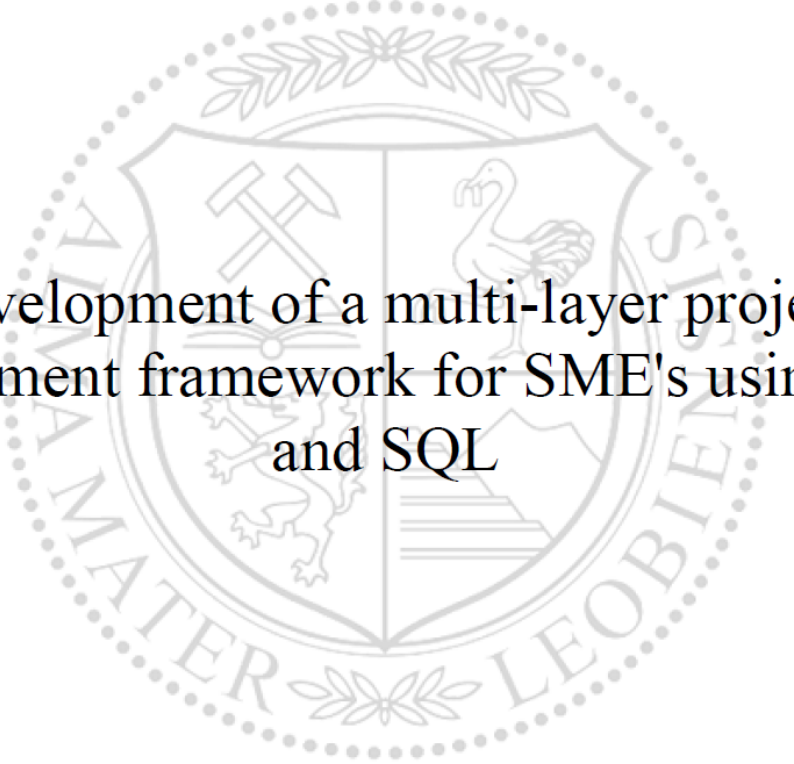




Chair of Metal Forming

Master's Thesis



Development of a multi-layer project
management framework for SME's using PHP
and SQL

Hans - Jörg Schmölder, BSc

March 2021



MONTANUNIVERSITÄT LEOBEN

www.unileoben.ac.at

EIDESSTÄTTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 10.03.2021

A handwritten signature in black ink that reads 'Hans-Jörg Schmölzer'.

Unterschrift Verfasser/in
Hans - Jörg Schmölzer

Acknowledgment

I would like to take this opportunity to thank all those who supported and motivated me during the preparation of this Master's thesis.

First of all, I would like to thank DDipl.-Ing. Benjamin Ralph, who supervised and examined my master's thesis. I would like to express my sincere thanks for the helpful suggestions and constructive criticism during the preparation of this thesis.

I also like have to thank Dipl.-Ing. Macel Sorger, who supported me in the implementation and extension of the existing software. I would like to thank him for the helpful advice regarding the programming and his time for the implementation.

Special thanks go to all the participants in my survey, without whom this thesis could not have been written. I would like to thank them for their willingness to provide information and their interesting contributions and answers to my questions.

I would also like to thank my fellow students Dominik Dax and Markus Berger, who supported me with a lot of patience, interest and helpfulness. I would like to thank them for the numerous interesting debates and ideas, which have contributed significantly to the fact that this Master's thesis is available in this form.

Finally, I would like to thank my parents, who made my studies possible through their support.

Hans-Jörg Schmölzer

Leoben 18.03.2021

Abstract

The interconnection of people, machines and products in the fourth industrial revolution (I4.0) is opening up more opportunities for managing projects. In response to these developments, project management must evolve as a result of the possibilities of direct integration of machines and process data. The possibilities of automated connections therefore pose new challenges for the interfaces between the different layers.

In order to be successful with digital project management, a multi-layered networked solution that uniformly notes the machine data, activities of the employees and the times of these activities is required. Especially for SMEs, the cost factor for the development and support of such a solution must also be taken into account. Another point of digitally supported project management is the security of the data input. These must be protected against unauthorised manipulation and extraction.

The objective of this thesis was to develop an end-to-end, adaptable solution for the integration of production and process data into a project management tool, with particular emphasis on a user-friendly graphical user interface. For the purpose of cost-effectiveness open source solutions, specifically PHP and SQL, were used.

By using the software mentioned above, a multi-layered project management tool linked to the machinery has been created and put into use. By using this system, it is now possible to present the administration of projects, machines and employees in a simple and concise way. This leads to a significant reduction in administrative effort. Through the automated connection of essential machine data, such as operating and idle hours, it is also possible to optimise maintenance intervals. This leads to a further increase in efficiency in the area of maintenance and enables the use of modern predictive maintenance algorithms in the future.

Kurzfassung

Durch die Vernetzung von Menschen, Maschinen und Produkten in der vierten industriellen Revolution (I4.0) werden dem Projektmanagement mehr Möglichkeiten eingeräumt. Als Reaktion auf diese Entwicklungen muss sich das Projektmanagement, durch die Möglichkeiten der direkten Integration von Maschinerien und Prozessdaten, weiterentwickeln. Durch die automatisierte Anbindung ergeben sich neue Herausforderungen an die Schnittstellen zwischen den verschiedenen Layern.

Um mit einem digitalen Projektmanagement erfolgreich zu sein, benötigt man eine mehrschichtige vernetzte Lösung, welche die Maschinendaten, Aktivitäten der Mitarbeiter und die Zeiten dieser Aktivitäten einheitlich notiert und für eine Auswertung aufbereitet. Gerade bei KMU's ist hierbei zusätzlich der Kostenfaktor für die Entwicklung und Betreuung einer solchen Lösung zu berücksichtigen. Ein weiterer Punkt des digital unterstützten Projektmanagements ist die Sicherheit der eingetragenen Daten. Diese müssen vor unautorisierter Manipulation und Auslesen geschützt werden.

Ziel dieser Arbeit war es daher, eine durchgehende, adaptierbare Lösung für die Integration von Produktions- und Prozessdaten in ein Projektmanagement Tool, unter besonderer Berücksichtigung von grafisch ansprechenden Benutzer Oberfläche zu entwickeln. Um den Kostenfaktor gerecht zu werden, wurden Open Source Lösungen, im speziellen PHP und SQL, eingesetzt.

Durch den Einsatz oben genannter Programme wurde ein mehrschichtiges, mit dem Maschinenpark verbundenes, Projektmanagement Tool erstellt und zum Einsatz gebracht. Durch die Verwendung dieses System ist es nun möglich, die Administration von Projekten, Maschinen und Mitarbeitern einfach und kompakt darzustellen. Dies führt zu einer signifikanten Verringerung von administrativen Aufwand. Durch die automatisierte Anbindung von essentiellen Maschinendaten, wie beispielsweise Betriebs- und Leerlaufstunden, ist es zudem möglich, die Wartungsintervalle zu optimieren. Dies führt zu einer weiteren Effizienzsteigerung im Bereich der Instandhaltung und ermöglicht zukünftig den Einsatz von modernen vorausschauende Wartungsalgorithmen.

Table of content

Affidavit	I
Acknowledgment	III
Abstract	IV
Kurzfassung	V
List of figures	VII
List of code snippets	IX
List of abbreviations	X
1 Introduction	1
2 Fundamentals	3
2.1 Project management	3
2.2 Database software system	4
2.3 Graphical user interface Environment	5
2.4 IT Security	5
2.5 Usability requirements	6
3 Development and implementation	7
3.1 Stakeholder analysis	7
3.2 Feasibility study	10
3.3 MySQL database	13
3.4 PHP	22
4 Results and discussion	65
5 Conclusion and outlook	83
References	X
Appendix	12
List of functions	12

List of figures

Figure 1: Result of the Stakeholder analysis.....	8
Figure 2 Roles of the project management tool	9
Figure 3 Flowchart of the connections of the tables	13
Figure 4 Tables of the project management tool.....	14
Figure 5 Attributes of the table <i>activitymac</i>	14
Figure 6 Attributes of the table <i>activityuser</i>	15
Figure 7 Attributes of the table <i>machines</i>	16
Figure 8 Attributes of the table <i>projects</i>	18
Figure 9 Attributes of the table <i>projecttypes</i>	19
Figure 10 Attributes of the table <i>users</i>	20
Figure 11 Attributes of the table <i>roles</i>	21
Figure 12 Flowchart of the PHP pages invisible for the user.....	22
Figure 13 Flowchart of the php pages visible for the user.....	23
Figure 14 General overview of the PHP pages	23
Figure 15 Header section of the project management tool.....	25
Figure 16 Navigation menu with different roles	27
Figure 17 Exemplary footer of the project management tool.....	27
Figure 18 Flowchart Login possibilities	28
Figure 19 Display of the last activities.....	31
Figure 20 Input mask for new activities.....	32
Figure 21 Input mask of activity update	35
Figure 22 Flowcharts of the project related pages	38
Figure 23 Flowchart of new project creation	40
Figure 24 Flowchart of the update project.....	43
Figure 25 Drop down menu and start / end date	45
Figure 26 Flowchart project details	45
Figure 27 Flowchart project details	49
Figure 28 Flowcharts of the machine related pages	50
Figure 29 Flowchart of new project creation	52
Figure 30 Flowchart of the update machine.....	54
Figure 31 Flowchart machine details	56

Figure 32 Flowchart project details	59
Figure 33 Related pages of the employee section	60
Figure 34 Flowchart to register a new user	61
Figure 35 Flowchart of the update user process	63
Figure 36 Multi-layer architecture of the chair of metal forming.....	65
Figure 37 Visible pages for the users	66
Figure 38 Index page with login for user	66
Figure 39 home.php of Administration	67
Figure 40 Difference of activities page for admin (left) and user (right).....	68
Figure 41 Input mask for new user activities.....	68
Figure 42 Activity Update page for administration.....	69
Figure 43 Project page of admin (left) and user (right).....	70
Figure 44 New project page.....	70
Figure 45 Update project page	71
Figure 46 Project detail page.....	72
Figure 47 Project Detail page with involved items.....	73
Figure 48 Finished projects detail page	74
Figure 49 Machines overview page	75
Figure 50 Machine page for new machines with input mask	76
Figure 51 Machines update page with input mask	77
Figure 52 Machine detail page	78
Figure 53 Machine detail list with involved items	79
Figure 54 Machine archive page	80
Figure 55 Employee page of an admin (left) and user (right)	81
Figure 56 Employee sign up page	81
Figure 57 Employee page of an admin (left) and user (right)	82

List of code snippets

Code snippet 1 Static header functions	24
Code snippet 2 Function <i>checkUser(...)</i>	24
Code snippet 3 Navigation menu admin view	25
Code snippet 4 Function <i>checkAccess(...)</i>	26
Code snippet 5 Function <i>loginUser(...)</i>	29
Code snippet 6 Function <i>displayProjects(...)</i>	30
Code snippet 7 Function <i>displayMachines(...)</i>	30
Code snippet 8 Function <i>displayActivity(...)</i>	32
Code snippet 9 Transfer page for create activity	33
Code snippet 10 Function <i>createActivity(...)</i>	34
Code snippet 11 Additional updates of <i>createActivity(...)</i>	35
Code snippet 12 Transfer page of <i>activityUpdate.php</i>	36
Code snippet 13 Function <i>updateActivity(...)</i>	37
Code snippet 14 SQL statement of Function <i>displayProjects(...)</i>	39
Code snippet 15 Data display of fetched SQL data	39
Code snippet 16 Transfer page for a new project	41
Code snippet 17 Function <i>createProject(...)</i>	42
Code snippet 18 Function <i>updateProject(...)</i>	44
Code snippet 19 Transfer function of <i>projectActive.php</i>	46
Code snippet 20 Functions of <i>projectDetail1.php</i>	46
Code snippet 21 Function <i>involvedEmployee(...)</i>	47
Code snippet 22 Function <i>involvedMachines(...)</i>	48
Code snippet 23 Function <i>displayMachines(...)</i>	51
Code snippet 24 Function <i>createMachine(...)</i>	53
Code snippet 25 Function <i>updateMachine(...)</i>	55
Code snippet 26 Transfer function of <i>machineDetail.php</i>	57
Code snippet 27 Machine detail PHP functions	57
Code snippet 28 Function to show involved projects	58
Code snippet 29 Function <i>createUser(...)</i>	62
Code snippet 30 Function <i>updateUser(...)</i>	64
Code snippet 31 Function <i>inactiveUser(...)</i>	64

List of abbreviations

ERP	...	Enterprise Resource Planning
GUI	...	Graphical User Interface
HMI	...	Human Machine Interface
IoT	...	Internet of Things
MES	...	Manufacturing Execution System
PHP	...	Hypertext Preprocessor
PMT:CMF	...	Project Management Tool of the Chair of Metal Forming
SME	...	Small and Medium-sized Enterprises
SQL	...	Structured Query Language

1 Introduction

The interconnectivity, automation, machine learning, and real-time data is in the focus of the fourth industrial revolution (I4.0). Industry 4.0 combines physical production and operations with smart digital technology, machine learning and Big Data to create a more holistic and connected ecosystem for businesses, their people, machines and products. In response to these developments, project management must evolve as a result of the possibilities of direct integration of machines and process data. The possibilities of automated connections therefore pose new challenges for the software interfaces between the different layers. [1]

In an SME, the combination of lack of staff and specific knowledge may cause staff to be assigned to project management tasks on top of their jobs. As a result, project manager activities are seen more as coordination activities and not as a strategic, integral unit. This results in a lack of dynamism and sustainability with the holistic management of the required processes until the end. To counteract this aspect, it is necessary to create and support a complete, integrated and agile project management. In addition to the creation of the project phases, iterative process steps are introduced in order to meet the requirements of the customer and also to be able to react agilely to changes. [2]

To achieve this, the used project management tools have to be changed from a single recording tool, which is only available when the employee is available, to a server-based tool. The advantage of this is, that tools are always available and provide a customised output for each employee. This enables direct tracking of projects with smaller status meetings and generates faster completion. [3]

In order to ensure this, SMEs are dependent on smaller, mostly not standardised solutions which, in addition to low cost intensity, also offer a high degree of flexibility and expandability. These conditions lead to open source software support because, as a rule, these products can be downloaded free of charge. Additional costs such as training, maintenance and support are sunk costs. Companies pay for it, regardless of whether the software is open source or closed source. Furthermore, open source software offers a high degree of customisation possibilities through access to the code itself. Detailed customisations can usually be made with limited resources. [4]

As a reason a digital project management tool for SMEs must consist of a cost-efficient and user friendly management tool. This integrates the production network and the shop floor as well as providing an overview of the projects for the administration and the project managers. The purpose of this thesis is to develop a cost-efficient, agile and digital project management tool for administration and shop floor use. It contains information about the machinery, the employees and projects of the chair of metal forming at Montanuniversität Leoben.

The key points of this thesis are the development of a database for data storage of the shop floor sensor data and the administrative data of the projects, including development of a graphical environment that allows the input and tracking of data from the shop floor and projects from any PC in the network. Finally, an existing Python interface will be connected with the developed data base.

2 Fundamentals

This chapter gives a brief description of the basics of project management and digital media. IT supporting industrial processes represents an important key to competitive advantage. Production - the shop floor - should be seamlessly networked as an integrated overall system with corporate planning - the top floor. This is the only way to evaluate production on the basis of current production data and make it more productive. In order to achieve this, the production data must be connected to corporate planning or project management via a digital interface and the data must be available for further assessments. In addition, this supports the coordination of corporate goals and production key figures through close interaction between top and shop floor. The central prerequisite for this is production data integration. [5]

In most SMEs, the machinery has grown over decades and generally works with heterogeneous control systems. A frequent first task before using IT-supported processes is therefore to overcome these barriers. For this purpose, data suppliers and consumers can be connected via various production data integration tools. This applies to plant and machine controls as well as measuring devices or testing equipment. Secondly, it is possible to connect machine data via sensors and actuators and by means of fieldbus controllers. Thirdly, machine operators and maintenance staff can enter or correct data. Regardless of the technology, production data integration takes place in the three steps of providing data, reading in data via customised software and interpreting the data. In order to archive this, SMEs tend to use open source solutions for the described layers of data generation. [6]

In the context of this thesis, the creation of a digital project management tool is elaborated, which unifies the interaction of the data from the production as well as the storage of this data for all project members and facilitates their evaluation.

2.1 Project management

The main tasks of a project management are to assess, to plan, to enable and facilitate, conduct and coordinate, supervise and control and to document the activities of a project. In order to facilitate this for the project manager, it is necessary to be able to access the data of the product and to have it displayed in a daily updated and uniform manner. [7]

To store the production data, an interface with the production network and a database is necessary. Furthermore, a general summary of the activities of the individual machines and employees in relation to the projects is necessary. The provided tools and their data should support the three main questions of project management. [8] These are:

1. Where are we?
2. Where have we planned to be?
3. How can we come back on track?

The first point is aimed at the activities carried out in a project. This is achieved and displayed with the help of the machine data and the stored activities of the employees. The second point relates to the project work plan, which is limited by the completion date. This is supported by the project data and its completion date. The last point aims at achieving the objectives in case of unscheduled delays. This requires machine data such as maintenance and staff availability. [9]

The most important point in all of this, however, is the usability of the project management tool in order to generate project-, machine- and employee-related data on the one hand and to be able to retrieve this data in a comprehensible and user-friendly way on the other. For these purposes, the Chair of Metal Forming represents a typical SME in the metal processing industry.

2.2 Database software system

In connection with the question ("Where are we?") from section 2.1, all existing data sources must be brought together and stored. In order to operate an efficient and flexible project management, the search for the required data and its processing must be simple and fast. For this reason, it is necessary to network all data in a cluster of databases and to be able to retrieve them uniformly.

For international companies, it can be advantageous to separate these databases from each other in order to make the required data available everywhere. This is not feasible for SMEs because they usually operate regionally and do not have the necessary IT infrastructure. For this reason, open source applications are an essential advantage for these companies. [10]

The advantages of a server-based database system, apart from availability and processing speed, are extension and spatial independence, interactivity and, if used correctly, actuality.

2.3 Graphical user interface Environment

The point mentioned in section 2.1 “Where should we be” is made realised by the possibility of appointments. These need to be presented in a clear and concise manner to provide the best possible support. The proposed system is designed to be extensible and expandable to improve efficiency and user engagement in terms of usability and flexibility. In addition, it should be accessible on any device of the network without installation. Due to these constraints, a server-based software solution was chosen for the environment. This offers the possibility of almost unlimited extensions as well as the necessary security aspects to prevent access by unauthorised users.

An addition is that this system avoids a proprietary solution that offers only one possibility to connect machines, but with the respective capabilities further machines and databases may be linked.

2.4 IT Security

The required security settings include user identification as well as the individual input options within the tools. This ensures that projects can be entered and edited correctly and reliably. Appropriate measures must be taken to fulfil these conditions when the tool is created.

Open source software (OSS) means that it is possible for anyone to use and change all parts of the software. Therefore, OSS is more secure because vulnerabilities are usually visible to everyone. OSS gets more attention overall, which means more testing, more frequent bug fixing and better resilience. Thus, OSS solutions benefit from a level of security that most commercial vendors cannot match. The flexibility and adaptability of OSS means that different programs, in this case PHP, SQL and Python, can interact with each other to build a multilingual system across the required layers. [11]

2.5 Usability requirements

By integrating different OSS, a production network consisting of different sensors and performance meters can be linked together to manage and track projects. Active condition monitoring makes it easier to calculate the maintenance dates of the machines and move them to existing downtimes. This generates the basis for a predictive maintenance analysis algorithm that accesses the machine data and can adjust the maintenance dates of the machines based on the data in the projects. [12]

Due to the linking of shop floor and top floor, the collection of data in the previously mentioned layers can lead to a lack of acceptance and adoption for the tool by the employees. [13]

For this reason, a stakeholder analysis is carried out in order to involve all stakeholders in the development of the project management tool and to generate the required and desired data on the one hand and to present it correctly and accurately on the other.

3 Development and implementation

This section covers the points of the development and implementation of the overall project management tool (PMT:CMF). The first stage is a stakeholder analysis, the second a feasibility study and finally the development of a suitable solution for the connection of the machine data and project activities. Because a project can be blocked by stakeholders to such an extent that it fails, mostly due to a lack of awareness and necessity, the start point is the stakeholder analysis. [14]

In this process, all employees of the Chair of metal forming are interviewed and the data necessary for continuous processing is collected. The goal is to obtain a consistent and useful project management interface, suitable for all potential user.

This is followed by a feasibility study, which serves as the basis for decision-making on the software to be used. Subsequently, the results shown in the feasibility study are presented and implemented.

3.1 Stakeholder analysis

This sub-chapter describes the stakeholder analysis conducted. The necessary steps of the stakeholder analysis can be summarised in identifying and prioritizing the stakeholders. The team at the chair of metal forming represents the main stakeholder group. In addition to the chair's management, the administration, technicians and student workers were also interviewed in order to obtain a consistent and holistic result. [15]

The people who are highly influential and highly interested in the project, the so-called players, who need to be fully involved, are represented by the head and the assistants of the chair. They are the main beneficiary of the integrated machine and project data within the PMT:CMF and are in essential contacts to ensure that everything works. The high influential, less interested people, are stakeholders which need to be satisfied, but not so much that they get bored. This group is represented by the technicians. They have to enter their details into software, but are not interested in the technical details of the PMT:CMF and its contents. The next group are the people with low power and high interest who need to be adequately informed about the project and the outcome. This group delivers the most ideas of the content and it is important to ensure that no major problems arise with this group. This group is represented by the administration, as they act in a supporting role in the details. The last group of people are those with low influence and less interest. This group is represented by the

student workers at the chair of metal forming. They have to work with the tools and know how to use them in order to perform their tasks adequately. [15]

According to the interviews, the most important points for the project management tool are consistency, connectivity, efficient data structures, high level of detail and easy visualisation as well as no change of software for data input and presentation. The participants were divided into three groups after the interviews.

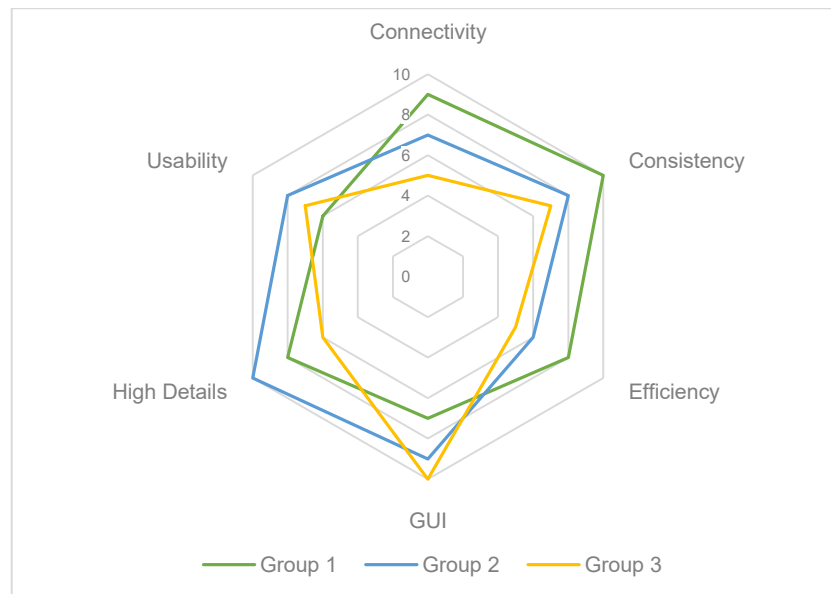


Figure 1: Result of the Stakeholder analysis

As shown in figure 1 the stakeholders are divided into three groups. Group one represents the Head of the Chair of metal forming and the assistants. The second group represents the administration staff and the technical staff and the third group represents the student workers at the chair. Furthermore, one subsection is dedicated to the feasibility study. The existing python scripts need to have a connection with the project management tool. In addition, it was decided not to use cost-intensive software for the tasks.

3.1.1 Layer of Project Management Tool

According to the stakeholder analysis, it has become clear that certain users should not be able to access all data of machines and projects. This resulted in the creation of roles within the tool. The roles are assigned when the users are signed in and can only be changed by someone with an administrator role. The roles/views are applied immediately after logging into the tool. The first difference is the navigation menu, which is displayed for the respective users and changed according to their role and function. A further difference is that non administration user can only enter or change activities for themselves. Project costs and budgets are only

visible in the detailed project view for the responsible project manager, for every other user a percentage value is shown. Furthermore, a user cannot view any personal data other than himself/herself. Based on the stakeholder analysis, the following roles and corresponding rights have been identified:

	ADMIN	PROFESSOR	ASSISTANT	TECHNICIAN	PHD STUDENT	STUDENT
SEE FINANCE	YES	YES	YES	NO	NO	NO
SEE MACHINE DETAILS	YES	YES	YES	YES	YES	NO
SEE PROJECT DETAILS	YES	YES	YES	NO	OWN	NO
CREATE PROJECT	YES	YES	YES	NO	NO	NO
CREATE MACHINE	YES	YES	NO	NO	NO	NO
CREATE USER	YES	YES	NO	NO	NO	NO
UPDATE MACHINE	YES	YES	YES	YES	NO	NO
UPDATE PROJECT	YES	YES	YES	OWN	OWN	NO
UPDATE ACTIVITIES	FOR ALL	FOR ALL	OWN	OWN	OWN	OWN
CREATE ACTIVITIES	FOR ALL	FOR ALL	YES	YES	YES	YES

Figure 2 Roles of the project management tool

As shown in figure 2, the role of professor and administrator can view all possible entries and also change them. In addition, these roles can also create machines, projects and employees in the system. Finally, the holders of these roles can also create activities for all. For ease of reading, the role of professor is equated with that of administrator in the thesis and is not enumerated further.

The role of the assistant has the same authorities as the role of the professor, except for the creation of users and machines. However, an assistant cannot create or alter activities for all employees.

The technician role allows editing of the machines created in the system. This includes not only the entries but also the detailed views of the machines.

The PhD student role can be entered as the main responsible for a project but cannot create one by himself.

The student role can only create its own activities and cannot access machines or projects.

In this thesis, a distinction is only made between admin and user for the sake of easier readability when the positions are the same. The rights of the special roles are explained with those of the admin.

3.2 Feasibility study

In addition to the results of the stakeholder analysis, further points were added. On the one hand, this concerns the expandability of a project management tool with regard to further calculations and details, and on the other hand, the storage of the machine data must be guaranteed. In order to make the data storage and the graphical processing as cost-effective as possible, open source software was used for the creation. In addition, derived from the stakeholder analysis, the software must fulfil the following conditions:

- Executable on a server
- Manual and automated data storage up to two years
- Security against unauthorised entry
- Possibility to link with Python
- Expandable
- Attractive design
- In-house

Because the software used should be low cost, open source software was chosen for the creation of the systems and because of the versatile tasks, the choice fell on 2 types of software. On the one hand, a database management system with a connection to the existing Python scripts and on the other hand, a system for displaying the data with safety measurements and pleasing presentation. For this purpose, the databases InfluxDB and MySQL were compared as database management systems for the project, machine and employee data. For the presentation and manipulation of the relevant user data, the systems of PHP and JavaScript were compared.

Python is often used as a support language for software developers, testing, controlling and in many other ways. Also at the chair of metal forming Python is used for subroutines and support activities. For this reason, it is mandatory for the chosen software to have an easy connector to Python.

To cover all the above points, the two databases InfluxDB and MySQL were compared. The database management systems must manage the activity data of the machines and employees in addition to the project, machine and employee data. For the display and manipulation of the relevant user data, the systems of PHP and JavaScript were compared. The main focus of both systems is on their interoperability and extensibility.

3.2.1 Database management systems

The InfluxDB time series platform supports the user to build software without the need to provision infrastructure and manage clusters. The InfluxDB Cloud gives users the ability to start collecting metrics serverless. It provides the ability to create a dashboard for real-time IoT monitoring and analytics display. [16]

InfluxDB includes a paid plan as well as a free plan and it is designed for quick queries and monitoring of active machines. However, with its free version, it is not designed to store data for a long period of time and due to the cloud storage of the data, sensitive budgeting and machine cost data as well as employee contract data would leave the university's servers. [16]

A SQL database is a storage and management system that can store, sort, select and retrieve large amounts of information. Instead of putting all the data in a single big table, the SQL database stores its data in separate tables. That sort of database is called a relational database. The database structures are organized like files, which are optimized for operation speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. The SQL database consists of two parts, on the one hand there are the structured tables with primary keys and information and on the other hand the database management system, the interface with which the database tables can be administered. The data is kept on the chair's own server, which means that all data remains the property of the chair and cannot be interfered with by third parties. [17]

Due to the possibility of an own server on which the database runs and the data is stored, and in addition that data can be stored for more than two years, the SQL database management system has been chosen for data storage. By choosing SQL, the administration program *phpMyAdmin* has been selected. It provides support for a vast number of operations with MySQL and common operations can be done directly from the user interface. Another Advantage is that there are predefined interfaces with Python, which can be used to interact with the database.

3.2.2 Content display

JavaScript is by definition a programming language in the broader sense and a scripting language in the narrower sense, as the name suggests. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. The script should be contained in or referenced by an HTML document, so that the code can be rendered by the browser. This means that a web page can contain programs that interact with the user and controlling the browser. JavaScript depends on the performance of the executing computer and the bandwidth. And all data inserted in the script is possible to be seen by the user. [18]

PHP code is executed by a so-called interpreter on the server and sent back to the client as HTML output, in contrast to client-side programming languages such as JavaScript, which are executed on the side of the web browser (the client). There are three main areas where PHP scripts are used. The first area is server-side scripting. To use PHP, a PHP parser, a network server and a browser are required. The server must be set up with a PHP installation and the browser is needed to view the programmed PHP pages. It is possible to run PHP scripts with only a parser, meaning without a browser or server. These applications are often used for simple text processing tasks. These are called command line scripts because they are executed and handled directly with the operating system. In principle, it is possible to run a stand-alone application with PHP, but there are more sophisticated languages that allow this to be done better. One advantage of PHP applications is that they can be run on multiple platforms. [19]

A major advantage is that for the user the PHP code is invisible, because of the fact that it runs on the server. Also the PHP programmed pages' don't depend on the performance of the executing computer but on the bandwidth at the server.

Since data security is a significant aspect, the scripting language PHP is used for the development of the project management tool. For this case, an interpreter has been installed on the server.

Another advantage of this solution is that the chair's homepage already works with PHP, making it easier to integrate the PMT:CMF.

3.3 MySQL database

This chapter describes the settings of the MySQL database tables and their primary key relations. The database with the name *projectmanagement* is set up with the tool *phpMyAdmin*, which is a OSS, written in PHP and intended to handle the administration of the SQL database. The database itself is in *utf8mb4_general_ci* format, where the performance of the *utf8mb4_general_ci* shows that it's faster at comparing and sorting. A minor disadvantage of the *utf8mb4_general_ci* is that it cannot implement all Unicode sorting rules, which leads to undesired sorting in some situations, e.g. when using certain languages or characters. As far as Latin (i.e. "European") languages are concerned, there is not much difference between the Unicode collation and the simplified '*utf8mb4_general_ci*' collation in MySQL. [20]

The database at the PMT:CMF consists of interlinked tables which are divided in the topics of usability. The SQL database stores and displays the activities of the employees, the information about projects and machines, the sensor values of the machines and includes auxiliary tables for interconnection.

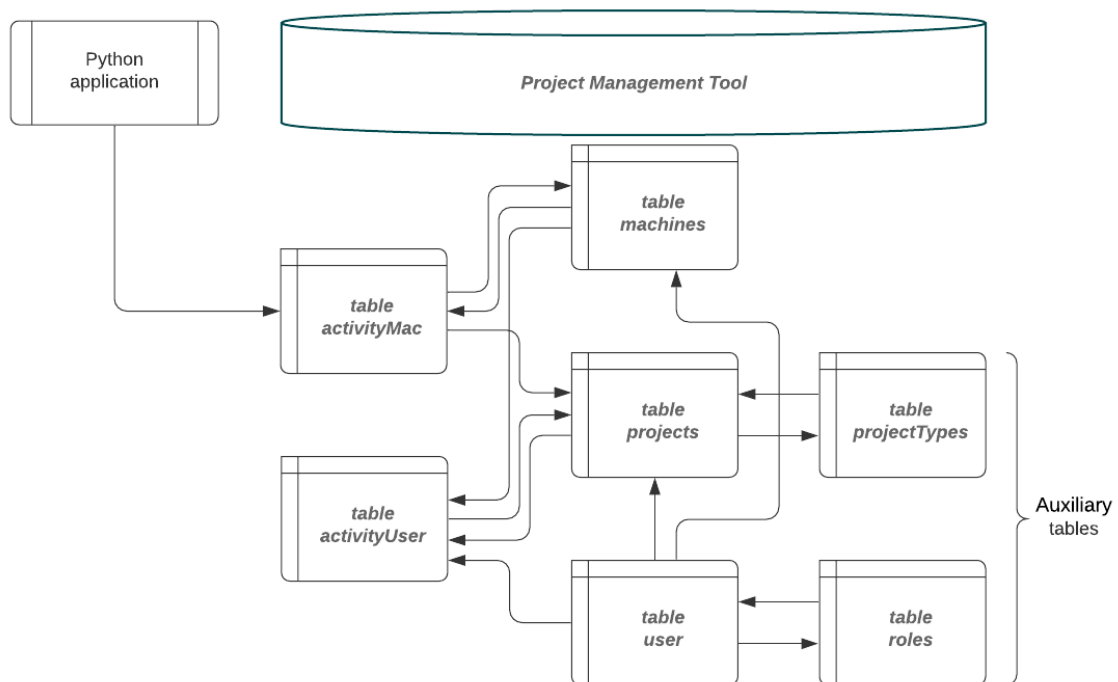


Figure 3 Flowchart of the connections of the tables

As displayed in figure 3 the tables are linked via their primary keys to reduce the probability of errors and to obtain a better response when selecting the tables.

3.3.1 Data Tables

In this subsection the SQL tables are described in detail. For better manageability, the tables were divided into their data areas and named accordingly. To address the structure of the tables, each table has its own auto incrementing id column. This column is used when linking attributes of one table to another one. The advantage of these unique identifiers is that entries can be changed in one place and are valid for all linked connections. The included tables with their detailed attributes are listed below.

table	Type	Collation	size
<input type="checkbox"/> activitymac	InnoDB	utf8mb4_general_ci	16.0 KiB
<input type="checkbox"/> activityuser	InnoDB	utf8mb4_general_ci	16.0 KiB
<input type="checkbox"/> machines	InnoDB	utf8mb4_general_ci	16.0 KiB
<input type="checkbox"/> projects	InnoDB	utf8mb4_general_ci	16.0 KiB
<input type="checkbox"/> projecttypes	InnoDB	utf8mb4_general_ci	16.0 KiB
<input type="checkbox"/> roles	InnoDB	utf8mb4_general_ci	16.0 KiB
<input type="checkbox"/> user	InnoDB	utf8mb4_general_ci	16.0 KiB
7 tables	InnoDB	utf8mb4_general_ci	112,0 KiB

Figure 4 Tables of the project management tool

Figure 4 shows the tables in the *phpMyAdmin* tool, with the displayed tables representing the main structure of the PMT:CMF. The empty database with all created tables and the inner structure has a volume of approximately 112 kB.

3.3.1.1 Tables *activitymac* and *activityuser*

The first two tables are the tables *activityuser* and *activitymac*. These two tables are the work tables of the PMT:CMF. All project-related activities of the employees and machines are stored in these two tables. The table *activitymac* obtains its data through a script written in Python. This data is generated by the sensors of the machines and transmitted to the database.

Name	Type	Collation	Attributes	Null	Default	Comments	Extra
actMID 🔑	int(32)			No	None		AUTO_INCREMENT
actMMachine	int(4)			No	None		
actMSensor	int(8)			No	None		
actMTime	datetime(4)			No	None		
actMValue	float			No	None		

Figure 5 Attributes of the table *activitymac*

As shown in figure 5, *actMID* is the primary key of the table and is automatically incremented when a new entry is added. The key can reach a 32-digit integer value.

The machine is set in the next column, only the *actMMachine* is stored here as an integer value and has a limit of 4-digits. This column is the connection to the Machines of the chair of metal forming.

To register the date and time of a sensor input, the column *actMTime* has been created with the attribute *datetime(4)*. It consists of the date and the time.

The final column of the table is the *actMValue* column. Here the actual sensor value is stored.

Employees enter data directly into the table *activityuser* via a form on the PHP page. The table *activitymac* obtains its data through an application written in Python. This data is generated by the sensors of the machines and transmitted to the database.


Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<i>actUsID</i> 	int(32)			No	None		AUTO_INCREMENT
<i>actUsName</i>	varchar(256)	utf8mb4_general_ci		No	None		
<i>actUsUser</i>	int(8)			No	None		
<i>actUsProject</i>	int(8)			No	None		
<i>actUsMachine</i>	int(8)			No	None		
<i>actUsStartDate</i>	datetime			No	None		
<i>actUsEndDate</i>	datetime			No	None		
<i>actUsNoUser</i>	tinyint(1)			No	None		
<i>actUsLog</i>	datetime			Yes	NULL		

Figure 6 Attributes of the table *activityuser*

Figure 6 shows the corresponding nine columns of the *activityuser* table. Beginning with the *actUsID*, this is the primary key of the table and is automatically incremented when a new entry is added. The key can reach up to a 32-digit integer value.

The naming of an activity takes place in the column *actUsName*. This is of the type varchar with a possible size of 256 digits.

The next three columns, *actUsUser*, *actUSProject* and *actUsMachine*, have a limit of 8-digits integer. In this columns the corresponding IDs are stored in the table.

To register the start of an activity, the column *actUsStartDate* has been created with the attribute datetime. It consists of the start time and date of the activity.

Similar to the column described above, the end time and date of an activity is registered in the column *actUsEndDate*, which has the attribute datetime. It consists of the start time and date of the activity.

To input an activity without an employee at the machine, e.g. if an oven heats up components slowly over a whole day, the Boolean attribute in the *actUsNoUser* column is set to true. When a new activity is created, this column is set to false unless otherwise entered, or to zero (not NULL).

The final column of the table is the *actUsLog* column. The attribute of this column is datetime and is set to the current date and time when the activity is created.

3.3.1.2 Table machines

This table (figure 7) is used to store all relevant data about the machines, including the machine name, the machine type and the date of the last maintenance. Data for the Asset valuation are also included as well as the possibility to take the machine out of service.

Name	Type	Collation	Attributes	Null	Default	Comments	Extra
macID 🗝️	int(10)			No	None		AUTO_INCREMENT
macNumber	text	utf8mb4_general_ci		No	None		
macName	varchar(128)	utf8mb4_general_ci		No	None		
macNotes	text	utf8mb4_general_ci		Yes	'1'		
macType	varchar(128)	utf8mb4_general_ci		No	None		
macPurchaseDate	date			No	None		
macPurchaseCosts	double			No	None		
macHourRate	float			No	None		
macUtilisation	float			No	None		
macMaintInterval	float			No	None		
macMaintLast	date			No	None		
macAssetValue	double			No	None		
macInactive	tinyint(1)			No	0		
macLastUpdate	datetime			No	None		

Figure 7 Attributes of the table *machines*

Starting with the macID, which is the primary key of the table. When a new entry is added, it is automatically incremented. The key can reach an 8-digit integer value.

According to the stakeholder analysis the machines of the chair of metal forming have a number in the main ERP system. For this reason, the column *macNumber* contains this data with the type text, so that the numbers can be alphanumeric.

The *macName* column is for the internal name of the machine. The attribute of the column is a varchar with a 128-digit entry for alphanumeric entries. The main purpose is to achieve general readability.

The *macNotes* column is an additional text field for general information in text form. The entries are saved as text type to allow alphanumeric input.

The *macType* column refers to the type of the machine. The main purpose is to achieve general readability and administrating the machines.

To register the purchase date of the machine, the column *macPurchaseDate* with the datatype date is used.

To be able to calculate the yearly costs of the machine the purchase costs have to be noted. This is done in the column *macPurchaseCost* with the data type double.

The running costs of a machine is stored in the column *macHourRate*. The data attribute used for this column is float.

The column *macUtilisation* with the data type float represents the duration of the machine useage in years.

Maintenance is an essential part of the project management tool. The interval between maintenance dates is entered in the *macMaintInterval* column as the data type float.

In addition to the maintenance interval, the last maintenance is also entered; this is achieved in the *macMaintLast* column with the data type date.

the current book value of the machine is stored in the *macAssetValue* column. This is calculated with the utilization time and the purchase date.

To remove an active machine, the boolean variable is set to true in the *macInactive* column. The column is set to the value false by default, respectively zero (not *NULL*) when a new machine is created.

The final column of the table is the *macLastUpdate* column. The attribute of this column is from the data type datetime and is set by the *actUsEndDate* variable when a new activity is created.

3.3.1.3 Table projects and projecttypes

The next tables are the table *projects* (figure 8) and its auxiliary table *projecttypes* (figure 9). Thus, all relevant data on the projects are stored, including the project number, type and relevant data of the project. Also the budget and running costs for a project are listed here. The projects table contains the following twelve columns.


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	proID 	int(8)		No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	proNumber	text	utf8mb4_general_ci	No	None		
<input type="checkbox"/>	3	proName	text	utf8mb4_general_ci	No	None		
<input type="checkbox"/>	4	proType	int(8)		No	None		
<input type="checkbox"/>	5	proNote	text	utf8mb4_general_ci	No	None		
<input type="checkbox"/>	6	proContact	int(8)		No	None		
<input type="checkbox"/>	7	proBudget	double		No	None		
<input type="checkbox"/>	8	proCosts	float		No	None		
<input type="checkbox"/>	9	proStartDate	date		No	None		
<input type="checkbox"/>	10	proEndDate	date		No	None		
<input type="checkbox"/>	11	proFinished	tinyint(1)		No	0		
<input type="checkbox"/>	12	proLastUpdate	datetime		No	None		

Figure 8 Attributes of the table *projects*

When a new entry is made, the *proID* column is automatically increased. The column has been selected as the primary key. The attribute *auto_increment* generates a unique number automatically when a new record is inserted into the table. To ensure that the table does not enter a data overflow, an 8-digit integer was chosen as the primary key.

The project number is entered in the corresponding column *proNumber*. This is the external identifier for the chair of metal forming. Due to the fact that the number may be alphanumeric, the type char with a length of 16 characters. Following two project number examples from the chair of metal forming, one shows an alphanumeric entry and the other one the numerical entry:

- P056-F-56-07
- 105600

The attribute of the field *proName* is a text field which allows alphanumeric entries this allows the column to serve as the internal name of the project. The main purpose is to achieve general readability.

The *proType* column refers to the type of the project. Only an integer value is stored here, as the name of the type is stored in the support table *projecttype*. This entry affects the costs of the project and results in a column change in the user table. An example for this connection is shown in the PHP section.

The *proNote* column is an additional text field for general information in text form. For example, external contact persons can be entered here or whether a partial payment has already been initiated. The entries are saved as text to allow alphanumeric input.

To ensure that a project has an internal contact person who is also responsible for the project, the column *proContact* was added. The type of this column is integer with a length of 4-digits. The integer value entered here corresponds to the ID value of the user table.

The project budget is stored in the column *proBudget*. The attribute used for this column is double to allow the entry of large numbers and comma values.

The column *proCosts* contains the costs incurred after completion of the projects. The attribute used is double to break down the necessary information.

To register the start date of the project, the column *proStartDate* has been created with a date attribute. The input is set with the creation of a new project.

Similar to the column described above, the delivery date of the project is registered in the column *proEndDate*, which has the attribute date.

To indicate a completed project, the boolean attribute is set to true in the *proFinished* column. The column is set to the value false by default, respectively zero (not *NULL*) when a new project is created.

The final column of the table is the *proLastUpdate* column. The attribute of this column is datetime and is set to the current date and time when the project is created. This column is updated when a new activity is entered by the staff.

The table *projects* is supported by the auxiliary table *projecttypes* for the different project types. This auxiliary table is linked to the cost calculation of the employees working in different project types.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	typID	int(4)			No	None		AUTO_INCREMENT
2	typName	text	utf8mb4_general_ci		No	None		

Figure 9 Attributes of the table *projecttypes*

The figure above shows the attributes of the table *projecttypes*. This table is used for the different project types and the cost calculation. The table *projecttype* consists of the two columns *typID* and *typName*.

The column *typID* represents the primary key with automatic incrementation. The attribute *auto_increment* generates a unique number automatically when a new record is inserted into the table.

The name of the type is saved in the column *typName*. The attribute of the field is a text field which allows alphanumeric entries. The main purpose is to achieve general readability.

3.3.1.4 Tables user and roles

The next tables are the table *users* and its auxiliary table *roles*. Thus are used to store all relevant data about the employees. Including name, the role and the relevant dates for the employee. The table *roles* is derived directly from the analysis of the different layers previously carried out. The table *users* contains the following ten columns.


Name	Type	Collation	Attributes	Null	Default	Comments	Extra
usID 	int(5)			No	None		AUTO_INCREMENT
usName	varchar(64)	utf8mb4_general_ci		No	None		
usLogin	varchar(32)	utf8mb4_general_ci		No	None		
usContract	date			No	None		
usRole	int(5)			No	None		
usFee1	float			No	None		
usFee2	float			No	None		
usFee3	float			No	None		
usPwd	varchar(128)	utf8mb4_general_ci		No	None		
usLastLogin	date			No	None		

Figure 10 Attributes of the table *users*

Figure 10 describes the attributes of the users table. The column *usID*, with an 8-digit integer type, describes the primary key with an automatic increment. The auto increment generates a unique number automatically when a new record is created.

The name of the employee is stored in the column *usName* with the attribute of the type *varchar* with a length of 64 digits. This makes it possible to store alphanumeric entries, which is intended to improve the general readability.

The main purpose of the column *usLogin* is to enter the PHP pages and identify the user. The attribute of the field is of type *varchar* with a length of 32 digits. The employee login is stored in the column.

The column *usContract* has two purposes. The first one is to display the end of the contract of an employee and the other one is to reset the user password to an unknown one if the contract is ended so that the entries of user still exist when the user left the chair of metal forming.

Based on the stakeholder analysis, the roles for the users were created and inserted in the column *usRole* as an integer with 8-digits. The role is set by the administration when the user is created and it is for the different views inside the PHP project management tool.

As mentioned before the project type refers to different fees of the employees so the three columns *usFee* one to three contain these fees which are stored as a set of float values.

To secure the SQL entries and the PMT:CMF on the PHP pages, the user must enter a password. This password is set when a new user is created and is hashed by the PHP code and stored in the column *usPwD* with 128-digit varchar type.

The final column of the user table is the *usLastLogin* column. The attribute of this column is datetime and is set to the current date and time when a new user is created. This column is updated when a new activity is entered by the staff.

The auxiliary table *roles* supports the table *users* for the different views in the project management tool. This auxiliary table is linked to the security features of the PHP system.


Name	Type	Collation	Attributes	Null	Default	Comments	Extra
rolID 	int(3)			No	None		AUTO_INCREMENT
rolName	varchar(128)	utf8mb4_general_ci		No	None		

Figure 11 Attributes of the table *roles*

As shown in Figure 11 describes the attributes of the roles table. The column *rolID*, with a 3-digit integer type, describes the primary key with an automatic increment. This integer is used for the access of different pages of the project management tool.

The column *rolName* represents the name of the user role, which has the attribute varchar for alphanumeric entries. The main purpose of this field is to achieve general readability.

3.4 PHP

This chapter describes the graphical user interface and the processes involved within and behind the PHP layers. The advantage of using PHP is that pages can be accessed from any device connected to the network because the PHP files are stored at a server of the chair of metal forming. In order to meet the requirements, the HMI was built with PHP version 8.0.2 for a stable performance and security reasons.

This chapter contains two larger blocks, the first one describing the interaction of the methods and representations used with a flowchart and class diagrams. Secondly, in addition to the HMI, the code snippets and their application are described.

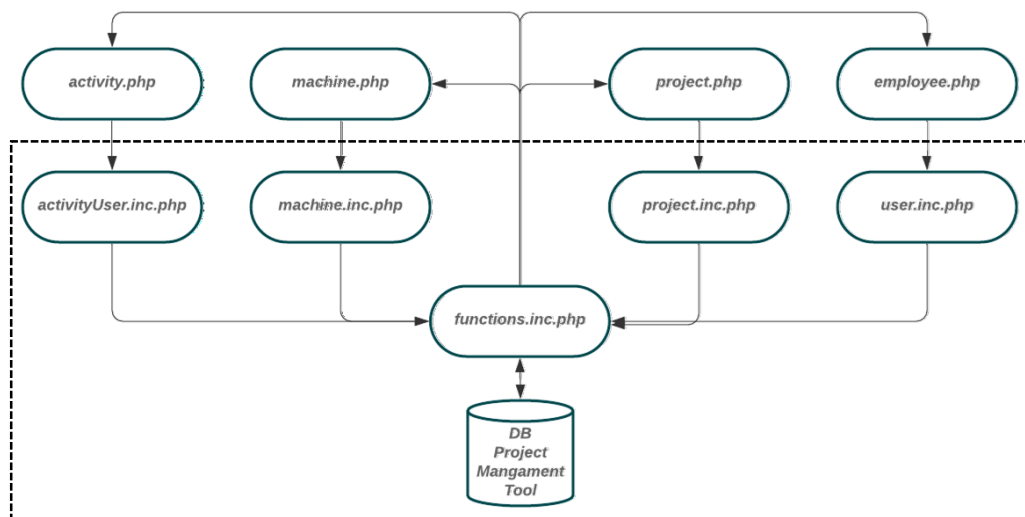


Figure 12 Flowchart of the PHP pages invisible for the user

Figure 12 shows the hidden pages for the user, which are connected with the function file and the database. The pages are divided into three types, of which the contents of two types are not directly visible to the user and the third type is the graphical user interface. The first invisible category is for the Cascading Style Sheets (CSS) where all the formats and styles for the display are stored. These files are stored in an own folder to separate them from the main pages. The second group consists of transfer pages that collect the data entered in the PHP forms, process it and transfer it to a *function.inc.php* file. All of this files contain only individual if statements and no visible parameters, so that they and the results are not displayed by the server to the user. As mentioned above each of the categories (Activities, Machines, Projects, Employees) has a transfer file, these are marked with the extension *.inc*.

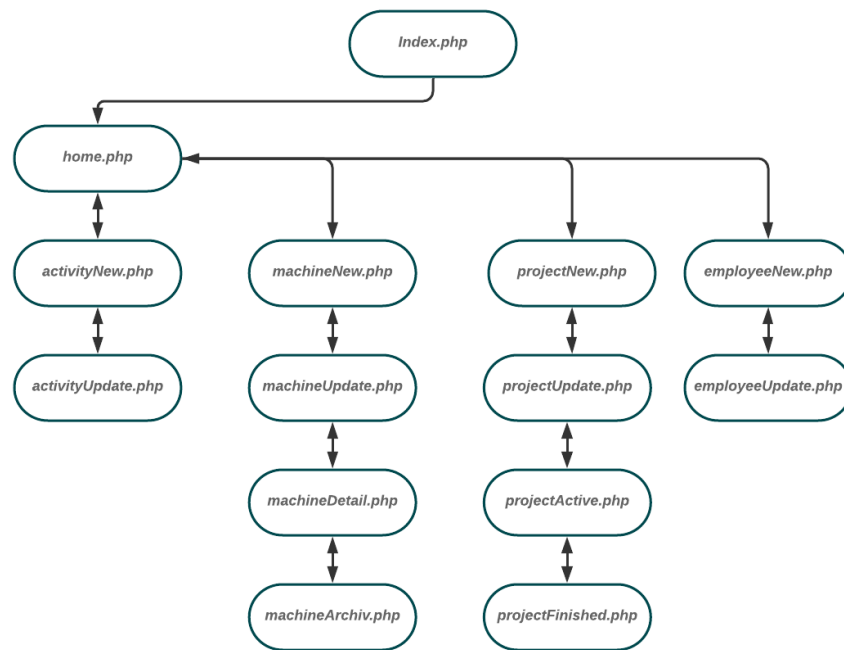


Figure 13 Flowchart of the php pages visible for the user

Due to the results of the stakeholder analysis, the division of the pages was made as shown in the figure 13. It shows the visible pages for all users. This third type follows two aspects, on the one hand the pages of this type contain instructions on how the retrieved data is displayed to the user in the browser. On the other hand, these pages contain functions which transferred to the *functions.inc.php* file to receive data from the database or to enter data to create or change it in the database.

3.4.1 Overview

All of the PHP pages are divided into four areas. Three of them, header, nav and footer, are static through the session of a user and the fourth is the main content area (body) where all information is displayed and entered.

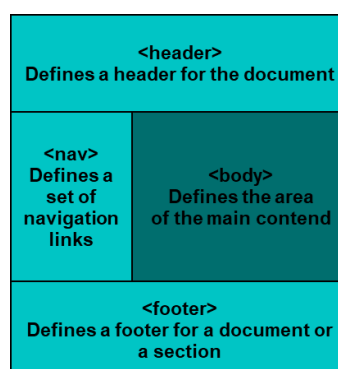


Figure 14 General overview of the PHP pages

Figure 14 displays the general sections of the PHP pages. The three static sections header, navigation and footer and the area for the content of the page named body. Anticipating, the navigation area is different for the user's roles. The header area consists of two sub files one which loads the whole format via cascading style sheet files and the other one displays the header of the pages. The setup file defines the loading order of the CSS files and setup.

```

<?php
    session_start();
    checkUser();
    $fullName = $_SESSION['usName'];
?>
<header>
    <div class="bluebox"></div>
    <div class="whitebox"><div class="ifutxt">CHAIR of METAL FORMING</div></div>
iv>
    <div class="alignleft" style="top: 25px;">
        </img>
    </div>
    <div class="alignleft" style="top: 90px;">
        <p> <?php greetings($fullName); ?></p>
    </div>
    <div class="alignleft" style="top: 65px;">
        <div class="logoutbox">
            <form action="includes/logout.inc.php" method="post">
                <button type="submit" name="logout-submit">
                    
                </button>
            </form>
        </div>
    </div>
</header>

```

Code snippet 1 Static header functions

Code snippet 1 displays the header section and its functions for identifying the user as well as the php function *greetings(\$fullName)*. The function *session_start()* is a built in PHP function which creates a new session or continues an existing one. PHP calls the open and read routines of the session memory function. This is used to create super global variables, which are available for each page and function. These variables are used for security reasons and for the different rights and views of the user.

```

function checkUser(){
    //Function to chek if user is logged in
    if(!isset($_SESSION['usID'])){
        header("location: ../PMTtoCMF/index.php?error=NoLogin");
        exit();
    }
}

```

Code snippet 2 Function *checkUser(...)*

The function *checkUser()* (code snippet 2) secures the page and, if the user didn't login properly, sends the user back to the login screen.

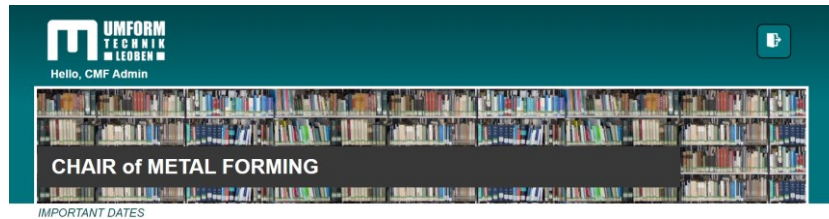


Figure 15 Header section of the project management tool

The header (figure 16) consists of the metal forming logo and a button to log out, as well as a time-dependent personalised greeting. Furthermore, the page title is shown with a background image of the metal forming library and the name of the current page.

The navigation bar is a user interface element within the PHP page that serves as a link to other areas of the tool. As already mentioned above, the navigation bar is part of the main template of the website, which means that it is displayed on most, if not all, pages. This means that regardless of which page is displayed, a user can use the navigation bar to visit other sections of the website. The navigation menu is divided into the categories Activities, Projects, Machines and Employees. The permitted views are shown from the user roles created in the SQL database.

```
<td valign="top" class="left">
  <div class="nav">
    <ul>
      <?php
        $access = checkAccess($conn, $_SESSION['usID']);
        $project = checkContact($conn, $_SESSION['usID']);
        $machine = checkMachine($conn, $_SESSION['usID']);
        $activity = checkActivity($conn, $_SESSION['usID']);
        if($access){
          echo '
            <li class="col1"><a href="home.php">IMPORTANT DATES</li>
            <li class="col2"><a href="activityNew.php">NEW ACTIVITY</a></li>
            <li class="col2"><a href="activityUpdate.php">UPDATE ACTIVITY</a></li>
            <li class="col3"></li>
            ...
          '
        }
      </?php
    </ul>
  </div>
</td>
```

Code snippet 3 Navigation menu admin view

The code snippet 3 shows the functions for retrieving the required variables *\$access*, *\$project*, *\$machine* and *\$activity*. These variables are used to map the individual areas for the user. All four functions use the variable *\$conn* for the active database connection and the as second

variable the user id from the super global variable of the logged in user. The logic of the functions for recording these variables is described below using the function *checkAccess(...)* as an example. (code snippet 4)

```
function checkAccess($conn, $user){
    //Function for access control on several pages
    $access = inputExists($conn, $user, $user, 'employee');
    $sql = "SELECT proContact
           FROM projects
           WHERE proContact = ".$access["usID"]."";
    $results=mysqli_query($conn, $sql);
    // check role of employee
    if($access["usRole"] == 2 || $access["usRole"] == 3 || $access["usRole"] == 101){
        return true;
    } else {
        return false;
    }
}
```

Code snippet 4 Function *checkAccess(...)*

The mentioned functions themselves call another key function of the tool, the function *inputExists(...)*. This function accesses the database and returns an associative array that corresponds to the fetched row or null if there are no more rows.

In order to specify this, the function *checkAccess(\$conn, \$_SESSION['usID'])* searches the role of the logged-in user to see if he or she has admin rights. The function *checkContact(\$conn, \$_SESSION['usID'])* checks the projects for the ID passed as the project contact. The function *checkMachine(\$conn, \$_SESSION['usID'])* checks the role of the user whether he has the right to change and maintain machines in the tool.

Finally, the function *checkActivity(\$conn, \$_SESSION['usID'])* checks the existence of executed activities of the user. All four functions are used to restrict the user's view in the event of a negative result, thus enabling selective user control through the pages.

IMPORTANT DATES	
NEW ACTIVITY	
UPDATE ACTIVITY	
PROJECTS	
NEW PROJECT	
UPDATE PROJECT	
ACTIVE PROJECTS	
FINISHED PROJECTS	
MACHINES	
NEW MACHINE	
UPDATE MACHINE	
DETAIL MACHINE	
MACHINE ARCHIVE	
EMPLOYEES	
NEW EMPLOYEE	
UPDATE EMPLOYEE	
	IMPORTANT DATES
	NEW ACTIVITY
	PROJECTS
	UPDATE PROJECT
	ACTIVE PROJECTS
	FINISHED PROJECTS
	MACHINES
	EMPLOYEES
	UPDATE EMPLOYEE

Figure 16 Navigation menu with different roles

In figure 16 above, two possible views of the navigation menu are shown. On the left side, the whole navigation menu is displayed. This is the view of the administration staff, so that they can alter every project and machine entry. On the right side the navigation menu shows the view of a project contact. This employee can see the project he is working for as well as the details of active and finished projects.

The footer, which is located at the bottom of each page, contains information about the current version of the tool as well as the mission statement of the chair of metal forming. In addition, the database connection is closed using the php internal function `mysqli_close($conn)`.

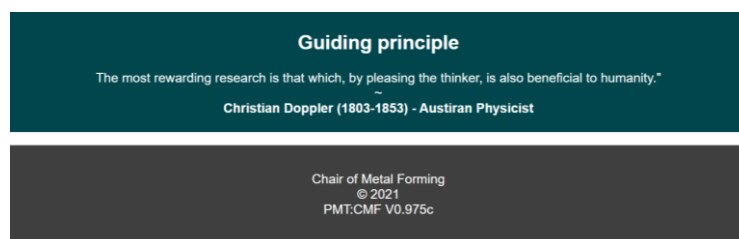


Figure 17 Exemplary footer of the project management tool

Figure 17 shows the footer of the PMT:CMF. In addition to the basic principle of the chair, the current version number of the tool is displayed.

3.4.2 index.php

The *index.php* is a PHP file that represents the entry point of a website or application. It is a file used for templates that contain a mixture of codes that are delivered as PHP code. The index page is the first security barrier of the tool. To log in to the tool, a login name and password must be entered. Both must be created in advance by an administrator in order to gain access. If this has been done so far, the function *loginUser(\$conn, \$username, \$pwd)* is used to initiate the access. This function uses the function *inputExists(\$conn, \$username, \$username, 'employee')* to check whether the user is stored in the database and, if available, outputs the relevant associated array. The PHP internal function *password_verify(\$pwd, \$input["usPwd"])* then checks the validity of the entered password. The PHP internal encryption is used to make the password not visible. There are several possibilities how the login can end as shown below:

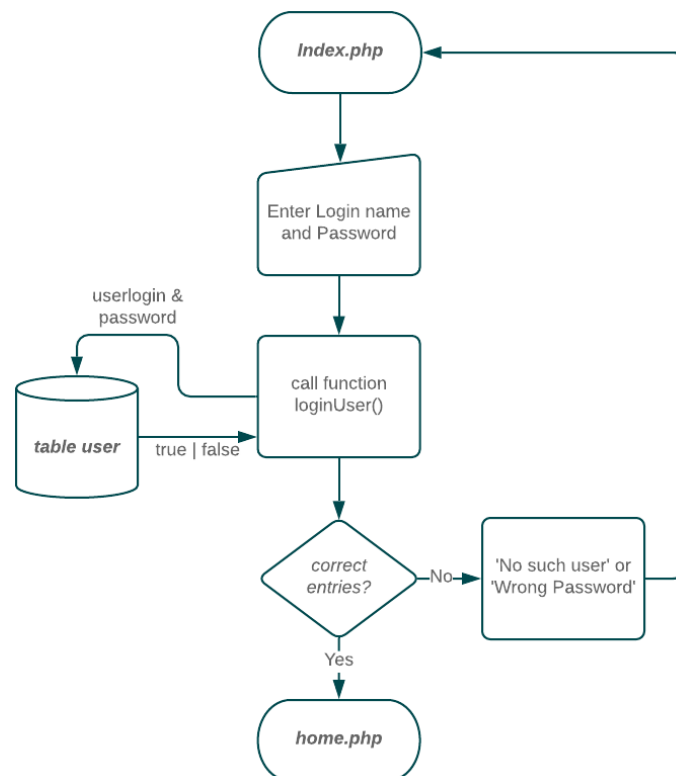


Figure 18 Flowchart Login possibilities

As shown in the flowchart in figure 18, the user will not be redirected if the wrong name or password is entered. In addition, the user is made aware of the incorrect procedure. When the user enters his correct login and password, the attributes *usID*, *usName* and *usRole* from the database are saved as global variables and his last login date is transferred to the database. The user is then redirected to the main overview of the project management tool.

```

function loginUser($conn, $username, $pwd){
    $input = inputExists($conn, $username, $username, 'employee');
    if($input === false){
        header("location: ../index.php?error=NoSuchUser");
        exit();
    }
    // verify Pwd
    $pwdCheck = password_verify($pwd, $input["usPwd"]);
    if($pwdCheck == false){
        header("location: ../index.php?error=WrongPwd");
        exit();
    } else if ($pwdCheck == true){
        //Open session for roles
        session_start();
        $userID = $input["usID"];
        $_SESSION['usID'] = $input["usID"];
        $_SESSION['usLogin'] = $input["usLogin"];
        $_SESSION['usName'] = $input["usName"];
        $_SESSION['usRole'] = $input["usRole"];
        $lastLog = "UPDATE user
                    SET usLastLogin = '".date("Y-m-d")."'
                    WHERE usID = ".$userID.";";
        mysqli_query($conn, $lastLog);
        inactiveUser($conn);
        //Close connection
        mysqli_close($conn);
        header("location: ../home.php");
    } else {
        header("location: ../index.php?error=NoInput");
        exit();
    }
}
}

```

Code snippet 5 Function *loginUser(...)*

Code snippet 5 shows the function *loginUser(\$conn, \$username, \$pwd)*. The input variables consist of the connection to the Database, the entered username and the password of the user. First, the function verifies with the already explained function *inputExists(...)* whether the user name exists in the database. If there is a similar username the function returns the row with the user information. The internal function *password_verify(\$pwd, \$input["usPwd"])* checks if the passwords are equal. Afterwards key variables like *usID*, *usLogin*, *usName* and *usRole* are stored as session super global variables and the user is transmitted to the page *home.php*.

3.4.3 home.php

The page *home.php* represents the global overview of the project management tool. The content section contains the most important dates about the projects, the machines and especially for administrators the contract end dates of the employees. This data is created directly from the SQL database with the help of display functions and displayed on the *home.php* page.

```
function displayProjects($conn, $activ){
    //Function to display the projects

    $access = checkAccess($conn, $_SESSION['usID']);
    $sql = "SELECT proID, proNumber, proName, typName, usName,
        proBudget, proStartDate, proEndDate, proLastUpdate
    FROM projects
    INNER JOIN user ON projects.proContact = user.usID
    INNER JOIN projecttypes ON projects.proType = projecttypes.typID
    WHERE proFinished = ".$activ."
    ORDER BY proEndDate DESC
    LIMIT 15;";
    $result = mysqli_query($conn, $sql);
    if (mysqli_num_rows($result) >0)
```

Code snippet 6 Function *displayProjects(...)*

Code snippet 6 illustrates how the data is selected by the function *displayProjects(\$conn, \$activ)* for projects. The variable *\$active* stands here for the search if the displayed items are actual running projects or already finished projects. Also seen here are the inner joins of the SQL table. This is necessary to improve the overall readability of the output data.

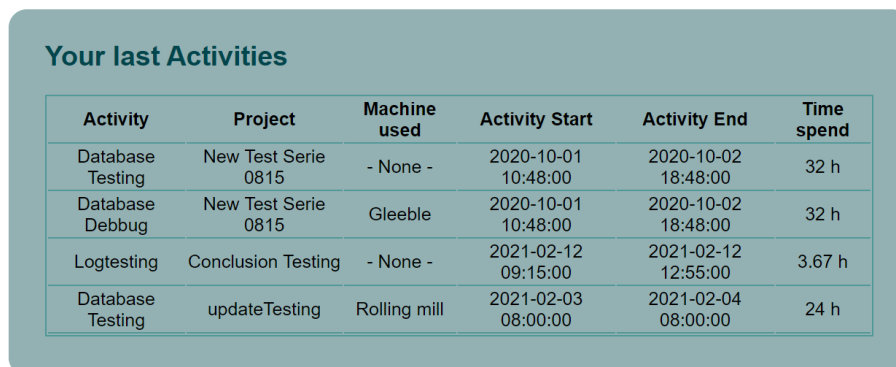
```
function displayMachines($conn, $activ){
    $access = checkAccess($conn, $_SESSION['usID']);
    if ($access){
        $sql = "SELECT * FROM machines
            WHERE macInactive = ".$activ." AND macID > 1
            ORDER BY macMaintLast DESC ;";
    } else {
        $sql = "SELECT macNumber, macName, macType, macMaintLast,
            macMaintInterval, macLastUpdate FROM machines
            WHERE macInactive = ".$activ." AND macID > 1
            ORDER BY macMaintLast DESC ;";
    }
    $result = mysqli_query($conn, $sql);
    if (mysqli_num_rows($result) > 0){
```

Code snippet 7 Function *displayMachines(...)*

Code snippet 7 explains how the data is selected by the function *displayMachines(\$conn, \$activ)* for projects. To display active or inactive machinery the variable *\$active* defines the output. Here is shown how the code differentiates between the users. User, for which the variable *\$access* is true, get all the machine data displayed, but user without get the short list with the next maintenance data.

3.4.4 activitiesNew.php

The activities page is about entering the activities carried out for projects with or without machines. This section is divided into the creation and updating of activities. The page *activityNew.php* allows a user to add new tasks to the projects. The display shows the last 15 tasks of the user. Below this entry the input form that transfers the tasks to the database is displayed. The page distinguishes between administrators and other users. An admin can independently enter activities for all members of the team. According to the Stakeholder analysis, users can only create activities in their own name.



Activity	Project	Machine used	Activity Start	Activity End	Time spend
Database Testing	New Test Serie 0815	- None -	2020-10-01 10:48:00	2020-10-02 18:48:00	32 h
Database Debbug	New Test Serie 0815	Gleeble	2020-10-01 10:48:00	2020-10-02 18:48:00	32 h
Logtesting	Conclusion Testing	- None -	2021-02-12 09:15:00	2021-02-12 12:55:00	3.67 h
Database Testing	updateTesting	Rolling mill	2021-02-03 08:00:00	2021-02-04 08:00:00	24 h

Figure 19 Display of the last activities

Figure 19 shows the display of the last 15 activities of the user. Including an activity name, corresponding project, used machines and the required time of the activity.

```

function displayActivity($conn){
    $user = $_SESSION['usID'];
    $access = checkAccess($conn, $user);
    if($access){
        $sql = "SELECT actUsName, usName, proName, macName, actUsStartDate,
                actUsEndDate
                FROM activityuser
                INNER JOIN user ON activityuser.actUsUser = user.usID
                INNER JOIN machines ON activityuser.actUsMachine = machines.macID
                INNER JOIN projects ON activityuser.actUsProject = projects.proID
                LIMIT 20;";
    } else {
        ...
    }
    echo '...';
    $result = mysqli_query($conn, $sql);
    if(!$result){
        $result = 0;
    }
    if(mysqli_num_rows($result) > 0){
        while($row = mysqli_fetch_assoc($result)){
            $hours = dateDifference($row["actUsStartDate"],
                                $row["actUsEndDate"], 2);
            ...
        }
    }
}

```

Code snippet 8 Function *displayActivity(...)*

Code snippet 8 shown above illustrates how the data is selected by the function *displayActivity(\$conn)*. It is also shown how the code differentiates between the users. As already mentioned before (code snippet 6) the function *checkAccess(...)* checks the user access to the sensible machine data. If check is successful, the display will shorten the list and only the next maintenance data is shown.

The page *activitiesNew.php* to allow the user to create a new task in his name. All entries are mandatory, otherwise the task cannot be saved.

Figure 20 Input mask for new activities

As shown in figure 20, a user can create tasks in his/her name. In addition to a name for the task performed, the duration must also be entered by means of the start and end time of the tasks. If the user has worked without a machine, he can set this within the machine selection by selecting None. If a user is not operating a machine for a task while the machine is running, this can be entered in the PMT:CMF via the 'No user' checkbox. The activity is submitted with the button named send, to the transfer file *activities.inc.php*. After checking the entries are transferred to the database using the function *createActivity(\$conn, \$aName, \$uName, \$pName, \$mName, \$actStart, \$actEnd, \$noUser)*.

```

if(isset($_POST["activity-submit"])){
    if($_POST["proName"] == 0){
        header("location: ../activityNew.php?error=NoProject");
        exit();
    }
    $uName = $_POST["actUser"];
    $aName = $_POST["actName"];
    $pName = $_POST["proName"];
    $mName = $_POST["macName"];
    $aDateS = $_POST["actStartDate"];
    $aDateE = $_POST["actEndDate"];
    $aTimeS = $_POST["actStartTime"];
    $aTimeE = $_POST["actEndTime"];
    if(!empty($_POST["cbNoUser"])){
        $noUser = 1;
    } else {
        $noUser = 0;
    }
    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';
    $actStart = date('Y-m-d H:i:s', strtotime("$aDateS $aTimeS"));
    $actEnd = date('Y-m-d H:i:s', strtotime("$aDateE $aTimeE"));
    createActivity($conn, $aName, $uName, $pName, $mName, $actStart, $actEnd, $noUser
);
    header("location: ../activityNew.php?error=none");
    exit();
}

```

Code snippet 9 Transfer page for create activity

In code snippet 9 the transfer file of *activityNew.php* is shown. Using the integrated *POST* method, the variables are read from the form of the *activityNew.php* page. It is important to note that *POST* requests are never cached, do not remain in the browser history, cannot be bookmarked and have no data length limit, resulting in the requirement of storing this kind of request to a variable as seen above.

```

function createActivity($conn, $aName, $uName, $pName,
    $mName, $actStart, $actEnd, $noUser){
    // Prepare sql statement
    $sql = "INSERT INTO activityuser(actUsName, actUsUser,
        actUsProject, actUsMachine, actUsStartDate, actUsEndDate,
        actUsNoUser, actUsLog)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
    $stmt = mysqli_stmt_init($conn);
    // Check if input is valid
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../activityNew.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    // update the last entry
    $lastActivity = date("Y-m-d H:i:s");
    $user = inputExists($conn, $uName, $uName, 'employee');
    // Bind parameter to statement and execute
    if(mysqli_stmt_bind_param($stmt, "siiissis" ,
        $aName, $user["usID"], $pName, $mName, $actStart,
        $actEnd, $noUser, $lastActivity) == false){
        header("location: ../activityNew.php?error=stmt_bind_param_failed?... ");
        exit();
    }

    if(mysqli_stmt_execute($stmt) == false){
        header("location: ../activityNew.php?error=execute_failed?... ");
        exit();
    }
    ...
}

```

Code snippet 10 Function *createActivity(...)*

In code snippet 10, the function *createActivity(\$conn, \$aName, \$uName, \$pName, mName, \$actStart, \$actEnd, \$noUser)* with its steps to connect to the database with the SQL statement is shown. After connection, the function gets checked with the internal function *mysqli_stmt_prepare(...)* and afterwards the entered variables are linked to the statement using the internal function *mysqli_stmt_bind_param(...)*. This ensures that the input does not damage, delete or render the database unusable through injection. In addition to that, if there is an error with the binding process, an error message is shown and the creation of the task is stopped. Finally, the statement is executed and sent to the database using the internal function *mysqli_execute(...)*.


```

function createActivity($conn, $aName, $uName, $pName,
...

//update Project
$logUpdate = date("Y-m-d H:i:s", strtotime($actEnd));
$lastLog = "UPDATE projects
            SET proLastUpdate = '$logUpdate.'
            WHERE proID = '$pName.'";

//update Machine
$lastLog = "UPDATE machines
            SET macLastUpdate = '$logUpdate.'
            WHERE macID = '$mName.'";

//Close DB connection
mysqli_stmt_close($stmt);
mysqli_close($conn);
}

```

Code snippet 11 Additional updates of *createActivity(...)*

As shown in code snippet 11, after the execution of the statement the tables *projects* and *machines* are updated to the latest dates of the activities. The end time / date of an activity is directly inserted into the tables.

3.4.5 activityUpdate.php

In addition to the creation page for activities, it is possible to modify activities afterwards. This is made possible with the page *activityUpdate.php*. Again, a distinction is made between admin and user. Users can only change their own activities while admins can access and change any activity.

Figure 21 Input mask of activity update

The update activity shows the same display as mentioned in the page *activityNew.php*. As shown in Figure 21, this is the input form for a user, where above the user name already is

placed on the form and the user isn't able to change it. Also, as mentioned before, the page distinguishes between administrators and other users so that an admin can alter activities for all members of the team.

```

if(isset($_POST["activity-update"])){
    $aID = $_POST["actUsID"];
    if($aID == 0){
        header("location: ../activityUpdate.php?error=NoActivity");
        exit();
    }

    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';

    //Get old activity data
    $input = inputExists($conn, $aID, $aID, 'activity');

    $userID = $input["actUser"];
    $aName = $input["actName"];
    $pName = $input["proName"];
    $mName = $input["macName"];
    $actStart = $input["actStartDate"];
    $actEnd = $input["actEndDate"];
    $noUser = $input["actNouser"];
    if (!empty($_POST["actUser"])){
        $userID = $_POST["actUser"];
    }
    if (!empty($_POST["actName"])){
        $aName = $_POST["actName"];
    }
    } ...

```

Code snippet 12 Transfer page of *activityUpdate.php*

As shown in code snippet 12, the transfer of *activityUpdate.php* using the PHP integrated `$_POST([])` method, takes the variables from the form of the update page. The activity id is first checked in the database and previous inputs are loaded into existing variables. Afterwards the input forms are checked if they are empty. If not, new data will be submitted to the update function. The update itself is done by the function `updateActivity($conn, $aID, $aName, $uName, $pName, $mName, $actStart, $actEnd, $noUser)`.

```

function updateActivity($conn, $aID, $aName, $uName,
    $pName, $mName, $actStart, $actEnd, $noUser){
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $user = inputExists($conn, $uName, $uName, 'employee');
    $update = " UPDATE activityuser
        SET  actUsName =?, actUsUser =?, actUsProject =?,
            actUsMachine =?, actUsStartDate =?, actUsEndDate =?,
            actUsNouser=?, actUsLog='".date("Y-m-d h:a:s")."'
        WHERE actUsID =?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../activityUpdate.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    $lastActivity = date("Y-m-d H:i:s");
    if(mysqli_stmt_bind_param($stmt, "siiissisi",
        $aName, $user["usID"], $pName, $mName, $actStart, $actEnd,
        $noUser, $lastActivity, $aID) == false){
        header("location: ../activityUpdate.php?error=stmt_bind_param_failed?...");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../activityUpdate.php?error=query_failed?...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    ...
}

```

Code snippet 13 Function *updateActivity(...)*

Code snippet 13 shows the update function and as already shown before in the previous subsection the function *mysqli_stmt_bind_param(...)*, which ensures that the update input does not damage, delete or render the database unusable through injection. Also if there is an error along the internal functions the update of the task is stopped and an error message is shown to the user. Finally, the statement is executed and sent to the database using the internal function *mysqli_execute(...)*. Just as shown in code snippet 11, after the instruction is executed, the projects and machines tables are updated to the latest dates of the activities. The end time / date of an activity is inserted directly into the tables.

3.4.6 projects.php

This section deals with the projects and their links within the tool. Each page is linked to the database to either retrieve data directly or save it. The main purpose of this section is to collect the chair's projects and provide required details.

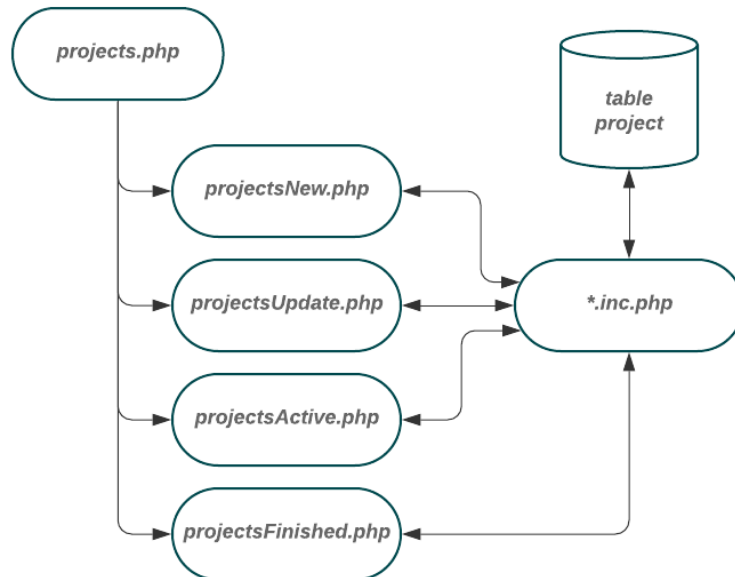


Figure 22 Flowcharts of the project related pages

As mentioned before and displayed in figure 22, all of the project pages are connected to the SQL table projects via **.inc.php* files. The main page *project.php* provides an overall overview of the projects. Here the user gets an update on all current active projects. In addition to the project number, a name and the contact person, the start date and the delivery date of the project can also be found here. A distinction is made here between users, project managers and administrators. Because financial data is sensitive, users can only see the relative status of budget and costs incurred as a percentage value. Only project manager and administrators are allowed to see the absolute numbers of the project budget and the costs already incurred. To archive this the function *displayProjects(\$conn, \$active)* implements a connection to the database and retrieves all necessary data about the projects.

For this purpose, a SQL statement is built, since this is not influenced by the user, the statement is not equipped with security measures against injection. This SQL statement connects the table project with the tables user and projecttypes to make it more readable for the user.

```

function displayProjects($conn, $activ){
    $sql = "SELECT proID, proNumber, proName, typName, usName, proBudget,
            proStartDate, proEndDate, proLastUpdate
            FROM projects
            INNER JOIN user ON projects.proContact = user.usID
            INNER JOIN projecttypes ON projects.proType = projecttypes.typID
            WHERE proFinished = ".$activ."
            ORDER BY proEndDate DESC
            LIMIT 15;";
    $result = mysqli_query($conn, $sql);
    if (mysqli_num_rows($result) >0){
        if ($access){
            echo "...

```

Code snippet 14 SQL statement of Function *displayProjects(...)*

As shown in code snippet 14, the attribute *\$active* is submitted with the function call. This attribute is matched with the *proFinished* column, allowing active or finished projects to be displayed. On the *project.php* page only active projects are shown, as this is more relevant for all users of the tool than if all projects are visible. The number of active projects was also limited to 15, as otherwise the overall overview according to the stakeholder analysis would be lost.

When an associative list of data is returned from the SQL database, the function prepares the entries for the user. An associative list is an abstract data type used here to manage key-value pairs and look up the value associated with a particular key.

```

...
while($row = mysqli_fetch_assoc($result)){
    $costsArray = costsProject($conn, $row['proID']);
    $costs = $costsArray[0] + $costsArray[1];
    $budget = number_format($row["proBudget"],2,".", " ");
    $costs = number_format($costs,2,".", " ");
    $start = date("d.m.Y", strtotime($row["proStartDate"]));
    $end = date("d.m.Y", strtotime($row["proEndDate"]));
    echo "
    <tr>
        <td>".$row["proNumber"]."</td>
        <td>".$row["proName"]."</td>
        <td>".$row["typName"]."</td>
        <td>".$row["usName"]."</td>
        <td>".$start."</td>
        <td>".$end."</td>
        <td>".$budget."</td>
        <td>".$costs."</td>
    </tr>";
} ...

```

Code snippet 15 Data display of fetched SQL data

In code snippet 15, the individual columns of the array are called and the respective values are stored at variables for display. These are then returned to the visible page using the command *echo*. As already mentioned, a distinction is made between admin and user for the costs. In addition, the admin also sees the finished projects on this page. As a result, the information can be accessed and used more quickly.

3.4.7 projectNew.php

To create a new project, the page *projectNew.php* has been created. This can only be done by an admin, as only he or she has access to the *projectNew.php* page. As shown in figure 22, the *projectNew.php* is not directly linked to the database, as the input data is transmitted to an inclusive file. Within this file, the input data is checked and afterwards transmitted to the *functions.inc.php* file for the creation of the new entry in the database.

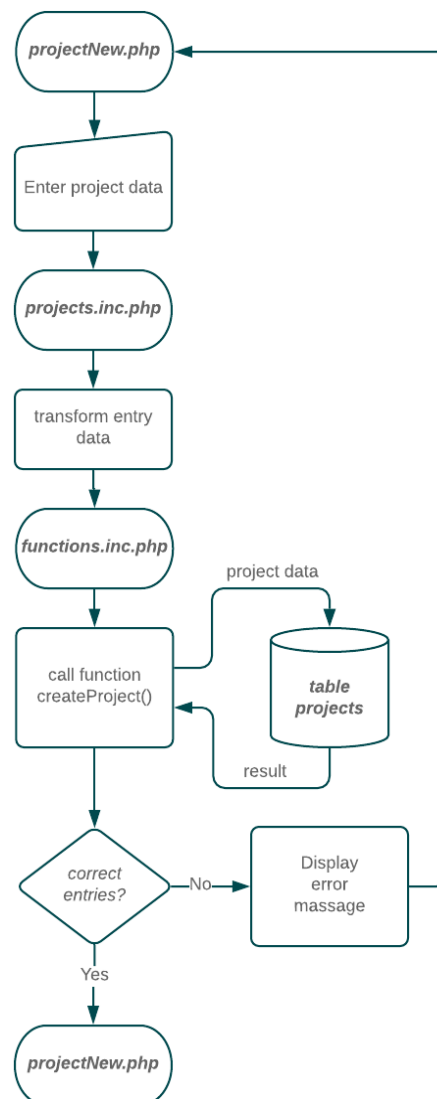


Figure 23 Flowchart of new project creation

Figure 23 shows the procedure for entering a new project. The data entered is passed to the `createProject($conn, $number, $name, $contact, $type, $budget, $startDate, $endDate, $note)` function via the `projects.inc.php` file. This function then transfers the entered data to the table `projects`. If the data is entered correctly, the user is informed that the project has been created. Otherwise, an error message is displayed with the respective problem.

```

if(isset($_POST["project-submit"])){
    $number = $_POST["proNumber"];
    $name = $_POST["proName"];
    $contact = $_POST["proContact"];
    $budget = $_POST["proBudget"];
    $startDate = $_POST["proStartDate"];
    $endDate = $_POST["proEndDate"];
    $note = $_POST["proNote"];
    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';
    createProject($conn, $number, $name, $contact, $type,
        $budget, $startDate, $endDate, $note);
    header("location: ../projectNew.php?error=none");
    exit();
}

```

Code snippet 16 Transfer page for a new project

This input fields are divided in three types. As shown in the first group of code snippet 16 the inputs project number and name are transferred. The second input group is the project manager and the project type both chosen by a dropdown menu and the third type are the project dates entered by time and date fields. The function `createProject($conn, $number, $name, $contact, $type, $budget, $startDate, $endDate, $note)` first creates the database connection and calls the function `inputExists(...)`. This function checks the database whether a similar project with the same number already exists.

```

function createProject($conn, $number, $name, $contact, $type, $budget, $startDate, $endDate, $note){
    // Check if Project already exists
    if(inputExists($conn, $name, $name, 'project') !== false){
        header("location: ../projectNew.php?error=ProjectExists");
        exit();
    }
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $sql = "INSERT INTO projects (proNumber, proName, proNote, proContact,
                                proType, proBudget, proStartDate, proEndDate,
                                proLastUpdate)
          VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
    // Check if input is valid
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../projectNew.php?error=stmt_prepare_failed?...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    // last update when insert is taken
    $lastUpdate = date("Y-m-d");
    // Bind parameter to statement and execute in SQL DB
    $res = mysqli_stmt_bind_param($stmt, "ssssidsss", $number, $name, $note, $contact, $type, $budget, $startDate, $endDate, $lastUpdate);
    if($res == false){
        header("location: ../projectNew.php?error=stmt_bind_param_failed?...");
        exit();
    }
    if(mysqli_stmt_execute($stmt) == false){
        header("location: ../projectNew.php?error=execute_failed?...");
        exit();
    }
    //Close DB connection
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
}

```

Code snippet 17 Function *createProject(...)*

In code snippet 17 the function *createProject(...)* is displayed. As mentioned in the subsections before, the function creates a SQL statement, then checks with the internal function *mysqli_stmt_prepare(...)* and links the entered variables to the statement using the function *mysqli_stmt_bind_param(...)*. Finally, the statement is executed and sent to the database using the internal function *mysqli_execute(...)*.

3.4.8 projectUpdate.php

To update a project, the user must access the *projectUpdate.php* page. However, this is only possible if the user has the status of a project manager or the role of an administrator. As shown in figure 22 the *projectUpdate.php* is not directly linked to the database.

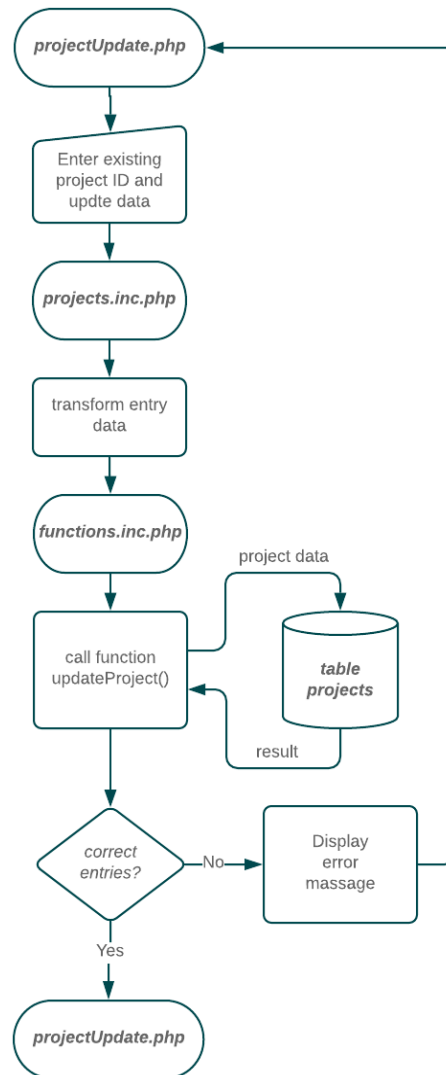


Figure 24 Flowchart of the update project

Figure 24 shows the procedure for updating a project. The data entered is passed to the function *updateProject(\$conn, \$pID, \$number, \$name, \$contact, \$type, \$budget, \$startDate, \$endDate, \$note, \$finished)* via the *projects.inc.php* file. This function then transfers the entered data to the table *projects*. If the data is entered correctly, the user is informed that the project has been updated successfully. Otherwise, an error message is displayed with the respective problem.

```

function updateProject($conn, $pID, $number, $name, $contact, $type,
    $budget, $startDate, $endDate, $note, $finished){
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $update = " UPDATE projects
        SET proNumber =?, proName =?, proContact =?, proType =?,
            proBudget =?, proStartDate =?, proEndDate =?, proFinished =?,
            proNote =?, proLastUpdate = '".date("Y-m-d h:a:s")."'
        WHERE proID =?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../projectUpdate.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    $budget = str_replace(',', '.', $budget);
    if(!mysqli_stmt_bind_param($stmt, "ssiidssii", $number,
        $name, $contact, $type, $budget, $startDate,
        $endDate, $note, $finished, $pID )){
        header("location: ../projectUpdate.php?error=stmt_bind_param_failed?...");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../projectUpdate.php?error=query_failed?...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if($finished == 1){
        $costsArray = costsProject($conn, $pID);
        $costs = $costsArray[0] + $costsArray[1];
        $fin = "UPDATE projects
            SET proCosts = ".$costs."";
        mysqli_execute($conn, $fin);
    }
    //Close DB connection
    mysqli_close($conn);
}

```

Code snippet 18 Function *updateProject(...)*

In code snippet 18 the function *updateProject(...)* with its steps to update a project is shown. The update statement searches for the *proID* in the database and selects the requested project. This procedure is set up as mentioned before to prevent damage to the database, the internal functions *mysqli_stmt_prepare(...)* and the function *mysqli_stmt_bind_param(...)* are used.

A special case occurs when the project is finished. The total costs are then calculated over again using the *costsProject(\$conn, \$pID)* function shown in code snippet and entered directly into the data field provided for this purpose in the database. The final steps of the function *updateProject(...)* are closing the statement and the connection to the database.

3.4.9 projectActive.php

The details of a project are especially important for milestones and project meetings. In addition to the basic project data, the persons and machines involved are also displayed. When selecting, the time period of the display can be selected. The page shows the active projects are displayed using the function `displayProjects($conn, $active)`. Below this, a drop-down menu is displayed using the function `ddltem($conn, 'project', 0)`. Next to it is the input option for the time span of the activities.



Figure 25 Drop down menu and start / end date

As shown in the preceding figure 25, after selecting the project, it is possible but not required to enter a start and end date before proceeding by clicking the button.

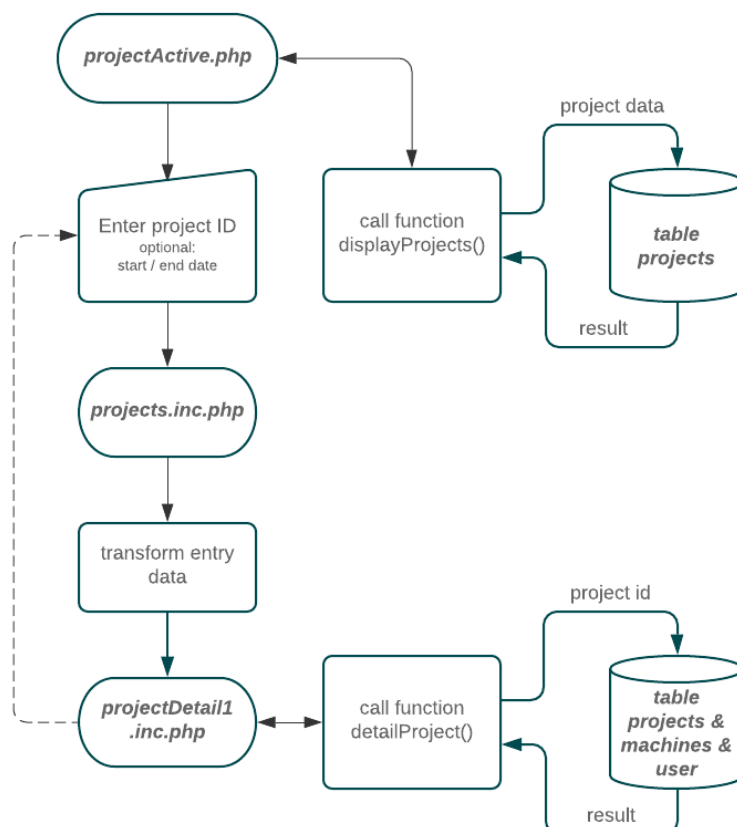


Figure 26 Flowchart project details

Figure 26 displays the procedure of the project details process. As already stated, the user has to choose a project id. When pressing the show display button in the GUI, the selected project id is submitted to the transfer page *project.inc.php*.

```

if(isset($_POST["choose-active"])){
    $id = $_POST["proID"];
    $start = $_POST["start"];
    $end = $_POST["end"];
    if($id == 0){
        header("location: ../projectActive.php?error=nothing");
        exit();
    }
    if(empty($start)){
        $start = '2000-01-01';
    }
    if(empty($end)){
        $end = '3000-01-01';
    }
    header("location: ../projectDetail1.php?id=".$id."
        &start=".$start."&end=".$end."");
    exit();
}

```

Code snippet 19 Transfer function of *projectActive.php*

As shown in code snippet 19, the transfer page catches the entry and the project id is transformed and displays the project details on a new page, *projectDetail1.php*. As mentioned before, if the user doesn't set a date, this is done automatically within this function.

```

<div class="signupbox">
    <?php if(isset($_GET['id'])){ detailProject($conn, $_GET['id']); }?>
    <table class="tableouter">
        <tr> <?php involvedMachine($conn, $_GET['id'],GET['start'],
            $_GET['end'], 'project');
        ?></tr>
        <tr> <?php involvedEmployee($conn, $_GET['id'], $_GET['start'],
            $_GET['end'], 'project'); ?></tr>
    </table class="tableouter">
</div class="signupbox">

```

Code snippet 20 Functions of *projectDetail1.php*

As shown above in code snippet 20, the output data is displayed in three tables. To fetch all necessary data and display it to the user, the *projectDetail1.php* page calls the function *detailProject(\$conn, \$project)*. The first table contains the general project data such as project number, type, contact person, start date, end date, days until completion, working hours of the employees and machines as well as notes entered.

```

function involvedEmployee($conn, $id, $start, $end, $type){
    if($type == 'project'){
        $sql = "SELECT actUsUser, usName, actUsProject, proNumber, proName,
                actUsMachine, macName, actUsStartDate, actUsEndDate
                FROM activityuser
                INNER JOIN user ON activityuser.actUsUser = user.usID
                INNER JOIN machines ON activityuser.actUsMachine = machines.macID
                INNER JOIN projects ON activityuser.actUsProject = projects.proID
                WHERE activityuser.actUsproject = ".$id."
                AND actUsEndDate >=".$start."
                AND actUsStartDate <=".$end."
                ORDER BY actUsStartDate;";

    }
    // Create a connection to the database
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        exit();
    }
    $result = mysqli_query($conn, $sql);
    if(!$result){
        $result = 0;
    }
    if(mysqli_num_rows($result) > 0){
        echo '...';
        while($row = mysqli_fetch_assoc($result)){
            $hours = dateDifference($row["actUsStartDate"], $row["actUsEndDate"], 2);
            $start = date("h.m - d.m.Y", strtotime($row["actUsStartDate"]));
            $end = date("h.m - d.m.Y", strtotime($row["actUsEndDate"]));
            echo '...';
        }
    } else {
        echo '<p> No employee activites.</p>';
    }
}

```

Code snippet 21 Function *involvedEmployee(...)*

Code snippet 21 illustrates the function *involvedEmployee(\$conn, \$id, \$start, \$end, \$type)* which selects the employees for the project from the activities and displays them in a table. The function starts with its steps to connect to the database with the SQL statement. The database then provides the requested columns and the function displays the data on the page. Afterwards the page *projectDetail1.php* starts the function *involvedMachine(\$conn, \$id, \$start, \$end, \$type)* with the same attempt.

```

function involvedMachine($conn, $id, $start, $end, $type){
    if($type=='project'){
        $sql = "SELECT actUsUser, usName, actUsProject, proNumber, proName,
                actUsMachine, macName, actUsStartDate, actUsEndDate
                FROM activityuser
                INNER JOIN user ON activityuser.actUsUser = user.usID
                INNER JOIN machines ON activityuser.actUsMachine = machines.macID
                INNER JOIN projects ON activityuser.actUsProject = projects.proID
                WHERE activityuser.actUsproject = ".$id."
                AND actUsEndDate >=".$start." AND actUsStartDate <=".$end."
                ORDER BY actUsStartDate;";
    }
    // Create a connection to the database
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        exit();
    }
    $result = mysqli_query($conn, $sql);
    if(!$result){
        $result = 0;
    }
    if(mysqli_num_rows($result) > 0){
        echo '...';
        while($row = mysqli_fetch_assoc($result)){
            $hours = dateDifference($row["actUsStartDate"], $row["actUsEndDate"], 2);
            $hours = number_format($hours, 2, ",", " ");
            $start = date("h.m - d.m.Y", strtotime($row["actUsStartDate"]));
            $end = date("h.m - d.m.Y", strtotime($row["actUsEndDate"]));
            echo '...';
        }
    } else {
        echo '<p> No machine activites.</p>';
    }
}

```

Code snippet 22 Function *involvedMachines(...)*

Code snippet 22 shows the function *involvedMachine(...)*. This function operates the same way as the function *involvedEmployee(...)*. The function selects the machines for the project from the activities and displays them in a table. The database then provides the requested columns and the function displays the data on the page.

3.4.10 projectFinished.php

The *projectFinished.php* page acts exactly the same as the *projectActive.php* page mentioned earlier in subsection 3.4.9. The only difference is the passing of a variable *\$active*, which the functions use to select which projects are displayed.

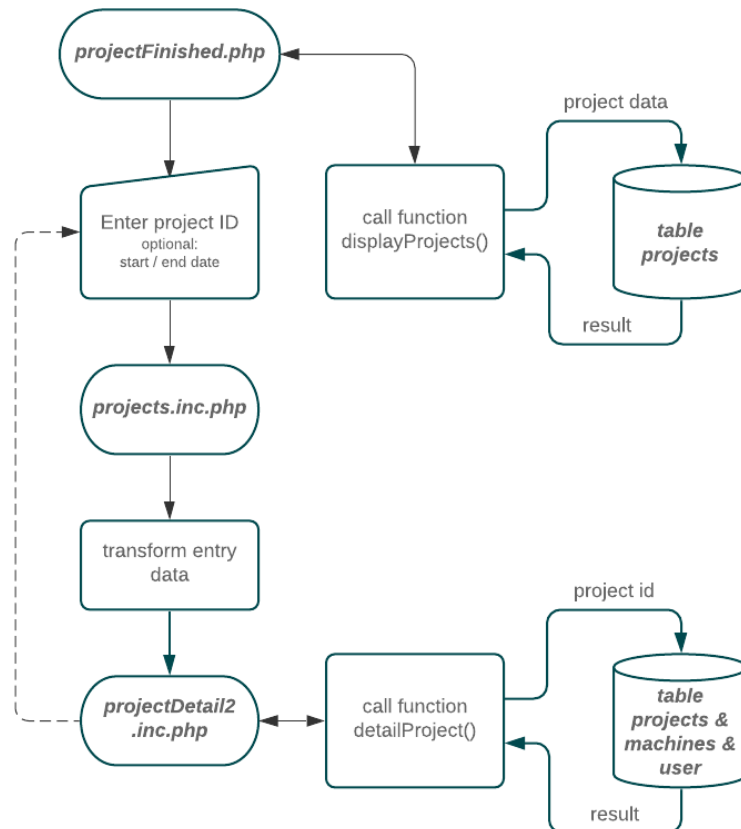


Figure 27 Flowchart project details

Figure 27 displays the procedure of the finished project detail process. As mentioned in the section before, stated the user must choose a project id. When pressing the show display button, the selected project id is submitted to the transfer page *project.inc.php*. Afterwards the procedure follows the same path as described in subsection 3.4.8 before.

3.4.11 Machines

This subsection covers the integration of the machines and their connections within the PMT:CMF with the SQL database. Each page is linked to the database to either retrieve data directly or save it. The main purpose of this section is to collect the machine data and their resulting working costs.

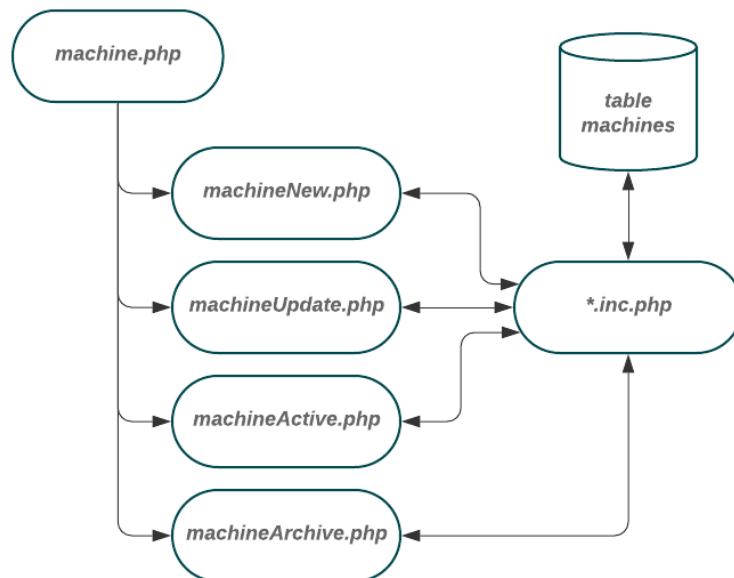


Figure 28 Flowcharts of the machine related pages

Figure 28 displays the interlink between the machine pages and the SQL table machines via a *machine.inc.php* file. The main *machine.php* page provides an overall overview of the machines. According to the stakeholder analysis, the user has the possibility to update on all maintenance dates, despite the machine names.

A distinction is made between users, project managers and administrators. Because financial data is sensitive, users can't see the purchase price and the actual asset value. Only technicians and administrators are allowed to see the absolute numbers of purchase price, hour rate and actual value. To do so, the function *displayMachines(\$conn, \$active)* retrieves all necessary data about the machines.


```

function displayMachines($conn, $activ){
    $access = checkAccess($conn, $_SESSION['usID']);
    if ($access){
        $sql = "SELECT * FROM machines
              WHERE macInactive = ".$activ." AND macID > 1
              ORDER BY macMaintLast DESC ";
    }
    $result = mysqli_query($conn, $sql);
    if (mysqli_num_rows($result) > 0){
        echo "...";
        while($row = mysqli_fetch_assoc($result)){
            $lastMaint = $row["macMaintLast"];
            $MaintInterval = $row["macMaintInterval"];
            $last = date("d.m.Y", strtotime($lastMaint));
            $next = date("d.m.Y", strtotime($lastMaint.'+'.$MaintInterval.'day'));
            $update = date("h:i d.m.Y", strtotime($row["macLastUpdate"]));
            $rate = number_format($row["macHourRate"],2,".", " ");
            $value = number_format($row["macAssetValue"],2,".", " ");
            echo "...";
        }
    } else {
        echo "NØ results";
    }
}

```

Code snippet 23 Function *displayMachines(...)*

As mentioned in code snippet 23, the attribute *\$active* is used which is submitted with the function call. This attribute is matched with the *macInactive* column, allowing active or inactive machines to be displayed. At the *machine.php* only active machines are shown. As the associative list *\$result* is returned from the SQL database, the function prepares the entries for the user. An associative list is an abstract data type used here to manage key-value pairs and look up the value associated with a particular key.

3.4.12 machineNew.php

To implement a new machine, the page *machineNew.php* has been created. This can only be done by an admin, as only he or she has access to the *machineNew.php* page. As shown in Figure 28 the pages is not directly linked to the database. The input data is transmitted to an inclusive file. There the input data is checked and afterwards transmitted to the *functions.inc.php* file for the creation of the new entry in the database.

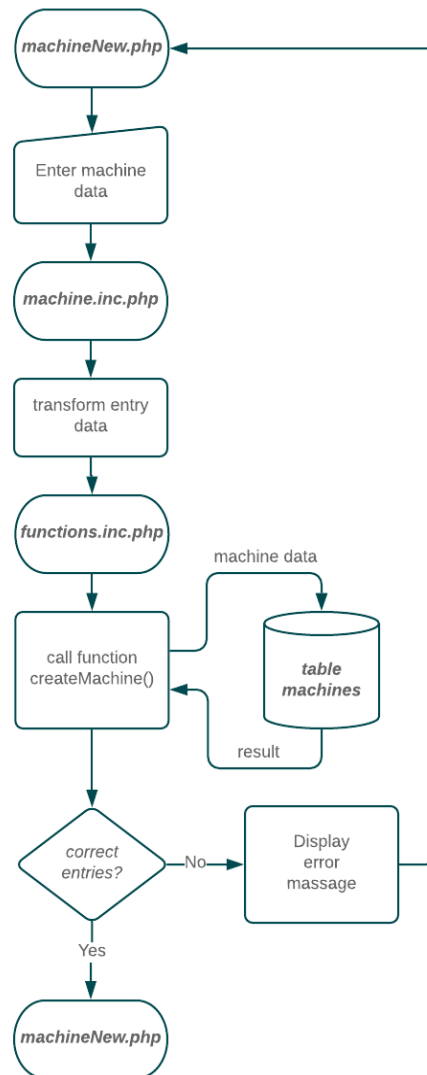


Figure 29 Flowchart of new project creation

Figure 29 shows the procedure for creating a new machine in the system. The entered data is passed to the function *createMachine(\$conn, \$number, \$name, \$type, \$note, \$purDate, \$purCost, \$hourRate, \$maintInterval, \$util)* via the *machines.inc.php* file. This function transforms the data and sends it to the table *machines*. If the data is entered correctly, the user gets the information that the machine has been entered in the system. Otherwise, an error message is displayed with the respective problem.

```

function createMachine($conn, $number, $name, $type, $note, $purDate,
                    $purCost, $hourRate, $maintInterval, $util){
    if(inputExists($conn, $name, $name, 'machine') !== false){
        header("location: ../machineNew.php?error=MachineAlreadyExists");
        exit();
    }
    $stmt = mysqli_stmt_init($conn);
    $sql = "INSERT INTO machines (macNumber, macName, macType, macNotes,
                                macPurchaseDate, macPurchaseCosts,
                                macHourRate, macMaintLast,
                                macMaintInterval, macUtilisation)
          VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../machineNew.php?error=stmt_prepare_failed?...");
        exit();
    }
    $maintLast = date(Y-m-d);
    if(!mysqli_stmt_bind_param($stmt, "ssssdisii", $number, $name, $type,
                                $note, $purDate, $purCost, $hourRate,
                                $maintLast, $maintInterval, $util)){
        header("location: ../machineNew.php?error=stmt_bind_param_failed?...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_execute($stmt)){
        header("location: ../machineNew.php?error=stmt_execute_failed?...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
}

```

Code snippet 24 Function createMachine(...)

In code snippet 24 the function *createMachine(...)* with its steps to connect to the database with the SQL statement, is visualised. To prevent damage, deletion or injection the SQL statements are checked with the internal function *mysqli_stmt_prepare(...)*. The internal function *mysqli_stmt_bind_param(...)* is used for the linking the entered variables to the statement and executed with the function *mysqli_execute(...)*. The already mentioned error messages are implemented as well as in the previous sections.

3.4.13 machineUpdate.php

To update machines, the user must access the page *machinesUpdate.php*. Only if the user has the role of a technician or the role of an administrator this is possible. As shown in figure 22 the *machineUpdate.php* is also not directly linked to the database.

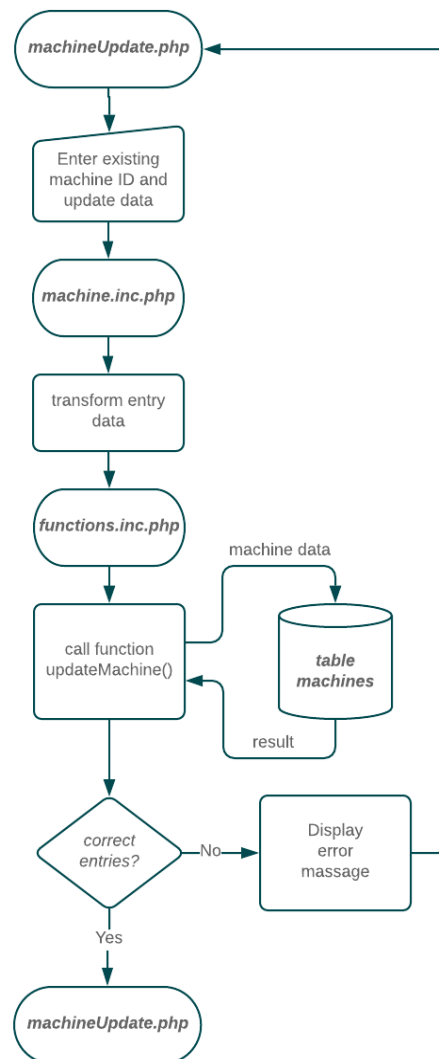


Figure 30 Flowchart of the update machine

In general, figure 30 shows the procedure for updating an existing machine. The data entered is passed to the function *updateMachine(\$conn, \$mid, \$number, \$name, \$type, \$note, \$purDate, \$purCost, \$hourRate, \$assetValue, \$util, \$maintInterval, \$maintLast, \$inactive)* via the *machines.inc.php* file. The entered data is transferred to the table *projects* with this functions. The user is informed that the project has been updated successfully, if the data is entered correctly, otherwise, an error message is displayed with the respective problem.

```

function updateMachine($conn, $mid, $number, $name, $type, $note,
    $purDate, $purCost, $hourRate, $assetValue, $util,
    $maintInterval, $maintLast, $inactive){
    $stmt = mysqli_stmt_init($conn);
    $input = inputExists($conn, $mid, $name, 'machine');
    $assetValue = assetCalc($conn, $purCost, $purDate);
    $update = "UPDATE machines
        SET macNumber=?, macName=?, macType=?, macNotes=?,
            macPurchaseDate=?, macPurchaseCosts=?, macHourRate=?,
            macAssetValue=?, macUtilisation=?, macMaintInterval=?,
            macMaintLast=?, macInactive=? ,
            macLastUpdate = '".date("Y-m-d h:a:s")."'
        WHERE macID=?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../machineUpdate.php?error=stmt_prepare_failed?... ");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_bind_param($stmt,"sssssdidiisii", $number, $name,
        $type, $note, $purDate, $purCost, $hourRate,
        $assetValue, $util, $maintInterval, $maintLast,
        $inactive, $input["macID"])){
        header("location: ../machineUpdate.php?error=stmt_bind_param_failed?...");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../machineUpdate.php?error=query_failed?...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    mysqli_close($conn);
}

```

Code snippet 25 Function updateMachine(...)

In code snippet 25 the function *updateMachine(...)* with its steps to connect to the database with the SQL statement is shown. The function *updateMachine(...)* operates in the same way as the function *updateProject(...)*. As mentioned in the section 3.4.11 before a SQL Statement is formed and the variables are bind with the internal function *mysqli_stmt_bind_param(...)*. In the same way as mentioned earlier, the user is informed of any errors that may have occurred when the function was run through or that the entry was correctly adopted.

3.4.14 machinesDetail.php

The details of a machine are especially important for evaluation meetings and project calculation. The page shows the machines, which are displayed by using the function *displayMachines(\$conn, \$active)*. Below this, a drop-down menu is displayed using the function *ddlItem(\$conn, 'machine', 0)*. Next to it the input option for a time span of the activities is given.

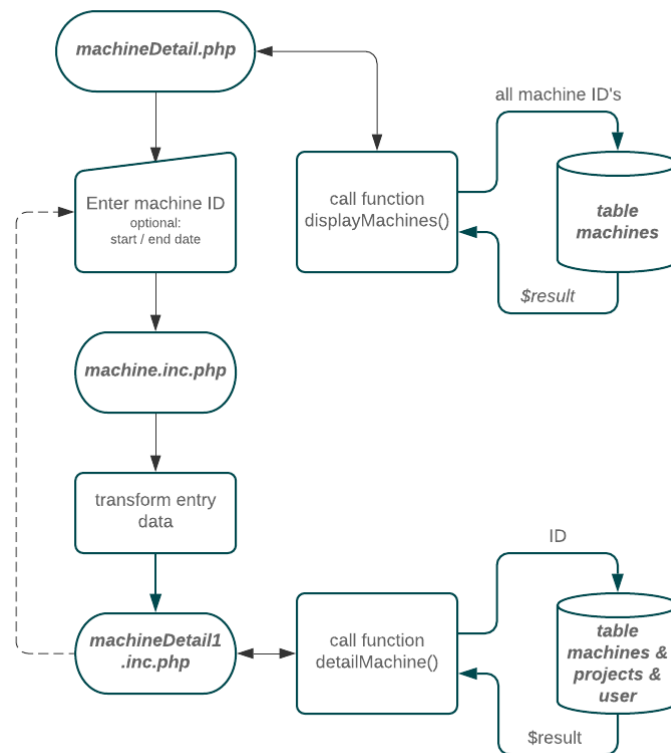


Figure 31 Flowchart machine details

Figure 31 shows the procedure of the detail machine process. As already stated the user has to choose a machine id to start it. When pressing the show display button, the selected id is submitted to the transfer page *machine.inc.php*.

```

if(isset($_POST["choose"])){
    $id = $_POST["macID"];
    $start = $_POST["start"];
    $end = $_POST["end"];
    if($id == 0){
        header("location: ../machineDetail.php?error=nothing");
        exit();
    }
    if(empty($start)){
        $start = '2000-01-01';
    }
    if(empty($end)){
        $end = '3000-01-01';
    }
    header("location: ../machineDetail1.php?...");
    exit();
}

```

Code snippet 26 Transfer function of *machineDetail.php*

As shown in code snippet 26, the transfer page passes the entry. Afterwards the id is transformed as mentioned in the previous section into a header for the new page, *machineDetail1.php*. If the user doesn't set a date the date is set here to display all entries of the item.

```

<div class="signupbox">
    <?php
        if(isset($_GET['id'])){
            detailMachine($conn, $_GET['id']);
        }
    ?>
    <table class="tableouter">
        <tr> <?php involvedProject($conn, $_GET['id'], $_GET['start'],
            $_GET['end'], 'machine'); ?> </tr>
        <tr> <?php involvedEmployee($conn, $_GET['id'], $_GET['start'],
            $_GET['end'], 'machine'); ?> </tr>
    </table class="tableouter">
</div class="signupbox">

```

Code snippet 27 Machine detail PHP functions

Code snippet 27 visualises the output data, which is then displayed in three tables. To fetch all necessary data and display it to the user the *machineDetail1.php* page calls the function *detailProject(\$conn, \$project)*. The first table contains the general machine data such as machine name, type its last maintenance, the next maintenance and the remaining days till that date, the costs and date of the machine purchase. The function creates the labour costs of the last and current year in addition to the labour hours of the last and current year, but only if the user is an admin.

The function *involvedEmployee(...)* which selects the employees from the activities and displays them in a table, is also used in this section. Code snippet 21 shows how the function starts with its steps to connect to the database with the SQL statement. The database then provides the requested columns and the function displays the data on the page. Afterwards the page *machineDetail1.php* starts the function *involvedMachines(...)* with the same attempt.

```
function involvedProject($conn, $id, $start, $end, $type){
    if($type=='machine'){
        $sql = "SELECT actUsUser, usName, actUsProject, proNumber, proName,
                actUsMachine, macName, actUsStartDate, actUsEndDate
        FROM activityuser
        INNER JOIN user ON activityuser.actUsUser = user.usID
        INNER JOIN machines ON activityuser.actUsMachine = machines.macID
        INNER JOIN projects ON activityuser.actUsProject = projects.proID
        WHERE activityuser.actUsMachine = ".$id."
        AND actUsEndDate >=".$start." AND actUsStartDate <=".$end."
        ORDER BY actUsStartDate;";
    }
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        exit();
    }
    $result = mysqli_query($conn, $sql);
    if(!$result){
        $result = 0;
    }
    if(mysqli_num_rows($result) > 0){
        echo '...';
        while($row = mysqli_fetch_assoc($result)){
            $hours = dateDifference($row["actUsStartDate"], $row["actUsEndDate"], 2);
            $start = date("h.m - d.m.Y", strtotime($row["actUsStartDate"]));
            $end = date("h.m - d.m.Y", strtotime($row["actUsEndDate"]));
            echo '...';
        }
    } else {
        echo '<p> No project activites.</p>';
    }
}
}
```

Code snippet 28 Function to show involved projects

As mentioned before at section 3.4.8 the function *involvedProject(...)* which selects the projects for the selected machine from the activities and displays them in a table as seen in code snippet 28. The function creates an SQL statement and connects to the database. With the statement, it fetches the rows of the table *activityuser* with the requested *\$id*. The variable *\$result* then provides the requested columns, which are then read and transformed into an expected expression so it can be displayed on the page.

3.4.15 machineArchiv.php

The page *machineArchiv.php* behaves exactly like the mentioned page *machineDetail.php* earlier in subsection 3.4.13. The only difference is the transfer value of the variable *\$active*, which the functions use to decide which projects are retrieved.

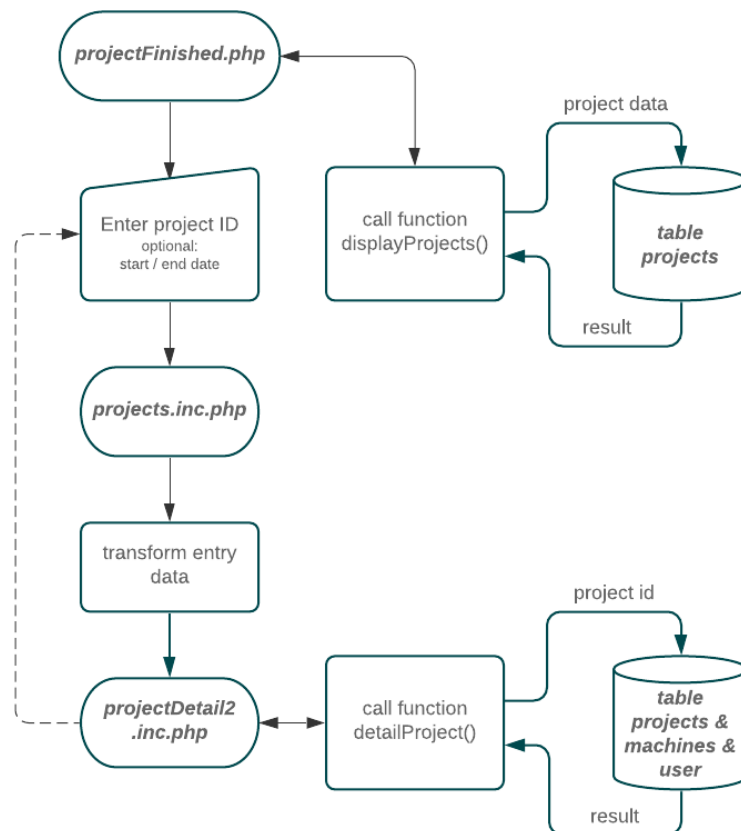


Figure 32 Flowchart project details

Figure 32 displays the procedure of the machine archive process. As already stated the user must choose an ID. When pressing the show detail button, the selected machine id is submitted to the transfer page *machine.inc.php*. Afterwards the procedure follows the same path as the subsection 3.4.13.

3.4.16 employee.php

In this section the employees' information is displayed, entered, stored and changed. Each page of the employee section is linked to the database via an *employee.inc.php* file to either create, retrieve, display or change data.

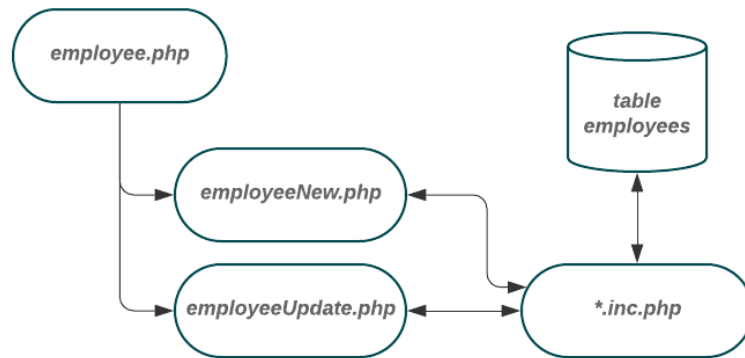


Figure 33 Related pages of the employee section

As shown in figure 33 above, the employee pages are linked by the *employee.inc.php* file to the database to prevent damage or injection. The main page displays all relevant data for an employee. To archive this the function *displayEmployee(\$conn, \$active)* implements a connection to the database and retrieves all necessary data about the user and displays the retrieved data to the viewer. Based on the stakeholder analysis, users only see their own data. An administrator is able to see all the registered user and their contract dates because they have to know when to talk to the employee about his contract and set a new date.

3.4.17 employeeNew.php

In accordance with the stakeholder analysis, only an administrator is allowed to register new employees. The entered data is transmitted to an inclusive file, *employee.inc.php*. Within this file, the data is checked and transmitted to the function *createUser(\$conn, \$name, \$login, \$contractEnd, \$role, \$cfec, \$afee, \$sfee, \$pwd)* of the *functions.inc.php* file for the creation of the new entry in the database .

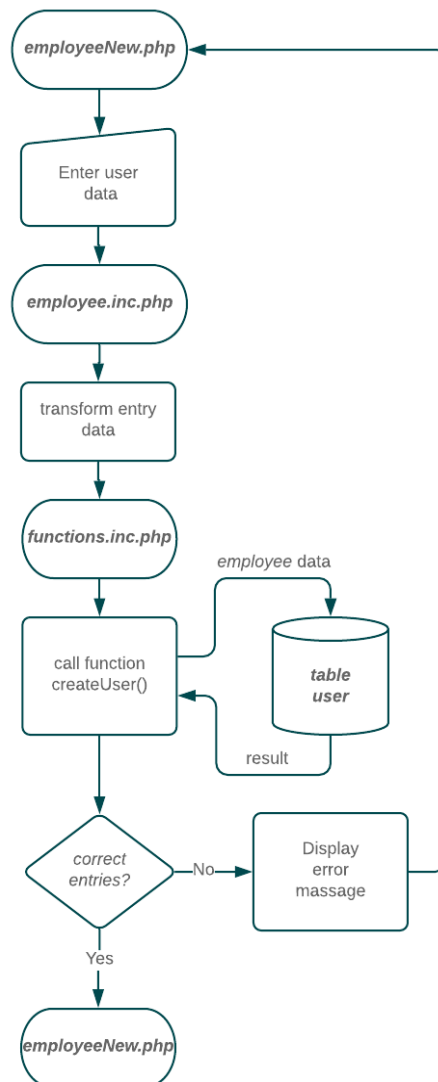


Figure 34 Flowchart to register a new user

Figure 34 shows the transfer of entered data with the function *createUser(...)* to the table *user*. If the data has been entered correctly, a notification is displayed, otherwise an error notification is displayed with the specific issue.

```

function createUser($conn, $name, $login, $contractEnd,
                  $role, $cfee, $afee, $sfee, $pwd){
    if(inputExists($conn, $login, $login, 'employee') !== false){
        header("location: ../employeeNew.php?error=EmployeeAlreadyExists");
        exit();
    }
    $stmt = mysqli_stmt_init($conn);
    $sql = "INSERT INTO user (usName, usLogin, usContract, usRole,
                            usFee1, usFee2, usFee3, usPwd, usLastLogin)
          VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../employeeNew.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    $hashedPwd = password_hash($pwd, PASSWORD_DEFAULT);
    $lastLogin = date("Y-m-d");
    if(!mysqli_stmt_bind_param($stmt, "sssiddss", $name, $login,
                              $contractEnd, $role, $cfee, $afee, $sfee,
                              $hashedPwd, $lastLogin)){
        header("location: ../employeeNew.php?error=stmt_bind_param_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_execute($stmt)){
        header("location: ../employeeNew.php?error=stmt_execute_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
}

```

Code snippet 29 Function createUser(...)

Code snippet 29 shows the function *createUser(...)*, which establishes a database connection to examine the database for an existing user with the *loginname*. If the request is incorrect, the function takes the user's entered password and scrambles the input with the internal *password_hash(...)* function. This approach has the advantage that even if the database is hacked, the user passwords are not shown in the tables. After this the statement is bound with the function *mysqli_stmt_bind_param(...)*, so that the statement can be inserted into the database with the function *mysqli_execute(...)*.

3.4.18 employeeUpdate.php

To upgrade the name or any other data of a user, an Admin can access the *employeeUpdate.php*. Based on the interviews from the stakeholder analysis, a distinction is also made at this point between user and admin. Every user is able to update its own name, login and password, while the admin can change all fields including roles and fees.

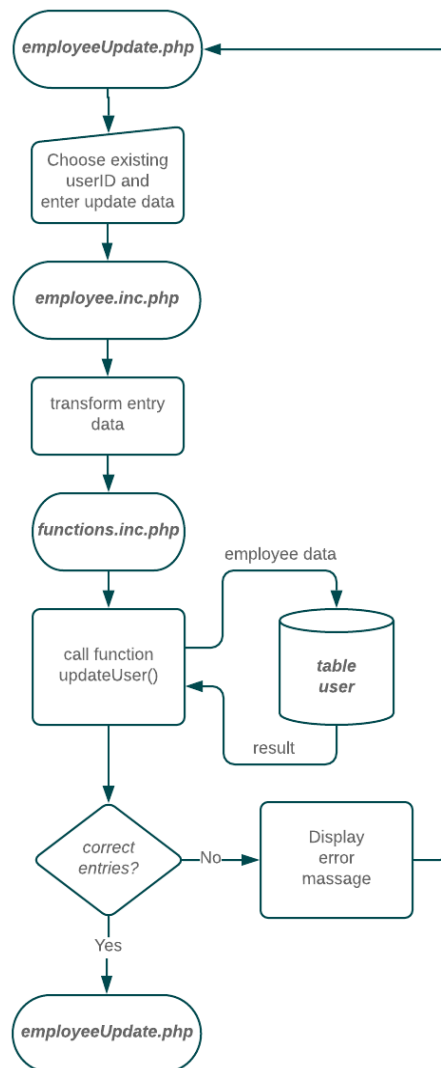


Figure 35 Flowchart of the update user process

As shown in Figure 35, the entered data is transferred to the users table using the function *updateUser(...)*. An error message is displayed with the corresponding problem if the data was entered incorrectly. The user is also informed if an entry is correct.

```

function updateUser($conn, $userID, $name, $login, $contractEnd,
                  $role, $cfee, $afee, $sfee, $pwd){
    $stmt = mysqli_stmt_init($conn);
    $hashedPwd = password_hash($pwd, PASSWORD_DEFAULT);
    $update = "UPDATE user
              SET usName=?, usLogin=?, usContract=?, usRole=?,
                usFee1=?, usFee2=?, usFee3=?, usPwd=?, usLastLogin=?
              WHERE usID=?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../employeeUpdate.php?error=stmt_prepare_failed...");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_bind_param($stmt, "ssiddssi", $name, $login, $contractEnd,
                              $role, $cfee, $afee, $sfee, $hashedPwd, $lastLogin,
                              $userID)){
        header("location: ../employeeUpdate.php?error=stmt_bind_param_failed?");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../employeeUpdate.php?error=query_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    mysqli_close($conn);
}

```

Code snippet 30 Function *updateUser(...)*

Code snippet 30 shown is the function *updateUser(...)* with its steps to create a SQL statement, connect to the database and update the variables in the database. A special case occurs when the users contract has ended.

```

function inactiveUser($conn){
    $stmt = mysqli_stmt_init($conn);
    $hashedPwd = password_hash('.*.*.', PASSWORD_DEFAULT);
    $sql = "UPDATE user
          SET usPwd = '". $hashedPwd. "'
          WHERE DATEDIFF(usContract, NOW()) < 0
          AND DATEDIFF(usContract, NOW()) > -180
          AND usID >= 2;";
    mysqli_query($conn, $sql);
}

```

Code snippet 31 Function *inactiveUser(...)*

With every user login the user table is searched for terminated contracts in the last six months. This is done at the end of the function *loginUser(...)* with the embedded function *inactiveUser(\$conn)* as shown in code snippet 31. This function updates every password of an inactive user to a random chosen one. This allows to let the contracts run out without an urgent need to change the user actively. If a user comes back, an admin can set a new contract end date and password to activate the account again.

4 Results and discussion

In this chapter, the results of the preceding code snippets are presented. The embedding in the whole system at the chair of metal forming is also shown and described. The figures shown in this chapter are created with the PMT:CMF version 1.212m. The digital project management tool is a part of a multi-layered networked solution for recording project, machine and employee data. According to the six-layer condition monitoring system of the chair of metal forming, the PMT:CMF collects this data using an open source solution with a MySQL database, prepares it and presents it to all stakeholders in an intuitive and understandable way with a server based PHP GUI. [21]

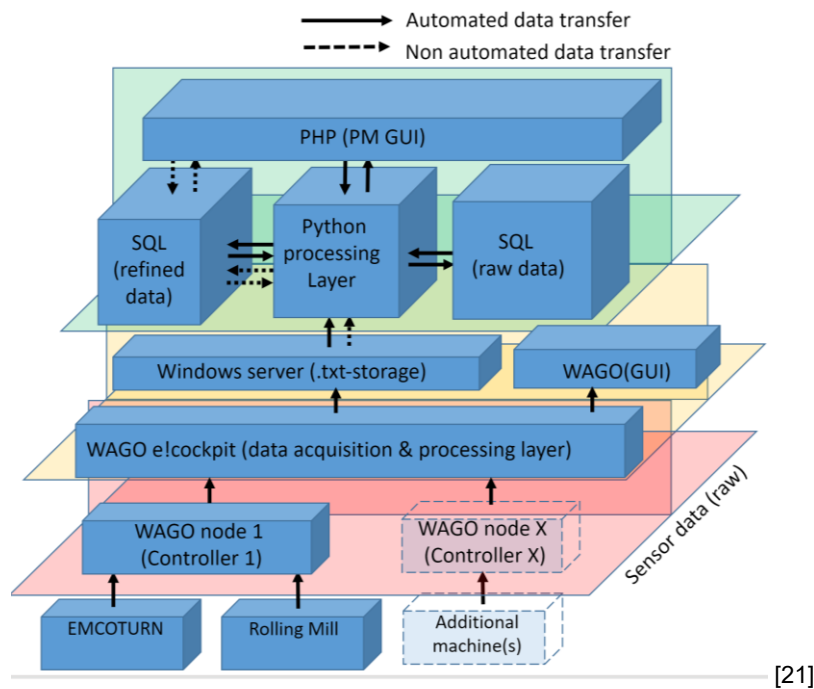


Figure 36 Multi-layer architecture of the chair of metal forming

As displayed in figure 36 above, the PMT:CMF is one of the two different Graphical User Interfaces which are developed to support respective workers. The PMT:CMF is a key component to support the chair of metal forming to implement condition monitoring to its machines. [21]

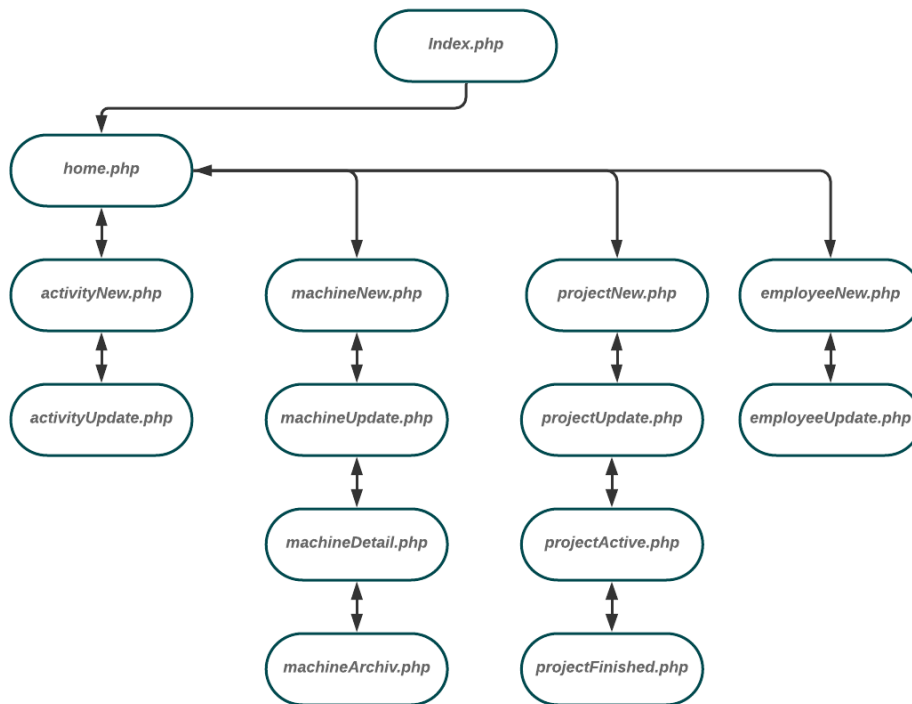


Figure 37 Visible pages for the users

As shown in figure 37, the tool is accessed via the *index.php* page. This is linked to the page of the chair of metal forming and is only available when it is accessed within the university network. If the user has a correct login and password, he/she is able to access the *home.php*, which allows a general overview over the projects and machines. The Graphical User Interface (GUI) of the PMT:CMF is presented on the next pages. Due to previous explanations, the details are kept brief.

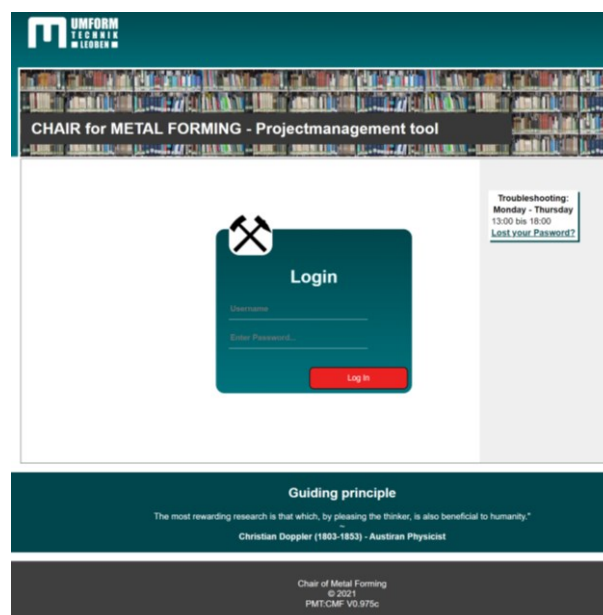


Figure 38 Index page with login for user

Figure 38 shows the GUI of the page *index.php* and the input fields for the users. On the right-hand side of the figure a note is displayed, indicating that if a user does not have his/her login details to hand, the link will take him/her to the chair's administration. The admins can tell the requesting users login name and set a new password. It is not possible to retrieve the password within the PHP pages or the SQL database, because the entered password gets hashed, when it is entered and only get decrypted if the user logs in the PMT:CMF *index.php* page.

UMFORM TECHNIK LEUBEN
Good Morning, CMF Admin

CHAIR of METAL FORMING

IMPORTANT DATES

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY

PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS

MACHINES
NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE

EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Projects

Project number	Project name	Project type	Contact	Start date	Delivery date	Budget	Costs
CMF123456	Conclusion Testing	Contract reasearch	Student	06.02.2021	06.02.2022	150 243,00 €	36,70 €
32467	Testing 5	Application reasearch	Dissertant	01.12.2020	17.03.2021	1 250,00 €	246,45 €
20210215002	Test update project	Application reasearch	Professor	08.02.2021	21.02.2021	9 000,00 €	0,00 €
34576	updateTesting	other	Professor	30.11.-0001	01.01.2021	147 852,00 €	1 069,30 €

Machine Maintenance

Machine number	Machine name	Machine type	Next maintenance	Last maintenance	Purchase date	Hour Rate [EUR]	Asset value [EUR]	Last activity end
0000	Personal Computer	PC	01.06.2021	01.02.2021	04.01.2021	1,10 €	0,00 €	03:00 12.02.2021
hello	EMCOTURN E65	Turn-Mill Center	23.12.2021	28.12.2020	07.08.2014	5,00 €	0,00 €	10:00 16.02.2021
147852	Gleeble	TMTS	18.04.2021	16.12.2020	15.10.2020	3,00 €	0,00 €	06:56 28.12.2020
123456	Forming aggregate	Former	24.02.2021	28.08.2020	28.12.2020	2,50 €	0,00 €	07:05 28.12.2020
741852	Water Jet ISK 2016	Cutter	13.02.2021	19.02.2020	15.12.2020	4,75 €	0,00 €	07:07 28.12.2020

Employee/Logs

Full name	Login name	Role	End of contract	Last Project
Student	TesterStud	Student	14.02.2022	30.11.-0001
Professor	TesterProf	Technichian	30.06.2022	30.11.-0001
Dissertant	TesterDis	Assistant	28.02.2022	30.11.-0001
Assistant	TesterAssist	Admin	01.01.2022	30.11.-0001
Administrator	TesterAdmin	Admin	31.05.2022	30.11.-0001

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT:CMF V1.212m

Figure 39 home.php of Administration

As shown in figure 39, the general information of the PMT:CMF is displayed on the *home.php* page. Additionally, completion dates of the projects and maintenance dates of the machines are shown.

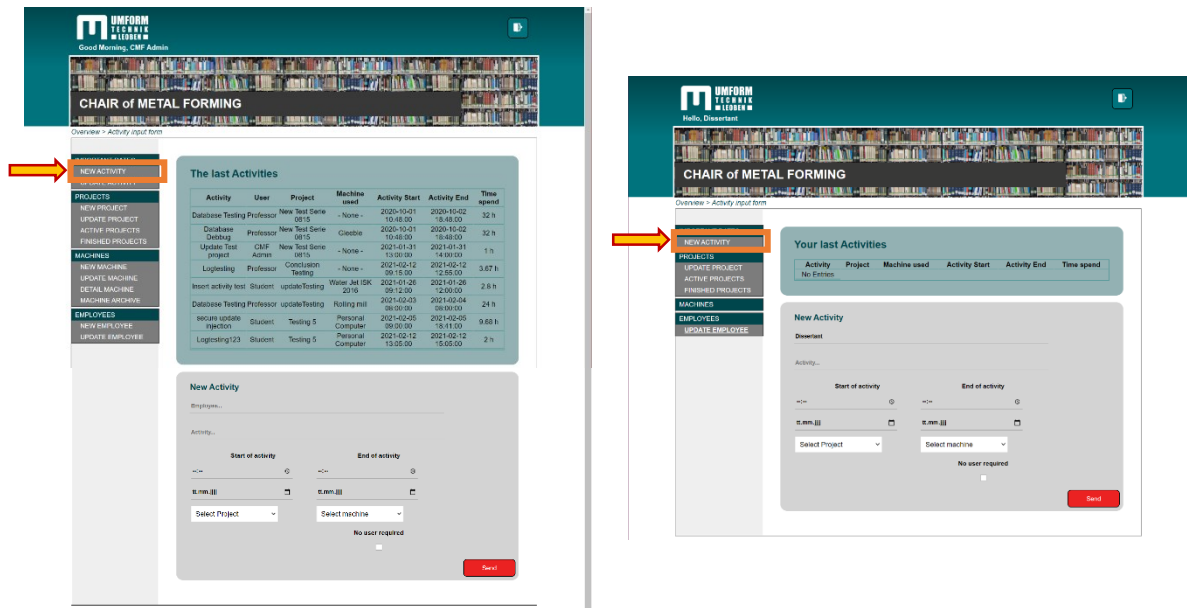


Figure 40 Difference of activities page for admin (left) and user (right)

As pictured in figure 40 the admin may enter activities for every signed user. The user, on the other hand, can only create activities in his or her own name. Another difference is the entered activities themselves. Here, the user only sees his own activities, whereas the admin sees all activities with machine and employee.

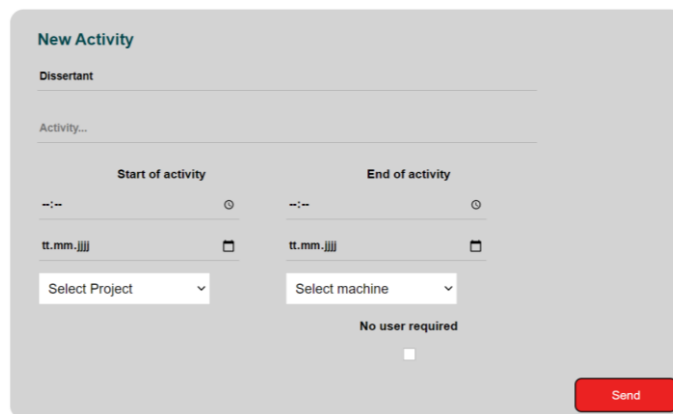


Figure 41 Input mask for new user activities

The entry of the activities requires that, in addition to the name of the activity, the start and end time are also entered. In addition, a project must be selected and, if applicable, the machine used must be selected. As mentioned before, an admin could also input a name of any employee.

UNIFORM
TECHNIK
LEBEN
Good Morning, CMF Admin

CHAIR of METAL FORMING

Overview > Activity update form

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY
PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS
MACHINES
NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE
EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Your last Activities

Activity	User	Project	Machine used	Activity Start	Activity End	Time spend
Database Testing	Professor	New Test Serie 0815	- None -	2020-10-01 10:48:00	2020-10-02 18:48:00	32 h
Database Debug	Professor	New Test Serie 0815	Gleeble	2020-10-01 10:48:00	2020-10-02 18:48:00	32 h
Update Test project	CMF Admin	New Test Serie 0815	- None -	2021-01-31 13:00:00	2021-01-31 14:00:00	1 h
Logtesting	Professor	Conclusion Testing	- None -	2021-02-12 09:15:00	2021-02-12 12:55:00	3.67 h
Insert activity test	Student	updateTesting	Water Jet ISK 2016	2021-01-26 09:12:00	2021-01-26 12:00:00	2.8 h
Database Testing	Professor	updateTesting	Rolling mill	2021-02-03 08:00:00	2021-02-04 08:00:00	24 h
secure update injection	Student	Testing 5	Personal Computer	2021-02-05 09:00:00	2021-02-05 18:41:00	9.68 h
Logtesting123	Student	Testing 5	Personal Computer	2021-02-12 13:05:00	2021-02-12 15:05:00	2 h

Update Activity

Select activity

Select Employee

Activity name:

Start of activity: End of activity:

Select Project Select machine

No user required

Send

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT.CMF V1.212m

Figure 42 Activity Update page for administration

The activity update page for an administrator has the same input possibilities as the create activity page shown in figure 42. An administrator is able to change every activity and may even change the employee of the activity.

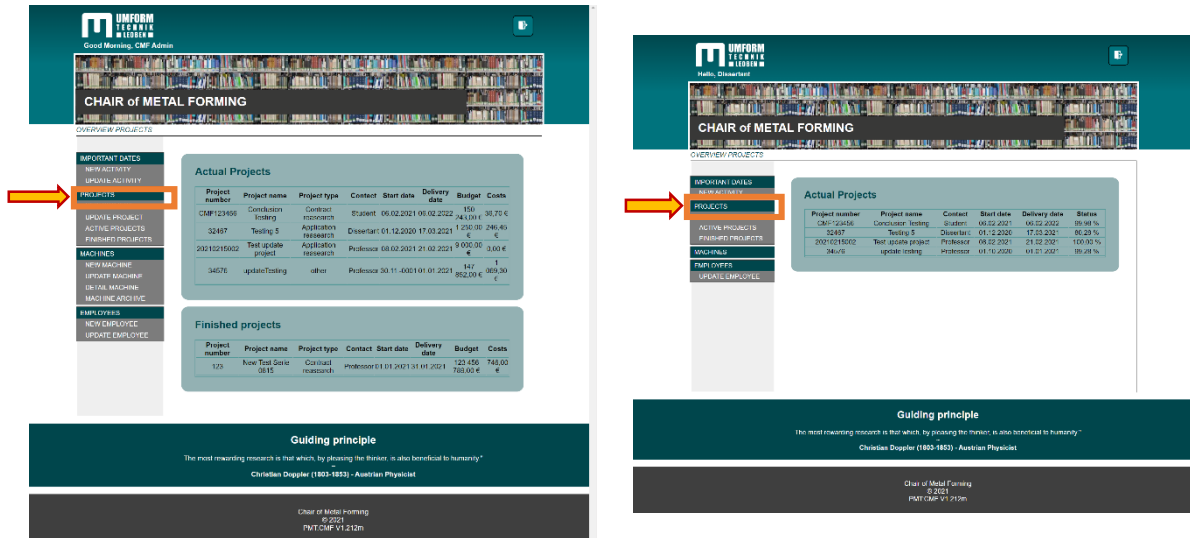


Figure 43 Project page of admin (left) and user (right)

As shown in figure 43, admins can view all projects on, as well as the budget and costs of each project. A user without project management functions sees a percentage status instead.

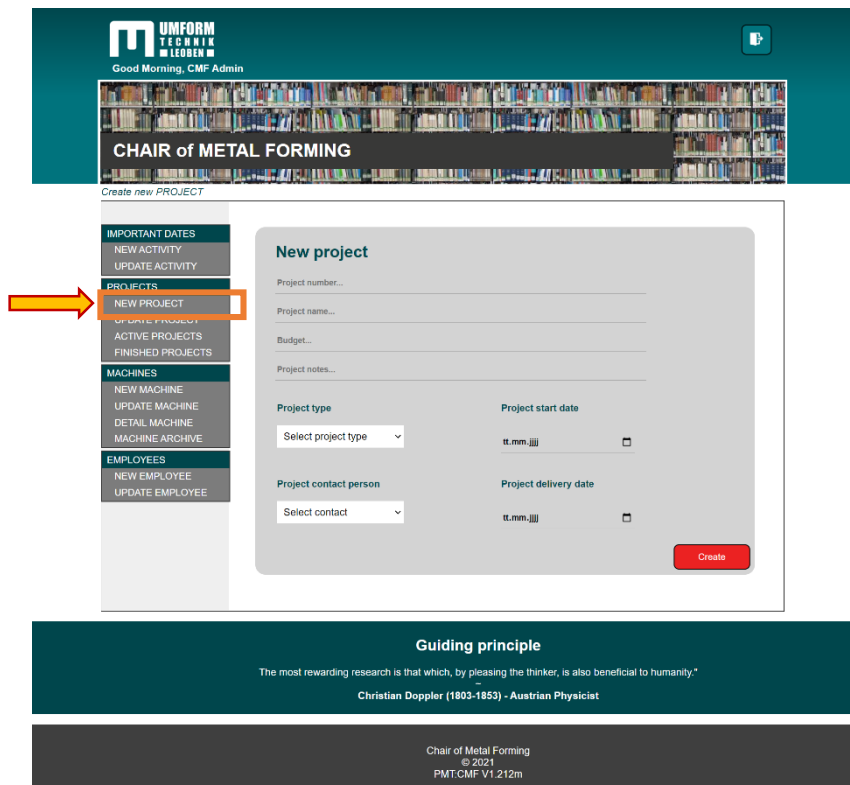


Figure 44 New project page

In coordination with the stakeholder analysis, administrators are tasked with creating new projects in the PMT:CMF. This is done using the ProjectNew.php page shown in figure 44. A predefined project number, name and budget are entered, then the project type and contact

person are selected using the drop-down menu. Finally, the start date and the end date of the project are entered. The entries are mandatory, otherwise the project cannot be created. Finally, with the Create button the input data is transferred to the system and stored in the database.

The screenshot displays the 'Update Project' interface. On the left, a sidebar menu lists various system functions, with 'UPDATE PROJECT' highlighted by an orange arrow. The main content area is divided into two sections: 'Actual projects' and 'Update Project'.

Actual projects table:

Project number	Project name	Project type	Contact	Start date	Delivery date	Budget	Costs
CMF123456	Conclusion Testing	Contract research	Student	06.02.2021	06.02.2022	150 243,00 €	36,70 €
32467	Testing 5	Application research	Dissertant	01.12.2020	17.03.2021	1.250,00 €	246,45 €
20210215002	Test update project	Application research	Professor	08.02.2021	21.02.2021	9.000,00 €	0,00 €
34576	updateTesting	other	Professor	30.11.-0001	01.01.2021	147 852,00 €	1 069,30 €

Update Project form:

- Select project name:** A dropdown menu labeled 'Select project'.
- Project number:** A text input field.
- Project name:** A text input field.
- Budget:** A text input field.
- Project notes:** A text input field.
- Project type:** A dropdown menu labeled 'Select project type'.
- Project start date:** A date picker labeled 't.mm.jjj' with a calendar icon.
- Project contact person:** A dropdown menu labeled 'Select contact'.
- Project delivery date:** A date picker labeled 't.mm.jjj' with a calendar icon.
- Project is finished:** A checkbox.
- Update project:** A red button at the bottom right.

At the bottom of the page, there is a 'Guiding principle' section with a quote by Christian Doppler (1803-1883) and a footer for the Chair of Metal Forming, © 2021, PMT.CMF V1.212m.

Figure 45 Update project page

To modify a project, administrators and project managers can do so using the input mask shown in figure 45. This GUI allows any aspect of the project to be changed. It is important to select the relevant project in order to change the correct entries. A modification of the project can be done at any time, as all details are automatically updated to the last status of the project. Another important point is the checkbox to conclude a project. All project data is calculated again and saved separately. This was requested by the stakeholders in order to obtain a reference value for future projects.

UNIFORM
TECHNIK
LEBEN
Good Morning, CMF Admin

CHAIR of METAL FORMING

Overview projects > ACTUAL PROJECT DETAIL

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY

PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS

MACHINES
NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE

EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Actual projects

Project number	Project name	Project type	Contact	Start date	Delivery date	Budget	Costs
CMF123456	Conclusion Testing	Contract research	Student	06.02.2021	06.02.2022	150 243,00 €	36,70 €
32467	Testing 5	Application research	Dissertant	01.12.2020	17.03.2021	1 250,00 €	246,45 €
20210215002	Test update project	Application research	Professor	08.02.2021	21.02.2021	9 000,00 €	0,00 €
34576	updateTesting	other	Professor	30.11.-0001	01.01.2021	147 852,00 €	1 069,30 €

Select project Start date End date Show Detail

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.*
~
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT:CMF V1-212m

Figure 46 Project detail page

Another stakeholder requirement is to show details of ongoing projects to support project meetings. As shown in figure 46, a project is selected by using the drop-down menu. It is possible to query the projects for specific time periods using the inputs displayed next to them. This influences the activities carried out and, accordingly, the staff and machines involved. A separate page has been created for the completed projects.

UMFORM
TECHNIK
LEBEN
Good Morning, CMF Admin

CHAIR of METAL FORMING

Overview projects > ACTUAL PROJECT DETAIL

ACTIVE PROJECTS

Project number	Project name	Project type	Contact	Start date	Delivery date	Budget	Costs
CMF 123456	Conclusion Testing	Contract reasearch	Student	06.02.2021	06.02.2022	150 243,00 €	36,70 €
32467	Testing 5	Application reasearch	Dissertant	01.12.2020	17.03.2021	1 250,00 €	246,45 €
20210215002	Test update project	Application reasearch	Professor	08.02.2021	21.02.2021	9 000,00 €	0,00 €
34576	updateTesting	other	Professor	30.11.-0001	01.01.2021	147 852,00 €	1 069,30 €

Select project: [dropdown] Start date: [input] End date: [input] Show Detail: [button]

32467 / Testing 5

Project type	Application reasearch
Project Contact	Dissertant
Start date	01.12.2020
Delivery date	17.03.2021
Days till delivery	13
Workhours employees	11,68 h
Workhours machines	11,68 h
Costs employees	233,60 €
Costs machines	12,85 €
Last activity	15:00 - 12.02.2021
Notes	Crative input

Involved machines

Name	Hours	Activity start	Activity end
Personal Computer	9,68	09.02 - 05.02.2021	08.02 - 05.02.2021
Personal Computer	2,00	01.02 - 12.02.2021	03.02 - 12.02.2021

Involved employees

Name	Hours	Start activity	End activity
Student	9,68	09.02 - 05.02.2021	08.02 - 05.02.2021
Student	2	01.02 - 12.02.2021	03.02 - 12.02.2021

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.*
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT:CMF V1.212m

Figure 47 Project Detail page with involved items

Figure 47 shows an example project with the corresponding entries. In addition to the basic project data, the remaining days until project completion are also displayed. Furthermore, the total costs incurred and the hours worked by employees and machines are displayed, hours worked without machines are not added. Notes on the project are also displayed and can be read out.

UMFORM
TECHNIK
LEIBNIZ
Good Morning, CMF Admin

CHAIR of METAL FORMING

Overview projects > FINISHED PROJECT DETAIL

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY

PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS
MACHINES
NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE

EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Finished projects

Project number	Project name	Project type	Contact	Start date	Delivery date	Budget	Costs
123	New Test Serie 0815	Contract reasearch	Professor	01.01.2021	31.01.2021	123 456 789,00 €	746,00 €

Select project Start date End date [Show Detail](#)

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.*
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT:CMF V1.212m

Figure 48 Finished projects detail page

Figure 48 shows the completed detail page, which displays the completed projects. The process of completing a project must be done manually by a project manager or an admin, as some data is transferred to the database that would otherwise be changed by changes to the fees. A finished project is selected using the drop-down menu. As mentioned before here again it is possible to query the projects for specific time periods using the inputs displayed next to them. The details displayed are identical to those of the active projects, with the difference that they no longer change due to staff and fee changes.

UMFORM
TECHNIK
LEOBEN
Good Morning, CMF Admin

CHAIR of METAL FORMING

OVERVIEW MACHINES

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY

PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS

MACHINES

NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE

EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Actual machines

Machine number	Machine name	Machine type	Next maintenance	Last maintenance	Purchase date	Hour Rate [EUR]	Asset value [EUR]	Last activity end
0000	Personal Computer	PC	01.06.2021	01.02.2021	04.01.2021	1,10 €	0,00 €	03:00 12.02.2021
hello	EMCOTURN E65	Turn-Mill Center	23.12.2021	28.12.2020	07.08.2014	5,00 €	0,00 €	10:00 16.02.2021
147852	Gleeble	TMTS	18.04.2021	16.12.2020	15.10.2020	3,00 €	0,00 €	06:56 28.12.2020
123456	Forming aggregate	Former	24.02.2021	28.08.2020	28.12.2020	2,50 €	0,00 €	07:05 28.12.2020
741852	Water Jet ISK 2016	Cutter	13.02.2021	19.02.2020	15.12.2020	4,75 €	0,00 €	07:07 28.12.2020

Inactive machines

Machine number	Machine name	Machine type	Next maintenance	Last maintenance	Purchase date	Hour Rate [EUR]	Asset value [EUR]	Last activity end
369528	Rolling mill	Rolling	12.02.2021	15.10.2020	15.10.2005	10,50 €	0,00 €	02:38 07.01.2021

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.
—
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT:CMF V1.212m

Figure 49 Machines overview page

To view all machine information admins and technicians can do this with the page shown in figure 49. Next to the machine number and name, the type and dates for the next maintenance are visualised. In addition to the active machines, the inactive machines are also displayed.

Figure 50 Machine page for new machines with input mask

At the *machineNew.php* page, shown in the figure 50, it is possible for the admin to add new machines to the project management tool. In addition to entering the specific machine number and a name, the type, notes and information on the purchase of the machine can also be entered. This includes the date of purchase, purchase price and machine hour cost rates, as well as maintenance data and the utilization of the machine.

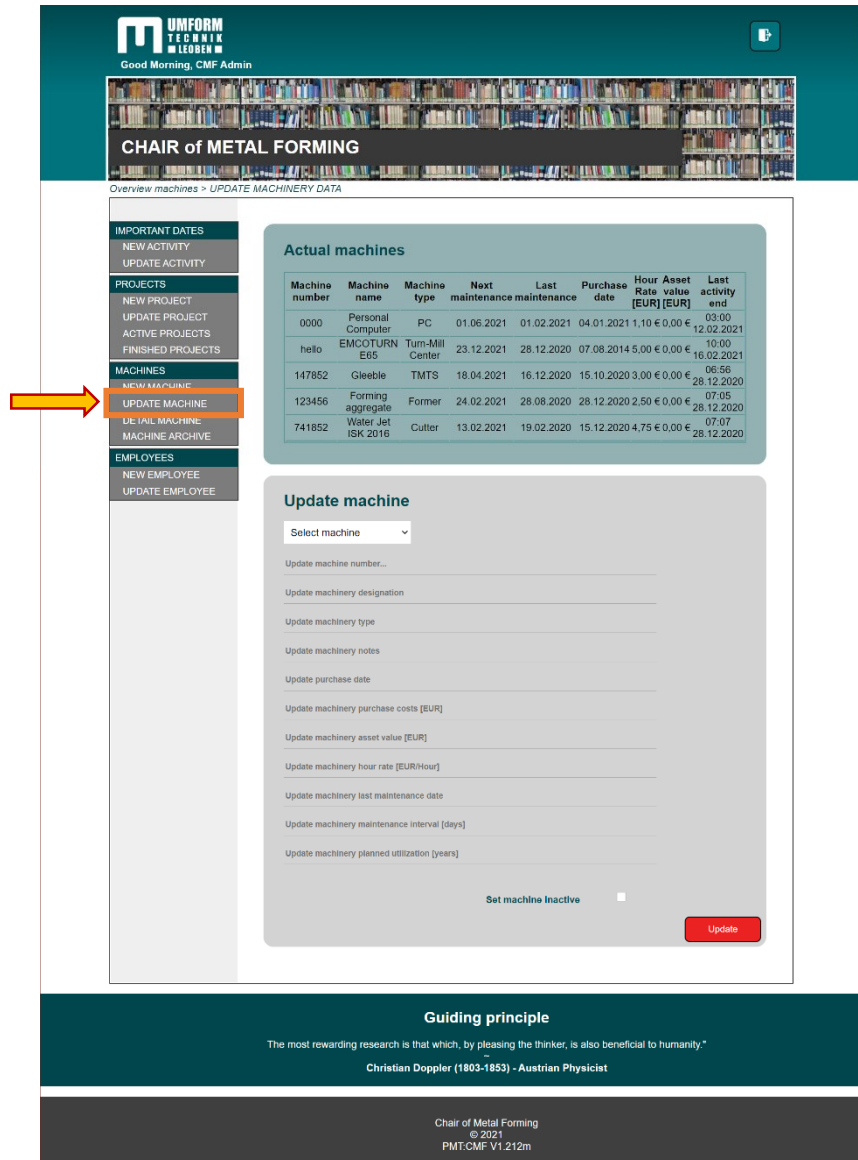
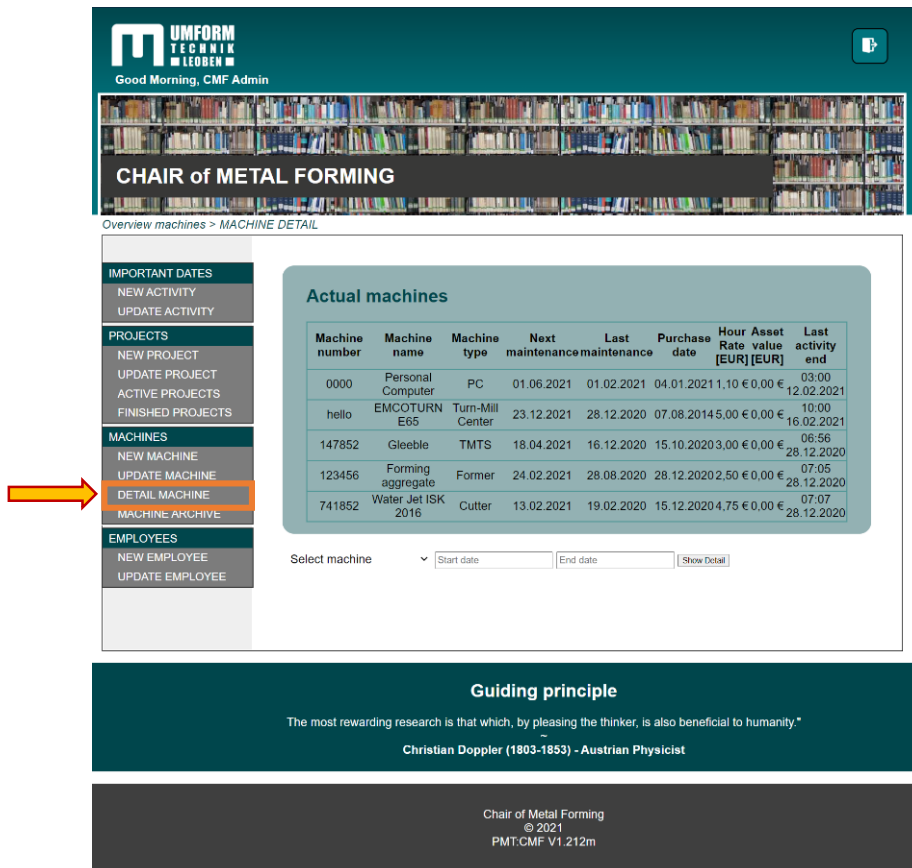


Figure 51 Machines update page with input mask

To update or change entries of a machine, administrators and technicians can do so using the input mask shown in figure 51. In the input mask shown, every aspect of the machine can be changed. It is important to select the appropriate machine to change the correct entries. A change to machine can be made at any time, as all entries are automatically updated in the activities to the latest status of the machine. Another important point is the checkbox to deactivate a machine. This is necessary in order to continue to display activities with the machine in the completed projects.



UMFORM
TECHNIK
LEOBEN

Good Morning, CMF Admin

CHAIR of METAL FORMING

Overview machines > MACHINE DETAIL

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY

PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS

MACHINES
NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE

EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Actual machines

Machine number	Machine name	Machine type	Next maintenance	Last maintenance	Purchase date	Hour Rate [EUR]	Asset value [EUR]	Last activity end
0000	Personal Computer	PC	01.06.2021	01.02.2021	04.01.2021	1,10 €	0,00 €	03.00 12.02.2021
hello	EMCOTURN E65	Turn-Mill Center	23.12.2021	28.12.2020	07.08.2014	5,00 €	0,00 €	10.00 16.02.2021
147852	Gleeble	TMTS	18.04.2021	16.12.2020	15.10.2020	3,00 €	0,00 €	06:56 28.12.2020
123456	Forming aggregate	Former	24.02.2021	28.08.2020	28.12.2020	2,50 €	0,00 €	07:05 28.12.2020
741852	Water Jet ISK 2016	Cutter	13.02.2021	19.02.2020	15.12.2020	4,75 €	0,00 €	07:07 28.12.2020

Select machine Start date End date [Show Detail](#)

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity.*
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT.CMF.V1.212m

Figure 52 Machine detail page

The machine details are displayed in the same way as described above for the projects. By selecting an active machine, the details can be called up as shown in figure 52. In the same way as mentioned before, certain time periods can be set to limit the activity display.

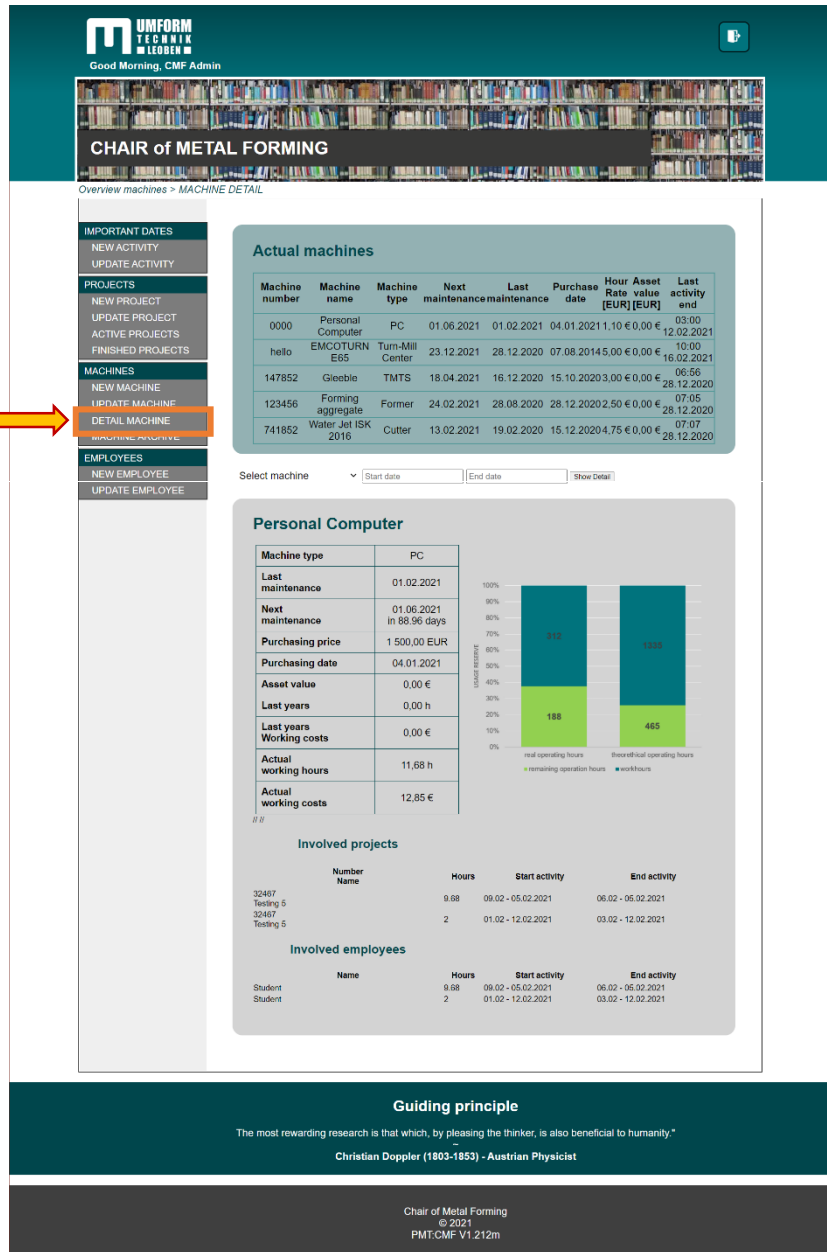


Figure 53 Machine detail list with involved hours items

Figure 53 shows an exemplary machine with the corresponding entries. In addition to the basic machine data, the remaining days until the next maintenance are also displayed. Furthermore, the asset value, the costs incurred and the working hours of the last and the current year are displayed and separated according to employees and machines. Notes on the machines are also displayed. The shown diagram is an input from a python script running on the chair's internal server. This script transfers the data to the database and creates the diagram, so that the detail page can display it accordingly to the machine name.

UNIFORM
TECHNIK
IM LEBEN

Hello, CMF Admin

CHAIR of METAL FORMING

Overview machines > MACHINE DETAIL

IMPORTANT DATES
NEW ACTIVITY
UPDATE ACTIVITY

PROJECTS
NEW PROJECT
UPDATE PROJECT
ACTIVE PROJECTS
FINISHED PROJECTS

MACHINES
NEW MACHINE
UPDATE MACHINE
DETAIL MACHINE
MACHINE ARCHIVE

EMPLOYEES
NEW EMPLOYEE
UPDATE EMPLOYEE

Machine archiv

Machine number	Machine name	Machine type	Next maintenance	Last maintenance	Purchase date	Hour Rate [EUR]	Asset value [EUR]	Last activity end
369528	Rolling mill	Rolling	12.02.2021	15.10.2020	15.10.2005	10,50 €	0,00 €	02:38 07.01.2021

Select machine [Show Detail](#)

Guiding principle
The most rewarding research is that which, by pleasing the thinker, is also beneficial to humanity."
~
Christian Doppler (1803-1853) - Austrian Physicist

Chair of Metal Forming
© 2021
PMT/CMF V1.212m

Figure 54 Machine archive page

Figure 54 shows the inactive machine details input page. The process of inactivating a machine must be done manually by a technician or an admin, as data is transferred to the database. An inactive machine is selected from the drop down menu. By using the items displayed next to it, it is possible to query a specific time period. The details displayed are identical to those of the active machines.

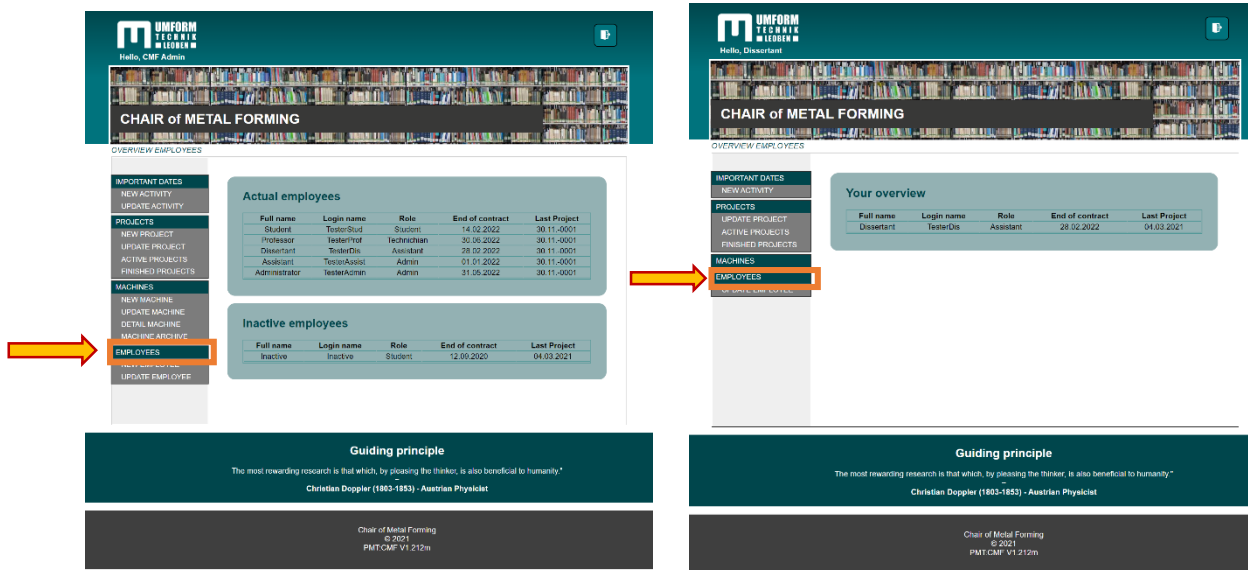


Figure 55 Employee page of an admin (left) and user (right)

The stakeholder analysis showed that the separation of personal data is necessary. As shown in Figure 55 above, an admin can see all users and their data regarding current contracts. The user, on the other hand, is only shown his own data.

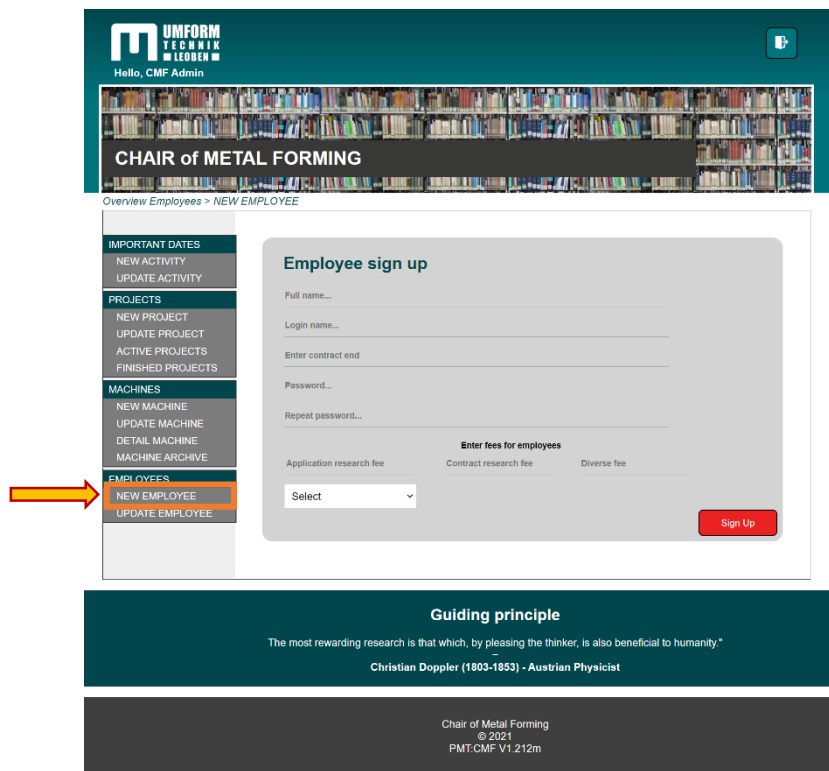


Figure 56 Employee sign up page

As determined by the stakeholder analysis, only admins are allowed to register new users. This is done with the input mask shown in the figure 56. In addition to the user name, a login name and an initial password is assigned. This must be entered repeatedly as a verification measure. Next, the end date of the contract is entered. After this, the administrator can create cost rates for the various project types. These are used for the activities carried out in order to provide the activities with the costs incurred. Finally, the role of the new user is defined, which regulates their access rights/views in the project management tool.

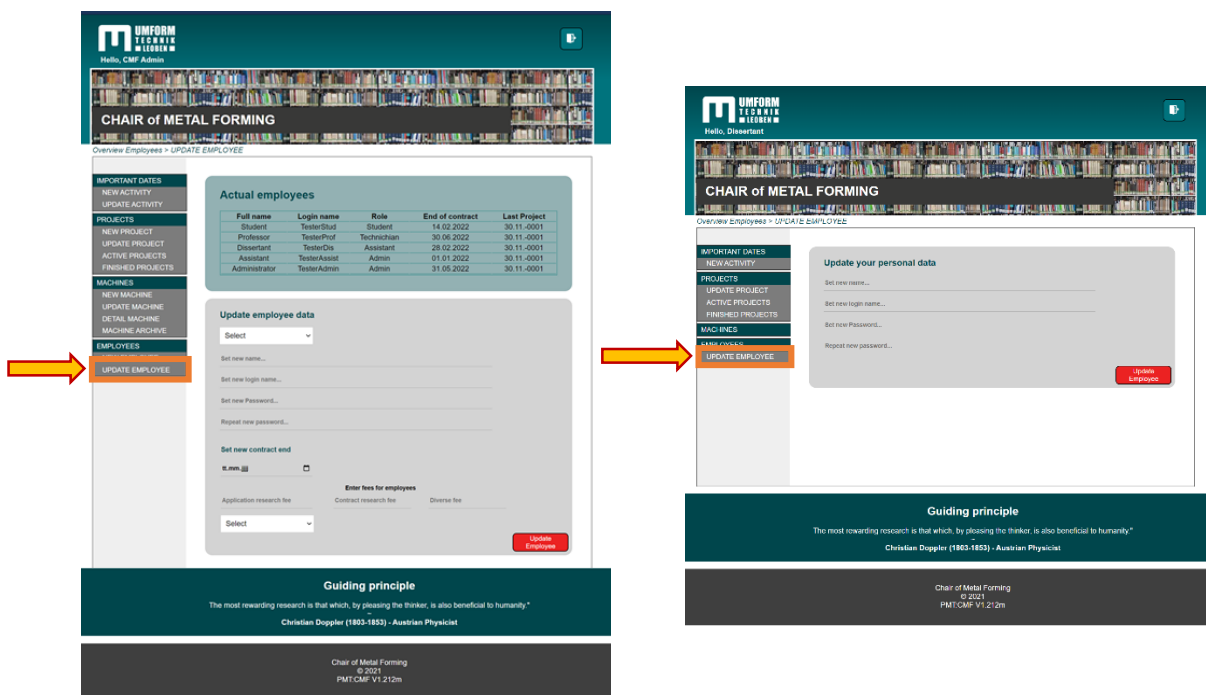


Figure 57 Employee page of an admin (left) and user (right)

Following previous definition, an admin can access any user data. This is also the case here on the update page shown in the figure 57. User without the administration role, can only access and change their own data records. The data regarding costs, contract duration and user roles cannot be changed by the user himself but by the admin. These changes take effect after a new logon to the system.

5 Conclusion and outlook

The digital project management tool (PMT:CMF) used, has enabled a multi-layered networked solution for recording project, machine and employee data. As shown in the stakeholder analysis, the activities as well as the times required for employees and machines used are also stored uniformly and processed for further evaluation. The tool supports the clear and uniform presentation of project data and enables a detailed listing of activities carried out, as well as machines used. Through the connection to the machine sensors, the direct machine consumption can be retrieved and converted into real costs with the help of a defined energy price.

As explained in the feasibility study, the open source solutions PHP and MySQL were used for the creation of this framework. The use of these solutions considerably reduces the costs and creates an end-to-end, adaptable solution. The PHP graphical user interface connects the sensor data of the machines with the project activities using the data storage in the MySQL database of the tool.

As the PMT:CMF goes live, further adaptations are made in the areas of user input and project tracking to achieve maximum effectiveness. As a result, inputs, displays and associated code snippets may change, due to the continuous improvement approach of the chair of metal forming.

The next steps in extending the project management tool could be to connect the tool directly to other machines. The connection is made via a Python script that is already in use for a machine at the institute. A connection to a higher-level MES system that can obtain data directly from the MySQL database is also being considered. The data entered in the project management tool is stored in the MySQL database and is to be read out and processed in this future system. This process of accessing the database and reading out the data as well as transferring it to a higher-level MES/ERP system is supported by a Python script.

For all this planned activities, this thesis and corresponding work serves as fundamental basis for further improvement and digitalisation.

References

- [1] J. Tupa, J. Simota, and F. Steiner, "Aspects of risk management implementation for Industry 4.0," *Procedia manufacturing*, vol. 11, pp. 1223–1230, 2017.
- [2] L. Safar, J. Sopko, S. Bednar, and R. Poklemba, "Concept of SME business model for industry 4.0 environment," *TEM Journal*, vol. 7, no. 3, p. 626, 2018.
- [3] C. Desmond, "Project management tools-software tools," *IEEE Engineering Management Review*, vol. 45, no. 4, pp. 24–25, 2017.
- [4] J. M. Müller and S. Däschle, "Business model innovation of industry 4.0 solution providers towards customer process innovation," *Processes*, vol. 6, no. 12, p. 260, 2018.
- [5] D. Mourtzis, E. Vlachou, N. Milas, and N. Xanthopoulos, "A cloud-based approach for maintenance of machine tools and equipment based on shop-floor monitoring," *Procedia Cirp*, vol. 41, pp. 655–660, 2016.
- [6] Z. Alansari *et al.*, "Challenges of internet of things and big data integration," in *International Conference for Emerging Technologies in Computing*, pp. 47–55.
- [7] A. Clarke, "A practical use of key success factors to improve the effectiveness of project management," *International journal of project management*, vol. 17, no. 3, pp. 139–145, 1999.
- [8] A. K. Munns and B. F. Bjeirmi, "The role of project management in achieving project success," *International journal of project management*, vol. 14, no. 2, pp. 81–87, 1996.
- [9] M. Pannu, Q. Salih, C. Yuen, Z. H. Li, and E. Tanu, "Web based Project Management Systems for small to midsize businesses," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, 112018, pp. 1233–1237.
- [10] *Deductive and object-oriented databases*: Elsevier, 1990.
- [11] J. Bishop, C. Jensen, W. Scacchi, and A. Smith, "How to use open source software in education," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pp. 321–322.
- [12] A. I. Vlasov, P. V. Grigoriev, A. I. Krivoshein, V. A. Shakhnov, S. S. Filin, and V. S. Migalin, "Smart management of technologies: predictive maintenance of industrial equipment using wireless sensor networks," *Entrepreneurship and Sustainability Issues*, vol. 6, no. 2, pp. 489–502, 2018.
- [13] R. Y. Zhong, L. Wang, and X. Xu, "An IoT-enabled real-time machine status monitoring approach for cloud manufacturing," *Procedia Cirp*, vol. 63, pp. 709–714, 2017.
- [14] R. Felkai and A. Beiderwieden, *Projektmanagement für technische Projekte*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015.

- [15] F. Ackermann and C. Eden, *Making strategy: Mapping out strategic success*, 2nd ed. London: SAGE, 2012.
- [16] M. Nasar and M. A. Kausar, "Suitability of influxdb database for iot applications," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 1850–1857, 2019.
- [17] S. N. Z. Naqvi, S. Yfantidou, and E. Zimányi, "Time series databases and influxdb," *Studienarbeit, Université Libre de Bruxelles*, p. 12, 2017.
- [18] D. Edler and M. Vetter, "The simplicity of Modern audiovisual web cartography: an example with the open-source JavaScript Library leaflet.js," *KN-Journal of Cartography and Geographic Information*, vol. 69, no. 1, pp. 51–62, 2019.
- [19] L. Welling and L. Thomson, *PHP and MySQL Web development*: Sams Publishing, 2003.
- [20] J. W. Krogh, *MySQL Connector/Python Revealed: SQL and NoSQL Data Storage Using MySQL for Python Programmers*. Berkeley, CA: Apress, 2018.
- [21] B. Ralph, M. Sorger, B. Schödinger, H.-J. Schmölzer, C. Stöckl, and M. Stockinger, "Development and implementation of a six-layer low cost resilient condition monitoring and predictive maintenance system," *Procedia manufacturing*, vol. 2021, 1-8 (submitted).

Appendix

To ensure the completeness of the thesis, all core functions are presented in alphabetical order in this section. As mentioned before, this is version V1.212m of the PMT:CMF.

List of functions

○ costsMachineYear(\$conn, \$id)	XIII
○ costsMachineActual(\$conn, \$id)	XIII
○ checkAccess(\$conn, \$user)	XIV
○ checkActivity(\$conn, \$user)	XV
○ checkContact(\$conn, \$user)	XV
○ checkMachine(\$conn, \$user)	XV
○ checkUser()	XV
○ createActivity(\$conn, \$aName, \$uName, \$pName, \$mName, \$sactStart, \$sactEnd, \$noUser)	XVI
○ createMachine(\$conn, \$number, \$name, \$type, \$note, \$purDate, \$purCost, \$hourRate, \$maintInterval, \$util)	XVII
○ createProject(\$conn, \$number, \$name, \$contact, \$type, \$budget, \$startDate, \$endDate, \$note)	XVIII
○ createUser(\$conn, \$name, \$login, \$contractEnd, \$role, \$fee, \$afee, \$sfee, \$pwd)	XVIII
○ costsProject(\$conn, \$projectId)	XIV
○ dateDifference(\$start, \$end, \$type)	XXI
○ ddlItem(\$conn, \$type, \$activ)	XXI
○ detailMachine(\$conn, \$machine)	XXII
○ detailProject(\$conn, \$project)	XXIII
○ displayActivity(\$conn)	XXV
○ displayEmployees(\$conn, \$activ)	XXVI
○ displayMachines(\$conn, \$activ)	XXVII
○ displayProjects(\$conn, \$activ)	XXIX
○ greetings(\$fullName)	XXX
○ inactiveUser(\$conn)	XXXI
○ inputExists(\$conn, \$id, \$name, \$type)	XXXI
○ invalideMail(\$email)	XXXII
○ invalidName(\$name)	XXXII
○ involvedEmployee(\$conn, \$id, \$start, \$end, \$type)	XXXIII
○ involvedMachine(\$conn, \$id, \$start, \$end, \$type)	XXXIII
○ involvedProject(\$conn, \$id, \$start, \$end, \$type)	XXXV
○ loginUser(\$conn, \$username, \$pwd)	XXXVI
○ pwdMatch(\$pwd, \$pwdrepeat)	XXXVII
○ updateActivity(\$conn, \$aID, \$aName, \$uName, \$pName, \$mName, \$sactStart, \$sactEnd, \$noUser)	XXXVII
○ updateMachine(\$conn, \$mid, \$number, \$name, \$type, \$note, \$purDate, \$purCost, \$hourRate, \$assetValue, \$util, \$maintInterval, \$maintLast, \$inactive)	XXXVIII
○ updateProject(\$conn, \$pID, \$number, \$name, \$contact, \$type, \$budget, \$startDate, \$endDate, \$note, \$finished)	XXXIX
○ updateUser(\$conn, \$userID, \$name, \$login, \$contractEnd, \$role, \$fee, \$afee, \$sfee, \$pwd)	XL

```

function costsMachineYear($conn, $id){
    /*
        function to return machine costs a an array with 4 entries
        Depreciation, last year's costs ,last year's working hours, running
        costs, this years hours
        returns array($depre, $coLastYear, $whLastYear, $coActual, whActual);
    */

    $sql = "SELECT actUsName, actUsProject, macName, macHourRate,
                actUsStartDate, actUsEndDate, actUsNoUser
            FROM activityuser
            INNER JOIN user ON activityuser.actUsUser = user.usID
            INNER JOIN machines ON activityuser.actUsMachine = machines.macID
            INNER JOIN projects ON activityuser.actUsProject = projects.proID
            WHERE activityuser.actUsMachine = ".$id." AND actUsStartDate > YEAR(CUR
RENT_TIMESTAMP)-1 AND actUsStartDate < YEAR(CURRENT_TIMESTAMP);";

    $activities = mysqli_query($conn, $sql);
    if(!$activities){
        $activities = 0;
    }

    $whMac = 0;
    $coMac = 0;
    $hours = 0;

    if(mysqli_num_rows($activities) < 0){
        $costsArray = array($coEmp, $coMac);
        return $costsArray;
    } else {
        while($row = mysqli_fetch_assoc($activities)){
            // Sum of working hours of project emp/mac
            $hours = dateDifference($row["actUsStartDate"],
                $row["actUsEndDate"], 2);

            $whMac = $whMac + $hours;
            $coMac = $coMac + $row["macHourRate"] * $hours;
        }
        $costsArray = array($coMac, $whMac);
        return $costsArray;
    }
}

function costsMachineActual($conn, $id){
    /*
        function to return machine costs a an array with 4 entries
        Depreciation, last year's costs, last year's working hours,
        running costs, this years hours
        returns array($depre, $coLastYear, $whLastYear, $coActual, whActual);
    */

```

```

$sql = "SELECT actUsName, actUsProject, macName, macHourRate,
           actUsStartDate, actUsEndDate, actUsNoUser
        FROM activityuser
        INNER JOIN user ON activityuser.actUsUser = user.usID
        INNER JOIN machines ON activityuser.actUsMachine = machines.macID
        INNER JOIN projects ON activityuser.actUsProject = projects.proID
        WHERE activityuser.actUsMachine = ".$id." AND actUsStartDate >
        YEAR(CURRENT_TIMESTAMP) ";

$activities = mysqli_query($conn, $sql);
if(!$activities){
    $activities = 0;
}

$whMac = 0;
$coMac = 0;
$hours = 0;

if(mysqli_num_rows($activities) < 0){
    $costsArray = array($coEmp, $coMac);
    return $costsArray;
} else {
    while($row = mysqli_fetch_assoc($activities)){
        // Sum of working hours of project emp/mac
        $whMac = $whMac + dateDifference($row["actUsStartDate"],
                                         $row["actUsEndDate"], 2);;
        $coMac = $coMac + $row["macHourRate"] * dateDifference(
            $row["actUsStartDate"], $row["actUsEndDate"], 2);;
    }
    $costsArray = array($coMac, $whMac);
    return $costsArray;
}
}

function checkAccess($conn, $user){
    //Function for access control on several pages
    $access = inputExists($conn, $user, $user, 'employee');
    $sql = "SELECT proContact FROM projects WHERE proContact = '
           ".$access["usID"]." ";
    $results=mysqli_query($conn, $sql);

    // check role of employee
    if($access["usRole"] == 2 || $access["usRole"] == 3 ||
        $access["usRole"] == 101){
        return true;
    } else {
        return false;
    }
}
}

```

```

function checkActivity($conn, $user){
    $sql = "SELECT actUsID FROM activityUser
           WHERE actUsUser = '". $user. "'";

    $sql = mysqli_query($conn, $sql);
    if(mysqli_num_rows($sql)>=1){
        return true;
    } else {
        return false;
    }
}

function checkContact($conn, $user){
    //Function for access control on project update
    $access = inputExists($conn, $user, $user, 'employee');
    $sql = "SELECT proContact FROM projects
           WHERE proContact = '". $access["usID"]. "'";

    $results=mysqli_query($conn, $sql);
    // check role of employee
    if(mysqli_num_rows($results)>=1){
        return true;
    } else {
        return false;
    }
}

function checkMachine($conn, $user){
    $sql = "SELECT rolID FROM user
           INNER JOIN roles ON roles.rolID = user.usRole
           WHERE user.usID = '". $user. "'";

    $sql = mysqli_query($conn, $sql);
    $results=mysqli_fetch_assoc($sql);
    // check role of employee
    if($results["rolID"] == 5){
        return true;
    } else {
        return false;
    }
}

function checkUser(){
    //Function to secure site
    if(!isset($_SESSION['usID'])){
        header("location: ../PMTtoCMF/index.php?error=NoLogin");
        exit();
    }
}

```

```

function createActivity($conn, $aName, $uName, $pName, $mName,
                      $actStart, $actEnd, $noUser){
    // Prepare sql statement
    $sql = "INSERT INTO activityuser(actUsName, actUsUser, actUsProject,
        actUsMachine, actUsStartDate, actUsEndDate, actUsNoUser, actUsLog)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
    $stmt = mysqli_stmt_init($conn);

    // Check if input is valid
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../activityNew.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    // update the last entry
    $lastActivity = date("Y-m-d H:i:s");
    $user = inputExists($conn, $uName, $uName, 'employee');
    // Bind parameter to statement and execute
    if(mysqli_stmt_bind_param($stmt, "siiissis", $aName, $user["usID"],
        $pName, $mName, $actStart, $actEnd, $noUser, $lastActivity) == false){
        header("location: ../activityNew.php?error=stmt_bind_param_failed?
            activity:". $aName. "?user:". $user["usID"]. "?project:". $pName. "?
            machine:". $mName. "?Start:". $actStart. "?End:". $actEnd. "?
            lonely:". $noUser. "?lastLog". $lastUpdate);

        exit();
    }
    if(mysqli_stmt_execute($stmt) == false){
        header("location: ../activityNew.php?error=execute_failed?
            activity:". $aName. "?user:". $user["usID"]. "?project:".
            $pName. "?machine:". $mName. "?Start:". $actStart. "?Time:".
            $actSTime. "?End:". $actEnd. "?Time:". $actETime. "?lonely:".
            $noUser. "?lastLog". $lastUpdate);

        exit();
    }
    //update Project
    $logUpdate = date("Y-m-d H:i:s", strtotime($actEnd));
    $lastLog = "UPDATE projects
        SET proLastUpdate = '". $logUpdate. "'
        WHERE proID = '". $pName. "'";

    if(mysqli_query($conn, $lastLog) == false){
        header("location: ../activityNew.php?error=update_project_failed?
            Date:". date("Y-m-d h:a:s"). "?Project:". $pName);

        exit();
    }

    //update Machine
    $lastLog = "UPDATE machines
        SET macLastUpdate = '". $logUpdate. "'

```



```

        WHERE macID = '". $mName. "'";

if(mysqli_query($conn, $lastLog) == false){
    header("location: ../activityNew.php?error=update_machine_failed?
        Date:".date("Y-m-d h:a:s")."?Project:". $mName);
    exit();
}
//Close DB connection
mysqli_stmt_close($stmt);
mysqli_close($conn);
}

function createMachine($conn, $number, $name, $type, $note, $purDate,
    $purCost, $hourRate, $maintInterval, $util){
    // Check if machine already exist
    if(inputExists($conn, $name, $name, 'machine') != false){
        header("location: ../machineNew.php?error=MachineAlreadyExists");
        exit();
    }
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $sql = "INSERT INTO machines (macNumber, macName, macType, macNotes,
        macPurchaseDate, macPurchaseCosts, macHourRate, macMaintLast,
        macMaintInterval, macUtilisation)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

    // Check if input is valid
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../machineNew.php?error=stmt_prepare_failed");
        exit();
    }
    // Set first dates
    $maintLast = date("Y-m-d");
    // Bind parameter to statement and execute in SQL DB
    if(!mysqli_stmt_bind_param($stmt, "ssssdisii", $number, $name, $type,
        $note, $purDate, $purCost, $hourRate, $maintLast, $maintInterval,
        $util)){
        header("location: ../machineNew.php?error=stmt_bind_param_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_execute($stmt)){
        header("location: ../machineNew.php?error=stmt_execute_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    //Close DB connection
    mysqli_stmt_close($stmt);
}

```

```

    mysqli_close($conn);
}

function createProject($conn, $number, $name, $contact, $type,
    $budget, $startDate, $endDate, $note){
    // Check if Project already exists
    if(inputExists($conn, $name, $name, 'project') !== false){
        header("location: ../projectNew.php?error=ProjectExists");
        exit();
    }
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $sql = "INSERT INTO projects (proNumber, proName, proNote, proContact,
        proType, proBudget, proStartDate, proEndDate, proLastUpdate)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

    // Check if input is valid
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("location: ../projectNew.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    // last update when insert is taken
    $lastUpdate = date("Y-m-d");

    // Bind parameter to statement and execute in SQL DB
    $res = mysqli_stmt_bind_param($stmt, "ssssidsss", $number, $name, $note,
        $contact, $type, $budget, $startDate, $endDate, $lastUpdate);
    if($res == false){
        header("location: ../projectNew.php?error=stmt_bind_param_failed");
        exit();
    }

    if(mysqli_stmt_execute($stmt) == false){
        header("location: ../projectNew.php?error=execute_failed?".
            $startDate."?".$endDate);
        exit();
    }
    //Close DB connection
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
}

function createUser($conn, $name, $login, $contractEnd, $role, $cftee,
    $afee, $sfee, $pwd){
    // Check if User already exist
    if(inputExists($conn, $login, $login, 'employee') !== false){
        header("location: ../employeeNew.php?error=EmployeeAlreadyExists");
        exit();
    }
}

```

```

}
// Create the statements for interaction with SQL
$stmt = mysqli_stmt_init($conn);
$sql = "INSERT INTO user (usName, usLogin, usContract, usRole, usFee1,
    usFee2, usFee3, usPwd, usLastLogin) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

// Check if input is valid
if(!mysqli_stmt_prepare($stmt, $sql)){
    header("location: ../employeeNew.php?error=stmt_prepare_failed");
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
    exit();
}

// hash pwd
$hashedPwd = password_hash($pwd, PASSWORD_DEFAULT);
$lastLogin = date("Y-m-d");
// Bind parameter to statement and execute in SQL DB
if(!mysqli_stmt_bind_param($stmt, "sssiddss", $name, $login,
    $contractEnd, role, $cfee, $afee, $sfee, $hashedPwd, $lastLogin)){
    header("location: ../employeeNew.php?error=stmt_bind_param_failed");
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
    exit();
}

if(!mysqli_stmt_execute($stmt)){
    header("location: ../employeeNew.php?error=stmt_execute_failed");
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
    exit();
}
//Close DB connection
mysqli_stmt_close($stmt);
mysqli_close($conn);
}

function costsProject($conn, $proID){
    /*
        function to return project costs a an array with 4 entries
        costs Employee, Machine, Workhours Employee, Workhour Machine
        returns array($coEmp, $coMac, $whEmp, $whMac);
    */
    $sql = "SELECT actUsName, usID, usName, actUsProject, proType, actUsMachine,
        macName, macHourRate, actUsStartDate, actUsEndDate, actUsNoUser
        FROM activityuser
        INNER JOIN user ON activityuser.actUsUser = user.usID
        INNER JOIN machines ON activityuser.actUsMachine = machines.macID
        INNER JOIN projects ON activityuser.actUsProject = projects.proID
        WHERE activityuser.actUsProject = ".$proID."";
}

```

```

$activities = mysqli_query($conn, $sql);
if(!$activities){
    $activities = 0;
}
$whEmp = 0;
$coEmp = 0;
$whMac = 0;
$coMac = 0;
$hours = 0;

if(mysqli_num_rows($activities) < 0){
    $costsArray = array($coEmp, $coMac, $whEmp, $whMac);
    return $costsArray;
} else {
    while($row = mysqli_fetch_assoc($activities)){
        // Sum of working hours of project emp/mac
        $hours = dateDifference($row["actUsStartDate"],
            $row["actUsEndDate"], 2);
        if(!$row["actUsNoUser"]){
            $whEmp = $whEmp + $hours;
        }
        if($row["actUsMachine"]>1){
            $whMac = $whMac + $hours;
        }
        // Sum of costs of project emp/mac depending on project type
        if($row["proType"]==1){
            //Contract Research
            $sql = "SELECT usFee1 AS fee
                FROM user
                WHERE usID = ".$row["usID"].".";";
        }elseif($row["proType"]==2){
            //Application Research
            $sql = "SELECT usFee2 AS fee
                FROM user
                WHERE usID = ".$row["usID"].".";";
        }elseif($row["proType"]==3){
            //Other
            $sql = "SELECT usFee3 AS fee
                FROM user
                WHERE usID = ".$row["usID"].".";";
        }
        $result = mysqli_query($conn, $sql);
        $costs = mysqli_fetch_assoc($result);
        $coEmp = $coEmp + $costs["fee"] * $hours;
        $coMac = $coMac + $row["macHourRate"] * $hours;
    }
    $costsArray = array($coEmp, $coMac, $whEmp, $whMac);
    return $costsArray;
}
}

```

```

function dateDifference($start , $end , $type){
    //type = 1 for days, 2 for hours, 3 for minutes, 4 for years
    $ts1 = strtotime($start);
    $ts2 = strtotime($end);
    $seconds_diff = $ts2 - $ts1;

    if( $type == 1){
        $days_diff = round($seconds_diff/86400,2);
        return $days_diff;
    }elseif($type == 2){
        $hours_diff = round($seconds_diff/3600,2);
        return $hours_diff;
    }elseif($type == 3){
        $minutes_diff = round($seconds_diff/60,2);
        return $minutes_diff;
    }elseif($type == 3){
        $year_diff = round($seconds_diff/(86400*365),0);
        return $year_diff;
    }else{
        return $seconds_diff;
    }
}

function ddItem($conn, $type, $activ){
    /*
    This function creates a drop down menu from the type
    and returns its ID as value
    */
    $access = checkAccess($conn, $_SESSION['usID']);
    if($type == 'machine'){
        $query = "SELECT * FROM machines
                WHERE macInactive = ".$activ."";

    } elseif ($type == 'project') {
        $query = "SELECT * FROM projects
                WHERE proFinished = ".$activ."
                ORDER BY proNumber ASC";

    } elseif ($type == 'employee') {
        if($active){
            $query = "SELECT * FROM user
                    WHERE usID > 1 AND DATEDIFF(usContract, NOW()) < 0
                    ORDER BY usName ASC";

        }else{
            $query = "SELECT * FROM user
                    WHERE usID > 1 AND DATEDIFF(usContract, NOW()) > 0
                    ORDER BY usName ASC";

        }

    } elseif ($type == 'role') {

```

```

    $query = "SELECT * FROM roles";
} elseif ($type == 'proType') {
    $query = "SELECT * FROM projecttypes";
} elseif ($type == 'activity') {
    if($access){
        $query = "SELECT * FROM activityuser";
    } else {
        $query = "SELECT * FROM activityuser WHERE actUsUser =
            ".$_SESSION['usID']."";
    }
}
$results=mysqli_query($conn, $query);
if($type == 'machine'){
    foreach ($results as $row){
        echo '<option value="'. $row["macID"].'">'. $row["macName"].'</option>';
    }
} elseif ($type == 'project') {
    foreach ($results as $row){
        echo '<option value="'. $row["proID"].'">'. $row["proNumber"].'
            </option>';
    }
} elseif ($type == 'employee') {
    foreach ($results as $row){
        echo '<option value="'. $row["usID"].'">'. $row["usName"].'</option>';
    }
} elseif ($type == 'role') {
    foreach ($results as $row){
        echo '<option value="'. $row["rolID"].'">'. $row["rolName"].'</option>';
    }
} elseif ($type == 'proType') {
    foreach ($results as $row){
        echo '<option value="'. $row["typID"].'">'. $row["typName"].'</option>';
    }
} elseif ($type == 'activity') {
    foreach ($results as $row){
        echo '<option value="'. $row["actUsID"].'">'.
            $row["actUsID"]." - ".$row["actUsName"].'</option>';
    }
}
return $results;
}

function detailMachine($conn, $machine){
    $detail = inputExists($conn, $machine, $machine, 'machine');
    if($detail > 0){
        // Machine infos and maintenance dates
        $name = $detail["macName"];
        $id = $detail["macID"];
        $lastMaint = date("d.m.Y", strtotime($detail["macMaintLast"]));
        $nextMaint = date("d.m.Y", strtotime($detail["macMaintLast"].'+

```

```

        . $detail["macMaintInterval"].'days'));
$purCosts = number_format($detail["macPurchaseCosts"], 2, ",", " ");
$pddate = date("d.m.Y", strtotime($detail["macPurchaseDate"]));
$asset = number_format($detail["macAssetValue"], 2, ",", " ");
$costsLastYear = costsMachineYear($conn, $machine);
$costsactual = costsMachineActual($conn, $machine);
echo '
    <h1>'.$name.'</h1>
    <table class="tableouter"><tr>
    <td>
        <table class="tableinner">
        <tr><th>Machine type</th> <td>'.$detail["macType"].'</td></tr>
        <tr><th>Last<br>maintenance</th> <td>'.$lastMaint.'</td></tr>
        <tr><th>Next<br>maintenance</th> <td>'.$nextMaint.'<br>in '
        .dateDifference('today',$nextMaint,1).' days</td></tr>
        <tr><th>Purchasing price</th> <td>'.$purCosts.' EUR</td> </tr>
        <tr><th>Purchasing date</th> <td>'.$pddate.'</td></tr>
        <tr><th>Asset value</th> <td>'.$asset.' €</td></tr>
        <tr><th>Last years<br>Working hours</th> <td>'
        .number_format($costsLastYear[0], 2, ",", " ').' h</td></tr>
        <tr><th>Last years<br>Working costs</th> <td>'
        .number_format($costsLastYear[1], 2, ",", " ').' €</td></tr>
        <tr><th>Actual <br>working hours</th> <td>'
        .number_format($costsactual[1], 2, ",", " ').' h</td></tr>
        <tr><th>Actual <br>working costs</th> <td>'
        .number_format($costsactual[0], 2, ",", " ').' €</td></tr>
        </table class="tableinner">
    </td>
    <td></img></td>
    </tr>
    </table class="tableouter">';
}
}

```

```

function detailProject($conn, $project){
    $detail = inputExists($conn, $project, $project, 'project');
    if($detail > 0){
        $sql = "SELECT proID, proNumber, proName, typName, usName,
            proBudget, proStartDate, proEndDate, proLastUpdate
            FROM projects
            INNER JOIN user ON projects.proContact = user.usID
            INNER JOIN projecttypes ON projects.proType =
                projecttypes.typID
            WHERE proID = '".$project."'";
        $stmt = mysqli_stmt_init($conn);
        mysqli_stmt_prepare($stmt, $sql);
        $result = mysqli_fetch_assoc(mysqli_query($conn, $sql));
        // Project date and time
        $number = $detail["proNumber"];
    }
}

```

```

$name = $detail["proName"];
$type = $result["typName"];
$note = $detail["proNote"];
$contact = $result["usName"];
// Sum of Workhours and Costs
$costsArray = costsProject($conn, $detail['proID']);
// returns array($coEmp, $coMac, $whEmp, $whMac) -> [0 - 3]
$costs = $costsArray[0] + $costsArray[1];
$startDate = date("d.m.Y", strtotime($detail["proStartDate"]));
$endDate = date("d.m.Y", strtotime($detail["proEndDate"]));
$finDate = date("d.m.Y", strtotime($detail["proEndDate"]));
if($detail["proFinished"] == 0){
    // Project days due the delivery
    $diff = dateDifference(date("d.m.Y") , $detail["proEndDate"] , 1);
} else {
    $diff = 'Project is finished.';
}
echo '
<h1>'. $number. " / ". $name. '</h1>
<table class="tableouter"><tr>
<td>
    <table class="tableinner">
    <tr><th>Project type</th> <td>'. $type. '</td></tr>
    <tr><th>Project Contact</th> <td>'. $contact. '</td></tr>
    <tr><th>Start date</th> <td>'. $startDate. '</td></tr>';
if(!$detail["proFinished"]){
    echo '
    <tr><th>Delivery date<br>Days till delivery</th>
    <td>'. $endDate. '<br>'. $diff. '</td></tr>';
} else {
    echo '
    <tr><th>Delivery date<br>Days till delivery</th>
    <td>Project finished<br>'. $finDate. '</td></tr>';
}
echo '
    <tr><th>Workhours employees</th> <td>'.
    number_format($costsArray[2],2,".", " ").' h</td></tr>
    <tr><th>Workhours machines</th> <td>'.
    number_format($costsArray[3],2,".", " ").' h</td></tr>
    <tr><th>Costs employees</th> <td>'.
    number_format($costsArray[0],2,".", " ').' €</td></tr>
    <tr><th>Costs machines</th> <td>'.
    number_format($costsArray[1],2,".", " ').' €</td></tr>
    <tr><th>Last activity</th> <td>'.date("H:i - d.m.Y",
    strtotime($detail["proLastUpdate"])).'</td></tr>
    <tr><th>Notes</th> <td>'. $note. '</td></tr>
    </table class="tableinner">
    </td>';
}else {
    echo 'No Result';
}

```



```

}
}

function displayActivity($conn){
    /*
    This function Displays Activity according to the user
    if the parameter matches with the sql database
    and if available returns the row of the requested inputs
    */
    $user = $_SESSION['usID'];
    $access = checkAccess($conn, $user);
    if($access){
        $sql = "SELECT actUsName, usName, proName, macName,
            actUsStartDate, actUsEndDate
            FROM activityuser
            INNER JOIN user ON activityuser.actUsUser = user.usID
            INNER JOIN machines ON activityuser.actUsMachine = machines.macID
            INNER JOIN projects ON activityuser.actUsProject = projects.proID
            LIMIT 20;";
    } else {
        $sql = "SELECT actUsName, usName, proName, macName,
            actUsStartDate, actUsEndDate
            FROM activityuser
            INNER JOIN user ON activityuser.actUsUser = user.usID
            INNER JOIN machines ON activityuser.actUsMachine = machines.macID
            INNER JOIN projects ON activityuser.actUsProject = projects.proID
            WHERE actUsUser = ".$user." LIMIT 15;";
    }
    echo '<tr>
        <th>Activity</th>';
    if($access){
        echo' <th>User</th>';
    }
    echo '<th>Project</th>
        <th>Machine used</th>
        <th>Activity Start</th>
        <th>Activity End</th>
        <th>Time spend</th>
        </tr>';
    $result = mysqli_query($conn, $sql);
    if(!$result){
        $result = 0;
    }
    if(mysqli_num_rows($result) > 0){
        while($row = mysqli_fetch_assoc($result)){
            $hours = dateDifference($row["actUsStartDate"],
                $row["actUsEndDate"], 2);

            if($access){
                echo "<tr>
                    <td>".$row["actUsName"]."</td>";

```

```

        <td>".$row["usName"]."</td>
        <td>".$row["proName"]."</td>
        <td>".$row["macName"]."</td>
        <td>".$row["actUsStartDate"]."</td>
        <td>".$row["actUsEndDate"]."</td>
        <td>".$hours." h</td>
    </tr>";
} else {
    echo "<tr>
        <td>".$row["actUsName"]."</td>
        <td>".$row["proName"]."</td>
        <td>".$row["macName"]."</td>
        <td>".$row["actUsStartDate"]."</td>
        <td>".$row["actUsEndDate"]."</td>
        <td>".$hours." h</td>
    </tr>";
}
}
} else {
    echo "<tr><td>No Entries</td></tr>";
}
}

function displayEmployees($conn, $activ){
    /*
    This function displays employee according to the user
    if the parameter matches with the sql database
    active describes if the Employee has a valid contract end
    */
    $access = checkAccess($conn, $_SESSION['usID']);
    if ($access){
        if($activ) {
            $sql = "SELECT * FROM user
                INNER JOIN roles ON roles.rolID = user.usRole
                WHERE DATEDIFF(usContract, NOW()) < 1 AND usID >= 2
                ORDER BY usName DESC;";
        } else {
            $sql = "SELECT * FROM user
                INNER JOIN roles ON roles.rolID = user.usRole
                WHERE DATEDIFF(usContract, NOW()) > 0 AND usID >= 2
                ORDER BY usName DESC;";
        }
    } else {
        if($activ) {
            $sql = "SELECT usName, usLogin, rolName, usContract, usLastLogin
                FROM user
                INNER JOIN roles ON roles.rolID = user.usRole
                WHERE DATEDIFF(usContract, NOW()) < 1
                AND usID =".$_SESSION['usID'].
                ORDER BY usName DESC;";
        }
    }
}

```

```

    }else{
        $sql = "SELECT usName, usLogin, rolName, usContract, usLastLogin
            FROM user
            INNER JOIN roles ON roles.rolID = user.usRole
            WHERE DATEDIFF(usContract, NOW()) > 0
            AND usID = ".$_SESSION['usID']."
            ORDER BY usName DESC;";
    }
}
$result = mysqli_query($conn, $sql);
echo "
    <tr>
        <th>Full name</th>
        <th>Login name</th>
        <th>Role</th>
        <th>End of contract</th>
        <th>Last Project</th>
    </tr>";
if (mysqli_num_rows($result) > 0){
    while($row = mysqli_fetch_assoc($result)){
        echo "
            <tr>
                <td>".$row["usName"]."</td>
                <td>".$row["usLogin"]."</td>
                <td>".$row["rolName"]."</td>
                <td>".date("d.m.Y", strtotime($row["usContract"]))."</td>
                <td>".date("d.m.Y", strtotime($row["usLastLogin"]))."</td>
            </tr>";
    }
}else {
    echo "<tr><td>No results</tr></td>";
}
}

function displayMachines($conn, $activ){
    /*
    This function displays machines according to the user
    if the parameter matches with the sql database
    and if available returns the row of the requested inputs
    */
    $access = checkAccess($conn, $_SESSION['usID']);
    if ($access){
        $sql = "SELECT * FROM machines
            WHERE macInactive = ".$activ." AND macID > 1
            ORDER BY macMaintLast DESC ";
    } else {
        $sql = "SELECT macNumber, macName, macType, macMaintLast,
            macMaintInterval, macLastUpdate FROM machines
            WHERE macInactive = ".$activ." AND macID > 1
            ORDER BY macMaintLast DESC ";
    }
}

```

```

}
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0){
    if($access){
        echo "<tr>
            <th>Machine number</th>
            <th>Machine name</th>
            <th>Machine type</th>
            <th>Next<br>maintenance</th>
            <th>Last<br>maintenance</th>
            <th>Purchase date</th>
            <th>Hour Rate<br> [EUR]</th>
            <th>Asset value<br> [EUR]</th>
            <th>Last activity end</th>
        </tr>";
        while($row = mysqli_fetch_assoc($result)){
            $lastMaint = $row["macMaintLast"];
            $MaintInterval = $row["macMaintInterval"];
            $last = date("d.m.Y", strtotime($lastMaint));
            $next = date("d.m.Y", strtotime($lastMaint.'+'. $MaintInterval.'da
ys'));

            $update = date("h:i d.m.Y", strtotime($row["macLastUpdate"]));
            $rate = number_format($row["macHourRate"],2,""," ");
            $value = number_format($row["macAssetValue"],2,""," ");
            echo "<tr>
                <td>".$row["macNumber"]."</td>
                <td>".$row["macName"]."</td>
                <td>".$row["macType"]."</td>
                <td>".$next."</td>
                <td>".$last."</td>
                <td>".date("d.m.Y", strtotime($row["macPurchaseDate"]))."<
/td>

                <td>".$rate." €</td>
                <td>".$value." €</td>
                <td>".$update."</td>
            </tr>";
        }
    } else {
        echo "<tr>
            <th>Machine name</th>
            <th>Machine type</th>
            <th>Next<br>maintenance</th>
            <th>Last<br>maintenance</th>
            <th>End Date <br>last activity</th>
        </tr>";
        while($row = mysqli_fetch_assoc($result)){
            $lastMaint = $row["macMaintLast"];
            $maintInterval = $row["macMaintInterval"];
            $last = date("d.m.Y", strtotime($lastMaint));

```

```

    $next = date("d.m.Y", strtotime($lastMaint.'+'.$maintInterval.'days'));
}
}
} else {
    echo "0 results";
}
}

function displayProjects($conn, $activ){
    //Function to display the projects
    $access = checkAccess($conn, $_SESSION['usID']);
    $sql = "SELECT proID, proNumber, proName, typName, usName, proBudget,
        proStartDate, proEndDate, proLastUpdate
        FROM projects
        INNER JOIN user ON projects.proContact = user.usID
        INNER JOIN projecttypes ON projects.proType = projecttypes.typID
        WHERE proFinished = ".$activ."
        ORDER BY proEndDate DESC
        LIMIT 15;";
    $result = mysqli_query($conn, $sql);
    if (mysqli_num_rows($result) >0){
        if ($access){
            echo "
                <tr>
                    <th>Project number</th>
                    <th>Project name</th>
                    <th>Project type</th>
                    <th>Contact</th>
                    <th>Start date</th>
                    <th>Delivery date</th>
                    <th>Budget</th>
                    <th>Costs</th>
                </tr>";
            while($row = mysqli_fetch_assoc($result)){
                $costsArray = costsProject($conn, $row['proID']);
                $costs = $costsArray[0] + $costsArray[1];
                $budget = number_format($row["proBudget"],2,""," ");
                $costs = number_format($costs,2,""," ");
                $start = date("d.m.Y", strtotime($row["proStartDate"]));
                $end = date("d.m.Y", strtotime($row["proEndDate"]));
                echo "

```

```

        <tr>
            <td>".$row["proNumber"]."</td>
            <td>".$row["proName"]."</td>
            <td>".$row["typName"]."</td>
            <td>".$row["usName"]."</td>
            <td>".$start."</td>
            <td>".$end."</td>
            <td>".$budget." €</td>
            <td>".$costs." €</td>
        </tr>";
    }
} else {
    echo "<tr>
        <th>Project number</th>
        <th>Project name</th>
        <th>Contact</th>
        <th>Start date</th>
        <th>Delivery date</th>
        <th>Status</th>
    </tr>";
    while($row = mysqli_fetch_assoc($result)){
        $costsArray = costsProject($conn, $row['proID']);
        $costs = $costsArray[0] + $costsArray[1];
        $status = (1-$costs/$row["proBudget"])*100;
        $start = date("d.m.Y", strtotime($row["proStartDate"]));
        $end = date("d.m.Y", strtotime($row["proEndDate"]));
        echo "
            <tr>
                <td>".$row["proNumber"]."</td>
                <td>".$row["proName"]."</td>
                <td>".$row["usName"]."</td>
                <td>".$start."</td>
                <td>".$end."</td>
                <td>".number_format($status, 2,","," ")." %</td>
            </tr>";
    }
}
} else {
    echo "<tr><td>0 results</tr></td>";
}
}

function greetings($fullName){
    $Hour = date('G');
    if ( $Hour >= 5 && $Hour <= 10 ) {
        $greetings = "Good Morning, ".$fullName;
    } else if ( $Hour >= 11 && $Hour <= 16 ) {
        $greetings = "Hello, ".$fullName;
    } else if ( $Hour >= 17 || $Hour <= 19 ) {
        $greetings = "Good Evening, ".$fullName;
    }
}

```

```

} else if ( $Hour >= 20 || $Hour <= 4 ) {
    $greetings = "Good Night, ".$fullName;
}
echo $greetings;
}

function inactiveUser($conn){
    $stmt = mysqli_stmt_init($conn);
    $hashedPwd = password_hash('1BUCpZr=H$YtR5iq1', PASSWORD_DEFAULT);
    $sql = "UPDATE user
        SET usPwd = '".$hashedPwd.'"
        WHERE DATEDIFF(usContract, NOW()) < 0
        AND DATEDIFF(usContract, NOW()) > -180
        AND usID >= 2;";
    mysqli_query($conn, $sql);
}

function inputExists($conn, $id, $name, $type){
    /*
        This function checks if the parameter matches with the sql database
        and if available returns the row of the requested inputs
    */
    if(empty($type)){
        exit();
    } elseif ($type == 'machine'){
        $sql = "SELECT * FROM machines WHERE macID = ? OR macName = ?;";
    } elseif ($type == 'project') {
        $sql = "SELECT * FROM projects WHERE proID = ? OR proName = ?;";
    } elseif ($type == 'employee') {
        $sql = "SELECT * FROM user WHERE usID = ? OR usName = ? OR usLogin = ?;";
    } elseif ($type == 'role') {
        $sql = "SELECT * FROM roles";
    } elseif ($type == 'activity'){
        $sql = "SELECT * FROM activityemp WHERE actID = ? OR actName = ?;";
    }
    // Create a connection to the database
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        exit();
    }
    // Set Statement for the database
    if ($type == 'employee') {
        mysqli_stmt_bind_param($stmt, "sss", $id, $name, $name);
    } else if ($type == 'project') {
        mysqli_stmt_bind_param($stmt, "ss", $id, $name);
    } else if ($type == 'activity') {
        mysqli_stmt_bind_param($stmt, "ss", $id, $name);
    } else if($type == 'machine'){
        mysqli_stmt_bind_param($stmt, "ss", $id, $name);
    }
}

```

```

// Execute Statement and search results
mysqli_stmt_execute($stmt);
$resultData = mysqli_stmt_get_result($stmt);
// Check if results are not 0
if($row = mysqli_fetch_assoc($resultData)){
    // $row is an array with named columns as result
    return $row;
}
else{
    $result = false;
    return $result;
}
// Close connection
mysqli_stmt_close($stmt);
}

function invalideMail($email){
    $result;
    if(!filter_var($email, FILTER_VALIDATE_EMAIL)){
        $result = true;
    }else{
        $result = false;
    }
    return $result;
}

function invalidName($name){
    $result;
    if(!preg_match("/^[a-zA-Z]*$/", $name)){
        $result = true;
    }else{
        $result = false;
    }
    return $result;
}

function involvedEmployee($conn, $id, $start, $end, $type){
    // $id ... object id
    // $type is who is asking
    if($type == 'project'){
        $sql = "SELECT actUsUser, usName, actUsProject, proNumber,
                    proName, actUsMachine, macName, actUsStartDate, actUsEndDate
                FROM activityuser
                INNER JOIN user ON activityuser.actUsUser = user.usID
                INNER JOIN machines ON activityuser.actUsMachine = machines.macID
                INNER JOIN projects ON activityuser.actUsProject = projects.proID
                WHERE activityuser.actUsproject = '". $id ."'
                AND actUsEndDate >='". $start ."' AND actUsStartDate <='". $end ."'
                ORDER BY actUsStartDate;";
    }elseif($type == 'machine'){

```



```

$sql = "SELECT actUsUser, usName, actUsProject, proNumber,
          proName, actUsMachine, macName, actUsStartDate, actUsEndDate
FROM activityuser
INNER JOIN user ON activityuser.actUsUser = user.usID
INNER JOIN machines ON activityuser.actUsMachine = machines.macID
INNER JOIN projects ON activityuser.actUsProject = projects.proID
WHERE activityuser.actUsMachine = ".$id."
AND actUsEndDate >= ".$start." AND actUsStartDate <= ".$end."
ORDER BY actUsStartDate;";
}
// Create a connection to the database
$stmt = mysqli_stmt_init($conn);
if(!mysqli_stmt_prepare($stmt, $sql)){
    //header("location: ../.index.php.?error=stmtExistFailed");
    exit();
}
$result = mysqli_query($conn, $sql);
if(!$result){
    $result = 0;
}
if(mysqli_num_rows($result) > 0){
    echo '
        <th><h2>Involved employees</h2></th></tr>
        <tr>
            <th>Name</th>
            <th>Hours</th>
            <th>Start activity</th>
            <th>End activity</th>
        </tr>';
    while($row = mysqli_fetch_assoc($result)){
        $hours = dateDifference($row["actUsStartDate"],
                               $row["actUsEndDate"], 2);
        $hours = number_format($hours, 2, ",", " ");
        $start = date("h.m - d.m.Y", strtotime($row["actUsStartDate"]));
        $end = date("h.m - d.m.Y", strtotime($row["actUsEndDate"]));
        echo '
            <tr>
                <td>'.$row["usName"].'</td>
                <td>'.$hours.'</td>
                <td>'.$start.'</td>
                <td>'.$end.'</td>
            </tr>';
    }
} else {
    echo '<p> No employee activites.</p>';
}
}

function involvedMachine($conn, $id, $start, $end, $type){
    // $id ... object id

```

```

// $type is who is requesting
if($type=='project'){
    $sql = "SELECT actUsUser, usName, actUsProject, proNumber,
                proName, actUsMachine, macName, actUsStartDate, actUsEndDate
    FROM activityuser
    INNER JOIN user ON activityuser.actUsUser = user.usID
    INNER JOIN machines ON activityuser.actUsMachine = machines.macID
    INNER JOIN projects ON activityuser.actUsProject = projects.proID
    WHERE activityuser.actUsproject = '". $id. "'
    AND actUsEndDate >='". $start. "' AND actUsStartDate <='". $end. "'
    ORDER BY actUsStartDate;";
}elseif($type == 'employee'){
    $sql = "SELECT actUsUser, usName, actUsProject, proNumber,
                proName, actUsMachine, macName, actUsStartDate, actUsEndDate
    FROM activityuser
    INNER JOIN user ON activityuser.actUsUser = user.usID
    INNER JOIN machines ON activityuser.actUsMachine = machines.macID
    INNER JOIN projects ON activityuser.actUsProject = projects.proID
    WHERE activityuser.actUsUser = '". $id. "'
    AND actUsEndDate >='". $start. "' AND actUsStartDate <='". $end. "'
    ORDER BY actUsStartDate;";
}
}
// Create a connection to the database
$stmt = mysqli_stmt_init($conn);
if(!mysqli_stmt_prepare($stmt, $sql)){
    //header("location: ../".index.php."?error=stmtExistFailed");
    exit();
}
$result = mysqli_query($conn, $sql);
if(!$result){
    $result = 0;
}
if(mysqli_num_rows($result) > 0){
    echo '
        <th><h2> Involved machines</h2></th>
        <tr>
            <th>Name</th>
            <th>Hours</th>
            <th>Activity start</th>
            <th>Activity end</th>
        </tr>';
    while($row = mysqli_fetch_assoc($result)){
        if($row["actUsMachine"]>1){
            $hours = dateDifference($row["actUsStartDate"],
                $row["actUsEndDate"], 2);
            $hours = number_format($hours, 2, ",", " ");
            $start = date("h.m - d.m.Y", strtotime($row["actUsStartDate"]));
            $end = date("h.m - d.m.Y", strtotime($row["actUsEndDate"]));
            echo '
                <tr>

```

```

        <td>' . $row["macName"] . ' </td>
        <td>' . $hours . ' </td>
        <td>' . $start . ' </td>
        <td>' . $end . ' </td>
    </tr>';
    }
}
} else {
    echo '<p> No machine activites.</p>';
}
}

function involvedProject($conn, $id, $start, $end, $type){
    // $id ... object id
    // $type is who is asking
    if($type=='user'){
        $sql = "SELECT actUsUser, usName, actUsProject, proNumber,
            proName, actUsMachine, macName, actUsStartDate, actUsEndDate
            FROM activityuser
            INNER JOIN user ON activityuser.actUsUser = user.usID
            INNER JOIN machines ON activityuser.actUsMachine = machines.macID
            INNER JOIN projects ON activityuser.actUsProject = projects.proID
            WHERE activityuser.actUsUser = '$id.'"
            AND actUsEndDate >= '$start.'" AND actUsStartDate <= '$end.'"
            ORDER BY actUsStartDate;";
    }elseif($type=='machine'){
        $sql = "SELECT actUsUser, usName, actUsProject, proNumber,
            proName, actUsMachine, macName, actUsStartDate, actUsEndDate
            FROM activityuser
            INNER JOIN user ON activityuser.actUsUser = user.usID
            INNER JOIN machines ON activityuser.actUsMachine = machines.macID
            INNER JOIN projects ON activityuser.actUsProject = projects.proID
            WHERE activityuser.actUsMachine = '$id.'"
            AND actUsEndDate >= '$start.'" AND actUsStartDate <= '$end.'"
            ORDER BY actUsStartDate;";
    }
    // Create a connection to the database
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        //header("location: ../.index.php.?error=stmtExistFailed");
        exit();
    }
    $result = mysqli_query($conn, $sql);
    if(!$result){
        $result = 0;
    }
    if(mysqli_num_rows($result) > 0){
        echo '
            <th><h2>Involved projects</h2></th>
            <tr><td>

```

```

        <tr>
            <th>Number<br>Name</th>
            <th>Hours</th>
            <th>Start activity</th>
            <th>End activity</th>
        </tr>';
while($row = mysqli_fetch_assoc($result)){
    $hours = dateDifference($row["actUsStartDate"],
                          $row["actUsEndDate"], 2);
    $start = date("h.m - d.m.Y", strtotime($row["actUsStartDate"]));
    $end = date("h.m - d.m.Y", strtotime($row["actUsEndDate"]));
    echo '<tr>
        <td>'.$row["proNumber"].'<br>'.$row["proName"].'</td>
        // <td>'.$hours.'</td>
        <td>'.$start.'</td>
        <td>'.$end.'</td>
    </tr></td>';
}
} else {
    echo '<p> No project activites.</p>';
}
}

function loginUser($conn, $username, $pwd){
    $input = inputExists($conn, $username, $username, 'employee');
    if($input === false){
        header("location: ../index.php?error=NoSuchUser");
        exit();
    }

    // verify Pwd
    $pwdCheck = password_verify($pwd, $input["usPwd"]);
    if($pwdCheck == false){
        header("location: ../index.php?error=WrongPwd");
        exit();
    } else if ($pwdCheck == true){
        //Open session for roles
        session_start();
        $userID = $input["usID"];
        $_SESSION['usID'] = $input["usID"];
        $_SESSION['usLogin'] = $input["usLogin"];
        $_SESSION['usName'] = $input["usName"];
        $_SESSION['usRole'] = $input["usRole"];

        $lastLog = "UPDATE user
            SET usLastLogin = '".date("Y-m-d")."'
            WHERE usID = ".$userID.";";

        mysqli_query($conn, $lastLog);
    }
}

```

```

        inactiveUser($conn);
        //Close connection
        mysqli_close($conn);

        header("location: ../home.php");
    } else {
        header("location: ../index.php?error=NoInput");
        exit();
    }
}

function pwdMatch($pwd, $pwdrepeat){
    $result;
    if($pwd !== $pwdrepeat){
        $result = true;
    }
    else{
        $result = false;
    }
    return $result;
}

function updateActivity($conn, $aID, $aName, $uName, $pName,
    $mName, $actStart, $actEnd, $noUser){
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $user = inputExists($conn, $uName, $uName, 'employee');
    $update = " UPDATE activityuser
        SET  actUsName =?, actUsUser =?, actUsProject =?,
            actUsMachine =?, actUsStartDate =?, actUsEndDate =?,
            actUsNouser=?, actUsLog='".date("Y-m-d h:a:s")."'
        WHERE actUsID =?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../activityUpdate.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    $lastActivity = date("Y-m-d H:i:s");
    if(mysqli_stmt_bind_param($stmt, "siiissisi", $aName, $user["usID"], $pName, $mName, $actStart, $actEnd, $noUser, $lastActivity, $aID) == false){
        header("location: ../activityUpdate.php?error=stmt_bind_param_failed?activity:". $aName ."?user:". $user["usID"] ."?project:". $pName ."?machine:". $mName ."?Start:". $actStart ."?End:". $actEnd ."?lonely:". $noUser ."?lastLog" . $lastUpdate);
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../activityUpdate.php?error=query_failed?ActID:". $aID);
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
    }
}

```

```

        exit();
    }
}
//update Project
$logUpdate = date("Y-m-d H:i:s", strtotime($actEnd));
$lastLog = "UPDATE projects
            SET proLastUpdate = '$logUpdate.'"
            WHERE proID = '$pName.'";
if(mysqli_query($conn, $lastLog) == false){
    header("location: ../activityNew.php?error=update_project_failed?
            Date:".date("Y-m-d h:a:s")."?Project:". $pName);
    exit();
}
}
//update Machine
$lastLog = "UPDATE machines
            SET macLastUpdate = '$logUpdate.'"
            WHERE macID = '$mName.'";
if(mysqli_query($conn, $lastLog) == false){
    header("location: ../activityNew.php?error=update_machine_failed?
            Date:".date("Y-m-d h:a:s")."?Project:". $mName);
    exit();
}
}
//Close DB connection
mysqli_stmt_close($stmt);
mysqli_close($conn);
}

function updateMachine($conn, $mid, $number, $name, $type, $note, $purDate,
    $purCost, $hourRate, $assetValue, $util,$maintInterval,
    $maintLast, $inactive){
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $input = inputExists($conn, $mid, $name, 'machine');
    // $assetValue = $buchwert * $Abschreibungssatz[in prozent]
    $assetValue = asstetCalc($conn, $purCost, $purDate);
    $update = "UPDATE machines
                SET macNumber =?, macName =?, macType =?, macNotes =?,
                macPurchaseDate =?, macPurchaseCosts =?, macHourRate =?,
                macAssetValue =?, macUtilisation =?, macMaintInterval =?,
                macMaintLast =?, macInactive= ? , macLastUpdate = '"
                .date("Y-m-d h:a:s")."'
                WHERE macID =?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../machineUpdate.php?error=stmt_prepare_failed?");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_bind_param($stmt,"ssssdidiisii", $number, $name, $type,
        $note, $purDate, $purCost, $hourRate, $assetValue, $util,
        $maintInterval, $maintLast, $inactive, $input["macID"])){

```

```

        header("location: ../machineUpdate.php?error=stmt_bind_param_failed?");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../machineUpdate.php?error=query_failed?");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    //Close DB connection
    mysqli_close($conn);
}

function updateProject($conn, $pID, $number, $name, $contact, $type, $budget,
    $startDate, $endDate, $note, $finished){
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $update = "UPDATE projects
        SET proNumber =?, proName =?, proContact =?, proType =?,
            proBudget =?, proStartDate =?, proEndDate =?, proFinished =?,
            proNote =?, proLastUpdate = '".date("Y-m-d h:a:s")."'
        WHERE proID =?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../projectUpdate.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    $budget = str_replace(',', '.', $budget);
    if(!mysqli_stmt_bind_param($stmt, "ssiidsssii", $number, $name, $contact,
        $type, $budget, $startDate, $endDate, $note, $finished, $pID )){
        header("location: ../projectUpdate.php?error=stmt_bind_param_failed?");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../projectUpdate.php?error=query_failed?");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if($finished == 1){
        // Sum of emp and mac costs
        $costsArray = costsProject($conn, $pID);
        // returns array($coEmp,$coMac,$whEmp,$whMac) -> [0 - 3]
        $costs = $costsArray[0] + $costsArray[1];
        $fin = "UPDATE projects
            SET proCosts = ".$costs."";
        mysqli_execute($conn, $fin);
    }
    //Close DB connection

```

```

mysqli_close($conn);
}

function updateUser($conn, $userID, $name, $login, $contractEnd, $role,
                  $cfee, $afee, $sfee, $pwd){
    // Create the statements for interaction with SQL
    $stmt = mysqli_stmt_init($conn);
    $hashedPwd = password_hash($pwd, PASSWORD_DEFAULT);
    $update = "UPDATE user
              SET usName=?, usLogin=?, usContract=?, usRole=?,
                  usFee1=?, usFee2=?, usFee3=?, usPwd=?, usLastLogin=?
              WHERE usID=?";
    if(!mysqli_stmt_prepare($stmt, $update)){
        header("location: ../employeeUpdate.php?error=stmt_prepare_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    if(!mysqli_stmt_bind_param($stmt, "sssiddssi", $name, $login, $contractEnd,
                              $role, $cfee, $afee, $sfee, $hashedPwd, $lastLogin, $userID)){
        header("location: ../employeeUpdate.php?error=stmt_bind_param_failed");
        exit();
    }
    if(!mysqli_execute($stmt)){
        header("location: ../employeeUpdate.php?error=query_failed");
        mysqli_stmt_close($stmt);
        mysqli_close($conn);
        exit();
    }
    //Close DB connection
    mysqli_close($conn);
}

```