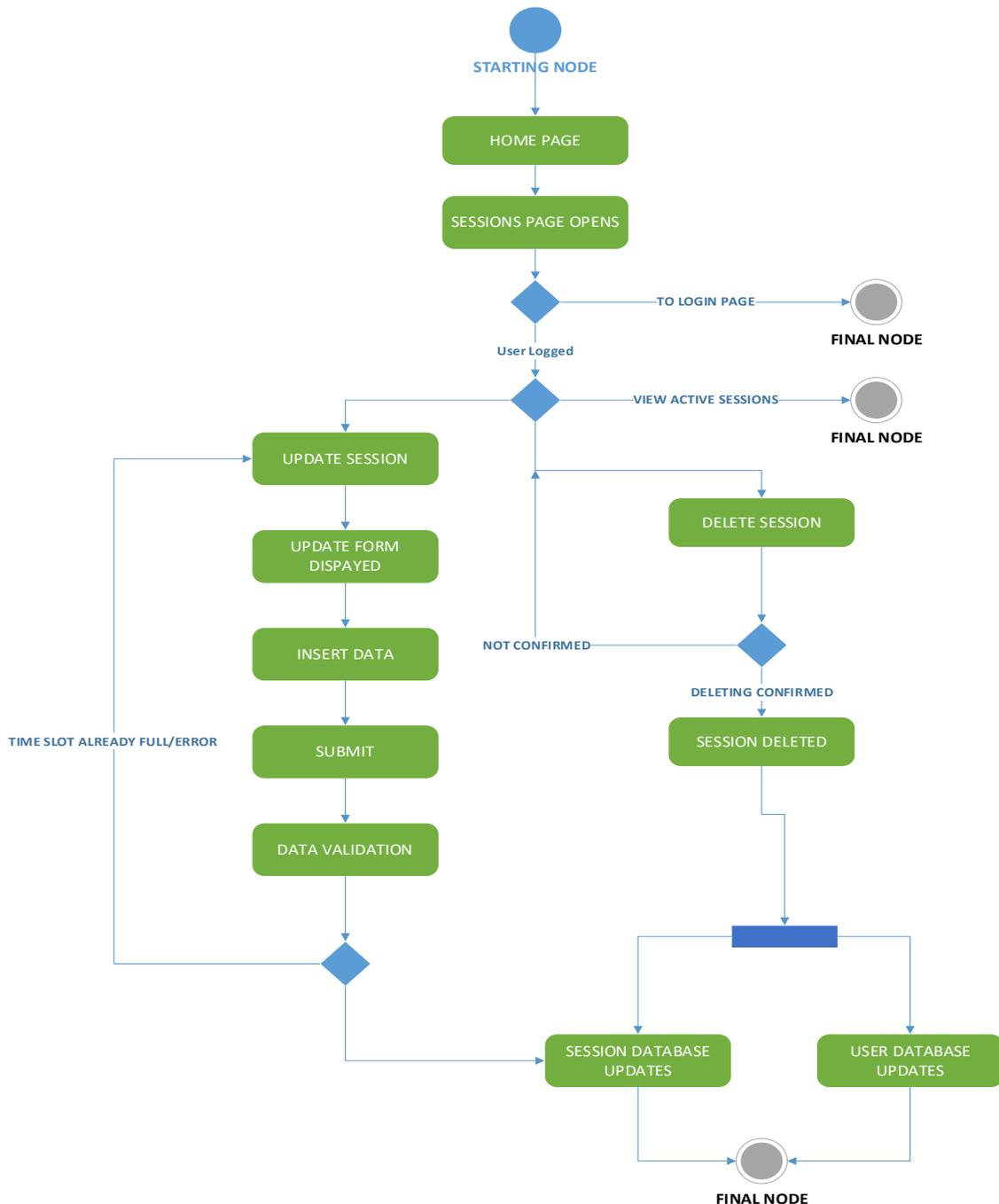


Figure 21 - Activity Diagram Session Editing

4.2.6 Uploading/Deleting Documentation

Only Administrators are allowed to manage documentation, therefore after having logged in and being recognized, they can access the Upload Page. From there a form is displayed, requesting to upload the actual file, and to specify filename and category, i.e. to which machine it belongs, being a Calibration Certificate.

After having uploaded a new document, an Administrator can also download it or delete it from the Documentation Page, which acts as general repository.

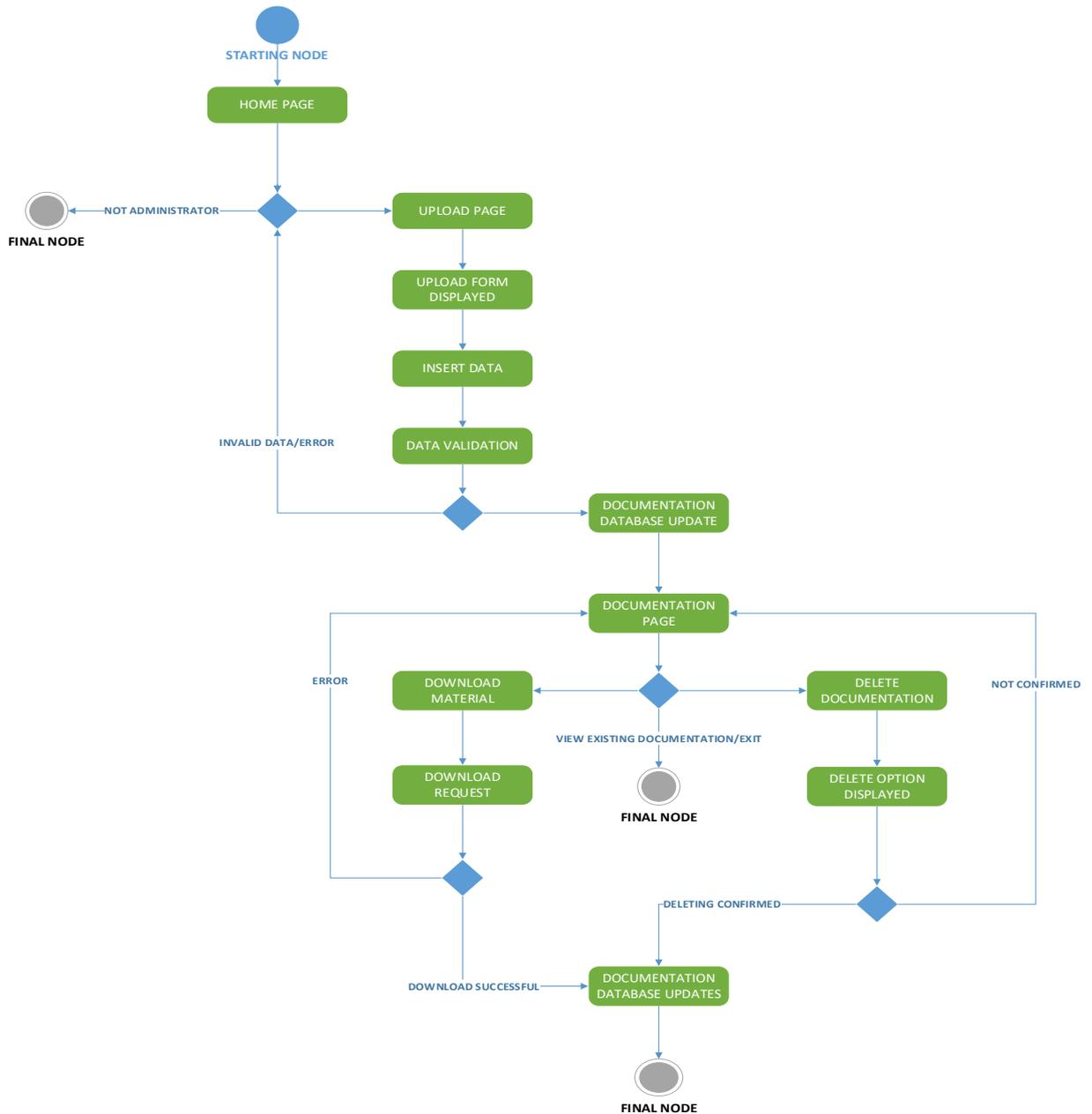


Figure 22 -Activity Diagram Documentation Management

4.3 Entity Relationship Model

After having defined HOW the System interacts with users, as shown before in the USE CASE, it's now important to define how such interactions are translated into relationships. This step is crucial since it shows how "entities" (as defined below) are stored in the website database. In order to do so, the following UML diagram is used, the Entity- Relationship Model.

Such model is composed of a series of symbols that are hereby listed:

-  and : entities in this context are abstract representations of existing objects; they simply exist and have no defined functions. In the database context, an entity is considered only an object for which data can be captured or stored. A weak entity is a sub category of an Entity and as such has no proper attributes.

- : are characteristic of an entity, and each entity can be described by one or more attributes.
- Connection: the modality of connection between entities and weak entities is called connection or relation. Connection can be classified into the following 3 types: one-to-one connection, one-to-many connection and many-to-many connection.

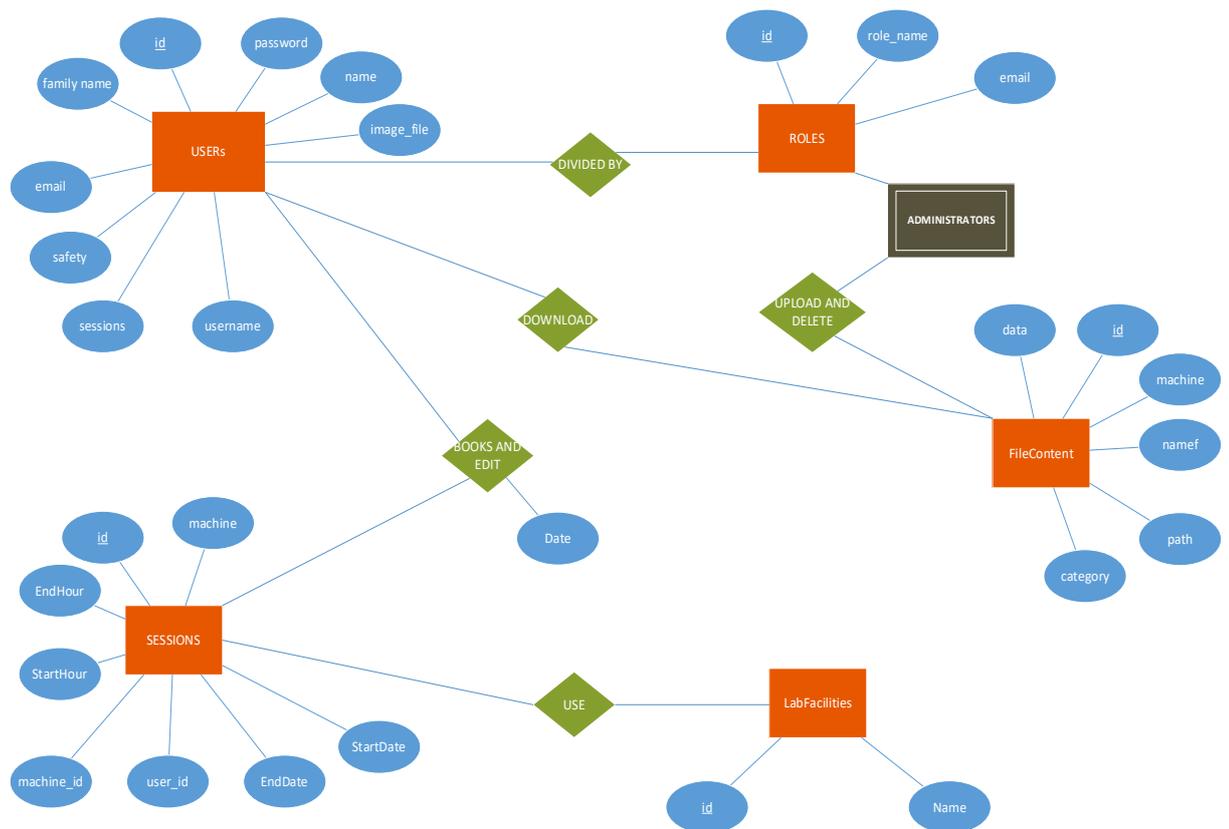


Figure 23 - Entity Relationship Model

All Entities (including weak entities) in the website represent the actors involved and as such, they are:

- **USERS:** all the users who access the Website, both internal and external to Politecnico. They are “Divided By their Role” as a user can be an Administrator, or a simple Student or Professor. They can also interact with the entities Sessions (if registered) and book or edit Sessions, and with FileContent to download documentation.
- **SESSIONS:** the entity represents each session that can be booked or edited by a generic (logged) user. Each session is booked for a single Facility and in a defined date/time.
- **ROLES:** each user, being external or internal to the University, is characterized by a role, which defines in principle what he is allowed to do in the Lab. Therefore, a connection exists with the entity Users.
- **ADMINISTRATORS:** is a weak entity of the entity ROLES, as it constitutes just one of many other possible roles. Being Administrators the only Users who

can upload/delete documentation, they have a relationship named “Upload and Delete” with FileContent.

- **LAB FACILITIES:** are the different machines and equipment currently present in the Lab and in the System; they are related to Sessions since each Session is effected by using one Lab Facility.
- **FILE CONTENT:** all Documentation which is possible to upload and download from the Website is described by this entity. It is related to Users for downloads and to Administrators for upload, editing and deleting.

4.3.1 Cardinalities

Table 4 - Cardinalities

Entity 1	Entity 2	Description	Type of relationship
<i>USERS</i>	<i>ROLES</i>	<ul style="list-style-type: none"> • A User can have multiple Roles • Multiple Roles can be assigned to one User 	1 to many
<i>USERS</i>	<i>FILE CONTENT</i>	<ul style="list-style-type: none"> • Each User can download multiple Files 	1 to many
<i>ADMINISTRATORS</i>	<i>FILE CONTENT</i>	<ul style="list-style-type: none"> • Each Administrator can upload or delete multiple files 	1 to many
<i>USERS</i>	<i>SESSIONS</i>	<ul style="list-style-type: none"> • Each User can book multiple Sessions 	1 to many
<i>SESSIONS</i>	<i>LAB FACILITIES</i>	<ul style="list-style-type: none"> • Each Session is booked for one Machine 	1 to 1

Chapter 5

Implementation

This chapter presents the Website implementation. After having designed the System in terms of functional requirements and concepts, now the Website is actually built in Mock-up form.

The following sections describe the relevant pieces of code that constitute the Website and better represent the analysis carried out so far. Finally, in the end pictures of the website at work are shown and described.

5.1 Tools Used for Implementation

This web application is composed employing the following frameworks: PyCharm as development environment, Python and HTML as programming languages, Flask (including SQLAlchemy as database module) as network service and Bootstrap and CSS as front end styling.

Furthermore, all frameworks and libraries are divided in two macro categories, Back-End and Front-End, depending by their role in the Website.

5.1.1 Back-End

The back-end refers to parts of a computer application or a program code that allows it to operate and that cannot be accessed by a user. Most data and operating syntax are stored and accessed in the back end of a computer system, which is also called the data access layer of the software.

5.1.2 IDE -Virtual Environment: PyCharm

PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for Python programming language. It has been developed by the Czech company JetBrains (formerly known as IntelliJ).

It provides code analysis, a graphical debugger, an integrated unit tester, integration with Version Control Systems (VCSes), and supports web development with Django, Flask, React, as well as data science with Anaconda.

The Community Edition is released under the Apache License for free, but for the sake of the project, the Professional Edition has been used, the license granted for educational purposes.

5.1.3 Programming Language: Python

Python is a programming language often used to build websites and software, and often used for Data analysis. It is a general purpose language; as such can be used for a variety of different programs but is not specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most used programming languages today. In fact, a survey conducted by industry analyst firm Red

Monk found that it was the most popular programming language among developers in 2020.

5.1.4 Web Framework: Flask

A Web Application Framework or simply a Web Framework is a collection of libraries and modules that enables web application developers to write applications without worrying about low-level details such as protocol, thread management, and so on.

Flask is a web application framework written in Python and developed by Armin Ronacher, who led a team of international Python enthusiasts called Pocco.

Flask is based on the Werkzeug WSGI toolkit and the Jinja 2 template engine, which are also both projects of Pocco.

5.1.5 Front-end

The front-end of a software program or website is everything that the user interacts with. From a user point of view, the front-end is synonymous with the user interface. From a developer standpoint, instead, it is the interface design and the programming that makes the interface actually work.

The primary goals of front-end development is to create a smooth or "frictionless" user experience. In other words the interface must be clear, intuitive and as simple as possible.

5.1.6 Programming Language: HTML and CSS

HTML is the language for composing the structure of Web pages.

It gives web designers the instruments to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

CSS is the language used to describe the presentation of Web pages, including colors, layout, and fonts. It allows developer to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent from HTML and can be used with any XML-based markup language.

Specifically, for this project the CSS extension Font Awesome has been used to render more stylish icons, further information in the link below³.

5.1.7 Library: Bootstrap

Bootstrap⁴ is a free and Open Sources CSS framework at responsive, mobile-first front-end web development.

It contains CSS and JavaScript based design templates for typography, layers, forms, buttons, navigation, and other interface components.

As of August 2021, Bootstrap has been recognized as the tenth most starred project on GitHub, with over 152,000 stars.

5.1.8 Templates

Each webpage in the website is based on a template, which is a separate file written in HTML and CSS. Such Files are rendered into pages following Routes instructions detailed in the following sections, as well as the graphic engine Jinja 2 for Flask. In particular Jinja 2 has been used to render the dynamic part of each page, represented inside curly brackets, as shown in the example below.

```
{% for newFile in new %} {% endfor %}
```

Jinja 2 also allows each web page to “inherit” from each other; this feature has been used to simplify the web design, defining a common Navigation Bar and a Layout. This has been achieved by creating a block structure as shown below. Each “block” is automatically included in a “child page”, instead what lies outside can be rewritten with new instructions. It is also possible to overwrite existing instructions in the child template, with the instruction “super()”, which in the following code, is used to implement the block “styles”.

```
{% block styles %}
  {{ super() }}
  <link rel="stylesheet"
        href="{{ url_for('static', filename='font-awesome/css/font-
awesome.min.css') }}">
  <link rel="stylesheet"
        href="{{ url_for('static', filename='style.css') }}">
{% endblock %}
```

Other web pages can then inherit from the base webpage as follows:

```
{% extends "layout.html" %}
{% extends "bootstrap/base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% extends 'NAVBAR.html' %}
```

³<https://fontawesome.com/>

⁴<https://getbootstrap.com>

5.2 Database

The database is implemented with a library in Flask called SQLAlchemy⁵. This library helped defining tables inside the database, using the same object-oriented approach used to define classes, being also able to create a database from scratch, all in the same environment. For the mockup a simple SQLITE3 database named “Test” has been used. For instance, the class User which represents a generic user in the website is the following:

```
class User(db.Model, UserMixin):
    __tablename__ = 'Users'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    family_name = db.Column(db.String(100), nullable=False)
    username = db.Column(db.String(100), unique=True, index=True)
    image_file = db.Column(db.String(20), nullable=False,
default='default.jpg')
    email = db.Column(db.String(100), unique=True, index=True)
    password = db.Column(db.String(200), nullable=False)
    safety = db.Column(db.Integer(), nullable=True, default=0)
    role_id = db.Column(db.Integer, db.ForeignKey('Roles.id'))
    sessions = db.relationship('Sessions', backref='author',
lazy=True)

    def __repr__(self):
        return 'User ({} , {}!)'.format(self.username, self.email,
self.image_file, self.safety)
```

A generic user is characterized by a series of attributes, which are his personal data (family name, name, username, email and password), by an image which is inserted by default, and by a specific attribute called “safety”. The latter is intended to be used for further implementations, his role is assessing if a specific user has downloaded the General Safety Instructions provided in the Home Page. Such information needs to be recorded and stored to prevent possible claims linked to harm or damage to Facilities and people. As shown in the Entity Relationship Diagram, the table User is linked to Roles and Sessions tables respectively. To this aim the attribute “role_id” behaves as a Foreign Key, referencing the table Roles. On the other hand the table Sessions is also

referenced by the attribute Sessions, this one linked using the method “Backref”. Both solutions lead to the same outcome.

The class has some supplementary methods in order to define the properties of the current user. They belong to the Flask-Login⁶ extension that helps manage user sign in sessions. With this extension is possible to limit the access to various pages of the site to only to the already signed in users.

Going forward, the class Roles represents all the possible roles that users can take inside the lab. Its attributes are a specific id for each role, a role name and a relationship established with the class Users.

```
class Roles(db.Model):
    __tablename__ = "Roles"
    id = db.Column(db.Integer, primary_key=True)
    role_name = db.Column(db.String(100), nullable=False)
    users = db.relationship('User', backref='role_name')
```

Another important class is called Sessions. This class represents the actual time slots available for the users to work on Lab Facilities, and as such is characterized by Times attributes requested to the user during the booking process, and by a unique id. The class has also two foreign keys, “user_id” and “machine_id” which are linked, respectively, to the User Table and the Lab Facilities Table.

In particular the last connection is implemented foreseeing future development and an increased number of facilities available in the Lab.

```
class Sessions(db.Model):
    __tablename__ = "Session"
    id = db.Column(db.Integer, primary_key=True)
    machine = db.Column(db.String(100), nullable=False)
    StartDate = db.Column(db.DateTime, nullable=False)
    EndDate = db.Column(db.DateTime, nullable=False)
    StartHour = db.Column(db.Integer, nullable=False)
    EndHour = db.Column(db.Integer, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('Users.id'),
        nullable=False)
    machine_id =db.Column(db.Integer,db.ForeignKey('machines.id'))

    def __repr__(self):
```

```
        return 'Session ({} , {}!)'.format(self.machine,
self.StartDate, self.EndDate, self.StartHour, self.EndHour)
```

The class Lab Facilities is used to represent the actual equipment in use in the Lab, each one with a proper id and Name; currently only the Nanoindenter and the Microscope are listed in the application.

```
class LabFacilities(db.Model):
    __tablename__ = 'machines'
    id = db.Column(db.Integer, primary_key=True)
    Name = db.Column(db.String, db.ForeignKey('documents.machine'))

    def __repr__(self):
        return 'Machines ({} )'.format(self.Name)
```

The last class is FileContent, and constitutes the actual repository for scientific documentation. In order to be used as an actual repository, the field “data” is set to be “Large Binary”, able to store the actual bytes constituting documents. The field “path” is used instead to store the reference to the position (pathway) of a specific document. In such way it is possible to query the database and find the right document requested by pressing the Download Button.

```
class FileContent(db.Model):
    __tablename__ = 'documents'
    id = db.Column(db.Integer, primary_key=True)
    machine = db.Column(db.String(100))
    namef = db.Column(db.String(200))
    path = db.Column(db.String)
    data = db.Column(db.LargeBinary)
    category = db.Column(db.String(300))

    def __repr__(self):
        return 'FileContent ({} , {}!)'.format(self.name, self.namef,
self.category)
```

⁵<https://www.sqlalchemy.org/>

⁶<https://flask – login.readthedocs.io/>

5.3 Forms

A web form, also called an HTML form, can be defined as an online page that allows input by a user. It is an interactive page that mimics a paper document, where users fill out particular fields. In the following lines are presented the most important forms used in the platform, all of which are managed through WTForms⁷. This choice is used to ease implementation, since it allows defining web forms as Python Classes.

As shown, each form possess a series of Attributes, each one constrained by one or more Validators. This returns an error each time a user tries to insert parameters which does not respect basic requirements such as Minimal Length for names, or allowed dominions for Emails. A button called “Submit” is present at the end of each form, its function is sending inserted data to the server to be processed. However, simply submitting data is not enough, since this does not produce any visible action in the web page; therefore “Return” functions are used in the view to redirect users to specific pages after form submissions.

5.3.1 Registration Form

In the Registration Form, used each time a new user requests registration to be able to use the system, a set of validating function called “def validate” are also present. They works querying the database (Class “Users”) to check if the username, email, and password inserted are equal to others already stored, and if any of them it is, display an error.

```
class RegistrationForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired(Length),
Length(max=50)])
    family_name = StringField('Family Name',
validators=[DataRequired(Length), Length(min=1, max=100)])
    username = StringField('Username',
validators=[DataRequired(Length), Length(min=4, max=20)])
    email = StringField('Email', validators=[DataRequired(Length),
Email(), ])
    password = PasswordField('Password', validators=[DataRequired(),
Length(min=6)])
    confirm_password = PasswordField('Confirm Password',
validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Sign Up')

    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
```

```

        if user:
            raise ValidationError('That username is already taken.
Please choose another one')

def validate_email(self, email):
    user = User.query.filter_by(email=email.data).first()
    if user:
        raise ValidationError('That email is already taken. Please
choose another one')

def validate_password(self, password):
    user = User.query.filter_by(password=password.data).first()
    if user:
        raise ValidationError('That password is already taken.
Please choose another one')

```

5.3.2 Reservation Form

With the Reservation Form a user can book a session for a specific machine; to do so he is required to insert the desired timeslot and the specific machine. Each timeslot is defined by a Start Date and an End Date respectively. These two fields are implemented as integers, ranging from 7 a.m. to 20 p.m.

```

class ReservationsMForm(FlaskForm):
    machine=StringField('Machine', validators=[DataRequired()])
    StartDate = DateField('First Day', validators=[DataRequired()])
    EndDate = DateField('End Day',validators=[DataRequired()])
    StartHour = SelectField('Choose starting hour', coerce=int,
choices=([i for i in range(8, 20)]))
    EndHour = SelectField('Choose ending hour', coerce=int,
choices=([i for i in range(8, 20)]))
    submit = SubmitField('Submit')

```

5.3.3 Upload Form

This form is presented only to Administrators, since they are the only users allowed to upload new files. It includes a File Field, which groups all allowed extensions, text, images and working tables respectively. In addition, to ensure information completeness, each single field must be mandatorily filled, and is subject to check by the Validator “Data Required”.

```
class UploadForm(FlaskForm):  
    namedocument = StringField('Filename',  
validators=[DataRequired()])  
    category = StringField('Category',  
validators=[DataRequired()])  
    machine = StringField('Machine',  
validators=[DataRequired()])  
    file = FileField('File', validators=[FileAllowed(['jpg',  
'png', 'pdf', 'txt', 'xms'])])  
    submit = SubmitField('Submit')
```

⁷<https://flask.palletsprojects.com/en/2.0.x/patterns/wtforms/>

5.4 Routes

The principal working logic of backend is routing between the html pages and the server. In Flask it is possible to route an URL to a specific function by using a specific “decorator”, as shown below.

```
@app.route('/register', methods=['GET', 'POST'])
```

The Get or Get/Post methods define whether the information is only being retrieved or also sent to the server. Every time the user requests the URL example.html the following part of code will be executed. The Platform must return a page to the browser, and to do that, it will use a command called “render template”, sending also the dynamic variables in the HTML code, like, the legend “New Book”.

```
render_template('New Session.html', form=form, legend='New Book')
```

Sometimes, data must be inserted automatically into a database when the web is launched, before any actual interaction with the user; this is shown in the section below. The instruction “db.drop_all()” is used at the beginning to clear the existing tables, which are immediately recreated anew by “db.create_all()”. After that, the database is populated with Administrator credential, as well as with the names of the Facilities which are currently being studied. This is done to let the platform administrator be already present in the database whenever the website is launched with no need for Registration. His password is also encrypted using “Generate.Password.hash” method, to prevent intrusions.

```
@app.before_first_request
def create_db():
    db.drop_all()
    db.create_all()
    password_ad = bcrypt.generate_password_hash('279856')
    user_admin = User(name='Michele', family_name='Magni',
username='admin',
                                email='ad@admin.com', password=password_ad)
    equipment1 = Machines(Name='Nanoindenter')
    equipment2 = Machines(Name='Microscope')
    db.session.add(equipment1)
    db.session.add(equipment2)
    db.session.add(user_admin)
    db.session.commit()
    user_query = User.query.filter_by(username="admin").first()
```

```
print(user_query.name)
```

Different pages can be rendered depending on predefined conditions. To illustrate this, it is possible to render the Log In page to a user who is not logged in and is trying to access the profile.html page.

In the route below, two completely different paths are followed depending on whether the request method is a GET or a POST. In the first case it means that the user is just asking for the registration page, and hasn't submitted the form yet. Once the user has submitted the form, the request will be a POST request and the user will be redirected to the login page following a successful registration or to the same page again in case of an unsuccessful one.

In addition, two functions called "send_email_us" and "send_email_ad" are defined. They are used after a successful registration to send two emails, one to the new User and one to the Lab Administrator. The former shows the user his credentials, the latter warns the administrator about the new registration by showing him the new username. For the sake of testing and debugging, both functions use "mailtrap.it" as email sender, while the email text is defined in two separate templates; this means that now the emails are sent to a "sandbox" (protected environment), instead that to real email addresses.

```
# email for User

def send_mail_us(to, subject, template, **kwargs):

    msg = Message(subject, recipients=[to],
sender=app.config['MAIL_USERNAME'])

    msg.html = render_template(template + '.html', **kwargs)

    mail.send(msg)

# email for Admin

def send_mail_ad(to, subject, template, **kwargs):

    msg = Message(subject, recipients=[to],
sender=app.config['MAIL_USERNAME'])

    msg.html = render_template(template + '.html', **kwargs)

    mail.send(msg)

@app.route('/register', methods=['GET', 'POST'])
def register():

    form = RegistrationForm()

    if form.validate_on_submit():

        hashed_password =
bcrypt.generate_password_hash(form.password.data)
```

```

        user = User(name=form.name.data,
family_name=form.family_name.data, username=form.username.data,
                    email=form.email.data, password=hashed_password)

        db.session.add(user)

        db.session.commit()

        send_mail_us(form.email.data, 'Your Registration has been
successful', 'Mail to user', name=form.name.data,
                    username=form.username.data,
password=form.password.data)

        send_mail_ad('michele.magni95@gmail.com', 'New User
Registered', 'Mail to admin', name=form.name.data,
                    username=form.username.data )

        flash("Your Account has been created", 'success')

        return redirect(url_for('login'))

return render_template('REGISTER.html', form=form)

```

In the section below, one of the most important routes of the website is reported, the one which allows a User to book Sessions. To do so, he must previously log in, and therefore must be registered. Being each booking intended for only one machine, the system must first check for possible times collisions. To do so it creates a variable called “collisions”, by querying the Table “Sessions”, using the instruction “Datetime” to combine each present recorded Start Date into minutes, and then using the resulting parameter to filter the records according with the Machine selected by the user. A following “for” cycle then starts, checking the two “if” conditions that identify an overlap and, in case of overlap, returns a message that redirects the user to the booking page.

```

@app.route('/Reservations/New', methods=['GET', 'POST'])
@login_required
def new_Session():
    form = ReservationsMForm()

    if form.validate_on_submit():
        collisions = Sessions.query.filter_by(
            StartDate=datetime.combine(form.StartDate.data,
datetime.min.time())).filter_by(
            machine=form.machine.data).all()

        print(len(collisions))

```

```

        for collision in collisions:
            if form.StartHour.data <= collision.EndHour and
(form.StartHour.data +
                                                    (form.EndHour.data -
form.StartHour.data)) > collision.StartHour:
                flash('Session already booked', 'danger')
                return redirect(url_for('book'))

            booked = Sessions(machine=form.machine.data,
StartDate=form.StartDate.data, EndDate=form.EndDate.data,
                                StartHour=form.StartHour.data,
EndHour=form.EndHour.data, author=current_user)
            db.session.add(booked)

            db.session.commit

            flash('Your Session has been booked!', 'success')

            return redirect(url_for('book'))

        return render_template('New Session.html', form=form, legend='New
Book'

```

5.5 Pages

5.5.1 Registration & Home Page

First of all, while accessing the platform, the User is shown the following page; this page constitutes the Front of the platform. By clicking on the button “To Main Page” the User can access the Home Page of the website, shown in Figure 25, where he can Register or Log In, compiling different forms, shown in Figure 26 and 27. The Login form requires email and password, and can be reached also by the Register Form Page, by clicking on the button “Sign In” bordered in red.



Figure 24 - Website Front

The Home Page, as well as any other page, includes a Navigation Bar to ease navigation. Furthermore the page has two links, bordered red, to DIGEP and Quality Engineering and Management Group websites respectively. A warning is also included; it reminds the Users to download and read General Safety Rules before entering the Sessions Page, by clicking on the nearby button.

The Home Page presents a section on the bottom which groups links (now empties); their purpose is pointing to Scientific Events such as Conferences in the field of Industrial Metrology, to better connect Politecnico di Torino with other research centers, as well as a section related to scientific literature in the same field.

A group of three buttons in the middle allows accessing Session Page, Lab Facilities and Contacts respectively.

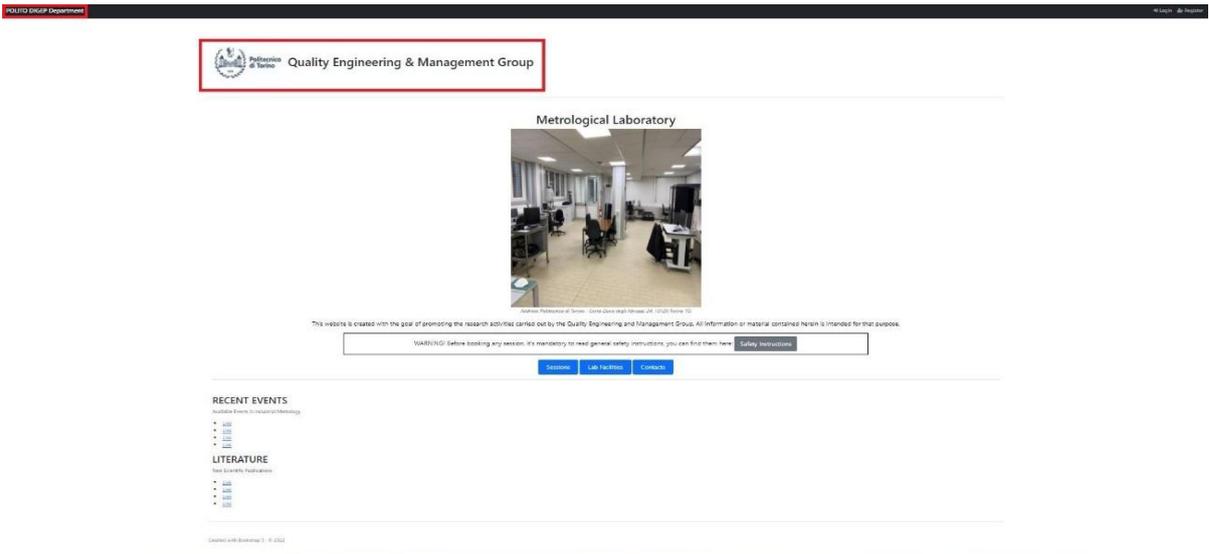


Figure 25 - Website Home Page

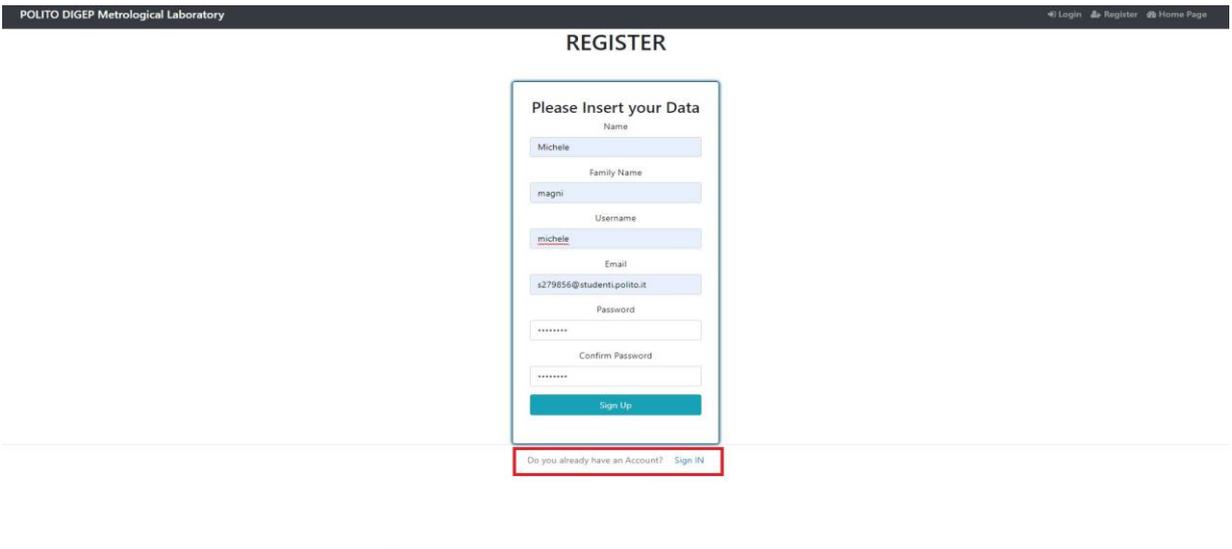


Figure 26 - Website Register Form

LOGIN

Email

Password

Remember Me

[Login](#)

Figure 27 - Login Form

In Figure 28 is shown the email platform Mailtrap.com which works as a sandbox for emails sent after Registration. In this example a generic “User1” receive his credentials.

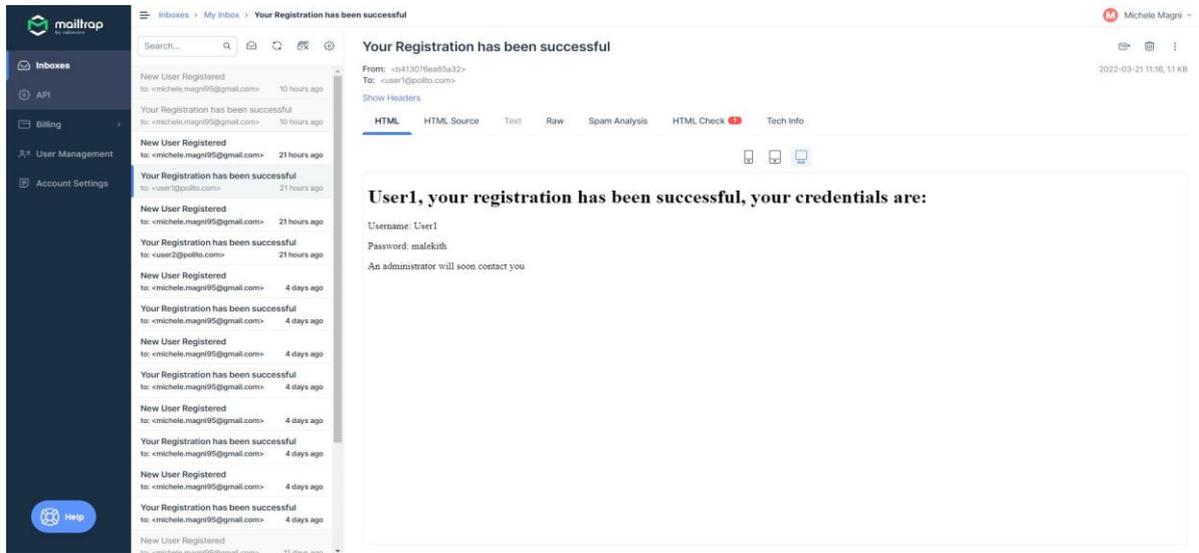


Figure 28 - Emails Sandbox

5.5.2 Lab Facilities/Nanoindenter

From the home page, by clicking on Lab Facilities Button, a user have access to the page shown below; in this page he can find a table in which each row describe a different facilities. As far as this thesis is concerned, only Nanoindenter and Microscope are present. By clicking on the link, the user can then access to the machine specific page; he can hereby find descriptions and eventually, a button will redirect him to the download page.

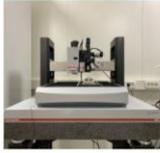
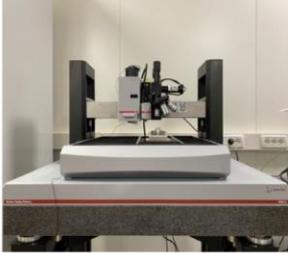
Machine	Model	Description	Photo	Link
Nanoindenter	NHT^3	Used for..		To Page
Microscope	Zygo 9000	Used for..		To Page

Figure 29 - Website Lab Facilities

DIGEP Metrological Laboratory Login Register Home Page

Anton Paar NANOINDENTER NHT^3



Nanoindentation, also called instrumented indentation testing, is a variety of indentation hardness tests applied to small volumes. Indentation is perhaps the most commonly applied means of testing the mechanical properties of materials. The nanoindentation technique was developed in the mid-1970s to measure the hardness of small volumes of material.

[Available Material](#)

Figure 30 - Website Nanoindenter

Material Name	Category	Download
G72IB002EN-B_MHT3_User_Manual.pdf	Manual	Download
G71IB002EN-B NHT3 User Manual.pdf	Manual	Download
G81IB001EN-A MST User Manual.pdf	Manual	Download

Figure 31 - Website Nanoindenter Material

5.5.3 Booking a new Session

To book a Session, a user must first registered and log in, then in the page Sessions, must click on the green button enlightened below. By clicking he is shown a form to compile, in which he must specify the desired Time Slot (Starting and Ending Date and Hours) and for which machine he is making his bookings.



Figure 32 - Website New Session

The image shows a form titled 'New Book' for booking a session. The form is contained within a light blue border. It includes the following fields:

- A dropdown menu labeled 'Select a Machine from the list' with 'Nanoindenter' selected.
- A 'First Day' field with a date input 'gg/mm/aaaa' and a calendar icon.
- An 'End Day' field with a date input 'gg/mm/aaaa' and a calendar icon.
- A 'Choose starting hour' dropdown menu with '8' selected.
- A 'Choose ending hour' dropdown menu with '8' selected.
- A green 'Submit' button at the bottom.

Figure 33 - Website Booking Form

The following popup is displayed at every new booking. If a user wants to proceed and actually reserve a session, he or she is declaring to have read the General Safety Instructions in the Lab (can be found in the home page); this feature is implemented to exempt Administrators and Politecnico from possible issues deriving from facilities misuses. Otherwise a user can simply delete his booking by clicking on the red button.

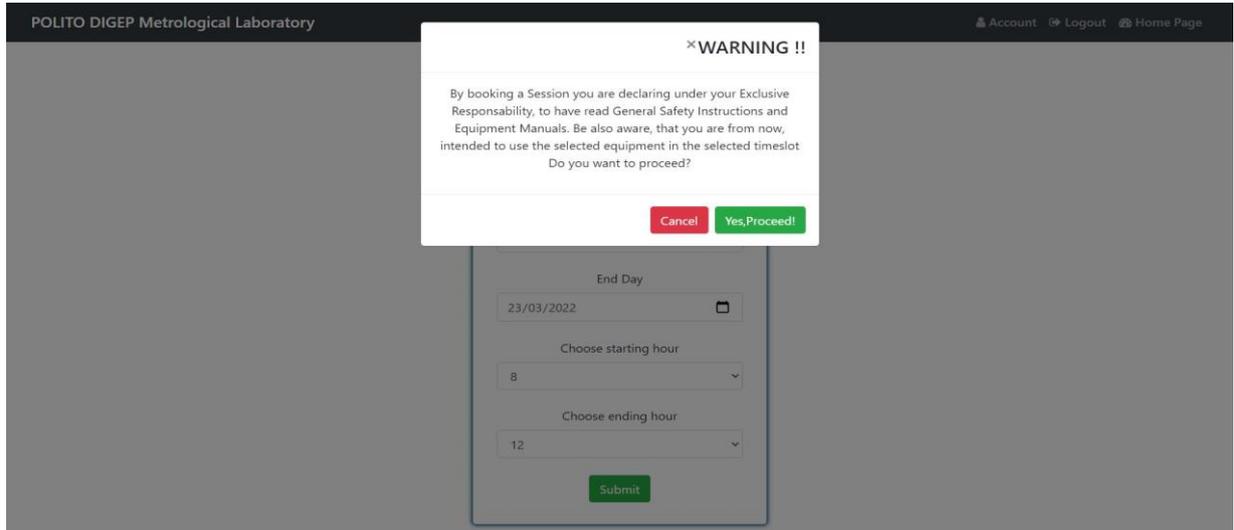


Figure 34 - Website Booking Popup

If no other sessions have already been booked for the same machine, in the same time slot, thus determining a time collision, a new session is recorded and displayed in the general Session Page, along with the bookers username and profile image. On the other hand, an error message is displayed, following redirection to the general Sessions page.

Author	Machine	StartDate	EndDate	StartHour	EndHour
User 1	Nanoindenter	2022-03-22 00:00:00	2022-03-23 00:00:00	8	11
User 1	Nanoindenter	2022-03-24 00:00:00	2022-03-24 00:00:00	8	13
User 1	Microscope	2022-03-25 00:00:00	2022-03-27 00:00:00	12	16
USER2	Microscope	2022-03-30 00:00:00	2022-03-30 00:00:00	14	18

Figure 35 - Website All Sessions

POLITO DIGEP Metrological Laboratory						Account	Logout	Home Page
Session already booked								
New Session								
Author	Machine	StartDate	EndDate	StartHour	EndHour			
admin	Microscope	2022-03-22 00:00:00	2022-03-23 00:00:00	13	15			
User1	Microscope	2022-03-23 00:00:00	2022-03-23 00:00:00	8	9			
User1	Nanoindenter	2022-03-24 00:00:00	2022-03-24 00:00:00	10	12			

Figure 36 - Website Session Already Booked

5.5.4 Uploading new Documentation

To upload new files in the website, an Administrators must first be logged in; his credentials are added in the database when it is launched so he does not need to formally register.

The shortcut “Uploads” shown below is implemented to appear only to a registered admin. In the following page a simple form is displayed, requesting Machine name, Category to which the document belongs and finally the actual file, which is uploaded by clicking on the Submit button.

POLITO DIGEP Metrological Laboratory		Uploads	Material	Account	Logout	Home Page
<h2>Upload New Documents</h2>						
Machine	<input type="text" value="Microscope"/>					
Category	<input type="text" value="Manual"/>					
Scegli file	Nessun file selezionato		<input type="button" value="Upload"/>			

Figure 37 - Website Upload Form

POLITO DIGEP Metrological Laboratory		Uploads	Material	Account	Logout	Home Page
Upload successful						
Documentation	Category	Download	Options			
NewView 9000 Operating Manual 0617_B.pdf	Manual	Download	Options			

Figure 38 - Website Upload Successful

5.5.5 Account Page

In this page, each user can change his personal info, such as username and email of reference, and can also upload a profile picture to substitute the default one, and see his bookings, clicking on “My Active Sessions”.

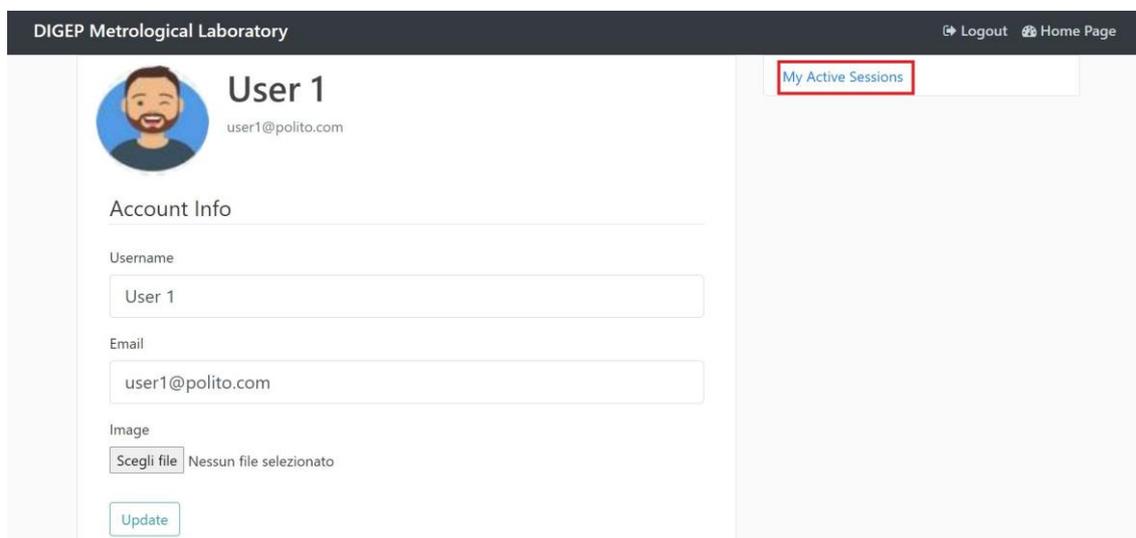


Figure 39 - Website Account Page

5.5.6 Editing an existing Session

To edit a session, a user can simply click on his name on a session he booked previously; each user is authorized to edit only session booked by himself.

User will be redirected to the following page, in which he can see all the bookings he made; clicking on Options will result in the second page below. This page presents each user with two options; he can update his booking by compiling another form as in Figure 33, or can delete his booking by clicking on the delete button below.



Machine	StartDate	EndDate	StartHour	EndHour	Options
Nanoindenter	2022-03-22 00:00:00	2022-03-22 00:00:00	11	13	Options

Figure 40 - Website Editing Reservation

POLITO DIGEP Metrological Laboratory Account Logout Home Page

User 1

[Update](#) [Delete](#)

Booker	Machine	StartDate	StartHour	StartHour	EndHour
User (User 1,user1@polito.com!)	Nanoindenter	2022-03-22 00:00:00	2022-03-22 00:00:00	11	13

Figure 41 - Website Editing Page

POLITO DIGEP Metrological Laboratory Account Logout Home Page

User 1

[Update](#) [Delete](#)

Warning!

Do you really want to delete? The action cannot be reversed!

[Cancel](#) [Yes, delete it!](#)

Booker	Machine	StartDate	StartHour	StartHour	EndHour
User (User 1,user1@polito.com!)	Nanoindenter	2022-03-22 00:00:00	2022-03-22 00:00:00	11	13

Figure 42 - Website Deleting Popup

Chapter 6

Conclusion and Future Developments

Thanks to the development of the proposed platform, is possible to significantly improve the DIGEP Metrological Lab functioning. Currently the lab is at a turning point. The Lab has worked without any structured management tool until now, but now it has been fully furnished and is fully operational, and an expansion and investment plan has started. Such event will bring about growing organizational complexity, which needs to be addressed in advance. Therefore, this work can constitute a simple and effective baseline in that direction. This model needs to be professionally developed into a full-scale system but the solution could be integrated into the DIGEP or in the Quality Engineering and Management Group website, could be enriched by adding other functions and eventually could constitute a model for labs and departments with similar organizational needs.

Such additional functions could be, but are not limited to:

- scheduling and managing the maintenance of the relevant equipment.
- proper definition of administrators.
- restricting registration to emails with a specific dominions.
- introducing a deadline after which sessions booked are “frozen” and cannot be edited or cancelled.
- possibility to delete a user account.
- organizing the database as a system of different folders, with the possibility of moving documentation between them.
- introducing different rights of access for different users, controlled by Administrators.

Appendix

1-Flask Modules

Beaker	1.11.0
Flask	2.0.2
Flask-Bcrypt	0.7.1
Flask-Bootstrap	3.3.7.1
Flask-Bootstrap4	4.0.2
Flask-Helper	1.2.7
Flask-Login	0.5.0
Flask-Mail	0.9.1
Flask-Reuploaded	1.2.0
Flask-SQLAlchemy	2.5.1
Flask-Uploads	0.2.1
Flask-WTF	1.0.0
Flask-Webhelpers	0.1
FormEncode	2.0.1
Genshi	0.7.5
Jinja2	3.0.3
MarkupSafe	2.0.1
Paste	3.5.0
PasteDeploy	2.1.1
PasteScript	3.2.1
Pillow	9.0.0
Pygments	2.10.0
SQLAlchemy	1.4.27
SQLAlchemy-Utils	0.37.9
Tempita	0.5.2
WTFormValidation	1.0.0
WTForms	3.0.0
WTForms-Alchemy	0.17.0
WTForms-Components	0.10.5
WTForms-Ext	0.5
WebError	0.13.1
WebHelpers	1.3
WebOb	1.8.7

Werkzeug	2.0.1
Werkzeug-Raw	0.0.2
bcrypt	3.2.0
blinker	1.4
certifi	2021.10.8
cfffi	1.15.0
charset-normalizer	2.0.8
click	8.0.3
colorama	0.4.4
decorator	5.1.0
dnspython	2.1.0
dominate	2.6.0
email-validator	1.1.3
greenlet	1.1.2
idna	3.3
infinity	1.5
intervals	0.9.2
itertoolz	0.5
itsdangerous	2.0.1
numpy	1.22.1
pandas	1.4.0
pip	21.2.3
pyparser	2.21
python-dateutil	2.8.2
pytz	2021.3
requests	2.26.0
setuptools	57.4.0
simplejson	3.17.6
six	1.16.0
transaction	3.0.1
urllib3	1.26.7
validators	0.18.2
visitor	0.1.3
werkzeug-auth-middleware	0.1.2
zope.interface	5.4.0

2-Imported Functions

```
from flask import Flask, render_template, redirect, url_for, flash, request, send_file, abort
from flask_bootstrap import Bootstrap
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import login_user, current_user, logout_user, login_required, LoginManager
from PIL import Image
from io import BytesIO
from datetime import datetime
import os
import secrets
from flask_mail import Mail, Message
```

3 – Settings

```
# app instantiation
app = Flask(__name__)
app.config['SECRET_KEY'] = 'b407c1bbf047582ddasdcb97344e'

# app SQLAlchemy Database Settings
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///Test.db'
app.config['SQLALCHEMY_COMMIT_TEARDOWN'] = True
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Emails
app.config['MAIL_SERVER']='smtp.mailtrap.io'
app.config['MAIL_PORT'] = 2525
app.config['MAIL_USERNAME'] = 'b413076ea85a32'
app.config['MAIL_PASSWORD'] = 'f2b94cccf5bd19'
```

```

app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USE_SSL'] = False

# Uploads
UPLOAD_FOLDER = 'static/Files'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}

# Settings
mail = Mail(app)
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
Bootstrap(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

```

4 – Database

```

from flask_login import UserMixin
from app import db

class User(db.Model, UserMixin):
    __tablename__ = 'Users'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    family_name = db.Column(db.String(100), nullable=False)
    username = db.Column(db.String(100), unique=True, index=True)
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    email = db.Column(db.String(100), unique=True, index=True)

```

```
password = db.Column(db.String(200), nullable=False)
safety = db.Column(db.Integer(), nullable=True, default=0)
role_id = db.Column(db.Integer, db.ForeignKey('Roles.id'))
sessions = db.relationship('Sessions', backref='author', lazy=True)
```

```
def __repr__(self):
    return 'User ({} , {}!)'.format(self.username, self.email, self.image_file, self.safety)
```

```
class Roles(db.Model):
```

```
    __tablename__ = "Roles"
    id = db.Column(db.Integer, primary_key=True)
    role_name = db.Column(db.String(100), nullable=False)
    users = db.relationship('User', backref='role_name')
```

```
class Sessions(db.Model):
```

```
    __tablename__ = "Session"
    id = db.Column(db.Integer, primary_key=True)
    machine = db.Column(db.String(100), nullable=False)
    StartDate = db.Column(db.DateTime, nullable=False)
    EndDate = db.Column(db.DateTime, nullable=False)
    StartHour = db.Column(db.Integer, nullable=False)
    EndHour = db.Column(db.Integer, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('Users.id'), nullable=False)
    machine_id = db.Column(db.Integer, db.ForeignKey('machines.id'))
```

```
def __repr__(self):
    return 'Session ({} , {}!)'.format(self.machine, self.StartDate, self.EndDate, self.StartHour,
self.EndHour)
```

```

class LabFacilities(db.Model):
    __tablename__ = 'machines'
    id = db.Column(db.Integer, primary_key=True)
    Name = db.Column(db.String, db.ForeignKey('documents.machine'))

    def __repr__(self):
        return 'Machines {}'.format(self.Name)

class FileContent(db.Model):
    __tablename__ = 'documents'
    id = db.Column(db.Integer, primary_key=True)
    machine = db.Column(db.String(100))
    namef = db.Column(db.String(200))
    path = db.Column(db.String)
    data = db.Column(db.LargeBinary)
    category = db.Column(db.String(300))

    def __repr__(self):
        return 'FileContent ({},{})'.format(self.name, self.namef, self.category)

```

5 – Forms

```

from flask_wtf import FlaskForm

from wtforms import SubmitField, StringField, PasswordField, BooleanField, DateField,
SelectField

from wtforms.validators import DataRequired, ValidationError, Length, Email, EqualTo

from flask_wtf.file import FileField, FileAllowed

from app import User

class RegistrationForm(FlaskForm):

```

```

name = StringField('Name', validators=[DataRequired(Length), Length(max=50)])

family_name = StringField('Family Name', validators=[DataRequired(Length), Length(min=1,
max=100)])

username = StringField('Username', validators=[DataRequired(Length), Length(min=4,
max=20)])

email = StringField('Email', validators=[DataRequired(Length), Email(), ])

password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])

confirm_password = PasswordField('Confirm Password', validators=[DataRequired(),
EqualTo('password')])

submit = SubmitField('Sign Up')

```

```

def validate_username(self, username):

```

```

    user = User.query.filter_by(username=username.data).first()

```

```

    if user:

```

```

        raise ValidationError('That username is already taken. Please choose another one')

```

```

def validate_password(self, password):

```

```

    user = User.query.filter_by(password=password.data).first()

```

```

    if user:

```

```

        raise ValidationError('That password is already taken. Please choose another one')

```

```

def validate_email(self, password):

```

```

    user = User.query.filter_by(email=email.data).first()

```

```

    if user:

```

```

        raise ValidationError('That email is already registered. Please choose another one')

```

```

class LoginForm(FlaskForm):

```

```

    email = StringField('Email', validators=[DataRequired(Length), Length(min=4, max=50)])

```

```

    password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])

```

```
remember = BooleanField('Remember Me')
```

```
submit = SubmitField('Login')
```

```
class UpdateAccountForm(FlaskForm):
```

```
    username = StringField('Username', validators=[DataRequired(Length), Length(min=4, max=20)])
```

```
    email = StringField('Email', validators=[DataRequired(Length), Length(min=6)])
```

```
    picture = FileField('Image', validators=[FileAllowed(['jpg', 'png'])])
```

```
    submit = SubmitField('Update')
```

```
class UploadForm(FlaskForm):
```

```
    namedocument = StringField('Filename', validators=[DataRequired()])
```

```
    category = StringField('Category', validators=[DataRequired()])
```

```
    machine = StringField('Machine', validators=[DataRequired()])
```

```
    file = FileField('File', validators=[FileAllowed(['jpg', 'png', 'pdf', 'txt', 'xms'])])
```

```
    submit = SubmitField('Submit')
```

```
class ReservationsMForm(FlaskForm):
```

```
    machine = StringField('Machine', validators=[DataRequired()])
```

```
    startDate = DateField('First Day', validators=[DataRequired()])
```

```
    endDate = DateField('End Day', validators=[DataRequired()])
```

```
    startHour = SelectField('Choose starting hour', coerce=int, choices=([i for i in range(8, 20)]))
```

```
    endHour = SelectField('Choose ending hour', coerce=int, choices=([i for i in range(8, 20)]))
```

```
    submit = SubmitField('Submit')
```

6 – Routes

```
# Routes
```

```
@login_manager.user_loader
```

```
def load_user(user_id):
```

```
return User.query.get(int(user_id))
```

```
@app.before_first_request
```

```
def create_db():
```

```
    db.drop_all()
```

```
    db.create_all()
```

```
    password_ad = bcrypt.generate_password_hash('279856')
```

```
    user_admin = User(name='Michele', family_name='Magni', username='admin',  
                      email='ad@admin.com', password=password_ad)
```

```
    equipment1 = LabFacilities(Name='Nanoindenter')
```

```
    equipment2 = LabFacilities(Name='Microscope')
```

```
    db.session.add(equipment1)
```

```
    db.session.add(equipment2)
```

```
    db.session.add(user_admin)
```

```
    db.session.commit()
```

```
    user_query = User.query.filter_by(username="admin").first()
```

```
    print(user_query.name)
```

```
@app.route('/')
```

```
def home():
```

```
    return render_template('Front.html')
```

```
@app.route('/Contacts')
```

```
def contacts():
```

```
    return render_template('Contacts.html')
```

```
@app.route('/main')
```

```
def main():
```

```
    return render_template('HomePage.html')
```

```

@app.route('/facilities')
def facilities():
    return render_template('Machines.html')

@app.route('/Nanoindenter')
def nano ():
    return render_template('Nanoindenter.html')

@app.route('/Microscope')
def micro ():
    return render_template('Microscope.html')

# email for User
def send_mail_us(to, subject, template, **kwargs):
    msg = Message(subject, recipients=[to], sender=app.config['MAIL_USERNAME'])
    msg.html = render_template(template + '.html', **kwargs)
    mail.send(msg)

# email for Admin
def send_mail_ad(to, subject, template, **kwargs):
    msg = Message(subject, recipients=[to], sender=app.config['MAIL_USERNAME'])
    msg.html = render_template(template + '.html', **kwargs)
    mail.send(msg)

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data)

```

```

    user = User(name=form.name.data, family_name=form.family_name.data,
username=form.username.data,

        email=form.email.data, password=hashed_password)

    db.session.add(user)
    db.session.commit()

    send_mail_us(form.email.data, 'Your Registration has been successful', 'Mail to user',
name=form.name.data,

        username=form.username.data, password=form.password.data)

    send_mail_ad('michele.magni95@gmail.com', 'New User Registered', 'Mail to admin',
name=form.name.data,

        username=form.username.data )

    flash("Your Account has been created", 'success')

    return redirect(url_for('login'))

return render_template('REGISTER.html', form=form)

```

```
@app.route('/login', methods=['GET', 'POST'])
```

```
def login():
```

```
    if current_user.is_authenticated:
```

```
        return redirect(url_for('main'))
```

```
    form = LoginForm()
```

```
    if form.validate_on_submit():
```

```
        user = User.query.filter_by(email=form.email.data).first()
```

```
        if user and bcrypt.check_password_hash(user.password, form.password.data):
```

```
            login_user(user)
```

```
            return redirect(url_for('main'))
```

```
        else:
```

```
            flash('Login Failed! Please check your Username and Password', 'danger')
```

```
    return render_template('LOGIN.html', form=form)
```

```
def save_picture(form_picture):
```

```

random_hex = secrets.token_hex(8)
_, f_ext = os.path.splitext(form_picture.filename)
picture_fn = random_hex + f_ext
picture_path = os.path.join(app.root_path, 'static/profilePic', picture_fn)

output_size = (80, 80)
i = Image.open(form_picture)
i.thumbnail(output_size)
i.save(picture_path)

return picture_fn

@app.route('/account', methods=['GET', 'POST'])
@login_required
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        picture_file = save_picture(form.picture.data)
        current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash('Your account has been updated', 'success')
        return redirect(url_for('account'))
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
    image_file = url_for('static', filename='profilePic/' + current_user.image_file)
    return render_template('Account.html', title='Account', image_file=image_file, form=form)

```

```

def allowed_file(filename):
    return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/material')
def material():
    new = FileContent.query.all()
    return render_template('AllDocumentation.html', new=new)

@app.route('/upload', methods=['GET', 'POST'])
def upload():
    form = UploadForm()
    if form.validate_on_submit():
        newFile = FileContent(category=form.category.data, machine=form.machine.data)
        db.session.add(newFile)
        db.session.commit()
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('No file part', 'danger')
            return redirect(request.url)
        file = request.files.get('file')
        if file and allowed_file(file.filename):
            newFile = FileContent(category=form.category.data, machine=form.machine.data,
            namef=file.filename,
                data=file.read())
            db.session.add(newFile)
            db.session.commit()
            flash('Upload successful', 'success')
            return redirect(url_for('material'))

```

```

return render_template('materials.html', form=form)

@app.route('/material/<int:doc_id>')
def document(doc_id):
    filew = FileContent.query.get_or_404(doc_id)
    return render_template('Document.html', filew=filew, doc_id=doc_id)

@app.route('/Nanoindenter1')
def nanod():
    fnano = FileContent.query.filter_by(machine='Nanoindenter')
    return render_template('NanoDoc.html', fnano=fnano)

@app.route('/Microscope1')
def microd():
    fmicro = FileContent.query.filter_by(machine='Microscope').all()
    return render_template('MicroDoc.html', fmicro=fmicro)

@app.route('/download/<int:id>', methods=['GET'])
def download(id):
    down = FileContent().query.filter_by(id=id).first()
    return send_file(BytesIO(down.data), as_attachment=True,
attachment_filename=down.namef)

@app.route('/downloadSafety')
def downloadsaf():
    path = 'Safety Instructions.pdf'
    return send_file(path, as_attachment=True)

@app.route('/Reservations')

```

```

def book():
    if current_user.is_authenticated:
        sessions = Sessions.query.all()
        return render_template('AllSessions.html', sessions=sessions)
    else:
        flash('You must be logged it', 'danger')
        return redirect('/login')

@app.route('/Reservations/New', methods=['GET', 'POST'])
@login_required
def new_Session():
    form = ReservationsMForm()
    if form.validate_on_submit():
        collisions = Sessions.query.filter_by(
            StartDate=datetime.combine(form.StartDate.data, datetime.min.time())).filter_by(
            machine=form.machine.data).all()
        print(len(collisions))
        for collision in collisions:
            if form.StartHour.data <= collision.EndHour and (form.StartHour.data +
                (form.EndHour.data - form.StartHour.data)) > collision.StartHour:
                flash('Session already booked', 'danger')
                return redirect(url_for('book'))

        booked = Sessions(machine=form.machine.data, StartDate=form.StartDate.data,
            EndDate=form.EndDate.data,
                StartHour=form.StartHour.data, EndHour=form.EndHour.data,
            author=current_user)
        db.session.add(booked)
        db.session.commit()

```

```

flash('Your Session has been booked!', 'success')
return redirect(url_for('book'))

return render_template('New Session.html', form=form, legend='New Book')

@app.route('/Reservation/<session_id>')
def session(session_id):
    session = Sessions.query.get_or_404(session_id)
    return render_template('Session.html', session=session)

@app.route('/Reservations/<int:session_id>/update', methods=['GET', 'POST'])
@login_required
def update_session(session_id):
    session = Sessions.query.get_or_404(session_id)
    if session.author == current_user or current_user.email == "ad@admin.com":
        form = ReservationsMForm()
        if form.validate_on_submit():
            collisions = Sessions.query.filter_by(
                StartDate=datetime.combine(form.StartDate.data, datetime.min.time())).filter_by(
                machine=form.machine.data).all()
            print(len(collisions))
            for collision in collisions:
                if form.StartHour.data <= collision.EndHour and (form.StartHour.data +
                    (form.EndHour.data - form.StartHour.data)) >
collision.StartHour:
                    flash('Session already booked', 'danger')
                    return redirect(url_for('book'))
            session.machine = form.machine.data
            session.StartDate = form.StartDate.data

```

```

    session.EndDate = form.EndDate.data
    session.StartHour = form.StartHour.data
    session.EndHour = form.EndHour.data
    db.session.commit()
    flash('Reservation changed', 'success')
    return redirect(url_for('session', session_id=session_id))
elif request.method == 'GET':
    form.machine.data = session.machine
    session.StartDate = form.StartDate.data
    session.EndDate = form.EndDate.data
    session.StartHour = form.StartHour.data
    session.EndHour = form.EndHour.data
    return render_template('New Session.html', form=form, legend='Update Book')
else:
    abort(403)

@app.route('/Reservations/<int:session_id>/delete', methods=['GET'])
@login_required
def delete_sessions(session_id):
    session = Sessions.query.get_or_404(session_id)
    if session.author == current_user or current_user.email == "ad@admin.com":
        db.session.delete(session)
        db.session.commit()
        flash('Your Booking has been deleted!', 'success')
        return redirect(url_for('book'))
    else:
        abort(403)

@app.route('/material/<int:delete_id>/delete', methods=['GET'])

```

```
@login_required
```

```
def delete_material(delete_id):  
    deletef = FileContent.query.get_or_404(delete_id)  
    if current_user.email == "ad@admin.com":  
        db.session.delete(deletef)  
        db.session.commit()  
        flash('Document Delete!', 'success')  
        return redirect(url_for('material'))  
    else:  
        abort(403)
```

```
@app.route('/user/<string:username>')
```

```
def user_sessions(username):  
    user = User.query.filter_by(username=username).first_or_404()  
    sessions = Sessions.query.filter_by(author=user)  
    return render_template('User_Sessions.html', sessions=sessions, user=user)
```

```
@app.route('/machine/<string:machine>')
```

```
def machine_users(machine):  
    sessions = Sessions.query.filter_by(machine=machine).all()  
    return render_template('Machine_User.html', sessions=sessions, machine=machine)
```

```
@app.route('/logout')
```

```
def logout():  
    logout_user()  
    return redirect('/')
```

```
if __name__ == '__main__':  
    app.run(debug=True)
```

Acknowledgements

Sono stati 6 anni impegnativi, stressanti ma ricchi di soddisfazioni, e ora passiamo alle cattive notizie... sono finiti! E questa tesi è stata il coronamento. Per svolgere questo importante lavoro è stata necessaria costanza e dedizione, ma non sono mai stato solo grazie a coloro che mi hanno sempre supportato.

Voglio ringraziare sentitamente il professor Gianfranco Genta ed il professor Luca Mastrogiacomo per le indicazioni e l'aiuto fornitomi durante lo svolgimento della tesi, così come il dr. Giacomo Maculotti per i continui ed utili feedback.

Sono orgoglioso di aver potuto contribuire, attraverso lo sviluppo di questa tesi, al miglioramento dell'Università che è stata una parte importante della mia vita in questi anni.

Un grazie speciale va alla mia famiglia per il continuo supporto ed incoraggiamento e per aver sempre creduto in me anche nei momenti più difficili che ho dovuto affrontare.

Ho avuto la fortuna di condividere il mio percorso universitario con persone speciali, soprattutto durante gli ultimi due anni: voglio ringraziare i miei compagni di gruppo e studio, Chiara, Arianna, Mirna, Mohan, Faezeh e Natallia con i quali ho condiviso innumerevoli ore di studio in videocall, e caffè, nonché William, Giovanna, Henrique e Catalin per le serate trascorse insieme.

Un grazie a Luca, Emanuele e Matteo, amici fin dalle superiori, per le serate bovine trascorse ad assaggiare hamburgers: a stomaco pieno si ragiona meglio!

Un grazie ad Andrea e Filippo che conosco ormai fin dalle elementari.

Infine, ringrazio la mia ragazza, Giulia, anche lei studentessa al Politecnico, per il suo incrollabile e contagioso entusiasmo, e per avermi spronato a dare il meglio!

Bibliography

Alexander Aronowitz, “Flask Framework Cookbook: Building Web Applications with Flask “ 2nd Edition, 2021.

Allen Downey, “Think Python”, Green Tea Press, Needham, Massachusetts,2017.

Jon Duckett, “HTML e CSS. Progettare e costruire siti web”, APOGEO,2021.

Miguel Grinberg, “ Flask Web Development”, O’Reilly Media Inc, 2014.

Marakas, O’Brian “Introduction to Information Systems”, 16th Ed, 2021.

Sitography

<https://www.codecademy.com/article/back-end-architecture>

<https://flask-sqlalchemy.palletsprojects.com/en/2.x/>

<https://flask.palletsprojects.com/en/2.0.x/#user-s-guide>

<https://getbootstrap.com/>

https://www.ictshore.com/software-design/web-application-structure/attachment/sfw0001-01-web_application_structure/

<https://www.idef.com/>

<https://www.jetbrains.com/help/pycharm/quick-start-guide.html>

<https://www.uml-diagrams.org/>