



**BRNO UNIVERSITY OF TECHNOLOGY**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

**FACULTY OF INFORMATION TECHNOLOGY**

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

**DEPARTMENT OF INFORMATION SYSTEMS**

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

**DATABASE SYSTEM FOR LABORATORY EQUIPMENT  
TRACKING**

DATABÁZOVÝ SYSTÉM PRE SLEDOVANIE LABORATÓRNYCH ZARIADENÍ

**BACHELOR'S THESIS**

BAKALÁŘSKÁ PRÁCE

**AUTHOR**

AUTOR PRÁCE

**DENIS HELIENEK**

**SUPERVISOR**

VEDOUCÍ PRÁCE

**Ing. VLADIMÍR BARTÍK, Ph.D.**

BRNO 2019

## Bachelor's Thesis Specification



21463

Student: **Helienek Denis**  
Programme: Information Technology  
Title: **Database System for Laboratory Equipment Tracking**  
Category: Databases

Assignment:

1. Compare possible options and choose a database, which can run on a virtual server.
2. Analyze requirements for an information system for laboratory equipment tracking, consisting of a mobile application with a QR code reader and a frontend/backend application on the virtual server for the administration of the equipment, which will also create PDF files with calibration lists and generate QR codes.
3. Design the structure of the database and applications mentioned in the 2nd item.
4. Implement the applications and test them on a suitable dataset.
5. Summarize achieved results and suggest possible extensions of the project.

Recommended literature:

- Ujbányai, M.: Programujeme pro Android. Grada Publishing, 2012, ISBN 978-80-247-3995-3.
- Naramore, E., Gerner, J. et al: PHP 6, MySQL, Apache: Vytváříme webové aplikace. Computer Press, 2009. ISBN: 978-8-0251-2767-4.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Requirements for the first semester:

- Items 1 to 3.

Detailed formal requirements can be found at <http://www.fit.vutbr.cz/info/szz/>

Supervisor: **Bartík Vladimír, Ing., Ph.D.**

Head of Department: Kolář Dušan, doc. Dr. Ing.

Beginning of work: November 1, 2018

Submission deadline: May 15, 2019

Approval date: October 26, 2018

## Abstract

The goal of this thesis is service delivery of a database system for laboratory equipment tracking for *Porsche Engineering Services, s.r.o.* The mentioned service has been delivered by the following service management framework guidelines with the focus on software engineering aspects. The final product is an information system implemented in Django, associated with the mobile application implemented in Swift programming language and uses the MariaDB database. This system establishes better inventory management of the laboratory and offers many possibilities for future extension. The entire service contributes to time and costs savings for the company.

## Abstrakt

Cieľom tejto práce je doručenie služby v podobe databázového systému pre sledovanie laboratórnych zariadení pre *Porsche Engineering Services, s.r.o.* Uvedená služba bola doručená nasledovaním usmernení frameworku manažmentu služieb so zameraním na aspekty softvérového inžinierstva. Finálnym produktom je informačný systém implementovaný v Django s mobilnou aplikáciou implementovanou v Swift a využíva databázu MariaDB. Tento systém vytvára lepšiu správu zariadení laboratória a ponúka mnoho možností pre rozšírenie. Celá služba prispieva k úsporám času a nákladov spoločnosti.

## Keywords

service management, software engineering, web application, database system, ITIL, Django

## Klíčová slova

manažment služieb, softvérové inžinierstvo, webová aplikácia, databázový systém, ITIL, Django

## Reference

HELIENEK, Denis. *Database System for Laboratory Equipment Tracking*. Brno, 2019. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vladimír Bartík, Ph.D.

## Rozšířený abstrakt

Cielom tejto práce je doručenie IT služby, ktorej hlavným produktom je informačný systém s databázou pre sledovanie laboratórnych zariadení. Úlohou tejto služby je zlepšenie správy samotných zariadení ako aj následne sprostredkovanie zozbieraných dát za účelom ich ďalšieho využitia. Potreba pre túto službu vznikla na základe požiadavkou vo firme *Porsche Engineering Services, s.r.o.* v Prahe. Táto firma je známy automobilový výrobca, ktorý investuje do rôznych odvetví vývoja a výskumu, a preto by malo takéto riešenie nepriamo viesť ku lepším ekonomickým výsledkom a celkovo k väčšej konkurencieschopnosti na trhu. Mimo požadované vlastnosti ponúka systém taktiež nové možnosti jeho využitia či ďalšieho rozšírenia. V prípade úspešného doručenia a zavedenia systému do prevádzky je teda možné tento systém, respektíve celú doručovanú IT službu, rozšíriť na globálnu úroveň spoločnosti. Toto rozšírenie by prinieslo ešte lepšie výsledky, keďže by bol systém využitý vo viacerých pobočkách firmy po celom svete.

Riešenie tejto práce začína úvodom do problematiky manažmentu IT služieb, ktorej cieľ je priblížiť populárnu sadu doporučení pri dodávaní IT služieb s názvom **ITIL**. Táto sada doporučení – ITIL – je rozdelená do viacerých logických celkov a snaží sa zahrnúť všetky dôležité aspekty manažmentu IT služieb od rôznych princípov až po samotný priebeh vytvárania softvérového produktu. Keďže sa jedná len o sadu doporučení a rozličné organizácie po svete poskytujú rozličné IT služby, je v tejto kapitole vyzdvihnutých iba zopár doporučení, najmä tie ktoré úzko súvisia s doručovaním služby pre vyššie spomínanú firmu. Nasledováním štruktúry ITIL-u sú na začiatku identifikované tri pre systém dôležité role užívateľov, a to inžinieri pracujúci v laboratóriách, lídri laboratórií starajúci sa o zariadenia a manažéri, ktorých hlavnou úlohou je manažovanie spomenutých zamestnancov. Každý z týchto užívateľov má svoje požiadavky na systém, ktoré musia byť splnené. Ďalej sú identifikované pravidlá firmy, ktoré musí doručenie služby spĺňať – bezpečnosť, branding a neporušenie softvérových licencií. Vývoj softvéru je podľa ITIL-u rozdelený do týchto štádií – analýza požiadavkou, architektúra a dizajn produktu, implementácia, testovanie a zavedenie na firemný server. V neposlednej rade je prevedená biznisová a finančná analýza, ktoré majú slúžiť ako základné potreby pre budúce rozšírenie služby.

Po problematike manažmentu IT služieb je práca zameraná na konkrétne štádiá softvérového inžinierstva. Prvým je analýza požiadavkou, ktoré sú rozdelené do štyroch celkov. Biznisové (obchodné) požiadavky boli získané od manažéra projektu a určujú aké má služba príležitosti, ciele, závislosti, riziká a akú má víziu. Znalosti nadobudnuté analýzou týchto požiadavkou nám dávajú lepšiu predstavu o produkte z perspektívy firmy a jej obchodných cieľov. To je základný predpoklad pre úspešné zavedenie produktu. Ďalej sú analyzované požiadavky získané od budúcich užívateľov produktu. Vďaka ich podnetom môže byť produkt lepšie uspokojovať ich potreby, a tým zabezpečiť bezproblémovú použiteľnosť informačného systému. Užívateľské požiadavky sú znázornené pomocou diagramov použitia. Na základe všetkých získaných podnetov sú vypracované funkcionálne a nefunkcionálne požiadavky na systém. Tie popisujú čo systém musí robiť a taktiež aj ako dobre to musí robiť.

Výstup analýzy je využitý pri vytváraní architektonických pohľadov na systém a prijímaní dizajnových rozhodnutí. Keďže architektúra je náročná disciplína a jej kvalitné spracovanie je nevyhnutné pre ďalší progres vývoja, je najprv potrebné uviesť jej problematiku, vďaka ktorej lepšie dokážeme porozumieť všetkým zúčastneným stranám, ktoré potrebujú architektonické pohľady. Tým pádom vieme presnejšie adresovať vytvorenú architektúru. Z týchto dôvodov je načrtnutý pohľad na systém z najvyššej úrovne a komunikácia medzi jednotlivými modulmi. Následne je na základe predošlej analýzy vypracovaný zjednodušený

dizajn databáze kde sa nachádzajú potrebné entity a vzťahy medzi nimi. Posledným krokom je vybratie technológií pre celý systém a zdôvodnenie výberu databázy.

S vypracovanou architektúrou a prijatými dizajnovými rozhodnutiami je ďalším krokom implementácia samotného systému. Tá je logicky rozdelená na grafické rozhranie webovej stránky, jej užívateľovi skrytú funkcionálnu a mobilnú aplikáciu spojenú so systémom. V časti pojednávajúcej o grafickom rozhraní sú poskytnuté ďalšie pohľady na štruktúru, dekompozíciu a špecifické ústrania webovej stránky. Podrobnejšie je rozpísané ako boli využité zvolené technológie a ako sa prikladal dôraz na užívateľský zážitok. Za týmto celkom nasleduje bližší popis a ďalšie pohľady na užívateľovi skrytú funkcionálnu webovú stránku. V nej je taktiež rozpísané ako boli využité zvolené technológie a špecifické riešenia ako autorizácia či nahrávanie súborov okrem iného. Popis mobilnej aplikácie sa zaoberá najmä jej spracovaním videa a komunikáciou s webovou stránkou. Na záver tejto kapitoly sú navrhnuté ďalšie možné funkcionality k implementácii, ktoré nebolo možné v čase vývoja implementovať najmä kvôli interným firemným procesom či časovej náročnosti.

Posledné štádiá softvérového inžinierstva – testovanie a zavedenie systému – zabezpečujú kvalitu služby a jej použitie. Pri testovaní bol dôraz najmä na užívateľské testovanie, ktoré je rozdelené do dvoch sekcií. V prvej sa jednalo o testovanie produktu na pravidelných stretnutiach s užívateľmi kde mohli poskytnúť svoju spätnú väzbu ešte počas skorého vývoja. Pri druhej vlne testovaní bol systém zavedený na firemný server, databáza naplnená testovacími dátami, a tak mali používatelia možnosť experimentovať so systémom zo svojich počítačov v dlhšom časovom období. Vzniknuté problémy boli nahlásené a postupne vyladené. Taktiež boli poskytnuté návrhy na akceptačné testovanie a úskalia zavádzania do ostrej prevádzky systému.

Na záver je možné zhrnúť, že požadovaný systém bol riadne navrhnutý, implementovaný a otestovaný pre jeho ďalšie použitie vo firme. Rovnako boli poskytnuté možné rozšírenia tohto produktu pre zvýšenie jeho kvality, a tým aj efektívnosti, ktorú pre firmu prináša. Pre jeho rozšírenie na globálnu úroveň spoločnosti – teda mimo Pražskú pobočku – je doporučené hlbšie zameranie sa na poskytovanie IT služieb a to najmä z dôvodov nastavenia stratégie, prevádzky a kontinuálneho zlepšovania služby tak ako bolo v tejto práci dôkladné zameranie sa na aspekty softvérového inžinierstva služby.

# Database System for Laboratory Equipment Tracking

## Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Vladimír Bartík, Ph.D. The supplementary information was provided by Ing. Tomáš Haubert, Ph.D., team leader at *Porsche Engineering Services, s.r.o.* All the relevant information sources, which were used during the preparation of this thesis, are properly cited and included in the list of references.

.....  
Denis Helienek  
May 15, 2019

## Acknowledgements

Special thanks to Ing. Tomáš Haubert, Ph.D. for providing valuable support at *Porsche Engineering Services, s.r.o.* and Ing. Vladimír Bartík, Ph.D. for supervision at *Brno University of Technology*.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Service management</b>	<b>4</b>
2.1	Guiding principles . . . . .	5
2.2	Governance . . . . .	6
2.3	Service value chain . . . . .	6
2.4	Continual improvement . . . . .	7
2.5	Practices . . . . .	7
<b>3</b>	<b>Analysis of requirements</b>	<b>9</b>
3.1	Business requirements . . . . .	10
3.2	User requirements . . . . .	12
3.3	Functional requirements . . . . .	15
3.4	Non-functional requirements . . . . .	16
3.5	The output of the analysis . . . . .	17
<b>4</b>	<b>Architecture and design decisions</b>	<b>18</b>
4.1	Architecture insight . . . . .	18
4.2	Top layer architecture . . . . .	22
4.3	Database design . . . . .	23
4.4	Selected technologies and database comparison . . . . .	25
4.5	Architecture process checkpoint . . . . .	27
<b>5</b>	<b>Implementation</b>	<b>28</b>
5.1	Website user interface . . . . .	28
5.2	Website business logic . . . . .	34
5.3	Mobile application . . . . .	42
5.4	Extendability of implementation . . . . .	43
<b>6</b>	<b>Testing and deployment</b>	<b>45</b>
6.1	Development testing . . . . .	45
6.2	User testing . . . . .	46
6.3	Deployment insight . . . . .	47
6.4	Extendability of testing and deployment . . . . .	48
<b>7</b>	<b>Conclusion</b>	<b>49</b>
	<b>Bibliography</b>	<b>50</b>

<b>A</b>	<b>Content of the attached CD</b>	<b>51</b>
<b>B</b>	<b>Selected screenshots from the web application</b>	<b>52</b>
B.1	Landing page . . . . .	52
B.2	Equipment management page . . . . .	53
B.3	Equipment detail page . . . . .	54
B.4	Management page . . . . .	55



# Chapter 1

## Introduction

The purpose of this thesis was set by a description of the requirements of the company *Porsche Engineering Services, s.r.o* located in Prague. This company is an automobile manufacturer heavily investing in its research and development in order to stay competitive against other manufacturers. This approach means any cost-cutting software or innovative solutions are highly appreciated. That is the background of their demand for the database system to track equipment in laboratories to improve internal organisation and workflow of employees within the laboratories.

However, to deliver the mentioned software, it is needed to capsule it into service. Omitting this decision would lead to the lower overall quality of the system. Due to this reason, this thesis focuses on service delivery in general apart from just the crucial software development stages such as analysis, architecture views, design decisions, implementation, testing or deployment.

Besides the requirements, an opportunity occurred for various features which could also enhance the effectiveness and bring valuable benefits to the table. These ideas of extendability could help the system to reach a global status throughout the company. Gaining this status would lead to even better business results for the whole company in total and of course for the particular branch in Prague.

Here is the list of chapters and their content for better navigation:

- **Service management** - insight into the service management discipline and why it is relevant to deliver a successful project
- **Analysis of requirements** - description of different requirements for the product
- **Architecture and design decision** - insight into the architecture of the project followed by design decisions
- **Implementatation** - most important details of the implemented system
- **Testing and deployment** - the chapter describes what tests were performed and how was the system deployed
- **Conclusion**

## Chapter 2

# Service management

Service management is a discipline designed to meet the needs of customers by creating value for them. Organisations in the modern world benefit by IT-enabling those services that in turn expand their IT service management (ITSM) capabilities. To guarantee that ITSM hangs around in well-fashioned habits, produces and delivers successful services, many organisations adopt established ITSM guidance frameworks. This chapter gives insight into the currently most popular framework according to *Forbes Insights survey* [1] - ITIL. Furthermore, it is logically divided into sections structured as the key components of the newest ITIL v4 [2] called *Service value system* as shown in figure 2.1 and tries to explain how these components are related to this project.

We should remember that the concept of ITIL gives a rundown of ITSM and provides us with guidelines. Organisations and their services vary across the world, and therefore we selected only the reasonable subcomponents of this framework for our purposes that aim towards future extendability of the project.

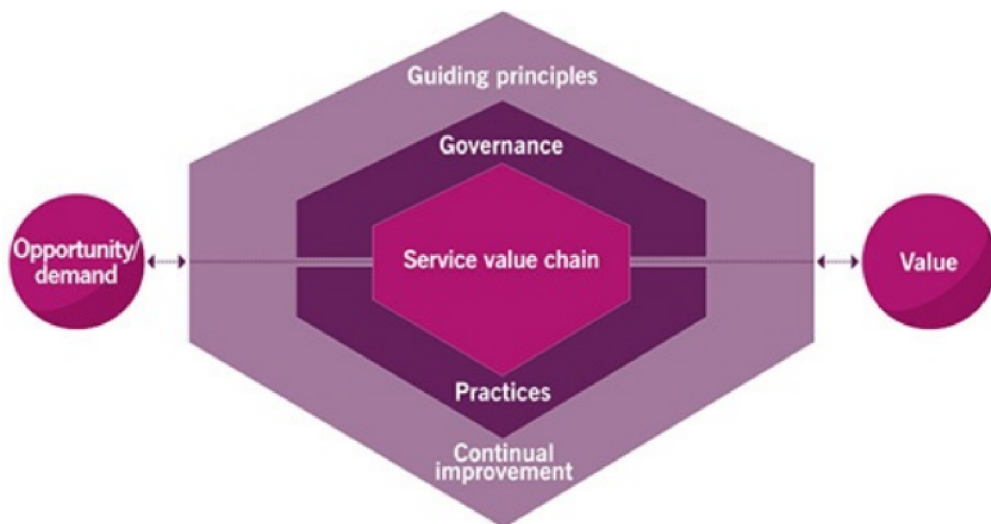


Figure 2.1: The service value system. Taken from [2].

## 2.1 Guiding principles

The ITIL guiding principles help an organisation in all circumstances to embody ideas of service management in general. These guidelines are also reflected in other philosophies such as Agile or DevOps. Therefore, these approaches are compatible and easily applicable in the ITIL v4 framework. ITIL v4 lists the following principles:

- Focus on value
- Start where you are
- Progress iteratively with feedback
- Collaborate and promote visibility
- Think and work holistically
- Keep it simple and practical
- Optimize and automate

Although all these principles are important to a certain degree, for our project, we have picked only a few of them for a closer look. This selection was mainly based on consultations with the company representatives.

### Focus on value

To create a valuable service for stakeholders, we have to know who is being served. The same principle also appears in the software architecture process [3], which tells us how to identify stakeholders for addressing architecture views in the form of diagrams, graphs and other visual representations. This architecture process is later described in chapter 4. Identified stakeholders involved in our service are:

- Engineers that work in laboratories
- Laboratory leaders in charge of keeping track of each lab equipment
- Managers that manage laboratory leaders

Once we identified the stakeholders, we need to understand their perspective of value by recognising what service should help them to do and how should the service help them achieve their goals. That's why we need to make an analysis of their requirements which is described in chapter 3 and the design it as described in chapter 4. By fulfilling these requirements, our service will bring an increased productivity, reduced costs and new opportunities.

### Progress iteratively with feedback

Due to the fact that no improvement occurs in a vacuum, gathering and applying feedback ensures the effectiveness of software engineering processes as well as service value delivery. Iterating over these activities returns better flexibility, faster response time to business needs and raises the overall quality of work. Thus every described phase in this thesis is in an iterative manner in collaboration with all the stakeholders.

## 2.2 Governance

One of the ITIL Foundation's [2] key messages to governance is that as every organisation is directed by a governing body, it is crucial to make sure that organisation's practices work in line with the direction given by the governing body. A governing body is accountable at the highest level for performance and compliance in the organisation. That means service delivery and software engineering processes in our case have to comply with policies set in *Porsche Engineering* located in Prague. Below is the list of items which are most relevant to our service:

- Branding
- Security
- Software License Compliance

It is mandatory to follow them and evaluate activities associated with them throughout this thesis for easier future extendability of the service inside the company.

## 2.3 Service value chain

*Service value chain* is the central element of the *Service value system* already presented in figure 2.1 and outlines the key activities to facilitate value realisation through the creation of products and services. Up to some level, it is somehow similar to the well-known waterfall model as described in [8] but in an iterative way. *Service value chain* is shown in figure 2.2. It consists of six activities - *Plan*, *Improve*, *Engage*, *Design and transition*, *Obtain/Build*, *Deliver and support*, *Products and services*, *Demand* and *Value*. As these guidelines are universal and different products need different streams, we should select and adapt some of them for our purposes.

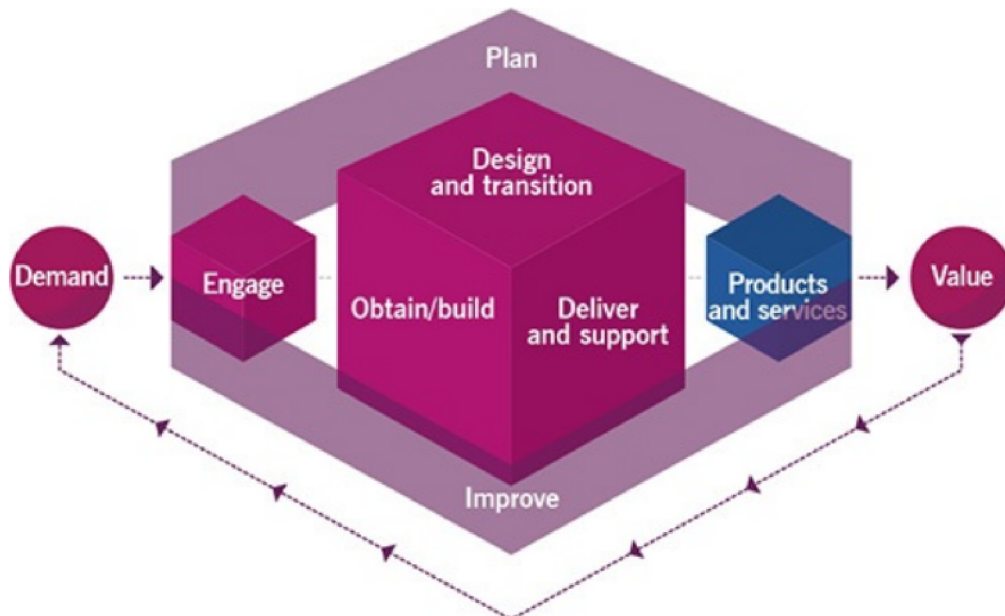


Figure 2.2: The service value chain. [2]

## Plan

The plan has to ensure a mutual understanding of the vision, and for that, we have to gather and **analyse** the current laboratory workflow of our listed stakeholders and their requirements.

## Improve

The improve value chain activity ensures continual improvement, so we are obliged to process product performance information and stakeholders' feedback gathered through **testing**.

## Design and transition

The purpose of this activity guarantees that the product meets the expectations by producing **architecture views** and **design decisions** and by presenting these views before the actual implementation.

## Obtain/Build

This activity includes the **implementation** of service components. In our project, this is a database system encapsulating a web application, a mobile application and a database connected to them.

## Deliver and support

The last activity we adopted is aimed towards the **deployment** of the system on the allocated company virtual server.

## 2.4 Continual improvement

The continual improvement model covers the entire service value system to maximise the effectiveness of services. ITIL provides a set of steps for this improvement model by defining a business vision, mission, a set of goals, objectives, measurable targets and improvement plans. However, as the formal usage of the continual improvement model would slow down the delivery process of the expected system, it is simplified to a high-level reminder of a sound thought process to ensure improvements are adequately managed. In consequence, some continual improvement aspects might appear in service delivery. As the framework itself suggests, critical judgement should always be applied when using this model [2].

## 2.5 Practices

Management practice is a set of organisational resources designed for performing work or accomplishing an objective [2]. ITIL categorises them in three groups - *general management practices*, *service management practices* and *technical management practices*. There are 34 practices in total making the management a core part of the framework. Not all of them are applicable to every project. Therefore, we have chosen only two of them through collaboration with the company representatives. First practice - *Business analysis* - is here to give us the first close view on the current *Inventory management* workflow in the laboratories of *Porsche Engineering* with its location in Prague. The second practice is

named *Service financial management* and brings us basic insight into the financial resources associated with the service. We consider that at least this basic overview is mandatory in a business environment for further extendability and success of the project.

## Business analysis

As we already mentioned, there are three stakeholders' groups - **Engineers, Laboratory leaders** and **Managers**. Furthermore, we have another two elemental components. **Laboratory** has its unique name with their respective laboratory leaders. In these laboratories, we can find a significant amount of various **Equipment** used by engineers such as a laser, multimeters or solder to name a few. A lot of these tools are rather sophisticated and come with a variety of specifications which are needed to be tracked for business purposes. They usually have their serial numbers, different kinds of documents belonging to them and calibration standards.

Apart from the regular sharing of equipment between local colleagues inside one building all the engineers from the whole company - not limited to Prague location of *Porsche Engineering* only - may borrow any device or tool located in any other office around the world while on business trips. This workflow in laboratories is prone to errors in many cases. Firstly, it is not uncommon for employees to spend time searching for a device which was borrowed and not returned. Secondly, keeping track of expiration periods of all devices is decentralised and usually kept around the office in several computers and documents. Lastly, the lack of adequate organisation at this layer does not produce any additional information for improved planning. In conclusion, we identified three main business opportunities that can be improved or reached by the system - **time savings, sophisticated inventory management** and **new extendability** possibilities based on gathered data.

## Service financial management

Financial management for our project takes into account three main areas to support the decision-making regarding the delivered system.

**Expenses** could be divided into *direct* and *indirect* expenses. *Direct expenses* count with salaries of allocated people in service, where in our case that would be primarily developers and maintainers of the system. Furthermore, they include software and hardware costs of the system such as servers or mobile devices. *Indirect expenses* take into consideration *operating expenses* associated with the system. These could be replacements of broken devices, software licenses and so on. For proper calculation of expenses, in general, it would be needed to perform deep research of the system from a financial point of view which is out of the scope of this thesis.

**Revenues** in our service are generated by the time savings of the engineers working in the laboratories by using the system. For example, time saved by searching for equipment through the system rather than using other slower methods. Same as the expenses, these calculations would have to be done by more in-depth financial analysis.

Last area - **Return on investments** - could be done in the end after previous revenues and expenses analysis. Its equation would look alike in formula 2.1.

$$return = investments / (revenues - expenses) \quad (2.1)$$

## Chapter 3

# Analysis of requirements

Once the service definition was set, there was an essential need to begin a more in-depth analysis of the requirements in order to get more meaningful input for the future design of the system. During the whole process of analysing it was essential to keep in mind all the needs of all stakeholders and at the same time having the system cost efficient and reasonable. Analysis of this thesis follows some of the basic principles from the book focused on software requirements [9]. Therefore, the whole chapter contains all relevant aspects from three distinct levels as shown on figure below 3.1 with additional nonfunctional requirements to identify what does the system have to provide.

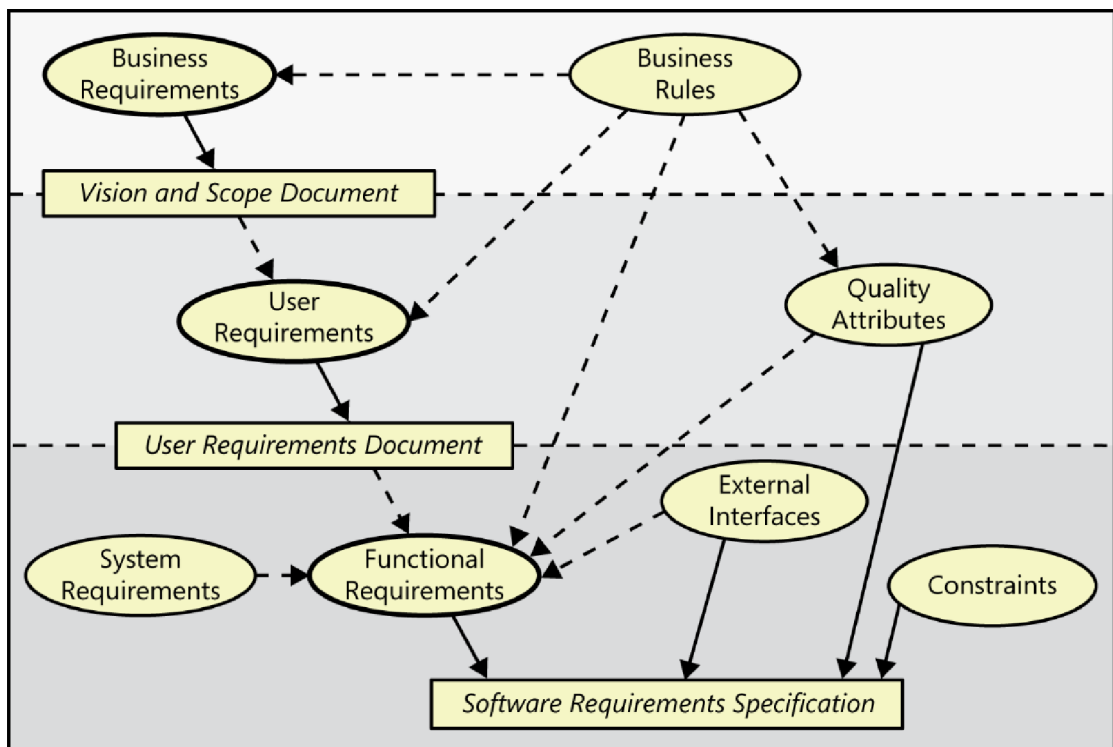


Figure 3.1: Relationships among requirements. [9]

### **3.1 Business requirements**

Business requirements were raised by the manager responsible besides of other duties for laboratory management. There was a lack of proper organisation in the equipment tracking, so it was necessitated to write down issues accompanied with the product vision. Main business issues are:

- decentralised document control
- loss of time spent on finding devices
- no additional data about workflow

Solving these issues should bring document organisation resulting in better control over calibration expirations, more convenient access to manuals of devices, saving the time of employees and further input for data processing. Furthermore, other business aspects that are divided into subsections below provide more points of view for better understanding of the service that is being delivered.

#### **Business opportunity**

This project solves internal laboratory management problems and therefore it is very specific. Hiring external companies for making such a tailored solution would be more costly and less effective because having an employee onboard to maintain this system ends up in faster response time to future requests. Additionally, the system does not handle critical data neither is critical for a business to run so having a solution from an external company to secure its running on level of agreement is not needed. Another solution would be some open source programs which are usually made for general purposes and do not include specific needs which would have to be implemented additionally. The system runs on internal servers and devices resulting in full control and accessibility for troubleshooting and innovation.

#### **Business objectives**

The main business nonfinancial measurable objectives:

- all data of devices inserted and migrated to the database
- all relevant documents stored on the server
- records of equipment usage
- accessibility to all employees

#### **Business risks**

The main business risks which could appear:

- users acceptance of scanning QR code every time before using each device
- cost of servers to run the system
- cost of devices used for scanning QR codes



## **Vision statement**

A laboratory equipment tracking system is a complex information system which serves mainly for two groups of stakeholders. Firstly, it is aimed at employees - engineers working in laboratories - to save their time while finding equipment around the laboratories and acting as a single point of easy access for reaching all documents related either to a particular device or laboratory. Secondly, it provides managers with usage overview of each equipment and predicts the calibration expiration that results in more accurate planning and scheduling. Unlike the current approach of working in laboratories, this system brings future opportunities for new features such as statistics, reservation modules or any data processing.

## **Business dependencies**

The main business dependencies are:

- server for web application and database
- device to record QR code in every laboratory
- both human and technical resources for developing and maintenance of the system

## 3.2 User requirements

User requirements were gathered by interviewing laboratory leaders who also act as regular employees working in laboratories. This method was repeated iteratively several times until it sharpened their requirements into an appropriate design. Users were also able to give input about the design by interacting with the prototype of the application. Their needs were validated with their manager, and use case diagrams express the main functionalities of the system. Moreover, the diagrams below are divided by roles of users in the system.

### Engineer

An user acting as a engineer working in a laboratory interacts mainly with the tablet through the application to scan a QR code placed on equipment in a laboratory and track all requested details. Furthermore, by using the web interface, he can check whether the equipment is borrowed and if so he can check who borrowed this equipment. In case equipment has not been taken away by any coworkers – and he is still struggling to find it – he may find where it should be placed. The system also provides him with insight into documents and equipment details. Engineers cannot update any data through the website, only indirectly by scanning QR codes. This is illustrated on figure 3.2.

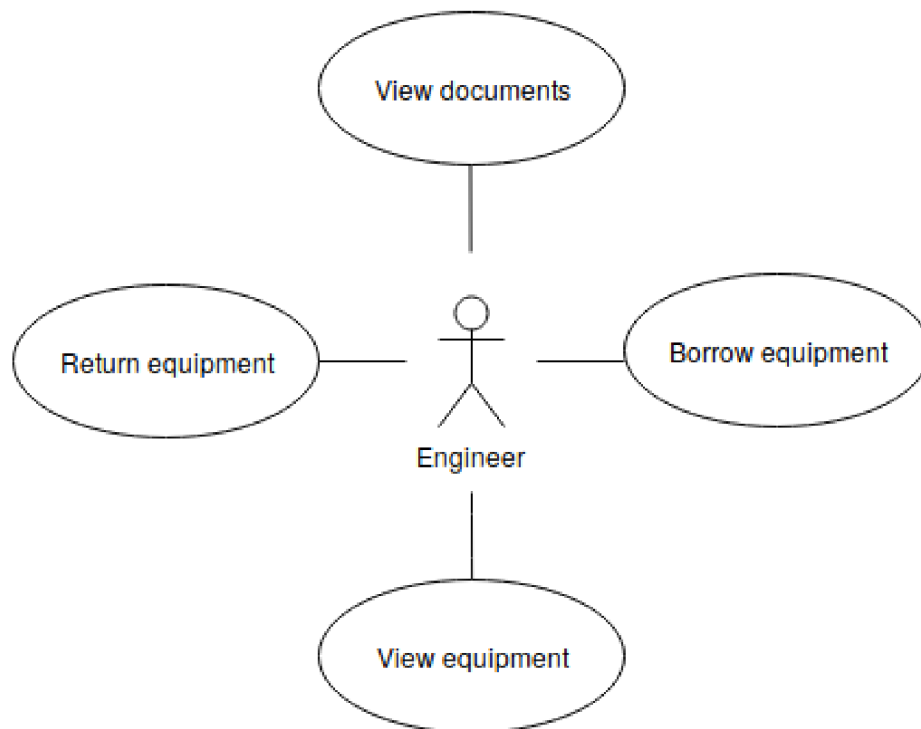


Figure 3.2: Simplified use case diagram of an engineer.

## Laboratory leader

On top of how an engineer is able to interact with the system, a laboratory leader can add either new equipment into the database through the web interface of the system or update existing ones in order to keep all equipment details current. This is illustrated on figure 3.3.

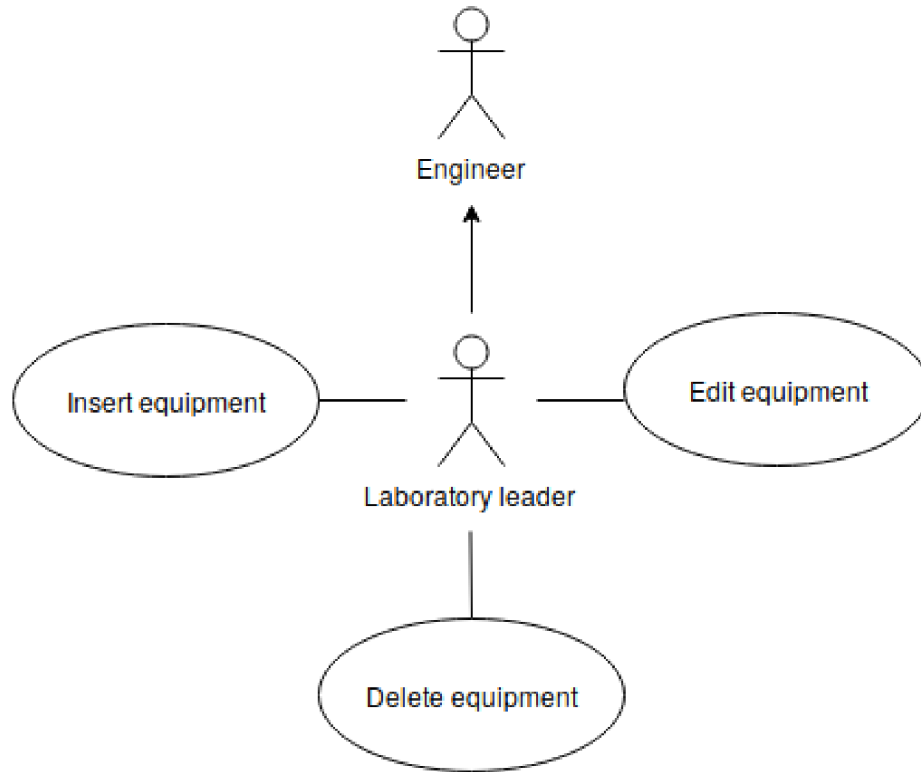


Figure 3.3: Simplified use case diagram of a leader.

## Manager

A manager has even more capabilities, including all which the laboratory leaders and engineers have. In addition, he can promote engineers to leader roles inside the system and vice versa – demoting their rights. He is also able to create new laboratories and upload new documents to the documents section (not to be mistaken with documents of equipment) such as laboratory rules. This is illustrated on figure 3.4.

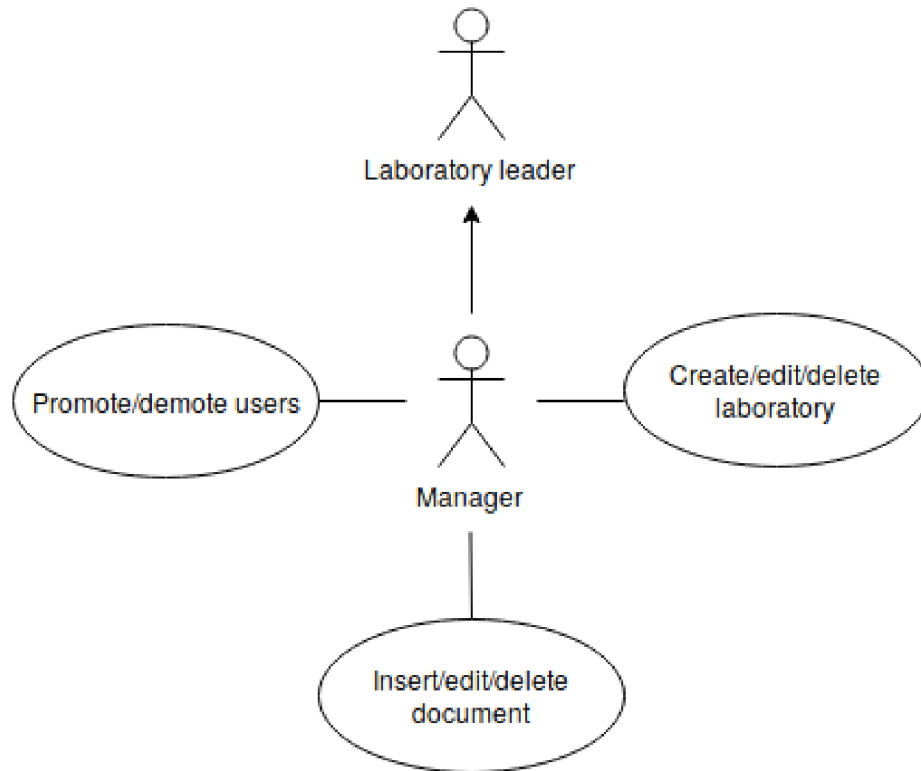


Figure 3.4: Simplified use case diagram of a manager.

### 3.3 Functional requirements

These requirements define *What the system does*. The most important functional requirements are organised by areas into sections below:

#### General

- The system shall be developed with agreed technologies described in the next chapter [4](#).
- The system must satisfy business and user requirements.

#### Web application

- The website application must have an authorisation system.
- The website application must provide equipment details and history for all users.
- The website application must provide a document section for all users.
- The website application must provide a list of all laboratories for all users.
- The website application must have a management module for managers and laboratory leaders with privileges according to use cases as shown on figures [3.3](#) and [3.4](#).
- The website application must provide all operations such as inserting, editing and deleting data to meet all actions described in use case diagrams in management module as shown on figures [3.3](#) and [3.4](#).
- The website application must generate QR codes for both users and equipment.
- The website application does not have to create PDF files with calibration lists as they are already provided by suppliers and can be easily uploaded to the document section of equipment that the web application must provide.
- The website application must provide an application interface for the mobile application to ensure it can record all required operations to the database.
- The website application shall have a profile section to provide all users their user details and QR codes.

#### Mobile application

- The mobile application must provide a user a log in ability by scanning his QR code.
- The mobile application must provide a user the ability to borrow or return equipment by scanning the QR code of equipment.
- The mobile application must run on iOS devices.

## Database

- The database must contain all the required data about the equipment<sup>1</sup>.
- The database must support the authorisation of the web application.
- The database must contain all the required data for file storage.
- The database shall follow its **Entity Relationship Diagram** as described in the next chapter.
- The database must keep all historical data associated with operations performed on equipment.
- The database must be relational.

## Infrastructure

- The system shall run on a Linux based operating system.
- The system shall be accessible within the internal network.
- The system must be able to store all the necessary documents.

## 3.4 Non-functional requirements

These requirements define *How well the system does things below*. The most important non-functional requirements are organised by areas into sections below.

### Availability

- The web application shall be available on all common browsers such as Chrome, Firefox, Explorer and Safari (both on computers and mobile devices).
- The mobile application must be available on all mobile devices in laboratories dedicated to this project.
- Every laboratory shall be equipped with mobile devices to ensure access to the system.
- The mobile application shall be available to download from the company application store.
- The system shall run uninterrupted except during planned downtimes for maintenance operations.

### Integrity

- The database shall be designed in a manner in which it avoids data inaccuracy.
- The database shall have its data backed up.

---

<sup>1</sup>Mentioned data can be seen in the ERD in the next chapter.

## Security

- The system shall have securely protected input fields.
- The system shall not have to keep any unnecessary data.
- The system shall follow company security policies.
- The system's most crucial components as server and database must be protected with strong passwords.

## Usability

- The mobile application should have an intuitive user interface.
- The website application should have an intuitive user interface.
- The system shall be regularly tested with users to improve overall user experience.

## Scalability

- The database shall be designed to allow for future growth.
- The web application shall be modular for better future scaling.
- The system shall be well documented for modifiability and reusability.

## 3.5 The output of the analysis

At the beginning of the analysis, we divided the requirements into four groups for further review. Business requirements help us meet business goals and objectives. They are essential for the project to stay competitive in the market and extend project lifetime. Their output shows us a bigger picture of the system with its objectives, opportunities, dependencies and risks, as well as it sets product vision with the aim of successful delivery. Analysis of user requirements is used for higher architecture quality, design decisions and to serve development needs. It expresses roles in the system in a comprehensive approach via use case diagrams for the first time. These diagrams predominantly support our understanding of employees and their behaviour in the system. Functional and non-functional requirements guide us what the system must and should do, contain and provide. Their output is essential for both architecture and implementation. They were also partly obtained by interviews with stakeholders, but on top of that, some of them are here as a result of technical and analytic decisions. The amount of their entities is ever-increasing and it is almost impossible to capture all of them, and therefore only the list of the most important is shown. Having all of these details and analytical outcomes allows us to start with the architecture and design of the system.

## Chapter 4

# Architecture and design decisions

Since a significant part of the structure of this documentation attempts to follow the well-known software process model [8] often used in software engineering up to some reasonable level as shown in figure 4.1, after defining and analysing requirements in chapter 3, the subsequent phase is to produce a design and architecture of the target system.

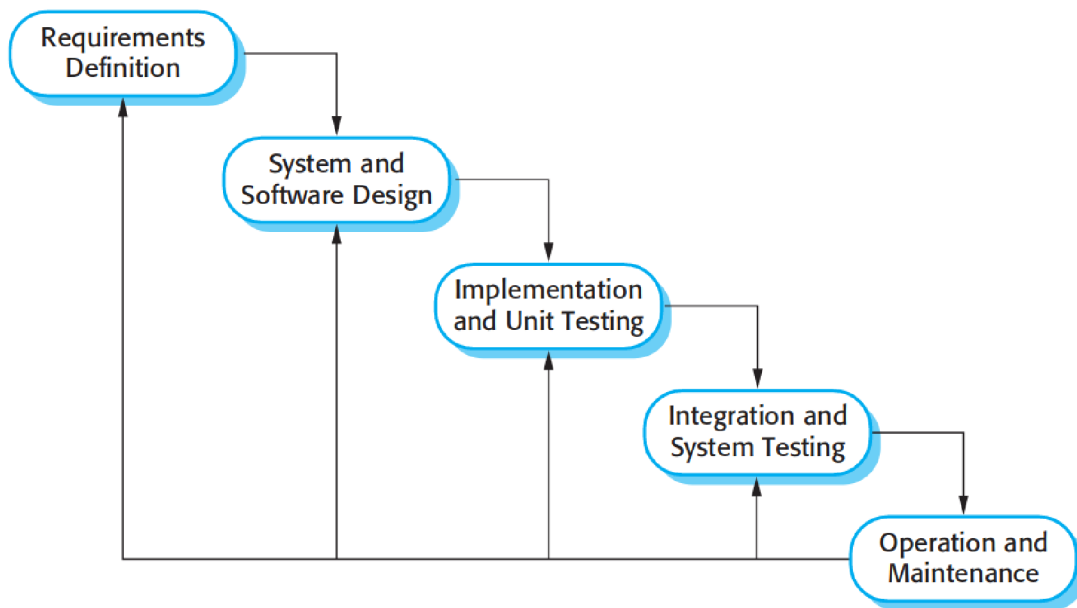


Figure 4.1: The waterfall model. [8]

### 4.1 Architecture insight

The process of crafting software architecture has an important role in the life cycle of software products. If the results of this phase would be interpreted incorrectly, it could consume future effort and time of all stakeholders involved. Hence, it is obligatory to understand what should be the output of such an architecture document. Before starting to apply methods such as layering, creating views or diagrams, it is inevitable to identify stakeholders correlated with the software product, their needs and provide it to them. These



are the people who work with the architecture documents and to whom it should be helpful firstly. As it soon turns out while creating such a document, architecture of a software product is a complex entity which cannot be described by a single view, table or diagram. Consequently, it comes naturally to use categorisation of views and so we have to shortly define several terms before we start with writing down stakeholders of the project.

## **Views**

Crafting architecture documentation consists of a package of design decisions that can be divided into the following three categories. These categories - views - help us to address specific information in documents more accurately to its audience. Each view has its styles to present entities and characteristics of the system.

### **Module views**

Modules are units of software with their attributes and relations. Views created over them provide us with information about:

- structure of modules
- functional dependencies
- relations among modules
- relations among data entities
- modifiability and portability

### **Allocation views**

These views represent the mapping between software elements and their environment which brings a better understanding of:

- build and deploy procedures
- resource allocations
- security
- reliability
- availability

### **Component-and-Connector views**

Components communicate with other components through connectors. This interaction through pathways brings us:

- the system runtime behaviour
- scalability
- functionality for the GUI through event-based styles

## Stakeholders and their needs

Before addressing the views, it is essential to recognise all crucial stakeholders in the system so we can address these views. Once they are defined, we can focus on their needs more closely.

### Business manager

Responsible for the success of the project, the business manager needs to understand the overall architecture of the application to meet business goals. His job duties are among others surrounded by planning, scheduling, dealing with constraints and managing the smooth progress of the project. Therefore his needs relate mainly to module and allocation views that provide him information about deployment, decomposition, dependencies and work assignments. The level of interest in details of views is shown in figure 4.2.

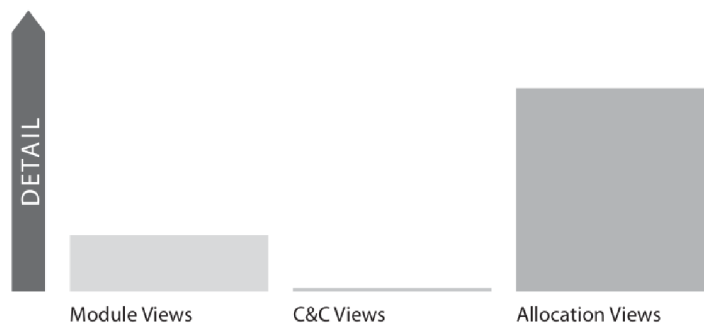


Figure 4.2: Needs of manager. [3]

### System administrator

Responsible for servers and software running on them. The system administrator is involved in many aspects such as understanding what database goals must be met, what software needs to be available for the system, network properties and overview of elements of architecture. Hence he requires the module view to see decomposition and dependencies, C&C views to see what runs on the infrastructure under his scope and allocation views for deployment and installation purposes. The level of interest in details of views is shown in figure 4.3.

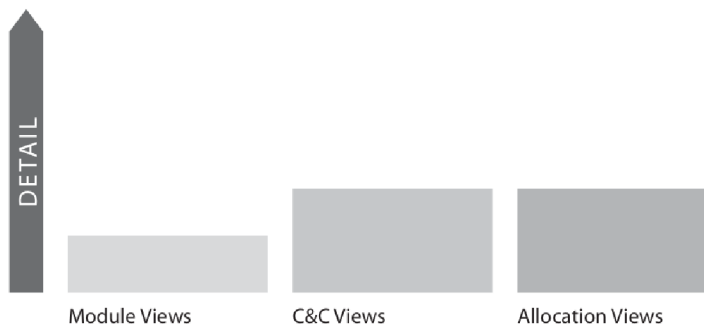


Figure 4.3: Needs of admin. [3]

## Developer

Responsible for developing the system, the developer requires all views to give him an overview of what is required to fulfil job duties of users properly. These views provide them blueprints to program, maintain, test and innovate the system. On top of that, developers need to be able to follow procedures for deployment, installation and understand the overview of the whole system. The level of interest in details of views is shown in figure 4.4.

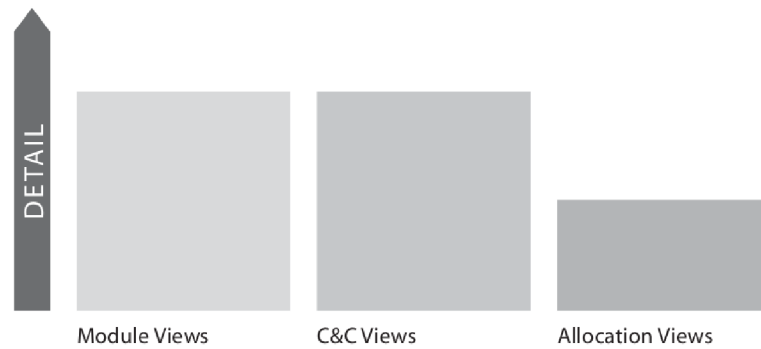


Figure 4.4: Needs of developer. [3]

## User

Users - mainly employees working in laboratories - are in the role of reviewers and can check whether their requirements were delivered. Therefore their main point of interest lays in allocation views so they can see deployment processes to the platforms with which they interact and C&C views to analyse results such as performance that can slow down their workflow. The level of interest in details of views is shown in figure 4.5.

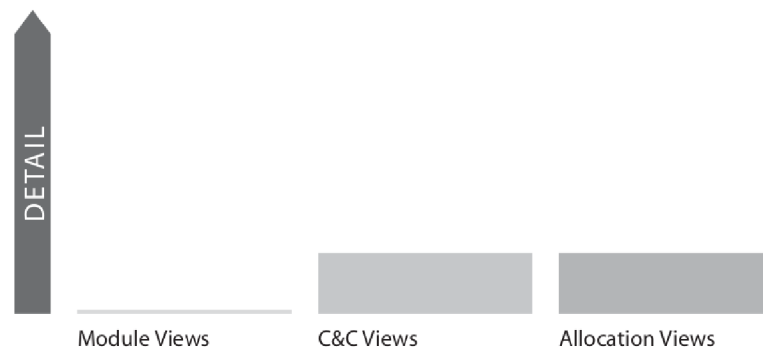


Figure 4.5: Needs of user. [3]

## Selected design and architecture views of the system

Once we identified all the stakeholders involved in the process of making the system, their individual needs of architecture and related documentation can be addressed in several views.

## 4.2 Top layer architecture

The simplistic architecture of the system infrastructure displays the main components of the system. As shown in figure 4.6, the website application and database system are being placed on the same server. It is necessary to mention that borders of the system also represent the same intranet (which means that the system is not reachable outside of this network) therefore not reachable from a public network.

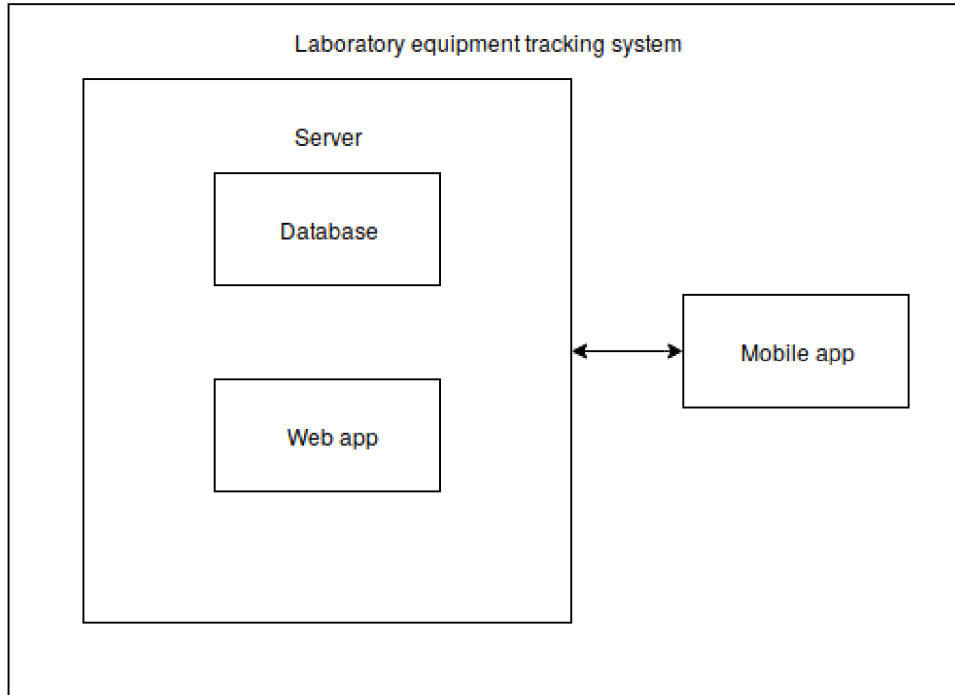


Figure 4.6: A simple architecture of the system infrastructure.

### Communication between modules

By evaluating the system infrastructure, we can notice there are two connections between modules although not all of them are seen at first sight in figure 4.6. The first one is hidden by packing web application and database into one shared module. This simplification allows us to focus on the system from a broader perspective and it is not essential as communication between database and web application is covered in the following chapters. A secondary and more interesting connection is between the web server in general and the mobile application. As they are not physically located at one location, communication between them is through the interface of the web application which could be a source of issues without proper care. The user has to be authorised before borrowing or returning equipment, and that means the order of these operations has to be handled safely at the web server side in order to avoid issues. Communication is illustrated in figure 4.7 with the timeline on the left side which indicates that operations found on the higher position were executed first.

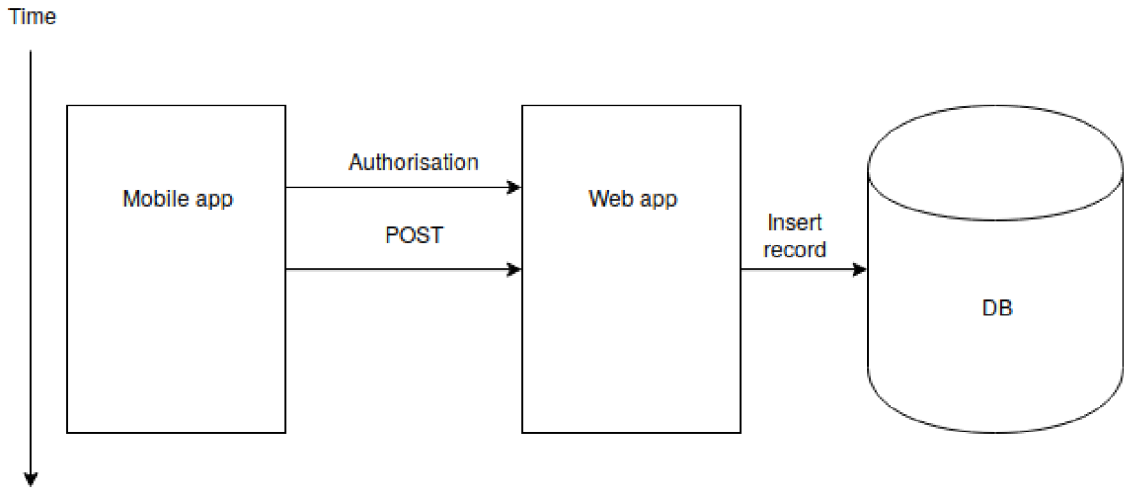


Figure 4.7: Communication between the mobile application and the web server.

### 4.3 Database design

Input for the process of designing a database came from requirements gathered in chapter 3 which were consequently analysed and interpreted by the ER diagram as shown in figure 4.8. This diagram was later on evaluated by all stakeholders and implemented. It covers all reasonable aspects of the system for informational purposes and at the same time preserves other additional elements hidden, such as framework-needed specific tables. This simplification delivers clear documentation of the database for its readers.

#### Main entities

##### Equipment

This entity holds relevant information about equipment as requested by laboratory leaders. It has to relate to a laboratory where it is stored. There are operations over this equipment recorded in the borrowing record. This entity has other relationships for more sophisticated attachment handling that are not covered by the ER diagram.

##### Laboratory

The laboratory consists of many types of equipment and has only one leader who is in charge to take care of the laboratory.

##### Employee

Employee entity keeps basic personal information and has a role attribute which indicates whether an employee is a leader. On top of the mentioned features employee is able to borrow equipment by creating borrowing records. This entity has other relationships for authorisation and informational purposes that are not covered by the ER diagram.

## Borrowing record

This entity serves to an employee for borrowing and returning purposes of equipment.

## ER Diagram

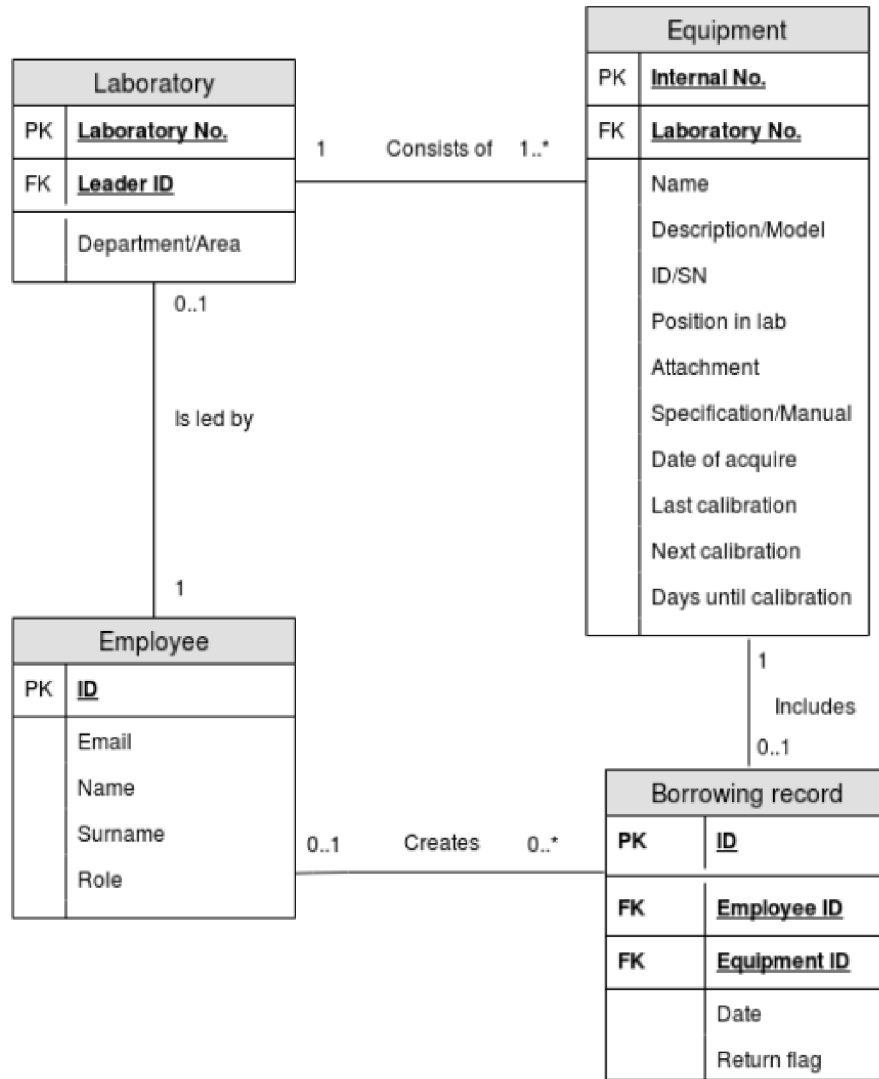


Figure 4.8: Simplified ER diagram.

## 4.4 Selected technologies and database comparison

The choice of technologies to adopt was rather open with some limitations set by the company with an eye toward company policies and business rules. To a great degree, we used mostly open source system technologies for a set of reasons such as:

- security
- lower costs of a new project
- popularity

### Web application

#### Frontend

The graphical interface of the web application is assembled with popular **Bootstrap** library comprised of HTML, CSS and Javascript. It encompasses an extensive list of components and uses a grid system for the purpose of delivering the responsive design of the final product. Along with the stated features, it is accompanied with easy to read and use documentation<sup>1</sup>. Brought up documentation comes in very handy and speeds up the entire development and maintenance process. It is worth to note that minor features of the graphical interface are shipped by Django framework even though Django is, for the most part, the core of the backend.

#### Backend

The business logic of the application is crafted using **Python** as the programming language. There is always a question and discussion about what programming language is most suitable for any software project. There is never one best decision, so the decision was made based on business objectives. These are concerned with the rapidly changing environment, and therefore they have no choice but to respond to new opportunities and challenges by faster development and deployment of the software [8]. Such an environment is well suited for Python due to simple syntax, satisfactory scalability, prototyping and fast development.

In an effort to comply with modern manners of web development and intensifying speed of development after selecting the programming language it was reasonable to apply web framework. Python offers a few choices to meet these needs. We picked **Django** for its built-in admin logic, scalability, first-class documentation<sup>2</sup> and popularity. Django offers many packages that help to solve issues and speed up development. It is also shipped with various security properties and operates well with databases.

The abovementioned benefit of Django operation with database liberates us from building raw SQL queries. Django provides us with a way to communicate with a database through its models and by taking this into account we have a database-abstraction API that lets us perform operations to the database no matter of the database type.

---

<sup>1</sup><https://getbootstrap.com/docs/4.3/getting-started/introduction/>

<sup>2</sup><https://docs.djangoproject.com/en/2.2/>

## Mobile application

Due to the fact that one of the requirements as mentioned in chapter 3 was to create a mobile application on iOS devices, programming language choice was strictly associated with the hardware. Both developments of backend and frontend had to be done in Xcode which is an IDE for developing iOS software by using full-stack language **Swift**.

## Infrastructure

The core of the infrastructure is the **Linux** operating system running on a virtual server. Linux goes well hand in hand with web development and used technologies in this project. It is open source, and there are no additional costs, so it was a logical choice to use it.

## Database comparison and choice

As the project is aimed more towards business goals such as the speed of delivery, benefits arising from the choice of programming language and its framework were in this project considered to be superior to detailed benefits of database choice. With the exception of unofficial workarounds<sup>3</sup>, Django does not support NoSQL<sup>4</sup> databases. Due to this reason, we have reduced the options only to SQL databases as crafting a custom workaround would slow down the development process and result in delayed project delivery. Nevertheless, a NoSQL database was not specifically demanded with respect to requirements.

Left with the option for relational databases, we had a much more relaxed time choosing a suitable option because Django handles backend connectivity very well. It isolates database backend to such a level that we do not have to even construct a single query in SQL. Thanks to its services, after successful configuration of database settings we can directly use its migration features to propagate changes we made to our entities. To fully reap the benefits of Django features, we persisted with officially supported database backends which include PostgreSQL, MySQL, SQLite and Oracle by avoiding 3rd-party backends.

From now on, none of our decisions which database backend to pick is final or fatal as they are comfortably commutable with the assistance of Django. With such freedom in choice, we have selected the open-source option to reduce costs at the beginning project. Until further demands arise, the database choice is **MariaDB** which offers high availability, security, interoperability and scalability<sup>5</sup>. After all, MariaDB is already used in the company environment so its pick would boost the speed of project delivery even more.

## Hardware

To complete a list of used technologies, hardware has to be also included. It is all derived from requirements on the mobile application. To develop an iOS application, purchase of iOS-powered device was indispensable. As only one application runs on this device getting the cheapest iPad was enough. Afterwards, the purchase of necessary ethernet and power cables was made, so that it would be possible to connect this iPad to the intranet given the fact that there is no WiFi in the laboratory.

---

<sup>3</sup><https://django-nonrel.readthedocs.io/en/latest/content/Django-nonrel%20-%20NoSQL%20support%20for%20Django.html>

<sup>4</sup><https://www.mongodb.com/nosql-explained>

<sup>5</sup><https://db-engines.com/en/system/MariaDB>



## 4.5 Architecture process checkpoint

At the beginning of the chapter, we gave insight into the process of making architecture and design decisions. Stakeholders involved in the system were recognised with their corresponding requirements. Some of the top layer architectural views and design decisions were put into place for them. A basic overview of the infrastructure together with connections inside this infrastructure was described and shown closer. In the best interest of the following proper designing and architectural approach, more in-depth designs of frontend and backend of applications were skipped due to the fact that development was following concepts of prototyping. Therefore, this part is documented closer in the following chapter 5. Nevertheless, database design could be formed based on gathered software requirements. Eventually, technologies for the project were chosen with deeper justification of the database selection.

# Chapter 5

## Implementation

Continuing through the waterfall software process model after system design and architecture is settled, we can look into implementation details. This chapter tries to give a description of all fundamental elements of the target system by providing different selected views on the system. Architectural views that can be found in this chapter follow the principles presented in the previous chapter. They are here to satisfy the requirements of stakeholders and complete their picture of the project. On the grounds that the implementation phase was carried out with the help of prototyping, specific views could not be made in the architect phase earlier. Among other things, it deals with three major topics - the user interface of the website, its domain logic and the mobile application as a whole. These topics cover all generic layers as shown in figure 5.1.

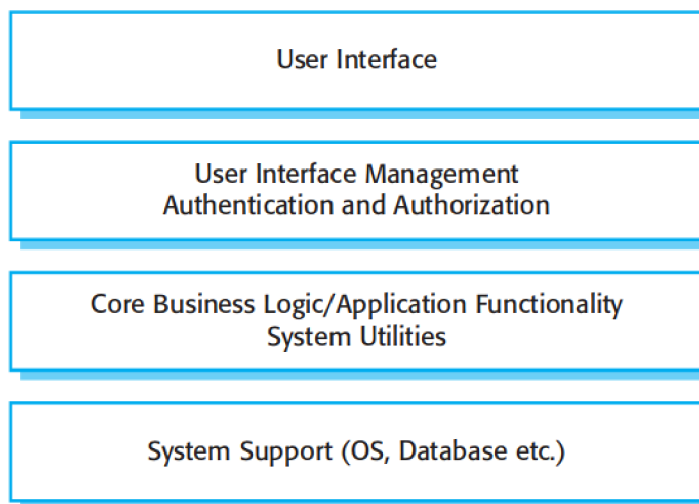


Figure 5.1: A generic layered architecture. [8]

### 5.1 Website user interface

As it was stated in the previous chapter, the user interface on the website is built with a Bootstrap library which uses HTML, CSS and Javascript. Apart from the Bootstrap documentation, a huge amount of information was found in book [7]. To follow good developing approach all files used to deliver graphical user interface are located in folders

named **static** and **templates** as can be seen in figure 5.2, so the project has a clear structure. Static folders include fonts, images, icons, CSS and JS files. On the other hand, templates contain just HTML files.

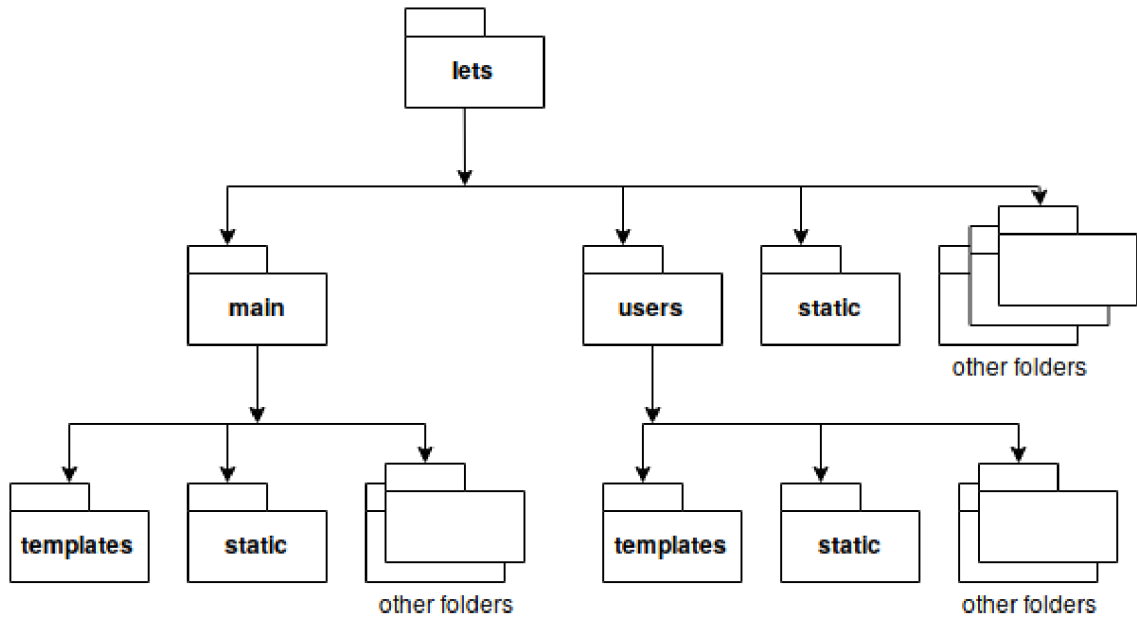


Figure 5.2: Structure of files used by the GUI.

## Website hierarchy

Figure 5.3 shows the hierarchy of the website omitting the authorisation pages. The **Dashboard** serves as a home page that provides access to all top-level modules. On the left side of the figure, we can see the less significant pages of the system. The **Help** page provides info for troubleshooting. The **About** page gives info about the product. The **Cookie policy**, **Data privacy policy** and **Legal Notices** pages are intended for future formal information. The **Profile** page shows basic user details with its history of records and personal QR code. Moving to the second subtree from the left side, we have the **Equipment Tracking** page that provides a list of equipment and availability of those entities. From this page, the user can continue to the **Equipment Details** page that provides details about equipment, its QR code, borrowing history and can store documents associated with the equipment. The subtree in the middle starts with the **Laboratories** page that displays a list of laboratories and a user can proceed to the **Laboratory Details** page that contains laboratory details and a list of equipment belonging to this laboratory. User can move from here to the neighbouring subtree, specifically to the **Equipment Details** page. The next subtree begins with the **Documents** page with a list of general-purpose documents. There a user can reach document details through the **Document Details** link. The management subtree on the right side starting with the **Manage** page is accessible only to managers and laboratory leaders. Managers can access all other pages in this module. The restricted **Leader Management** page for laboratory leaders serves for promoting and demoting employees. The second restricted **Laboratory Management** page for laboratory leaders provides an interface to create, edit and remove laboratories. The **Document Manage-**

**ment** page provides space for inserting and removing general-purpose documents. The last **Equipment management** page is for inserting, removing and editing equipment.

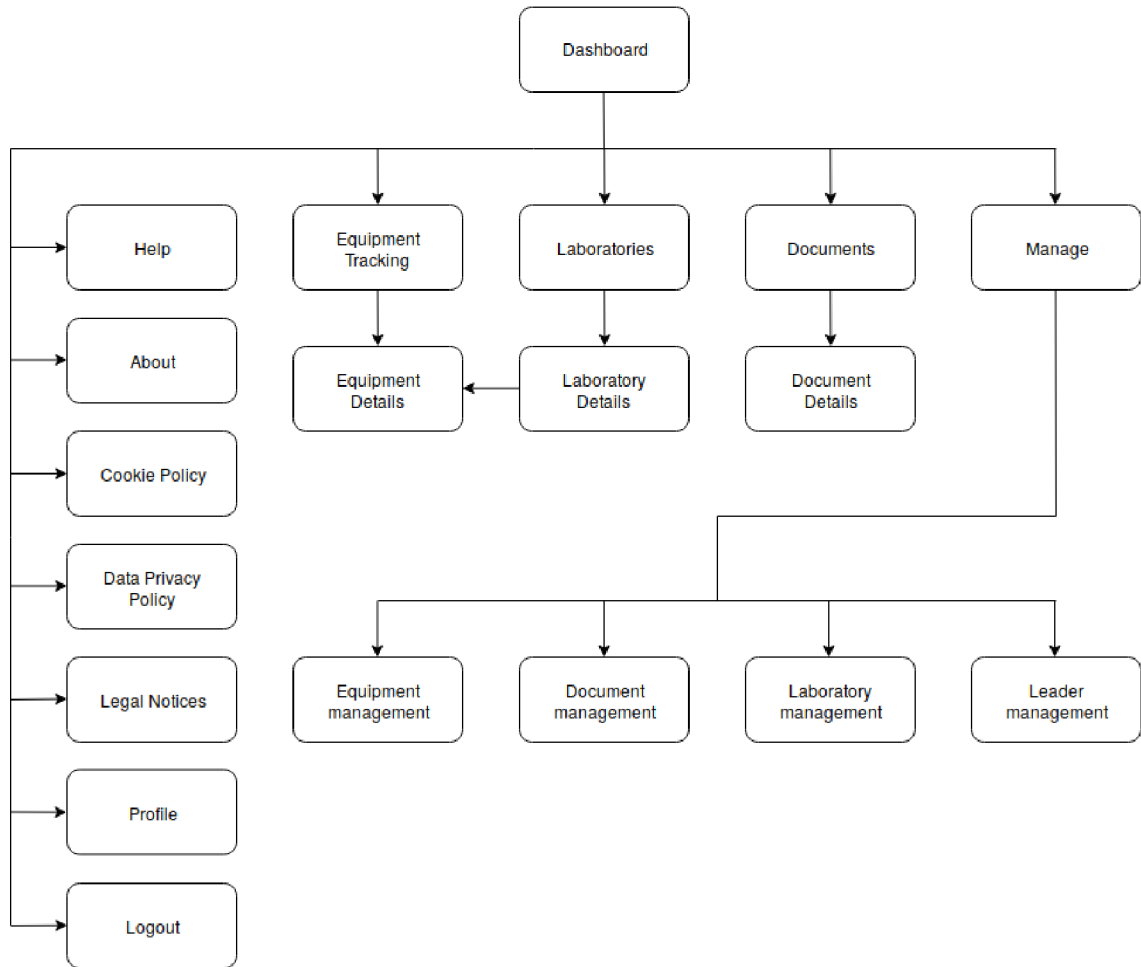


Figure 5.3: Website hierarchy after successful login.

## Wrapper

Except for the landing page of the system and two authorisation pages (login and registration) all pages include the same footer and header. Due to this fact, the wrapping method is used which brings to the table simplification and readability of the source code. The method is programmed in the file **wrapper.html** that includes all HTML elements as needed by the structure<sup>1</sup> starting by tags **html** and **head**. In the **body** tag after the navigation panel we have the **div** tag containing Django templating language<sup>2</sup> that is recognised and interpreted by the template engine and ends up rendering context. This context is then followed by a footer and ending of used tags. This wrapper allows us to use the same navigation panel and footer for all pages without recreating them. On top of that, we have each HTML file containing page separated. The process of wrapping is shown in figure 5.4. The elements bordered by dashed line are visible to the user. The wrapper consists of

<sup>1</sup><https://www.w3.org/TR/html401/struct/global.html>

<sup>2</sup><https://docs.djangoproject.com/en/2.2/topics/templates>

the navigation panel and the footer and wraps context requested by the user. Pages on the right side are dynamically changed as requested.

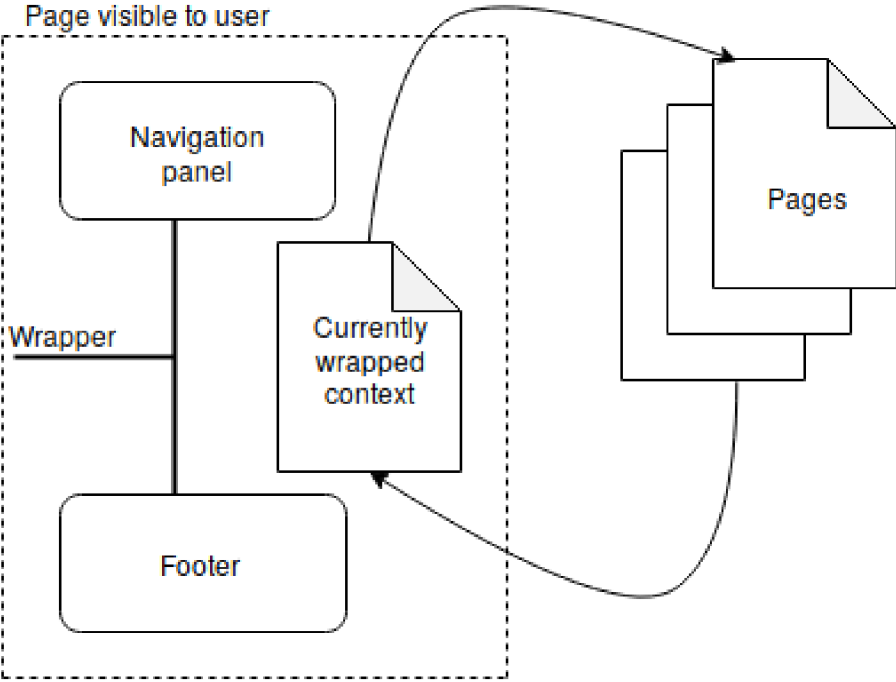


Figure 5.4: Process of wrapping website. Elements bordered by dashed line are visible to user.

## Selected Bootstrap components used in the system

The whole graphical user interface is built with bootstrap in combination with Django, and therefore we selected essential components that can be found in the system.

### Grid system

Every HTML page in the system is constructed using the grid system. The grid system divides every page up to 12 columns in each row with a focus on responsiveness of the graphical user interface of the system. The result of using this component is that the website is visually appealing on any kind of device. An example of the grid system in our graphical user interface is shown in figure 5.5.

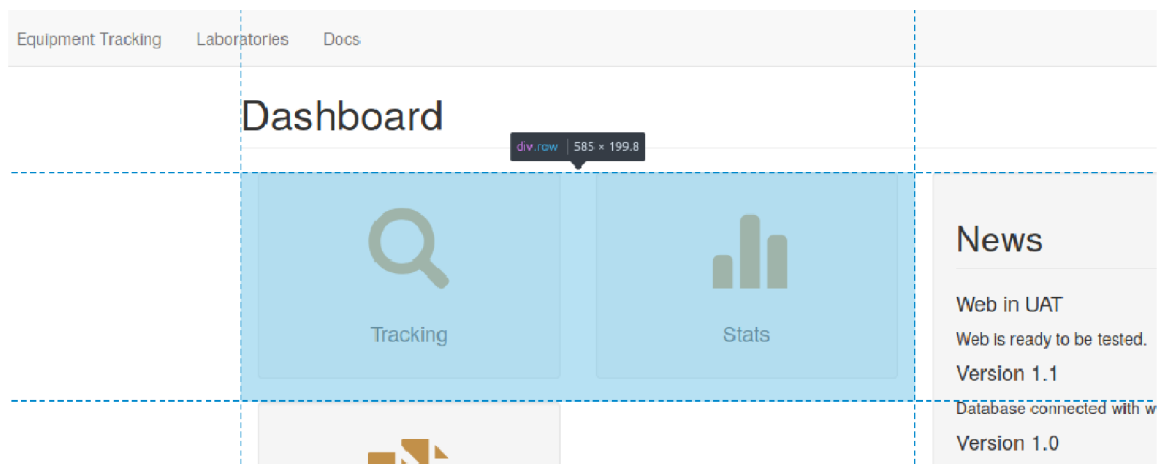


Figure 5.5: Example of the grid system.

### Navbar

Another component that can be seen on almost every page is the navigation bar. This navbar is implemented in the **wrapper.html** using the **navbar-default** class for applying style and effects. Its implementation is quite sophisticated as it also includes a bunch of Django templating code for pagination and minor graphical design features. Navbar of the system is provided in figure 5.6.

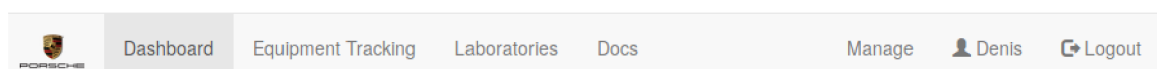


Figure 5.6: Navigation bar of the system.

### Glyphicons

The last bootstrap component is glyphicons. They are used to display icons. Although they might seem to be a non-remarkable part of the system, glyphicons enhance overall user experience and satisfaction which are crucial aspects for product success. They were used very thoughtfully on places to help a user navigate through the system.

## User experience

We have very inconspicuously touched the topic of user experience in the previous section related to glyphs. In today's world of a modern approach to the software development discipline, user experience is getting more and more attention. Due to these reasons, our system also follows several ideas to satisfy users. Feedback for the GUI was collected on a regular basis by interviews and observations.

## Brand presence

Logos are implemented in three locations. The first one was already shown in figure 5.6, second is located at the landing page, and the last one is in the form of a favicon<sup>3</sup>. Less noticeable is the presence of company colours as it can be clearly seen in figure 5.5 (gold colour).

## Copywriting

Omitting words and optimisation of sentences on the web site is another user experience aspect that makes useful content more prominent and reduces the noise level of the pages [5]. This idea is applied and implemented by using short but meaningful names all around the website.

## Navigation

The website meets the concept of clear hierarchies and navigation by providing the user with information about his location on the website at all places. All entities and elements of the website are logically and visually grouped. For example, buttons for navigation in figure 5.6. Furthermore, a user can easily see in which way he is nested and located on the website as shown in figure 5.7. The highlighted **Manage** button shows in which module user is and below the navigation bar interface provides information about the way how users are nested into the current package. Noticeable are clearly visible blue clickable buttons providing the way back through the hierarchy of the pages.

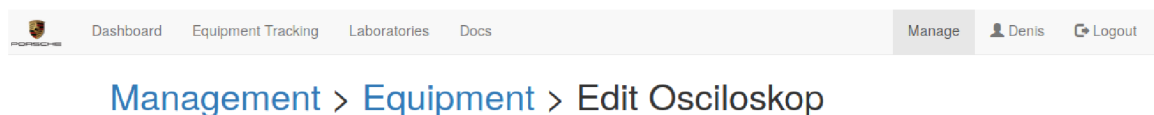


Figure 5.7: Example of how users are nested in the system.

## Communication with the user

The last notable idea of implemented user experience is communication with the user. After the user performs various actions, he has to be informed about the progress or results of his efforts. These responses are illustrated in figure 5.8. A user tried to sign in with incorrect credentials and is immediately informed by a striking red warning with relevant details about the failed operation.

<sup>3</sup><https://techterms.com/definition/favicon>

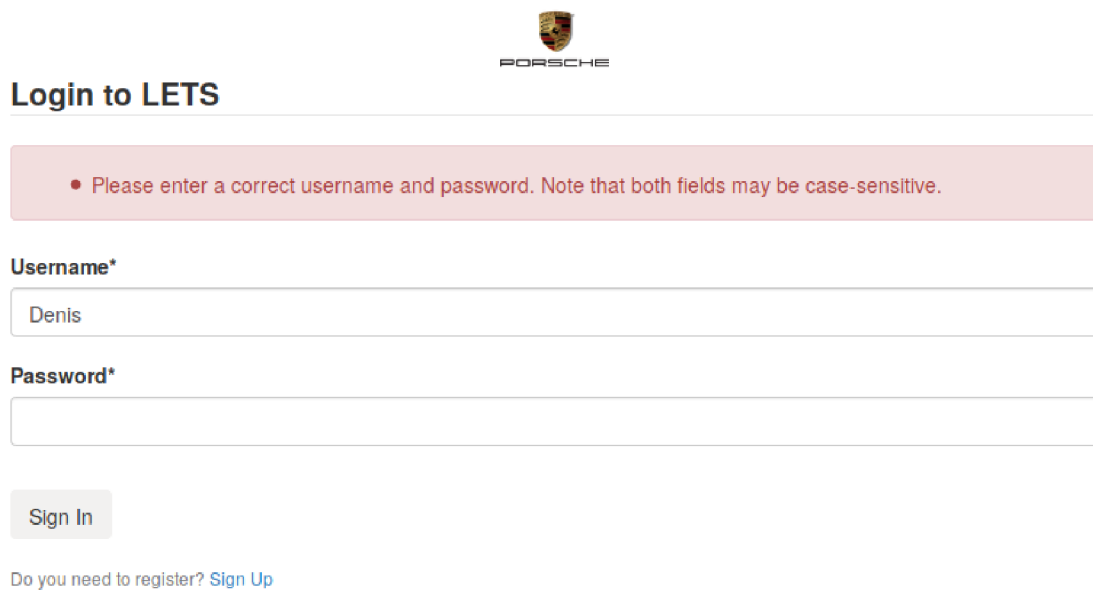


Figure 5.8: Example of how the system responds to a user during failed login operation.

## 5.2 Website business logic

The upcoming section describes the backend of the web application with an aim on Django framework used for its implementation. It provides another set of views for a better understanding of the characteristics and functionalities of the system. Domain logic of the application is branched into three main sections as exposed to view in figure 5.9. The **main** section contains almost all components and aspects of the website such as models to operate over the database, equipment tracking and pages provided to users. The **users** section address authorisation and authentication requirements. It incorporates pages related only with mentioned requirements such as **login.html**, **logout.html**, **register.html** and their **wrapper.html**. This section also cooperates with useful Django user authentication system. The third section - **lets** - is Django's python package that contains the necessary settings and configuration file. Figure 5.9 hides among other components in **other folders** the QR module<sup>4</sup> and folder dedicated for uploaded documentation. For further reading into the Django user authentication, configuration files or any other Django aspects take a look into [4].

### Selected Django components used in the system

Django is a complex framework offering many components for web development, and our system uses plenty of them. Just some of them were selected for a closer look. Apart from the first-class online documentation, a large amount of information was found in the book [4].

<sup>4</sup><https://django-qr-code.readthedocs.io/>



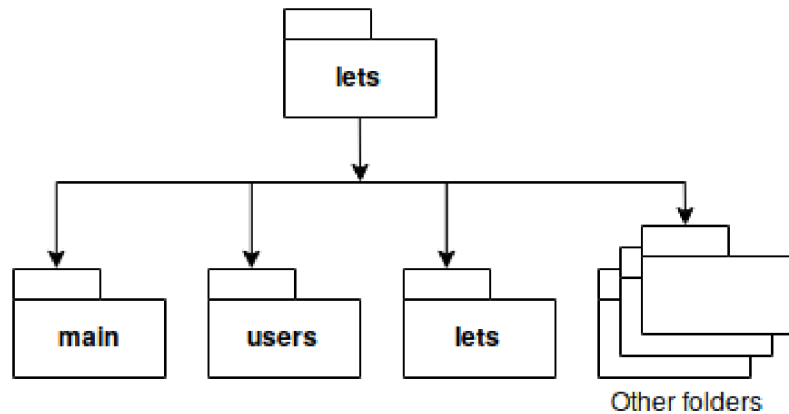


Figure 5.9: Structure of the files used by the backend.

## Views

Django tries to follow the MVC pattern, and due to that reason, views are its core part. The users section contains only one view compared to the rest in the main section, so this section describes and refers only to those in the main section. With the help of Django authorisation module, **views.py** secures all pages except the landing page from unauthorised access; thus non-authorised users cannot reach the system without providing the correct credentials. Functions in the beforementioned file accept user requests and return the demanded pages shipped together with data if needed. For the purposes of shipping data to users, it uses **Models** to access and retrieve objects from the database. A few of these functions use **Forms** with the aim to satisfy user POST requests. If the data in POST request are valid, it is further processed and saved into the database. Below is the list of used views and their purpose:

- **landing\_page** renders the landing page
- **dashboard** renders the dashboard page with user details
- **cookies** renders the cookies page
- **about** renders the about page
- **profile** renders the profile page with record objects
- **device** renders the equipment details page with equipment, attachment and record objects
- **equipment** renders the equipment list page with equipment and record objects
- **privacy** renders the privacy page
- **legal** renders the legal notices page
- **help** renders the help page
- **doc** renders the documents list page with document objects
- **management** renders the management module page

- **equipment\_management** renders the equipment management page with equipment objects and processes **EquipmentInsertForm**
- **equipment\_management\_remove** removes the equipment
- **equipment\_management\_attach\_remove** removes the equipment attachment
- **equipment\_management\_edit** renders the equipment edit page with equipment and attachment objects and processes **EquipmentEditForm** and **EquipmentAttachmentInsertForm**
- **document\_management** renders the document management page with document objects and processes **DocumentInsertForm**
- **document\_management\_remove** removes the document
- **people\_management** renders the leader management page with user objects
- **people\_management\_man\_up** promotes a user to manager
- **people\_management\_man\_down** demotes a manager
- **people\_management\_lead\_up** promotes a user to laboratory leader
- **people\_management\_lead\_down** demotes a laboratory leader
- **laboratory\_management** renders the laboratory management page with laboratory objects and processes **LaboratoryInsertForm** and **LaboratoryChangeForm**
- **laboratory\_management\_remove** removes the laboratory
- **laboratories** renders the laboratories list page with laboratory objects
- **labdetail** renders the laboratory detail page with laboratory and equipment objects
- **rest** encodes the POST request, throws useless metadata, validates an equipment and a user against database and finally inserts **borrowing record** into the database

## Models

Identical to views, models are exclusively located in the main section in the **modely.py** file. It defines the entities and attributes of the ER diagram. Unlike the diagram in chapter 4 it has more entities. Additionally, some of the entities are also added by the Django authorisation module. Python classes representing these entities - tables - also define relationships using keys and characteristics of the attributes such as field types, length or the ability of attributes being set to blank. Here is the list of all entities defined in the mentioned file:

- Laboratory
- Equipment
- EquipmentAttachment
- Record
- UserDetails
- Document

## Templates

Templates were partly covered in the previous section but from the graphical user interface point of view. As mentioned, Django tries to be a MVC framework but it is often called MTV where templates cover the presentation layer. These templates are filled with the data returned from views with the help of the Django template language. This template language performs many operations. Below is the list of selected templates with information about what mentioned language performs there:

- **wrapper.html** - dynamic change of content in the body container as already shown in 5.4, resolving URL names to highlight the active page, providing URL references and username
- **profile.html** - providing user details, shipping QR module, showing a history of records
- **manage\_people.html** - providing user details and reference to promote or demote users
- **manage\_lab.html** - providing forms to create, edit and remove laboratories
- **manage\_eq\_edit.html** - providing forms to edit equipment
- **manage\_eq.html** - providing forms to insert and remove equipment
- **manage\_doc.html** - providing forms to insert and remove documents
- **manage.html** - checking user privileges and based on it enables various management modules
- **landing\_page.html** - checking if a user is authenticated
- **laboratories.html** - providing a list of laboratories
- **labdetail.html** - providing laboratory details
- **equipment.html** - providing a list of equipment and their availability
- **docs.html** - providing a list of documents
- **device.html** - providing equipment and attachment details, shipping QR module
- **dashboard.html** - providing URL references for quick access

## Forms

Most of the forms are stored in the main section in the **forms.py** file. Other forms are associated with the authorisation module and are used in the users section. It uses defined models and provide users with a gateway to POST data into the database. All forms are secured with **csrf\_token** tag<sup>5</sup>.Below is the list of forms in usage:

- DocumentInsertForm
- EquipmentEditForm

---

<sup>5</sup><https://docs.djangoproject.com/en/2.2/ref/csrf/>

- EquipmentInsertForm
- EquipmentAttachmentInsertForm
- LaboratoryInsertForm
- LaboratoryChangeForm

## URLs

There are two main **urls.py** files for routing URLs to views by the Django URL dispatcher as shown in figure 5.11. The file located in the **lets** folder is the entry one to route requests between users and main sections. It contains paths to registration, login and logout pages. On top of that, it includes all paths defined in the second **urls.py** file located in the **main** folder. For routing purposes this entry file uses the Django authorised views feature. The second URL file handles routing inside the main section. In its URL patterns, we can also find a converter type to capture integer values for changeable routing. This converting is primarily used for two reasons. First one is to access the requested row in the database as it can be seen in figure 5.10 where number **17** in the URL indicated the **ID** of the equipment. The second usage is for inserting borrowing records by the mobile application. The mobile application accesses the website through converter named **rest** to authorise and perform an action.

 [127.0.0.1:8000/management/equipment/edit/17](http://127.0.0.1:8000/management/equipment/edit/17)

Figure 5.10: Converter capturing integer values in the URL - in this case number 17 which is an ID of equipment.

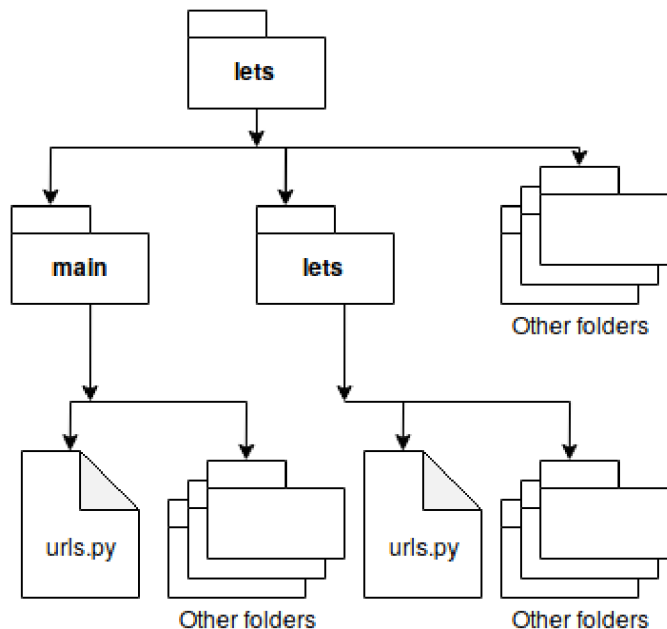


Figure 5.11: Structure of files used by the URL dispatcher.

## QR module

The system is dependent on the external Django QR module to transform text into QR code. Its location in the project folder hierarchy can be seen in figure 5.12. The purpose of the used QR module is to generate QR codes for equipment and users. This mission indicates its location in **profile.html** and **device.html**, where with the support of the Django templating language the QR module transforms equipment serial numbers and user emails to QR codes. These QR codes are subsequently used for authorisation of the user and equipment tracking objectives.

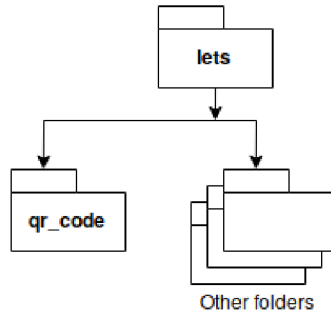


Figure 5.12: Location of the QR module.

An example of the output of the QR module can be seen in figure 5.13 on the profile details page where users can see their assigned QR code which can be further scanned with a mobile application for authorisation. Afterwards, the user can scan his equipment QR code to create a borrowing record.

Dashboard Equipment Tracking Laboratories Docs Manage Denis Logout

## Profile of Denis

Personal Info:

Name	Surname	Email
Denis	Hellenek	Denis.Hellenek@porsche-engineering.cz

QR code:

Figure 5.13: Generated QR code of a user in the profile page by the QR module.

## File upload

Upload of files is implemented with the help of the **Document** and **EquipmentAttachment** models and are divided into two folder categories: (It can be also seen in figure 5.14)

- Attachments folder (for attachments of equipment)
- Uploads folder (for document uploads accessible in **Docs** page)

The mentioned models contain path information to the folders. The **EquipmentAttachment** model is in relationship with the **Equipment** model. It allows users to upload manuals and specifications in the form of files or URLs associated with equipment. The template implementation is located in the `manage_eq_edit.html` file by use of the **EquipmentAttachmentInsertForm** form and the `equipment_management_edit` view. The **Document** model is standalone and not connected with any others. It allows users to upload general documents such as laboratory rules. The template implementation is located in the `manage_doc.html` file by use of the **DocumentInsertForm** form and the `document_management` view.

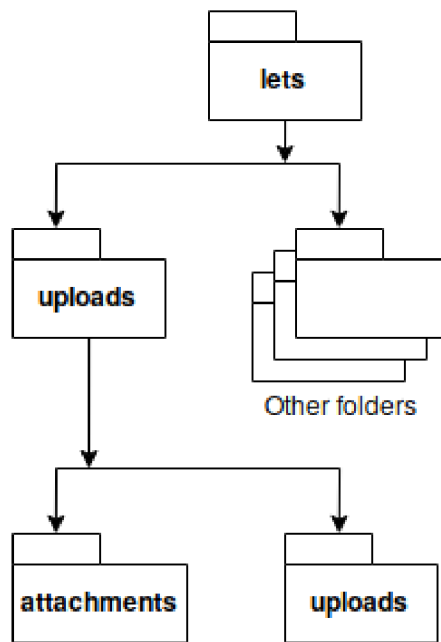


Figure 5.14: Structure of folders dedicated for uploaded files.

An example of document uploading can be seen in figure 5.15.

Insert new document

Name:	<input type="text"/>
Source:	<input type="button" value="Browse..."/> No file selected.

Figure 5.15: Interface for uploading documents.

## Crispy forms

Another external Django module is called **crispy forms**<sup>6</sup> used in registration and login pages for design purposes. These forms are implemented through Django's template language tags.

## Authorisation and authentication

The last selected module is Django authentication. It consists of a Django delivered **User** object including attributes such as **username**, **password**, **session** and **permission** details. Login, logout and registration pages are located in the users section. The only view in this section uses a form delivered by mentioned Django authentication module. All over the project, we can find applied methods inherited from the model delivered by Django such as **@login\_required**, **is\_superuser** or **is\_authenticated**. Authorisation flow of the user before entering the system is illustrated in figure 5.16. Starting at the landing page as shown on top of the image, the user continues to enter the system and is redirected to the login page where he can immediately login. If the user is the first time in the system, a user can register and process through the login page to the system after a successful login. Registration is for functionality and permission purposes, and therefore any employee who is connected to the company intranet can freely register.

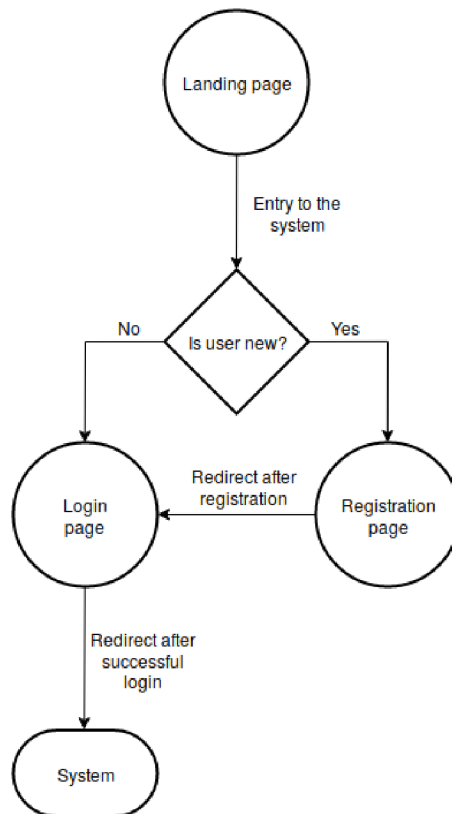


Figure 5.16: Authorisation flow before entering the system.

<sup>6</sup><https://django-crispy-forms.readthedocs.io/en/latest/>

## 5.3 Mobile application

Implementation of the mobile application is relatively small compared to the web application. The reason is that there was only a small number of requirements for the mobile application. Basically, its functionality is just to scan the QR code and send it to the web application by the POST method. Aside from that the entire domain logic is handled by the web application and database system. It was developed in the Swift programming language<sup>7</sup> in conjunction with Xcode IDE<sup>8</sup>. The developed program is aimed primarily at iPad<sup>9</sup> devices. These two tools are fundamental building blocks for creating an application in iOS<sup>10</sup> environment. To a large extent, the development of this application was influenced by web article [6] focused on building a QR scanner.

### Graphical user interface

The graphical user interface is assembled with two screens that are shown to the user. These screens are presented in figure 5.17 in the Xcode environment. The screen on the left side is the first screen a user sees after running the application. After the user presses the touchscreen of the device, he is redirected to the video-capturing screen designed to scan QR codes. Second, the video-capturing screen is shown on the right side of the same figure.

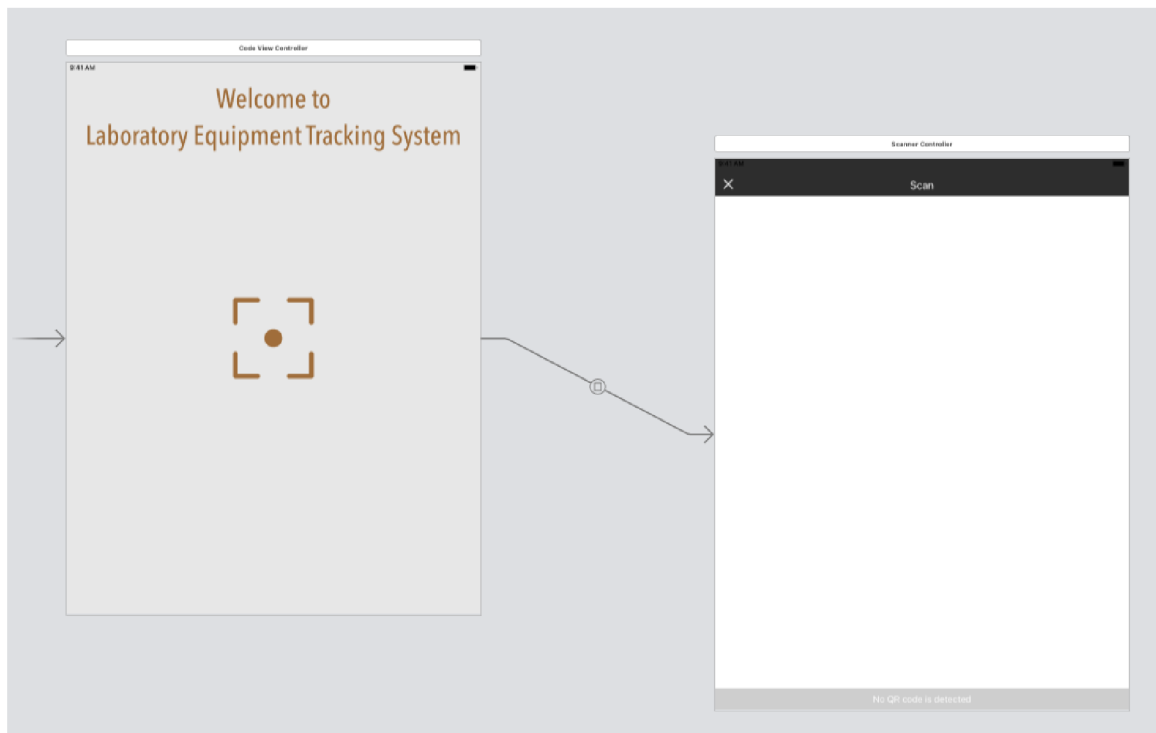


Figure 5.17: Two screens of the mobile application in Xcode environment.

<sup>7</sup><https://swift.org/>

<sup>8</sup><https://developer.apple.com/xcode/>

<sup>9</sup><https://www.apple.com/ipad/>

<sup>10</sup><https://www.apple.com/lae/ios/ios-12/>



## Backend of the mobile application

Each of the screens from figure 5.17 has its domain logic controller. The controller handling the video-capturing screen uses AVFoundation Framework<sup>11</sup> for scanning purposes. The application firstly allocates a camera of the device and links it to real-time capturing session. This session handles the flow of video data from the device. Using the **AVCaptureMetadataOutputObjectsDelegate** protocol we are able to process the captured QR code. By specifying **metadataObjectTypes**, the application gets QR metadata. The video captured by a chosen device is previewed on the screen with the help of **AVCaptureVideoPreviewLayer** after running the capture session. For better user experience the application provides the user with information about capturing the QR code by highlighting borders around the mentioned QR code. Once we reached the QR code, we send it towards the web application for further processing. This is done after we checked that the first scanned result is an email validated against **porscheEmailFormat** regular expression and the scanned equipment on the second place has suffix **letseq**. If everything is correct after validations, the **POST** request is created in the form of encoded URL and sent against server address defined in the variable **url** in the function **recordOperation**.

## 5.4 Extendability of implementation

We have followed principles of a architect mindset in this chapter by providing various views with different abstraction levels. In the web user interface, we have provided folder and page hierarchies for better decomposition understanding, specific wrapping method which acts as a core of the design layout, selected components of used frontend library with the closing of such an important topic as user experience is and its implementation in this project. Moving through website business logic, we could also see top layer decomposition of the backend, crucial Django components that run the application and communicate with the database, URL handling and its location in file hierarchy of the project, input gateway for users in the fashion of forms, QR module, file uploading and authentication system. In the end, we gave insight on a minor mobile application which is also part of the system.

All of these details give us relevant information about how the system is implemented with a better picture of the architecture and design of the system. Continuing with the waterfall method, we needed to cover other aspects in the project lifecycle like testing and deployment (as covered by the next chapter with the aim of achieving complete integration of the product into the production version). Before directly moving there, we provide a few suggestions for possible further implementation of features.

### LDAP authorisation

So far our system handles the entire process of authentication by itself. It might be suitable and perhaps beneficial to end-users to sign up into the system with their credentials that they use to sign up into company intranet. An approach like this would definitely increase overall user experience and satisfaction as they would not be forced to remember additional credentials. Django framework is ready for such scalability of the project, and it allows us to plug in other authentication methods such as LDAP. This idea has to go through the long company decision making process and therefore was not implemented.

---

<sup>11</sup><https://developer.apple.com/documentation/avfoundation>

## **Statistics**

Another useful extended feature could be to have automatically generated statistics about the state of the system and its data. This component could also be easily plugged in as Django is quite open to scalability and adding new modules. It would serve primarily to report a variety of graphs to illustrate the usage of equipment or workload in laboratories for example. Based on these outputs, management could sharpen their business decisions. Deeper business analysis is needed as input to this module and therefore was not implemented.

## **Reservation module**

Last suggested feature would serve to lock equipment against borrowings of it. These reservations could come in handy if some special equipment would be crucial to a certain project delivery under time pressure (with strict SLA for example), and therefore it would reserve locked equipment to be exclusively allocated to this project. More consultations and feedback from users is needed to implement this idea.

## Chapter 6

# Testing and deployment

Last but not least, the phases of development iteration<sup>1</sup> are testing and deployment. This chapter divides testing into two stages on the basis of typical segmentation [8] omitting **Release Testing** that was considered not to be required:

- Development testing
- User testing

The approach to deliver a high-quality product by testing was a mixture of all techniques mentioned above no matter their order or separation. The objective of this chapter is to show how it was being performed. The description of testing is followed by an insight into the deployment of the delivered product with an objective to give an overview of all constraints and required procedures. To a noticeable degree, it was influenced by Django deployment specifics and characteristics [4].

### 6.1 Development testing

Django's special 404 error page as described in [4] and Mozilla page inspector<sup>2</sup> supported development testing activities that were done during the development of the system. The first mentioned tool displays error pages filled with metadata, settings and traceback. These attributes help to find and fix issues and bugs in the backend of the system. This debugging tool can be enabled in **settings.py** by setting the boolean value to true. It is absolutely necessary to note and remember that this error page might carry sensitive data and therefore must be turned off in production systems especially in those environments that are exposed to the public.

In contrast with Django's backend debugging support, the Page inspector was almost exclusively used for debugging graphical user interface issues. It allows simple examination and testing of all HTML and CSS elements. Among others, the responsiveness of the website was tested with its **responsivity design mode**. Additionally, this tool also enables performance, storage and memory tests.

---

<sup>1</sup>Note that we are here referring to the development iteration, not to be confused with the software product lifecycle where are extra operation and maintenance steps. Also as might be expected there are other business aspects and decisions.

<sup>2</sup>[https://developer.mozilla.org/en-US/docs/Tools/Page\\_Inspector](https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector)

## Unit testing Django

Even though this project does not use Django provided unit tests, it is worth to mention that this framework is shipping them. However, the system is ready for size extension and count with unit tests as they are a crucial part in software engineering. For this purpose, in the **main** folder of the project is located file **tests.py** designed for test cases. These tests can be executed afterwards with the following command - **python3 manage.py test lets**.

## Component testing

Another approach used during development testing is called component testing. This technique was used to find possible errors and undesirable behaviour of the interacting objects within the bounds of the entire system. As previously explained, laboratory equipment tracking system has three mutually interconnected components - web application, mobile application and database. Web application and database conjointly occupy one server. On account of this reason, their intercommunication is unproblematic and managed by Django framework itself. Under other conditions interchange of data works between web and mobile communication endpoints. We had to test the web interface carefully in the form of URL links by RESTClient<sup>3</sup> for these reasons. Creating custom HTTP requests helped to tune this interface and in turn, connection of the mobile application went smooth.

## 6.2 User testing

Testing with the user representatives (solely with the laboratory leaders and people manager acting in the role of the product owner) was carried to completion on a routine basis in an informal style. The intention of the user testing was above all directed towards the achievement of completing the implementation of the requirements, and it was operated in two stages:

- Alpha testing
- Beta testing

### Alpha testing

We have performed the alpha testing process during meetings where users could see and test the system as it was being developed. Findings from these meetings helped identify problems in the early stages of the development. Regular feedback from users also shaped the product for its improved practical usage.

### Beta testing

Succeeding to build a reasonable prototype which was delivered with major issues resolved and improved while reflecting on its real usage, it was made available to users for their experiments with the system by deploying the application to provided virtual server. What's more, the prototype connected to the database was stocked with the dataset. As a result, other drawbacks were found and straightened out.

---

<sup>3</sup><https://addons.mozilla.org/en/firefox/addon/restclient/>

## 6.3 Deployment insight

Deployment of the system can be logically divided into three parts:

- Web application
- Database
- Mobile application

### Web application deployment

The web application was deployed onto the provided virtual Linux server by the company. For deployment purposes, several software dependencies had to be met, and in case they were missing they had to be additionally installed:

- Python3 developer package<sup>4</sup>
- Python's qrcode<sup>5</sup> and mysqlclient<sup>6</sup>
- Django and its QR code module and crispy forms that were already explained in previous chapters

### Database

MariaDB database was installed on the same virtual server as the web application and connected by configuring Django database settings. Also, a database user with credentials was created for accessibility purposes.

### Mobile application

The mobile application was installed on an iPad that is located in one laboratory. This iPad was connected to an intranet with the purchased ethernet cable. Users could experiment with the application by downloading QR codes from the website application and applying them in a laboratory. This setup also include a purchased power cable for the iPad, so users don't have to worry about its battery running out.

---

<sup>4</sup><https://pkgs.org/download/python3-devel>

<sup>5</sup><https://pypi.org/project/qrcode/>

<sup>6</sup><https://pypi.org/project/mysqlclient/>

## 6.4 Extendability of testing and deployment

Throughout this chapter, we have exhaustively interpreted how we approached and broke down testing stages, what debugging tools and how we used them for development testing, how user testing led to more valuable product and finally in which fashion was the system delivered. Now, we provide a suggestion about the possible extendability of these spheres for the future of the product.

### Testing

Omitted acceptance testing is on real-world production systems very important and reduces the risk of disruptive effects on business [8], and therefore it should be performed as illustrated in figure 6.1. The same suggestion applies to unit tests that should be likewise performed. That's why the prepared test file and guidance comes in handy.

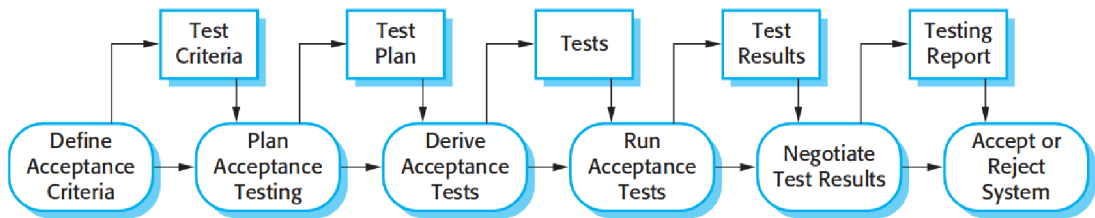


Figure 6.1: Example of process of acceptance tests. [8]

### Web deployment

Production system should first, and foremost aim for impenetrable security and reliability and therefore here is the list of recommended ideas to follow or consider during web deployment:

#### Django settings

- **SECRET\_KEY** setting must be confidential
- **DEBUG** must be set to false
- **ALLOWED\_HOSTS** should be evaluated
- **CACHES** should be evaluated

#### Other considerations

- **Database passwords** must be secured
- **File handling** should be evaluated
- **Database and files** should have proper back-up solutions
- **HTTPS** should be considered

## Chapter 7

# Conclusion

The primary objective of this thesis is to deliver a service for laboratory equipment tracking implemented as a database system to the Czech location of *Porsche Engineering*. The database system was successfully delivered and deployed to an internal company pre-production virtual server.

At the beginning of the thesis, we have gone through an introduction to service management in an effort to enlighten standards for first-class delivery of the services and their incorporated products.

Afterwards, the key requirements for the system were gathered from a reasonable group of user representatives. The collected data were further categorised and analysed to give us a more detailed picture of the expected system.

Moreover, the output of the analysis supplied information for the architecture phase of the thesis. Due to this fact, we were able to create diverse architecture views, establish optimal database design decisions and select the most suitable technologies for the system.

Furthermore, the system was developed by following the acknowledged architecture and design decisions. It was implemented in a modern approach by using current frameworks and principles that boost the desirable system characteristics.

In addition to the above-written, we have performed development tests with the use of different tools and a set of experiments with users. The produced findings contributed to increased quality of the system that could be further deployed into pre-production stage.

In conclusion, by going through an entire software engineering process, we managed to craft a system that enhances the internationally integrated engineering services of *Porsche* in its location in Prague. During several engineering phases, we previously suggested possible features that can be implemented, additional tests that can be executed and we have pointed at potential constraints in deployment to production. To extend this service to a global level - meaning to all *Porsche* branches around the world - a more in-depth focus on service management is recommended. Appropriate service strategy, transition, operations and continual improvements will bring the organisation achievement of new business objectives, and therefore should also be deeply researched similarly to the software engineering part.

# Bibliography

- [1] *Delivering value to today's digital enterprise: The state of IT Service Management*. Forbes Insights. 2017. [Online; accessed 30.04.2019]. Retrieved from: <https://www.bmc.com/content/dam/bmc/migration/pdf/Delivering-Value-to-Today%27s-Digital-Enterprise-FINAL.pdf>
- [2] Axelos: *ITIL Foundation: ITIL 4 Edition*. Stationery Office. 2019. ISBN 978-0113316076.
- [3] Clements, P.: *Documenting Software Architectures: View and Beyond, Second Edition*. Addison-Wesley. 2010. ISBN 978-0-321-55268-6.
- [4] George, N.: *Mastering Django: Core*. Packt Publishing. 2016. ISBN 978-1-78728-114-1.
- [5] Krug, S.: *Don't Make Me Think! A Common Sense Approach to Web Usability, Second Edition*. New Riders. 2006. ISBN 0-321-34475-8.
- [6] Ng, S.: *Building a Barcode and QR Code Reader in Swift 4 and Xcode 9*. [Online; accessed 30.04.2019]. Retrieved from: <https://www.appcoda.com/barcode-reader-swift/>
- [7] Rahman, S. F.: *Jump Start Bootstrap*. SitePoint. 2014. ISBN 978-0-9922794-7-9.
- [8] Sommerville, I.: *Software Engineering, Ninth Edition*. Addison-Wesley. 2011. ISBN 0-13-703515-2.
- [9] Wiegers, K.; Beatty, J.: *Software Requirements, Third Edition*. Microsoft Press. 2013. ISBN 978-0-7356-7966-5.



# Appendix A

## Content of the attached CD

- This documentation in PDF
- Source code of this documentation in LATEX
- Source code of the mobile application in Swift
- Source code of the web application in Django
- External modules used by Django
- Guiding information in README

## Appendix B

# Selected screenshots from the web application

### B.1 Landing page



Figure B.1: The landing page of the system.

## B.2 Equipment management page

### Management > Equipment

Insert new equipment

<b>Name:</b>	<input type="text"/>
<b>Description:</b>	<input type="text"/>
<b>Serial number:</b>	<input type="text"/>
<b>Position in lab:</b>	<input type="text"/>
<b>Acquire date:</b>	2019-05-12 13:04:31
<b>Last calibration:</b>	2019-05-12 13:04:31
<b>Next calibration:</b>	2019-05-12 13:04:31
<b>Laboratory:</b>	-----

Submit

Remove equipment



Equipment	Serial number	Edit	Remove
Oscilloskop	102249		

Figure B.2: The equipment management page of the system.

## B.3 Equipment detail page

### Equipment Tracking > Equipment details: Oscilloskop

Description:

R&SRTH1004

Specifications:

Name	SN	Position in Lab	Acquired on
Oscilloskop	102249	PEG-CZ-EEH	March 25, 2019, 9:19 a.m.

Calibration Info:

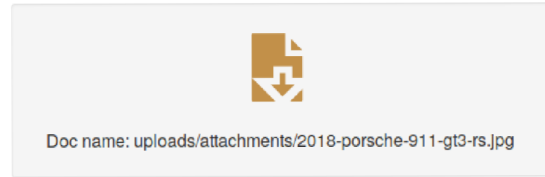
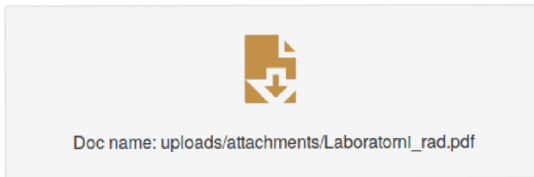
Last calibrated on	Next calibration date	Days left
March 25, 2019, 9:19 a.m.	March 25, 2019, 9:19 a.m.	9999999

List of attached URLs:

URL
<a href="https://www.rohde-schwarz.com/dk/prduct/rth-productstarpage_63493-156160.html">https://www.rohde-schwarz.com/dk/prduct/rth-productstarpage_63493-156160.html</a>

QR code (DISABLED):

Documents:



History:

Borrowed on:	Returned on:	Employee ID:
May 9, 2019, 1:30 p.m.	None	1
May 9, 2019, 1:44 p.m.	May 9, 2019, 1:44 p.m.	1

Figure B.3: The equipment detail page of the system.

## B.4 Management page

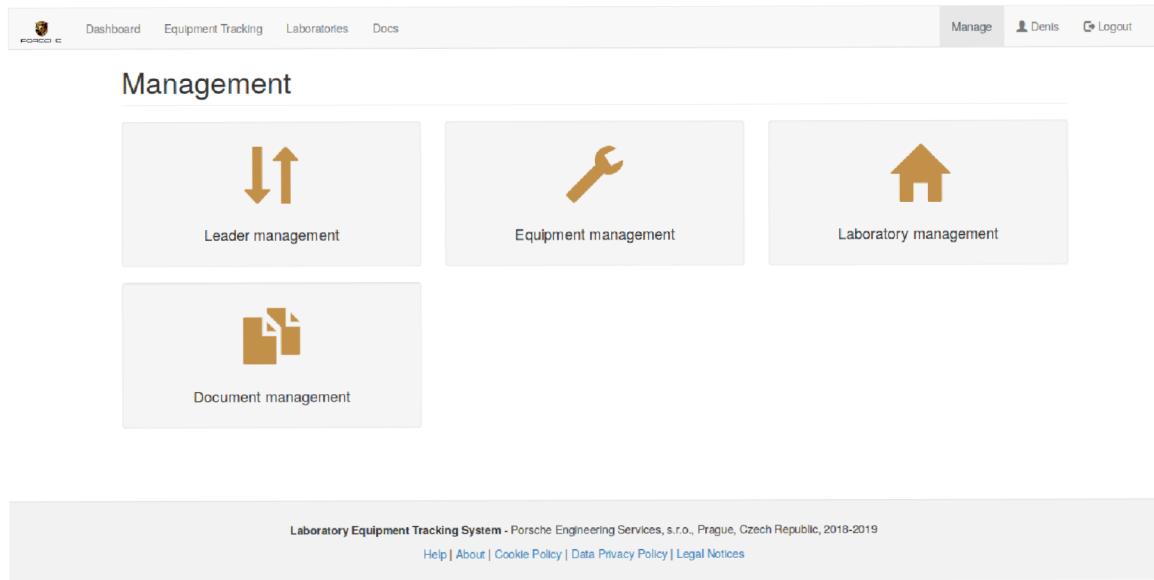


Figure B.4: The management page of the system.