# DESIGN REPORT

Teaching Distribution System

NOVEMBER 12, 2021

Authors:
Koen Reefman - s2137674
Xin Wan - s1996746
Ziwei Chen - s2096781

Supervisor:
Arend Rensink

UNIVERSITY OF TWENTE.

# Abstract

Currently, assigning tasks to teachers in the Applied Mathematics department is done by a single person. This person has been using a Microsoft Access database in order to achieve these goals. However, due to the amount of time this system has been in use, it has become very complex. The main problem is that this person is about to retire and someone else needs to assign the tasks to teachers. It was decided that it would be too time-consuming to explain the current system to new people, therefore a new system had to be designed. In this project, a new system was designed in order to assign teachers to tasks. This report explains the steps taken during the design process, which eventually led to the final product. All phases, starting with formulating requirements and ending with system tests, are discussed in detail. The system is designed with usability in mind, in order to help the future time table planners of the AM department as much as possible. In the future, this system could be implemented in other departments as well.

# Contents

# 1. Introduction

Scheduling teachers for certain educational needs is a very complex task, you have to take into account many factors such as the total number of teachers you have available, which limits the amount of choices you can make when assigning them. You also have to keep in mind that certain teachers are unavailable on certain days (for example, a teacher might only be available on Monday and Tuesday, or they are unavailable on a certain date). Now, the situation gets even more complicated when considering the Applied Mathematics department. Instead of only having to worry about their own courses, the teachers from the AM department also teach in courses for other departments (for example, a mathematics part of a computer science module).

Currently, the entire scheduling of all teachers is done by a single person in a Microsoft Access database. This person is the only person that fully understands the database, which brings some problems with it. The main problem is now that this person is going to retire and someone else has to take over all the scheduling tasks. However, to explain the entire database and make the new person understand would take a lot of time. This is where the new scheduling tool comes in, this tool should be more intuitive to understand as well as make it easier to schedule teachers for tasks.

In chapter 2, the general information about the project is discussed, including stakeholders and software requirements. This is followed by an explanation of the design process from start to finish, as well as the requirements obtained from the client. Chapter 5 and 6 detail the design choices made in the project and an explanation of the overall implementation of the project. Chapter 7 discusses several tests, such as user tests, and their findings. After this, the future of the system is discussed. Lastly, there is an evaluation of the entire project, including a reflection on the planning, cooperation and task division within the project.

# 2. General information

## 2.1 - Stakeholders

Before the entire process of the project is explained, it is useful to get a complete picture. Firstly, there are several stakeholders that have an interest in this project. Table 1 shows who they are, how large of an impact the project has on them, how large of an impact they have on the project and what exact interests they have in this project.

*Table 1) Description of the stakeholder in this project*

| Stakeholder | Impact of project on them (Low, medium, high) | Influence over the project (Low, medium, high) | What are the stakeholder's interests? |
|---|---|---|---|
| Time table planners | High | High | Having an easy to use scheduling tool. |
| Module coordinators | Medium | Low | Being able to see which teachers are scheduled for which tasks. |
| Teachers | Low | Low | Being able to be scheduled for tasks reliably. |

Table 1 shows that the most important stakeholder is the time table planners, which would benefit from this project the most, since it would allow them to schedule teachers to tasks more easily than the current system. The client of this project isn't specifically mentioned in the table, since their interests are aligned with that of the time table planners.

## 2.2 – Current system

To know which tasks need to be filled, an Microsoft Excel file is obtained from the scheduling office. Figure 1 shows such a file, which contains information about the date of the task, the type of task, and the course for which the task is assigned.

| | Vak | Sleutelveld | Dag | Begintij | Eindtij | Wee | Type | Datum | Groep |
|---|---|---|---|---|---|---|---|---|---|
| 2 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUSA027B6 | ma | 10:45 | 12:30 | 36 | HCR | ######### | |
| 3 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUS8EC162 | ma | 10:45 | 12:30 | 37 | HC | ######### | |
| 4 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUS8EC162 | ma | 10:45 | 12:30 | 38 | HC | ######### | |

*Figure 1) Small sample of the actual Excel file sent by the scheduling office*

This file is then imported into a Microsoft Access database located on the computer of the time table planner, of which there is only one. This database contains information about the courses of the tasks, such as the number of students, the amounts of certain tasks and which educations teach the certain course. The database also contains information about the teachers of the Applied Mathematics department, including their total available teaching time and limitations in terms of which days they are available for teaching tasks. The time table planner can then assign tasks to teachers using a series of forms (found on figure 2), which save this data in a table in the database. Eventually, when all tasks are assigned, the completed planning can be printed out and handed to the teachers, as well as sent back to the scheduling office.

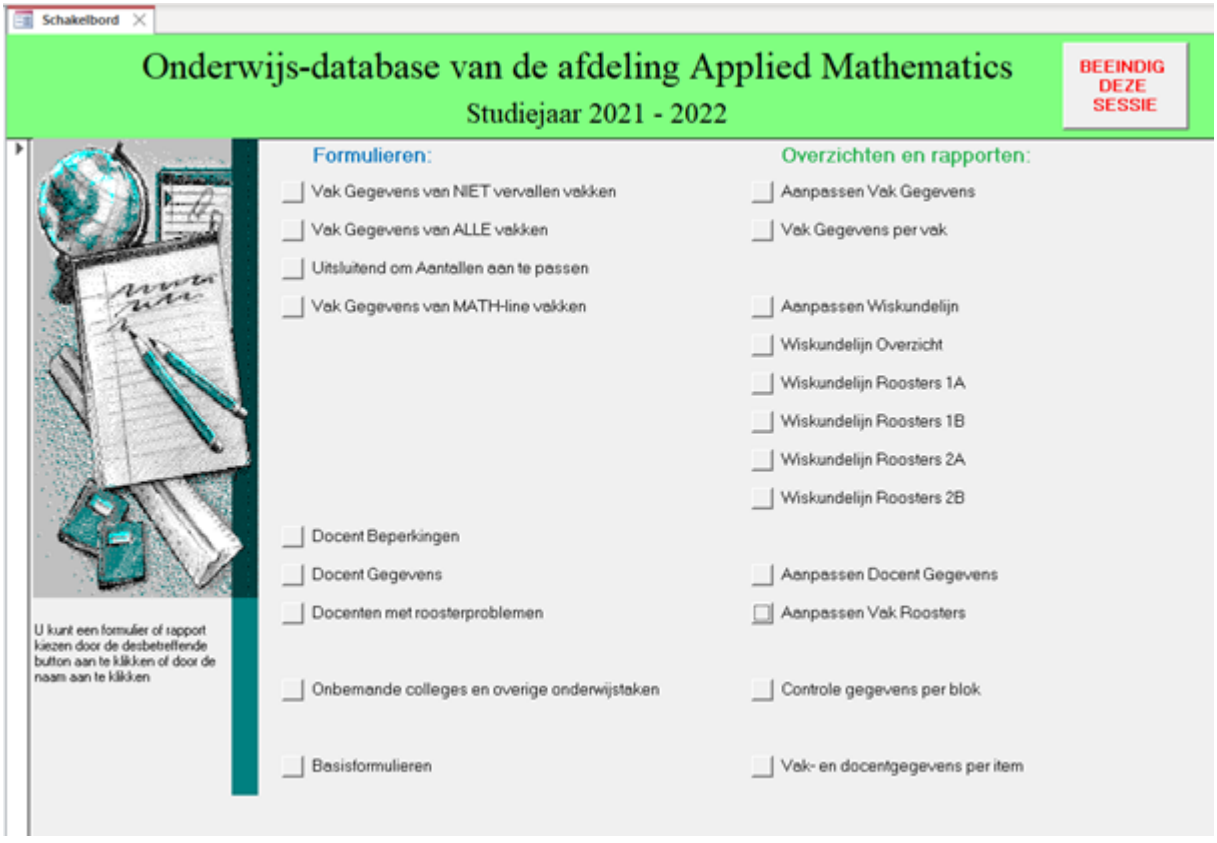


*Figure 2) The current system used to assign teachers to tasks*

## 2.3 – Software requirements

In order to get the project hosted on a University of Twente server, some software requirements were put in place. The university has guidelines ("Development Guidelines," 2021) in order to develop applications that should be hosted on their servers, which has information about what kinds of software should be used; However, a meeting was held with a representative of LISA in order to discuss the details of our project and the recommended software that should be used by us. These requirements were imposed on the project so that LISA can host the software on their servers and maintain it if necessary. The requirements can be split up into three categories: the front-end, the back-end and the database.

Starting with the front-end, there were a few choices in terms of frameworks that could be used (found in chapter 1.2 of the Canvas page), these included Angular (Jain, Mangal, & Mehta, 2015), as well as Bootstrap (Bootstrap, 2021). In the conversation with LISA, it was recommended that the bootstrap framework should be used.

For the back-end of the project, there were a few more software limitations. The first plan was to use PHP in order to handle all the back-end. However, in order to get the software hosted on a LISA server, Java needs to be used. This project uses the Spring Boot framework, together with Apache Maven (Miller, Vandome, & McBrewster, 2010) as the software management and comprehension tool. In order to be able to log in using UT accounts, OpenID or SAML (Security Assertion Markup Language) should be used.

As far as the database part of the requirements is concerned, there aren't many limitations. The Canvas page mentions that the main choices regarding databases are MySQL, PostgreSQL and OracleDB. After speaking with LISA, it was decided that the best option would be to use PostgreSQL, since they were most familiar with this.

# 3. Design Process

This chapter describes the steps taken in order to get the final product, divided into sub-chapters that discuss the different steps in detail. The sub-chapters are in chronological order, starting with getting the requirements, designing mock-ups of the system, creating prototypes, and ending with a description of the tests of the final product.

## 3.1 – Getting requirements

At the start of the project, several meetings with the client were held in order to determine what the client exactly expected of the final product. During the meetings, it became evident that the client did not really have an idea of what the final system should look like, so there were possibilities for own ideas. The client did have some requirements regarding the functionality of the system and what it should ultimately be able to do. These requirements are discussed in detail in chapter 4.1 of this report. It should be noted that more requirements can be obtained throughout the entire duration of the project, since it is an Agile development.

## 3.2 – Creating mock-ups

In order to create a project that meets the client's needs, the requirements that can be found in chapter 3.1 of this report are used to create mock-ups of the system. Mock-ups were created in code instead of using just a visual representation of the system, some buttons also had a bit of functionality to show the client what was possible for the final product. Several mock-ups with different options were presented to the client in order to gather possible improvements and more requirements. After presenting these different options to the client, decisions could be made about what would be the best course of action and what parts of the design could still be improved. During these meetings, the client was also able to add more requirements to the project, since it is an Agile project. The feedback of the client was implemented into the product, which ultimately led to a product that's closer to the client's expectations.

## 3.3 – Creating prototypes

During the project, biweekly meetings were held with the client in which progress was shown in the form of prototypes. These prototypes were shown in detail, which included showing what certain buttons or forms do and how the newly implemented features had an impact on the entirety of the system. The meetings were held in order to get feedback on the system, as well as gather any other possible requirements that should be added to the system. Useful feedback was gathered in these meetings, since the client could directly experience what newly implemented features did and how they could be used. This led to the client being able to give more detailed feedback regarding the look and feel of the product, ultimately leading to a product that was closer to the client's expectations.

## 3.4 – Testing

In order to test the system, user testing was done together with the client. For this, the client was given free reign over a test environment with mock data. Since the website was not hosted on a UT server at this point, real data could not be used in order to protect the data and not accidentally leak it online. The client was asked to perform a selection of tasks, ranging from importing tasks into the system to assigning teachers to tasks. For the tests, the client was encouraged to speak freely and criticize any flaws of the system. During the tests, the client was

observed by a tester, who wrote down comments made by the client, as well as write down any obvious non-verbal communications done by the client. These could include things like confusion on where to find a certain feature of the software. The results of the tests are discussed in detail in chapter 7 of this report.

# 4. Requirements

This chapter describes the different requirements that were obtained during the project, starting with stakeholder requirements, often referred to as user requirements of user needs, and ending with the system requirements that followed from these stakeholder requirements. The stakeholder requirements are based on two of the stakeholders mentioned in chapter 2.1: the time table planners/ administrators, and the module coordinators. There aren't any requirements for the teachers, since they have no impact on how the system functions, they are only interested in getting tasks assigned to them. The system requirements are derived from the stakeholder requirements and are used to define the functionality of the system.

## 4.1 – Stakeholder Requirements

Since this project is an Agile project, new stakeholder requirements could be added at any stage of the project. As mentioned in chapter 3, there were biweekly meetings with the client in which new stakeholder requirements could be added to the project. The requirements are all formatted according to the SMART guidelines (Doran, 1981).The detailed stakeholder requirements and a short description of them can be found in table 3 on appendix 1.

## 4.2 – System Requirements

In order to get requirements that could be used to implement features in the project, the stakeholder requirements were converted into concrete system requirements. The requirements were first thought of by the team and later requirements could be changed or added on advice of the client. In order to focus on the most important requirements, they were ordered according to the MoSCoW principle (Clegg & Barker, 1994), having the most important requirements in the 'must' category, while having the less important requirements further down the list. In order to determine which requirements were most important, a meeting was held with the client in order to rank them according to the client's interpretation of what was most important. The detailed system requirements, coiled with a short explanation, can be found in table 4 on appendix 1.

# 5. Design choices

This chapter describes the main design choices made on each part of the system (front-end, back-end and database), which help to illustrate the entire system later in the chapter. This chapter will only give a high-level description of the system, together with pictures. The low-level description of the system is explained in chapter 6 of this report.

## 5.1 – Global Design Choices

This project is designed to reduce the amount of labour for the time table planners of the Applied Mathematics department by combining the separate assignment processes to an integrated application. The system also prevents the user from making mistakes as much as possible. All functions and designs are based on the analysis of the client's requirements to make the application attractive and meet the user's needs.

The system aims to make the system easier to understand and more intuitive than the previous system. Therefore, the application focuses on the usability and efficiency of the system, coupled with a simplistic design. In order to develop this easy-to-use system, the client should be involved in all steps of the project. Because of this, the project used participatory design to ensure that the development at any stage of the project keeps up with the client's demands.

## 5.2 – Preliminary Design Choices

In order to create the project, some design choices need to be made before starting implementing the requirements. These design choices include the chosen programming language, used frameworks, libraries, and other choices made before the start of the implementation. These choices were made based on the pre-existing knowledge of the team members.

### 5.2.1 – Programming Languages

At first, the main choice for the back-end programming language was PHP. However, as mentioned in chapter 2.3 of this report, this wasn't allowed according to the LISA guidelines. Therefore, the choice for the back-end programming language was Java, since the team had a little bit of experience using this. As far as the programming languages for the front-end are concerned, there wasn't really a choice to be made, since the only language the team had any experience with was HTML. There was a bit of a choice to be made between JavaScript and Typescript; However, due to JavaScript being easy to learn, this was chosen instead of Typescript. For the database part, the choice was between MySQL and PostgreSQL. These two are very similar, having only minor differences in terms of syntax. On advice of LISA, mentioned in chapter 2.3, the choice was made to use PostgreSQL.

### 5.2.2 – Frameworks

The project uses the Spring framework, more specifically the Spring Boot framework. The Spring Boot framework is a popular and widely used framework used to develop REST APIs. The main reason this choice was made is because of the requirements set by LISA, coupled with the widespread use of this framework.

The Spring framework has many good features such as dependency injection, inversion of control and declarative programming (Rollbar Editorial Team, 2021) .Spring Boot is a sort of extension of the Spring framework, which removes the need for certain configurations that need to be set in the Spring framework. Therefore, Spring Boot is easier to use and makes it possible to program a stand-alone application with less requirements, configurations, and boilerplate code. Spring Boot also allows the embedding of Tomcat to the web-application directly. At the

same time, the framework provides other good features for security, which help prevent external attacks like SQL injection or cross-site scripting (XSS).

For the front-end, the Bootstrap framework was used to design the web-pages. The main reason for this choice was also because of the requirements set by LISA, found in chapter 2.3. Bootstrap is also quite a popular toolkit that includes HTML, CSS and JavaScript to build nice web-applications with fewer code (Bacinger, 2015). Based on these advantages, the project decided to use Bootstrap as the framework to design the front-end.

### 5.2.3 – Authentication

For the authentication part of the project, design choices didn't really have to be made. It was clear from the beginning that the UT Single Sign-on authentication system would eventually be used. However, since the team had no way to implement this at the start of the project, an external service by the name of Okta was used in order to simulate the login authentication. Both Okta, as well as the UT login system, work with SAML 2.0. In order to integrate the login system with the rest of the project, the Spring Security library was used. This library can be used in order to check permissions of logged in users by using certain configurations.

### 5.2.4 – Interaction between frontend and backend

The interaction between the front-end and the back-end is handled by sending POST and GET requests from the front-end to a specified URL in the back-end. POST requests are mainly used in order to submit user data to the back-end, while GET requests are mainly used to obtain data from the back-end that is subsequently used on the front-end. In our system, post requests are also used in order to execute, create, retrieve, update and delete operations for the database. These SQL operations are commonly referred to as CRUD, which are widely used in describing the basic operating functions of the database or persistence layer in a software system. RESTful services are used to implement the GET and POST requests since, according to the requirements of LISA, the Spring Boot framework should be used. These RESTful services are directly integrated in the Spring Boot framework, which has the advantage of good scalability and a clear structure, while also making the interaction between the front-end and back-end much simpler.

In RESTful services, the POST, DELETE, PUT and GET methods are used to get and modify resources given at a specified URL. The browser sends a request through the use of the AJAX function.

## 5.3 – System Design Overview

Now that the main design choices are explained, the individual web-pages can be discussed. These web pages are designed with the stakeholder- and system requirements from chapter 4 in mind. The webpages are individually explored and explained in detail. Images taken from the web pages can be found in Appendix 2.

### 5.3.1 – Login Page

The very first page the user is prompted with when using the website is the login page. This page is shown when a user tries to access a page without being logged in. Figure 11 on Appendix 2 shows this page, which consists of a simple button to click to login. This button will in turn

redirect the user to the connected authentication system. This is, as mentioned in chapter 5.2.3, ideally the UT Single Sign-on system. However, for the time being, the user is signed in by using an Okta account.

### 5.3.2 – Main Page

After successfully logging into the website, the user will be shown this main page. This page was designed to make the most used functionality of the website more readily available for the user. On this page, the user can assign teachers to tasks, assign tasks to teachers, as well as add a favourite teacher to a course so the system knows which teachers can teach which courses. These functions will be explained in more detail in sections 5.3.3 and 5.3.4. In addition to these functions, this page also shows the amount of unassigned tasks as well as the most recent changes in terms of assigned tasks. This can show if any changes have recently been made to the assigned tasks, for example in case a teacher gets ill and another teacher is now assigned to the task.

### 5.3.3 – Tasks Page

The tasks page contains all functions related to tasks. The most important functions, which are also on the main page, are assigning teachers to tasks and adding favourite teachers to a course. Assigning a teacher to tasks works by first selecting a task or package of tasks of the same course and type and subsequently selecting one or more teachers to assign to this task. Adding a favourite teacher to a course works quite similarly, the user first selects a course and selects one or multiple teachers he thinks are suitable for that course. It should be noted that teachers are also added to the favourite table whenever they are assigned to a certain task. The tasks page also contains a list of all tasks, coupled with information about which teacher is linked to that task. Lastly, there is an option to “import tasks”. As mentioned in chapter 2.2, the tasks are all obtained from the scheduling in a Microsoft Excel format. The “import tasks”, as the name suggests, imports all the tasks from the Excel file and puts them in the database.

### 5.3.4 – Teachers Page

The teachers page contains functions related to teachers. The most important feature of this page, which is also on the main page, is the ability to assign tasks to a teacher. In case a teacher has very little available time, it could be easier to assign tasks to him via that person’s time table than assigning that teacher to tasks. This way, the user can hand pick at which hour a teacher can perform a task. Other functions that are present on the teacher page are the function to add a new teacher to the system by providing the necessary details, as well as an easy way to indicate if a teacher is unavailable on a certain date (for example, due to other obligations). Lastly, there is an option to show the entire list of teachers and their details. From this detail menu, it is also possible to see a teacher’s detailed time table for each week of the current quarter.

### 5.3.5 – Settings Page

In order to provide the user with enough customizability, there is a settings page. This page contains options to enlarge the text on screen to increase readability of the text, as well as an option to turn the website to dark mode. This web page also contains two functional buttons. The first one is to back up the entire database onto the user’s personal computer, this is done to ensure that data doesn’t get lost easily in case something happens to the database. The second button is used to calculate the amount of compensation (in hours) a teacher gets for a certain task, which is used to handle the financial compensation given to a teacher.

# 6. System details

This chapter goes into detail about the design of the project, including diagrams for the database. Additionally, the design choices mentioned in the previous chapter are further explored and explained on a lower level, together with an overall low level explanation of all functionality of the product.

## 6.1 – Project Details

### 6.1.1 – Detailed Design Choices

#### 6.1.1.1 – Assign teachers to task

The first (and arguably most important) functionality that is explained is assigning teachers to a certain task. At first, this functionality was implemented in such a way that only one teacher could be assigned to a task, since it was only considered that task types could contain a single teacher. However, after discussion with both the client and within the group itself, it was mentioned that in some cases multiple teachers can be assigned to a single task. For example, for supervising an exam there are often multiple teachers. This led to some design choices/ alterations. At first, there was a button to assign a single teacher to a task, which later changed to checkboxes which could be used to select several teachers. After selecting multiple teachers, the submit button sends this data to the back-end and subsequently puts the data in the database using the methods described in chapter 5.2.4.

In addition to being able to add a teacher to a single task, the system provides a way to assign one (or more) teachers to a block of tasks of the same course and the same type. The grouping of the tasks on the front-end is done by a very simple JavaScript function, which collects tasks with the same course name and same type and groups them together in a table. When selecting teacher(s) for multiple tasks, the system sends these multiple tasks to the back-end and then adds them to the database individually.

#### 6.1.1.2 – Assign tasks to a teacher

It is also possible to assign tasks to a teacher, instead of the other way round. In order to assign tasks to a teacher, the user should first select the teacher and after that select the task that they want to assign to that specific teacher. In an early stage of the project, the user was shown all the unassigned tasks still left in the system, without considering if the teacher is available at that certain time. With this implementation, the user would not know in advance if there would be any conflicts in the planning. This implementation was shown to the client, which requested that a person’s time table would be shown, after which the user could assign tasks to a teacher in a specific time slot on the time table. This time table also contains information about how many hours of assigned tasks the teacher has, how many hours the teacher can teach in total, as well as any restrictions for specific dates. The current implementation works by selecting a teacher, after which that teacher’s time table is shown. In this time table, the user can assign a single task to a single time slot, while simultaneously seeing any restrictions they might have on their time table. After assigning a task to a teacher, the back-end (and subsequently database) will save this information using the methods described in chapter 5.2.4.

### 6.1.1.3 – Showing all tasks

The way that showing all tasks works is by having an SQL query retrieve information about all tasks from the database, coupled with information about whether that task has a teacher assigned to it yet. This information is then parsed by the back-end and subsequently shown on the front-end in a nicely formatted table. The user also has the opportunity to see the full task details for a certain task after clicking a button in the aforementioned table. If the user, for any reason, wants to change the teacher assigned to the currently selected task, there is an option that allows them to do so. Changing a teacher for a task works by using an SQL UPDATE statement to update the ”m_number” of the teacher assigned to the current task. After updating the assigned teacher, the last modified time of a task is also updated to reflect the recent changes to the task.

### 6.1.1.4 – Calculating Compensation of a task

To calculate the compensation awarded to a teacher for teaching a single task, a series of predetermined formulas and norms is used. The main problem with these formulas is that they apply to a block of tasks (for example, all tutorials for a certain course). In order to solve this issue, the total amount of compensation is divided by the number of tasks of that certain type, which gives a rough estimate of how many hours are awarded for a single task. The formulas that calculate this are located in the back-end and can be called from anywhere in the back-end. The way the system currently calculates all the compensations is by pressing a button, which goes through all the tasks in the database and subsequently calculates the compensation for each individual task. After implementing this function, it still takes a while to calculate the compensation for all tasks. To improve the speed of this function, the tasks could be grouped together by type and course name, since it drastically reduces the amount of calculations and calls to the database that need to be made (which significantly improves the speed).

### 6.1.1.5 – Importing Excel File

Importing the Microsoft Excel file into the database works by first having an upload button on the website. A JavaScript function is then coupled to this upload button, which checks if any files with an extension “.xls” or “.xlsx” have been uploaded. Files with other extensions will not be accepted. If a file with a correct extension has been uploaded to the website, a front-end JavaScript function will read all the lines of the Microsoft Excel file one by one and parse each row to a JSON string. This JSON string is sent to the back-end, using methods described in chapter 5.2.4, after the entire file is read. When the back-end Java function receives the JSON string, it will convert the JSON string into a list of tasks. This list of Java objects is subsequently saved in the database, resulting in all tasks from the Excel file being entered into the database.

### 6.1.1.6 – Add favourite teachers to different tasks

The functionality to add a favourite teacher to a certain course works in two ways: manually assigning them by using the provided buttons on the website, and automatically adding them as favourites after assigning the teachers to a task.

In the early stages of the project, it was requested that the system should remember which teachers were chosen for which tasks. This was first interpreted as having an algorithm that could recommend certain teachers for certain tasks based on factors such as previous teaching experience in that course. However, due to the limited information present in the database, this wasn’t feasible. Therefore, the second option was to have a way to remember which teachers are able to teach which courses. As mentioned before, this can be done in two ways. The first

way is to favourite teachers for a task manually, which can be done by the administrator on the website. This option was provided, since it was determined that the administrator/ timetable planner is the person who is most familiar with most teachers' skills. After selecting which teachers should be favoured for a certain course, this information is put into the database. The second way doesn't require the administrator to explicitly mention which teachers are the best for certain tasks, since it is done automatically when assigning a teacher to a task. After assigning the teacher to a task, the system also remembers that this teacher could teach the selected course.

#### 6.1.1.7 – Showing all teachers

The system has functionality to provide the administrator with details about all the teachers currently in the system. When clicking the button to show all teachers on the website, the user is shown some information to identify a teacher, such as the teacher's full name and m-number. At this point, the administrator can also select "full details" for a teacher, which shows all the details for a certain teacher in the database. These details are obtained by using a GET request to the back-end.

#### 6.1.1.8 – Adding a new teacher

The administrator of the system is also able to add a new teacher to the system via the website. This is done by providing the necessary details (such as their name, "m_number", amount of available hours, etc.) of a teacher to the system and clicking the submit button. If a teacher is successfully added to the database (by using the methods described in chapter 5.2.4), a small popup will notify the administrator of the success.

#### 6.1.1.9 – Change teacher details

Another design choice was made regarding the way to change a teacher's details. The choice was between having a button next to each individual piece of information or have a general button to change all the information of a teacher at once. In the beginning, there was a button next to each piece of information. This worked by first clicking the button, then changing the information in the text box, after which the information was sent to the back-end using a POST request in order to update the information in the database. The big drawback of this method was that a user would have to click a multitude of buttons if multiple detail fields of a certain teacher needed to be changed. Therefore, the choice was made to have one button that is used to update all fields of a teacher's data. With this method, the user can modify any number of fields in a teacher's information page. In order to submit the changes, the user can select the "save" button on the bottom right of the page, after which the data is sent to the back-end using a POST request, which in turn updates the changed fields in the database. This design choice solves the problem that a user has to click multiple buttons in order to change multiple fields of a teacher's data, which greatly improves the usability of the system.

#### 6.1.1.10 – Add certain unavailable date for teachers

Adding an unavailable date to a teacher is done easily by selecting the "Add An Unavailable Date" button on the website. The user is then prompted with a list of all teachers, with the option to add an unavailable date next to the teacher's name. This unavailable date is sent to the back-end using methods described in chapter 5.2.4. If this was successful, the administrator is notified via a simple popup message.

### 6.1.1.11 – Timetable

The timetable views are designed for showing the assignment situation of the teachers and allowing the planner to assign a task to a teacher on certain days. The project had two options to show the views of the timetables. The first option was to place a button in a separate teacher list. The second option was to place a link in the teacher details page. The project decided to use the second option to show the timetable of the teachers because the client prefers to see the timetable after browsing the details of the teacher. The project also decided to provide links to the timetable during the task assignment processes, which made it easy for the planner to clearly know the assignment situation of the teachers during the assignment processes.

### 6.1.1.12 – Recent changes

On the main page, the most recent changes of the system are displayed (meaning the tasks that were assigned to a teacher last). The choice here was to either display all tasks with a scroll bar or only display the five most recent changes. Every time a user enters the main page, the front-end sends a GET request through AJAX to the back-end to get these changes. Therefore, if a lot of tasks have been assigned, the system will have to load more and more tasks to display on the website, causing it to slow down significantly. For this reason, it was chosen to just display the five most recent tasks on the main page, together with a button to show all the recent changes on a new page if the user wants to find a certain task. Implementing it in this way will limit the amount of information that needs to be obtained from the back-end, which effectively solves the problem of the page loading slowly.

### 6.1.1.13 – Conflict notification

This design considers whether the user can click the link button on the conflict notification to resolve the conflict in the course arrangement. At the beginning, when designing the course conflict reminder, only consider that when assigning the teacher to the task, if there is a conflict (This teacher has already been assigned for an another task during the time period of this task), then a conflict reminder will pop up to inform the user that the teacher will have two tasks in the same time period, and then the user can close the reminder by clicking the close button, then choose other teachers to assign to this task.

This option is confirmed after the meeting with the client. The client wants to choose to enter the task detail page of the teacher conflicting task by clicking the link button after the task conflict pops up, and through the reassign teacher button on this page, he can reassign other teachers to the conflicting task, so that the teacher will become available during the conflicting time period of the task, afterward the user will be able to assign the teacher to the task that needs to be allocated. This design choice allows the user to assign the task in a much more flexible way. He can choose to assign another teacher to the task after seeing the task conflict notification, or let the teacher become free during the time period when the task needs to be assigned, and then assign the teacher to the task.

### 6.1.1.14 – Search functions

For the functionality of the search bars on several pages of the website, some choices were also made. The first working implementation of the search bars used the SQL LIKE function, which works by pattern matching a string. This idea was proposed by a team member based on previous experience with this function. However, this function was implemented in such a way that only one of the columns of a table could be searched at a time. Therefore, after suggestions

of other team members, this functionality was completely removed and replaced with a much simpler JavaScript function. This had the advantage of it being run client-side, which meant there were fewer requests being sent to the back-end, which in turn sped up the interaction time of the system. This also had an advantage of greatly reducing the code complexity, since the previous implementation of the SQL LIKE function used around 300 lines of code and the new implementation only uses around 30. Lastly, this new implementation allows for the user to search all the columns instead of being restricted to searching in one specific column.

### 6.1.1.15 –Login Design

The project only designed one “login” button on the Login page. After clicking on the button, it will redirect to the external identity platform (here is UT authentication system). So, it doesn’t allow the users to sign up or use other accounts. The project made this decision for security because only several people that are authorized by the AM department can login the system. As a result, other users should ask the AM department for permission.

### 6.1.2 – Backend Structure

The entire back-end of this project is based on LISA's requirements mentioned in chapter 2.3, the exact design pattern used can be found in figure 3. This chapter will describe the different blocks shown in this figure, as well as some additional explanations of classes within the project.

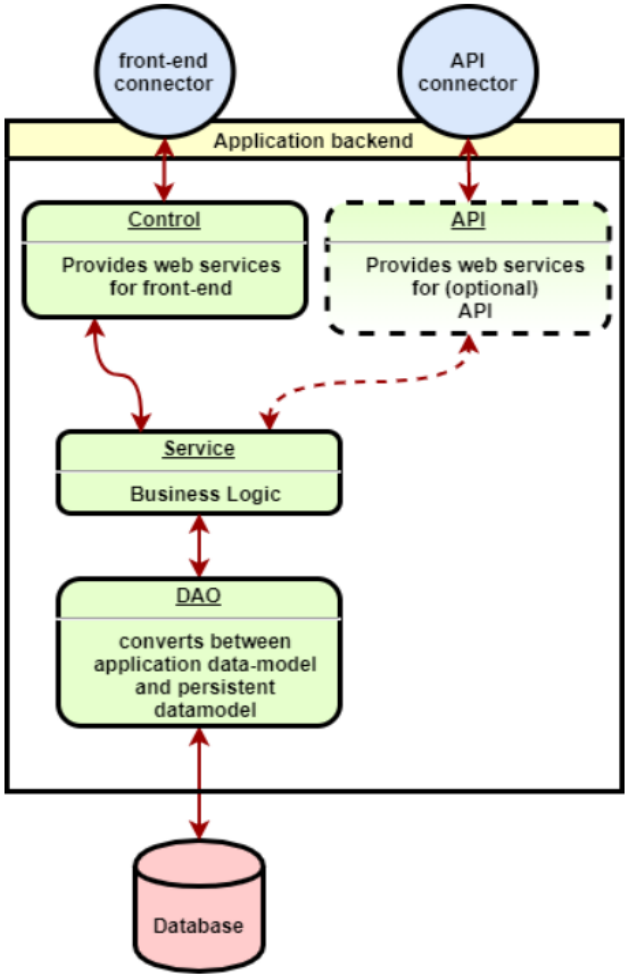


*Figure 3) Design pattern used in the backend (https://utwente-dev.instructure.com/courses/103/pages/1-dot-3-design-pattern?module_item_id=108)*

### 6.1.2.1 - Control

The controller provides the URL addresses of all RESTful services required by the front-end. The directory structure of this part of the back-end can be found in figure 4. The rest of this subchapter will shortly explain each class.

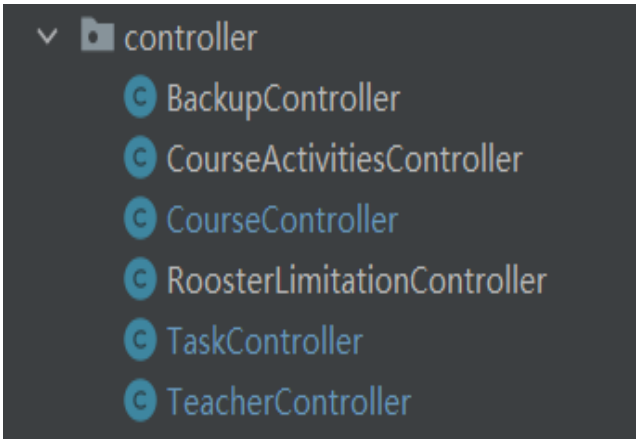


*Figure 4) Directory structure of the control part*

The BackupController java file contains the RESTful service URL for backing up the database, which means if the front-end sends a request to back up the database, this controller provides a URL for it.

The CourseActivitiesController java file contains all RESTful service URLs for calculating the total amount of hours a person gets compensated for a course based on its course name and type (e.g. lecture or tutorials), which means if the front-end sends a request to calculate the total amount of hours compensated for a course, this controller provides the URL for it.

The CourseController java file contains all RESTful service URLs for all operations related to the course (adding courses, seeing a list of all courses, etc.), except for calculating the amount of hours compensated, which is in the CourseActivitiesController. This means that, if the front-end sends a request to add a favourite teacher to a course, this controller provides the URL for it.

The RoosterLimitationController java file contains the RESTful service URLs to add and see the rooster limitations a certain teacher has, which means that if the front-end sends a request to add one (or more) rooster limitations to a teacher, this controller provides the URL for it.

The TaskController java file contains all Restful service URLs for all operations related to the task in the frontend, which means if the front-end sends the request to (for example) assign a teacher or multiple teachers to a task or display all unassigned tasks, then this controller provides the URLs for it.

The TeacherController java file contains all Restful service URL for all operations related to the teachers in the frontend, which means if the front-end sends a request to (for example) add a new teacher or change teacher details, then this controller provides the URLs for it.

### 6.1.2.2 - Service

The service mainly contains business logic and rules like methods, the directory structure of this part is shown in figure 5.

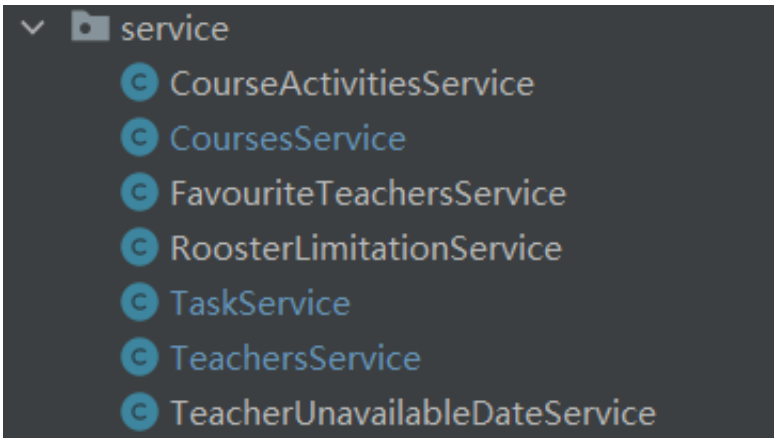


*Figure 5) Directory structure of the service part*

### 6.1.2.3 - DAO

The DAO (Data Access Object) contains all the CRUD operations related to the database. The directory structure of the DAO part of the project can be seen in figure 6. Table 2 contains all the files in the DAO part and their corresponding tables in the database.

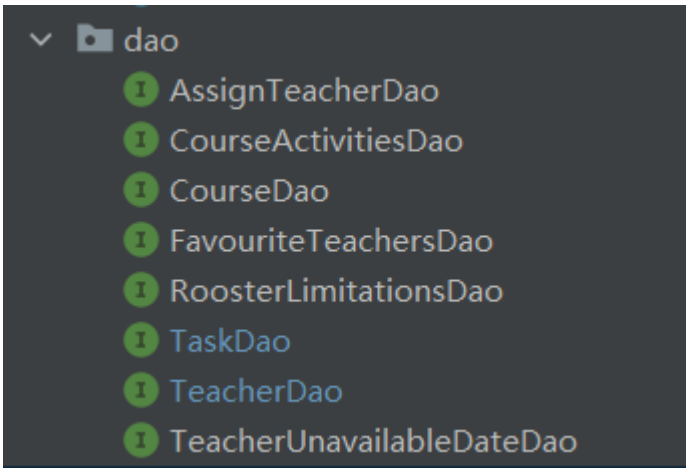


*Figure 6) Directory structure of the DAO part*

*Table 2) DAO classes and the database tables they access*

| File | Modifies: |
|---|---|
| AssignTeacherDao | assign_teachers |
| CourseActivitiesDao | course_activities |
| CourseDao | course_data |
| FavouriteTeacherDao | favourite_teachers |
| RoosterLimitationsDao | rooster_limitations |
| TaskDao | tasks_to_fill |
| TeacherDao | teacher_data |
| TeacherUnavailableDateDao | teacher_unavailable |

### 6.1.2.4 - Model

The model contains all the classes used to define the database tables. Each Java class corresponds to the object of each table in the database. The directory structure of the model part of the project can be seen on figure 7. Because we are using JPA, the names of the columns in each table are also defined in this JAVA class. The detailed description of each table in the database can be found in chapter 6.1.3 of this report.

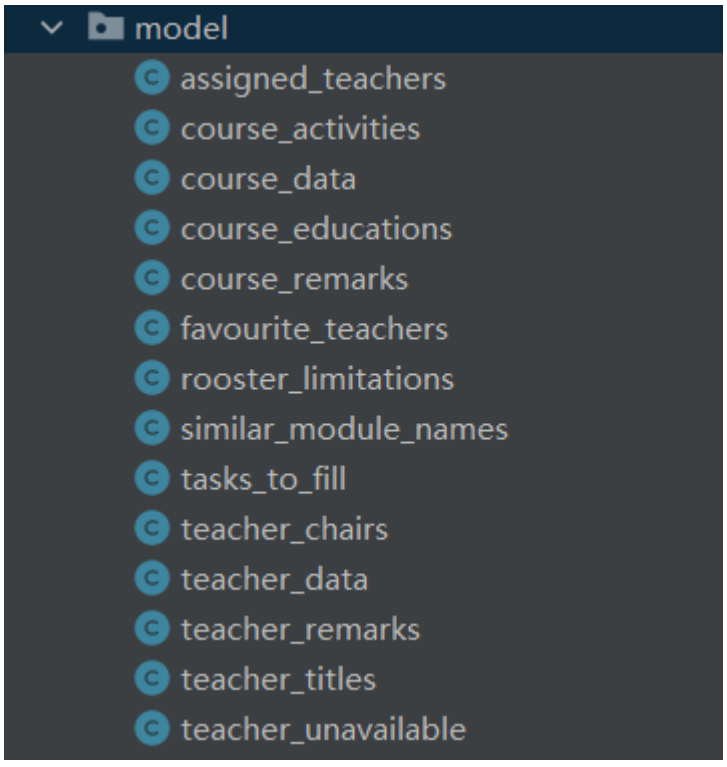


*Figure 7) Directory structure of the model part*

### 6.1.2.5 - Util

The Util folder provides additional tools and methods, such as the class to compute how many hours of compensation a teacher gets for a given task (Java class “hours_per_task” on figure 8). The util folder also contains a “TaskForm”, which is a Java object that is a list of tasks. This class is used to import the tasks from the Excel file in a much more efficient way than importing the tasks one by one. Lastly, there are Java classes that end with “PK”, these are the primary keys of the tables found in the model part (mentioned in chapter 6.1.2.4). When a table has a composite primary key, a class is created that holds this primary key. The full directory structure of the util file can be found in figure 8.

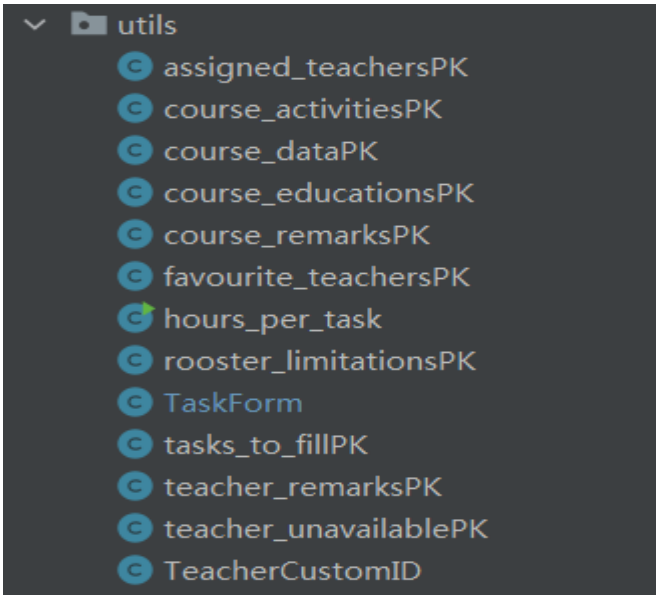


*Figure 8) Directory structure of the utils part*

## 6.1.3 - Database

In order to store the information about which teachers are assigned to which tasks, including detailed information about both teachers and tasks, a database was designed. This database design is loosely based on the MS Access database that is currently being used, which was mentioned in chapter 2.2. It should be noted that this database is not a replica of the original database, it just contains the most important information from the old system and puts it in a better structure. This subchapter is used to explain the individual tables, how they work together, and explain what design choices were made and how they impacted the database design. Throughout the explanations, figure 9 will be used as an illustration of the system.


*Figure 9) Illustration of the database structure used in the project*

### 6.1.3.1 – course_data

This table is used to store data about different courses given by the Applied Mathematics department, the most important columns in this table are code, subcode and module (name), which are used to identify different courses in the database. The ”code” column contains the module code, which is the same code that is used on Osiris. The column ”study_program” contains information on which study program this course belongs to (e.g. Calculus 2 for ME). Quarter indicates which quarter the course takes place (e.g. 2A), cluster has information on which cluster the course belongs to (M-line, which are a series of math courses). The ”ec” column indicates how many ECs or European Credits are awarded after completing the course. Type indicates if the course is mandatory or optional, ”standard_compensation” gives information about the amount of compensated hours for the course. ”Nr_of_students” and ”nr_of_exams” are self-explanatory. Changes and ”changes_explanation” hold information on if the course has any changes (e.g. course changes after switching to new course material). Lastly, the name column is used to have a shorter name in the database, since some course names include a lot of course codes.

### 6.1.3.2 – course_activities

The ”course_activities” table is linked to the ”course_data” table with a one-to-one relationship on (code, subcode, module). This table is used to indicate what kinds of activities, such as lectures, tutorials, etc. take place in the module and how many of them there are. One small design choice made in this table is the splitting up of three different types of tests (setting_up_t<*x*> and correcting_t<*x*>)m with the *x* ranging from 1 to 3. This is done in order to more easily calculate the amount of compensated hours awarded to a teacher for performing such a task.

### 6.1.3.3 – course_educations

This table holds information about different educations (e.g. Computer Science or Applied Mathematics) that teach a certain course, identified by code, subcode and module. Information about which year the course is taught is also included in this table.

### 6.1.3.4 – course_remarks

The “course_remarks” table contains, as the name implies, remarks about a certain course. This includes remarks for the course from previous years. The column “year” is used to indicate the year in which the remarks were made about the course.

### 6.1.3.5 – teacher_data

After having the information about the courses, this table holds information about the teachers of the Applied Mathematics department. This information includes the teacher’s full name. The column m_number contains a unique number for each teacher, which also acts as the primary key for this table. One design choice made in this table was to include both the column “chair” as well as the table “owverdeling”. In essence, they are almost the same thing; However, there is a small difference between them. Some teachers are linked to a certain chair, who handles their payments, but their hours are not registered on the overview of that chair. Another column that should be mentioned is “sylplus_id”, this table is used to link a teacher to the Syllabus Plus software, which is used to create the time tables. In some cases, it can happen that a person doesn’t have an “m_number” yet, the “sylplus_id” table then shows “m??” instead of the

person's "m_number". Arguably the most important column in this table is the "available_hours" column, which indicates the total amount of hours a person is able to work during the entire educational year. When assigning them to tasks, it is preferable to not assign them more hours than their available hours. The columns "function" and "type" are used to indicate what a person is capable of doing, for example the "function" column is used to indicate what kind of function a person has (e.g. Teaching Assistant or PhD student) and the type column is used to indicate whether the person has a temporary or a permanent contract. Lastly, the "remarks" column is used to easily indicate if there are any remarks for a certain teacher, which are subsequently stored in another table.

### 6.1.3.6 – teacher_remarks

As mentioned in the previous paragraph, this table is used to store all kinds of remarks for a certain teacher, which are stored in the column "remarks". This table also contains the columns "action" and "okay_action", which were taken from the original Microsoft Access database. These tables are used to indicate if certain actions need to be taken with the teacher at the end of the current educational year (for example, change the amount of available hours).

### 6.1.3.7 – rooster_limitations

In some cases, it can happen that a teacher is always unavailable on a certain day (for example due to other obligations like weekly meetings). In this case, "rooster_limitations" are added to the system, which will keep the system from assigning teachers to those hours. (Note: rooster means time table in Dutch) This table holds information about the days and hours that a teacher is unavailable (every week). A teacher can have zero, one or even multiple limitations.

### 6.1.3.8 – teaching_hours

In order for the system to know which specific hours a teacher is available, this table was created to hold information about the teaching hours. If a teacher is unavailable on certain hours of a day, this table helps to decipher those numbers. For example, hour 1 in the "rooster_limitations" table is equal to 8:45 – 9:30 in terms of real time.

### 6.1.3.9 – teacher_unavailable

It could also happen that a teacher is unavailable only on a certain date (for example due to having to attend a conference), this table is then used to indicate which date(s) a teacher is unavailable by linking that teacher's "m_number" to a specific date.

### 6.1.3.10 – teacher_titles

The "teacher_titles" table holds a list of titles that are accepted by the system, so when a new teacher is added to the system, their title should match one of the titles in this table. This table was implemented in order to keep consistent teacher titles instead of having multiple variations of the same title (Doctor, Dr., etc.).

### 6.1.3.11 – teaching_chairs

As mentioned in chapter 6.1.3.5, each teacher is linked to a certain chair who handles the payments of those teachers. This table contains the chair abbreviations, combined with their full names for clarity.

### 6.1.3.12 – favourite_teachers

Teachers have two ways in which they can end up in this table. As mentioned in chapter 5.3.3, teachers can be favored manually for a course, and a teacher is automatically added as a favourite when assigning them to a task. This table holds information about a course, which is linked to the course_data table by the code, subcode and module_name, and the teachers that are favored for that specific course.

### 6.1.3.13 – tasks_to_fill

In order to keep track of which tasks have been assigned and which ones haven't, the table "tasks_to_fill" was created to hold all the tasks. These tasks are directly imported via a Microsoft Excel file sent by the scheduling office, figure 10 shows a small snippet of such a file. The first two columns, "module_code" and "course_name", are both used to identify which course this task belongs to. The column "s_plus_code" is used in order to link this task to a certain course in the Syllabus Plus time table software. The columns "day", "start_time", "end_time", "week" and "date" are all used to determine on which date and which time a task takes place. The type of the task determines how much compensation an assigned teacher gets, based on pre-existing formulas and norms. This compensation is subsequently stored in the column "compensation". In order to identify a certain task, the column "task_id" was created. This column automatically increments every time a new task is added to the system, which allows for all the tasks in the Excel file to be imported. Lastly, the "last_modified" column is used to indicate when the task was last modified, this can be changed when inserting a new task into the database or when assigning a teacher to the task. This information is shown in the recent changes on the main page, described in chapter 5.3.2.

| | Code | Vak | SleutelveId | Dag | Begintij | Eindtij | Wee | Type | Datum | Groep |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUSA027B6 | ma | 10:45 | 12:30 | 36 | HCR | ######### | |
| 3 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUS8EC162 | ma | 10:45 | 12:30 | 37 | HC | ######### | |
| 4 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUS8EC162 | ma | 10:45 | 12:30 | 38 | HC | ######### | |
| 5 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUS8EC162 | ma | 10:45 | 12:30 | 39 | HC | ######### | |
| 6 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUS8EC162 | ma | 10:45 | 12:30 | 42 | HC | ######### | |
| 7 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUSAB5FB0 | ma | 10:45 | 12:30 | 40 | HC | ######### | |
| 8 | 202001115/202000610 | Calculus 1 for AT & EE 202001212/202001213 | #SPLUSAB5FB0 | ma | 10:45 | 12:30 | 41 | HC | ######### | |
| 9 | 202000610 | Calculus 1 for AT 202001212 | #SPLUS21255C | ma | 08:45 | 10:30 | 37 | OOO | ######### | |
| 10 | 202000610 | Calculus 1 for AT 202001212 | #SPLUS21255C | ma | 08:45 | 10:30 | 38 | OOO | ######### | |
| 11 | 202000610 | Calculus 1 for AT 202001212 | #SPLUS21255C | ma | 08:45 | 10:30 | 39 | OOO | ######### | |
| 12 | 202000610 | Calculus 1 for AT 202001212 | #SPLUS21255C | ma | 08:45 | 10:30 | 40 | OOO | ######### | |

*Figure 10) Small snippet of the actual Excel file used*

### 6.1.3.14 – similar_module_names

Since the Microsoft Excel file contains different course names than the ones that are in the database, the choice was made to have a table in between the "tasks_to_fill" and "course_data" tables, which links the two together. In order to link the two tables together, the course name in "tasks_to_fill" was compared to the course names in "course_data" and the most likely match was chosen as the right choice. This process was done automatically using a PostgreSQL module called pg_trgm, which is based on trigram matching to compare the two names. The module name from the Microsoft Excel file was then stored in the column "similar_module_name", while the actual module name is stored in the "actual_module_name" column. Other information about the actual module, like the code and subcode, is also stored in this table to quickly see which course a task belongs to.

### 6.1.3.15 – assigned_teachers

Since during the meetings it was brought up that multiple teachers could be assigned to a single task, this table was created. Before this table was created, it was only possible to assign a single

teacher to a task, since the assigned teacher data was stored together with the task in the “tasks_to_fill” table. Currently, this table is linked to the “tasks_to_fill” table by using the unique “task_id”. Multiple teachers can be assigned to a task by having multiple rows of the same “task_id” with different “teacher_mnumber” in the table.

# 7. Tests

In this chapter of the report,  A usability test will be explored and the result of this test will be explained. The first type of test is a usability test done together with the client to see if their needs are met.

## 7.1 – Test Plans

### 7.1.1 – Usability test

For the usability tests, several aspects of the system were tested together with the client. Table 5 on Appendix 3 shows these different aspects, together with a short explanation of the aspect, the expected outcome of the test and a risk level. The risk level indicates the importance of certain functionalities, being categorized in two levels: H(igh) and L(ow). In this case, the tests with a high level should be implemented without defects. The tests with a low level can have some small defects (won't negatively affect other functions). The tests for the functions that are indispensable for the core goal of the system (which in this case is the assignment of teachers to tasks) and the system security are assigned a high level. Other tests for extra functionality are assigned a low level.

The project used the test strategy based on the heuristics that were proposed by Nielsen and Molich (Nielsen & Molich, 1990). These can be summarized as follows: the users should always know the state of the current system situations by receiving feedback on their actions (e.g., a notification when a new teacher is successfully added to the database); the users should clearly understand the meaning of the words, situations and actions in the system (e.g., brief explanations to the functions and the names that are used in the system); the system should show less irrelevant information to the users to make the system clear and understandable; the system should provide the users documentation that can help the user easily find the information that they want and solve their confusions about the tasks.

The task planner of AM was asked to do some tests by following the instructions and scenarios presented by the tester. At the same time, he was also asked to provide his improvement advice and extra requirements according to the usage experience. This process is considered as the think-aloud protocol. After these processes, the test can be considered as passed. Only when the system designs meet the user's needs and the expectations provided by the heuristics, can it be considered as accepted.

## 7.2 Test Results

### 7.2.1 – Usability test

During the test done together with the client, several points of improvement were found. This chapter discusses the results obtained from the test, together with any (possible) solution(s) to the problems found during testing.

| Observation | (Possible) solution |
|---|---|
| When assigning a task, the client clicked the underlined name of a teacher instead of the "add" box to assign a teacher. | Change the description below the title of the page to explain how to assign a teacher to a task. |
| There is no information about the selected task when the user goes to the teacher assign page. (So the user has to remember a lot of information) | Include a bit of information about the selected task at the top of the page. |
| Front-end values do not update automatically when something is changed. | Use a JavaScript function to reload the necessary elements on a page. |
| Client wanted to assign multiple tasks at once to a teacher through the time table, which isn't possible. | |
| Popups when certain actions succeed are very inconsistent. | Smooth out the front-end and use the same popups throughout the entire system. |
| Client wanted to change the available days of a teacher, which isn't implemented. | Provide the ability to change the available days of a teacher on the teacher detail page. |
| The system should be able to present a printable time table to hand out to teachers. | Convert the assigned tasks from the database into a compact, readable format which can be printed. |
| The UI isn't user friendly, the user has to click the 'x' button multiple times to close all the popup windows. | When a user successfully performs an operation, the page will automatically close after two seconds. |

# 8. Future

This chapter describes the future of the project after the quartile has come to an end. This includes possible support for the system, as well as an implementation on a university-wide scale is discussed.

## 8.1 - Future additions

Although most of the main goals set at the beginning of the project were met, there are still some things that could be improved or added to the system. One of the main functions that should be implemented in the future, which was also mentioned during the user tests in chapter 7.2, is the ability to present a printable time table for individual teachers. This could be implemented by getting the assigned task data for a single teacher and converting it into a compact, readable format. Another improvement that could be made to the system is reducing the amount of requests sent to the back-end and the database. One example of this was already discussed in chapter 6.1.1.4, which mentioned that the amount of calls to the database for the calculation of compensation for a task could be reduced. One goal that unfortunately wasn't achieved within the duration of the project was that there is a separate view for module coordinators, who are only able to see certain things (for example which teachers are assigned to which tasks). Lastly, there needs to be a way to export the assigned teachers into a pre-defined format in Excel, which can be sent back to the scheduling department in order to assign teachers their time tables.

## 8.1 – Maintenance and Support of the Project

Since the project is currently implemented for the Applied Mathematics department at the University of Twente, it is advisable to host the software on a server of the UT. As mentioned in chapter 2.3 of this report, system requirements were put in place by LISA so they have the opportunity to maintain the current system. Since the system was developed with tools that the employees of LISA have expertise with, they should be able to maintain the system if needed. It is also safe to assume that the current implementation might still contain bugs that haven't been found during testing, these could also be solved by the employees of LISA.

## 8.2 – University-wide Usage

Although the current system is designed with only the Applied Mathematics department in mind, the system should be expandable to be used for other departments as well. By making some small changes to the current system, other departments could also use this tool to assign their teachers to their tasks. The advantage that other departments have over the AM department is that their teachers are often specialized in only a handful of topics instead of being able to teach a majority of the tasks, which also makes it easier to implement this for other departments.

# 9. Evaluation

## 9.1 – Planning

In order to ensure that deliverables were ready on time, the team used a SCRUM tool called Trello (Trello, 2021). The team's Trello board can be found at the following address: https://trello.com/b/8KD0VQ2H/planning. This board was used to indicate whether certain features were implemented or if they still needed to be done, as well as keep track of who implemented which feature. Meetings with the client and meetings with the supervisor were scheduled roughly every two weeks in order to discuss the progress made on the project, as well as get valuable feedback.

At the end of the module, several deliverables needed to be handed in. In order to finish these on time, the team set deadlines a little bit in advance of the actual deadlines to leave some room to read and possibly rewrite (parts of) the deliverables. This led to an overall higher quality of the deliverables.

## 9.2 – Responsibilities

Since the project group consisted of three students instead of the average of 4 or 5, the students each had more responsibilities. Each student has different levels of expertise and interests in different parts of the project, this was kept in mind while dividing the tasks among the team members. The following list gives a rough overview of which team member had which responsibilities:

Koen Reefman: Database Designer & Developer, Front-end Developer, Back-end Developer, Requirement Analysis, Testing, Poster Designer, Final Presentation Slides, Head of Communications, Editor in Chief.

Xin Wan: Front-end Developer, Back-end Developer, Head of Testing, Requirement Analysis, Peer-Meeting Presentations.

Ziwei Chen: Front-end Designer & Developer, Back-end Developer, Testing, Requirement Analysis, Head of Login Services, Peer-meeting Presentations, Final Deploy.

It should be noted that writing the deliverables was a team effort, which means that each team member contributed to these to a certain extent. A lower level division of tasks can also be found on the group's Trello board.

## 9.3 - Team Evaluation

During the project, the teamwork of the group wasn't the best at times. At the start of the project, this was caused by everyone working on separate parts and not needing to connect it together yet. Later in the project, due to a lack of communication between the team members, the teamwork also wasn't great. Even though all the systems worked together, the team members were not really cooperating and just implementing things on their own. This eventually led to a red card being presented to Xin Wan, due to the belief that he hadn't put in as much work as the other two group members. This was later solved by discussing it within the team and explaining what he had been doing.

## 9.3 – Final Result

This project resulted in a system that is efficient, easy-to-use, and understandable. It was created by involving the client in the design process by gathering feedback on working prototypes. Although there were some difficulties throughout the project (such as slow response times, as well as internal struggles), the main goals were achieved. All functions implemented in the system were based on requirements and feedback obtained from the client, therefore ensuring there are no unnecessary features in the project. This also ensured that the product met the expectations of the client.

## 9.4 – Conclusion

The meetings with both the client, as well as the group itself, ensured that the requirements of the client were met, and the deliverables were handed in on time. These meetings were crucial in distributing the workload over the entire duration of the project, as well as being able to consistently show new features.

The aim of the project was to create a system that helps the time table planners of the Applied Mathematics department assign tasks to teachers. The goal was to make the system provide the same functionality as the old system, while also making the system overall better to understand. Furthermore, the goal was to deploy the final product on a UT server; However, due to slow response times at the start of the project, as well as the group being with three people, this wasn't feasible in the given time frame, since there wasn't enough time to convert the JavaScript code into TypeScript code requested by LISA.

Finally, the project produced a working system which meets the requirements of the client. In the future, with some small adjustments, the system could be hosted on a UT server and can even be used by other departments as well.

Although there were many difficulties and unexpected situations during the entire project, some valuable lessons were still learnt.

# Bibliography


Bacinger, T. (2015). What is Bootstrap? A Short Bootstrap Tutorial on the What, Why, and How. Retrieved from https://www.toptal.com/front-end/what-is-bootstrap-a-short-tutorial-on-the-what-why-and-how

Bootstrap. (2021). Build fast, responsive sites with Bootstrap. Retrieved from https://getbootstrap.com/

Clegg, D., & Barker, R. (1994). *Case Method Fast-Track: A Rad Approach*: Addison-Wesley Longman Publishing Co. Inc.

Development Guidelines. (2021). Retrieved from https://utwente-dev.instructure.com/courses/103

Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. *Management review.*, *70*(11), 35-36. Retrieved from https://community.mis.temple.edu/mis0855002fall2015/files/2015/10/S.M.A.R.T-Way-Management-Review.pdf

Jain, N., Mangal, P., & Mehta, D. (2015). AngularJS: A modern MVC framework in JavaScript. *J. Global Research in Computer Science, 5*, 17-23.

Miller, F. P., Vandome, A. F., & McBrewster, J. (2010). *Apache Maven*.

Nielsen, J., & Molich, R. (1990). Heuristic Evaluation of User Interfaces. *Chi '90*, 249–256. doi:10.1145/97243.97281

Rollbar Editorial Team (2021). Spring Boot vs Spring MVC: How do They Compare? Retrieved from https://rollbar.com/blog/spring-boot-vs-spring-mvc-how-do-they-compare/

Trello. (2021). Retrieved from http://trello.com/

# Appendix 1

*Table 3) Stakeholder requirements*

| *Nr* | *Requirement* | *Explanation* |
|---|---|---|
| *1* | *As administrator, I want to log in and out of the system* | *An administrator has to be able to log in and out of the system.* |
| *2* | *As administrator, I want to assign certain teachers to certain tasks* | *An administrator has to be able to assign certain teachers to tasks that haven't been assigned to a teacher yet.* |
| *3* | *As an administrator, I want to be able to add teachers to the database* | *An administrator has to be able to add new teachers to the database when new teachers join the department.* |
| *4* | *As administrator, I want the system to recommend certain suitable teachers for certain tasks* | *Assigning teachers to tasks can be very time consuming for the administrator, therefore the system should recommend certain teachers that are qualified for the job (based on previous assignments to the same task)* |
| *5* | *As administrator, I want to see time tables for certain teachers* | *An administrator has to be able to see the schedule for a teacher in order to better plan the teacher for other tasks.* |
| *6* | *As administrator, I want to be notified if there are any conflicts in the planning* | *To ensure that teachers aren't scheduled for two tasks during the same time slot, the system has to notify the administrator of these conflicts.* |
| *7* | *As administrator, I want to know the total time needed for a course* | *An administrator has to know how much time is needed to perform certain tasks such as giving a lecture in order to make a good schedule.* |
| *8* | *As administrator, I want to see tasks which haven't been filled yet* | *The most important function the administrator has is to fill all open tasks. The system should notify the administrator in case certain tasks have not been filled.* |
| *9* | *As administrator, I want to prevent the data from being changed by the visitors* | *The administrator is the only person who can modify data in the system, other visitors such as the module coordinator should not be able to modify the data, only read it.* |

| | | |
|---|---|---|
| *10* | *As administrator, I want to save the most suitable teachers for each course.* | *An administrator knows which teachers are most suitable for certain tasks, the system should have a feature with which the administrator can save certain teachers for certain tasks.* |
| *11* | *As administrator, I want to know who is adjustable when there is no available teacher for a task.* | *When a certain task is not yet assigned to a teacher and there is no teacher available to be assigned, the administrator should receive suggestions from the system which other teachers could be adjusted to fill the task (i.e. swapping teachers)* |
| *12* | *As administrator, I want to make changes to the completed schedule.* | *An administrator should be able to make last minute changes to the schedule in case something unexpected happens (i.e. a teacher gets ill)* |
| *13* | *As administrator, I want to send notifications to teachers who have been assigned tasks.* | *An administrator should be able to send a teacher's schedule to that teacher every once in a while. This could also apply to small last minute changes to the schedule.* |
| *14* | *As administrator, I want a good backup of the database* | *An administrator should be able to back up the database manually. If the database gets corrupted somehow, the administrator can revert back to an older version. The system should also back up the database automatically.* |
| *15* | *As module coordinator, I want to log in and out of the system* | *A module coordinator has to be able to log in and out of the system.* |
| *16* | *As module coordinator, I want to see the latest tables of the teachers* | *A module coordinator has to be able to see the time tables of certain teachers.* |
| *17* | *As module coordinator, I want to see the latest tables of the courses* | *A module coordinator has to be able to see which teachers are assigned to which tasks for a certain course.* |
| *18* | *As module coordinator, I want to see the most recent changes to the arrangement* | *A module coordinator has to be able to easily see the most recent changes in the system. In case of illness, the module coordinator should be able to see which teacher got ill and which teacher now teaches the course.* |

*Table 4) System requirements*

| Nr | Requirement | Explanation |
|---|---|---|
| **MUST HAVE** | | |
| 1 | The system must have a mechanism which enables the administrator to log in and out of the system. | The mechanism that is used in our system for login is that it can link to the login system of the University of Twente, and the administrator can use their own usernames and passwords to login the system. |
| 2 | The system must have a mechanism which enables the administrator to schedule certain teachers for certain tasks. | On both the main page and task page, the administrator is able to pick an unassigned task and then click the button to jump to the all teachers page which can display all teachers list, then he is able to select one or multiple teachers by selecting the check boxes to assign to this task. |
| 3 | The system must have a mechanism which enables the administrator to see recommended teachers for certain tasks. | On both the main page and task page, the administrator is also able to select the radio button to see the recommended teachers for a task, the administrator is able to pick an unassigned task and then click the button to jump to the all teachers page which can display all teachers list, then he is able to select one or multiple teachers by selecting the check boxes to assign to this task. |
| 4 | The system must have a mechanism which enables the administrator to see time tables for certain teachers. | The administrator is able to click a button to see the time table of a teacher, this timetable page is able to display the detailed task allocation of this teacher for each working day. |
| 5 | The system must show the notification when there are conflicts in the planning. | When the administrator is assigning a teacher or teachers to a task A, if one of the teachers is already assigned to a task B which the time slot is conflicted to the task A, then a conflict page is able to pop up which notify the administrator that this teacher is already assigned to task B. |
| 6 | The system must have a mechanism that enables the administrator to change the completed arrangements | The administrator is able to change all the already assigned tasks (reassign the task to another one teacher or multiple teachers by clicking the button in the task details page.) |

| | | |
|---|---|---|
| 7 | The system must have a mechanism that enables the administrator to know the unassigned tasks | On both the main page and task page, the administrator is able to click the button to jump to the page that can see all unassigned tasks. |
| 8 | The system must have a good backup of the database | On the setting page, the administrator is able to back up the whole database by clicking the button. After clicking the button, the backup database file will be stored on the local laptop of the administrator. |
| 9 | The system must have a mechanism which enables the module coordinator to log in and out of the system. | This system is not implemented in this design, due to the time constraint and the small group size. |
| 10 | The system must have a mechanism which enables the module coordinator to see the latest time tables of certain teachers. | This system is not implemented in this design, due to the time constraint and the small group size. |
| 11 | The system must have a mechanism which enables the module coordinator to see the latest time tables of certain courses. | This system is not implemented in this design, due to the time constraint and the small group size. |
| **SHOULD HAVE** | | |
| 1 | The system should have a mechanism which enables the module coordinator to see the most recent changes to the schedule(s). | On the main page, the administrator is able to see the five recent changes of the task allocations, the administrator is also able to see more recent changes of the task allocations by clicking the button. |
| 2 | The system should have a mechanism which enables the administrator to notify teachers of their assigned tasks. | This system is not implemented in this design, due to the time constraint and the small group size. |
| 3 | The system should show the suggestion when there is no available teacher for a task. | This system is not implemented in this design, due to the time constraint and the small group size. |

| | | |
|---|---|---|
| 4 | The system should have a mechanism that enables the administrator to save the most suitable teachers for the course. | On the teacher page, the administrator is able to add one or multiple teachers to the favourite teachers table of the database by selecting one or multiple check boxes, each checkbox is represented to each teacher. |
| 5 | The system should have a mechanism that enables the administrator to know the total time needed for a course. | The administrator is able to see the total number of hours needed for a task on every page regarding the tasks. |
| 6 | The system must have a mechanism which enables the module coordinator to log in and out of the system. | This system is not implemented in this design, due to the time constraint and the small group size. |
| 7 | The system must have a mechanism which enables the module coordinator to see the latest time tables of certain teachers. | This system is not implemented in this design, due to the time constraint and the small group size. |
| 8 | The system must have a mechanism which enables the module coordinator to see the latest time tables of certain courses. | This system is not implemented in this design, due to the time constraint and the small group size. |
| **COULD HAVE** | | |
| 1 | The system could have the option to change font size on the website | This system is not implemented in this design, due to the time constraint and the small group size. |
| 2 | The system could have the option to change to dark theme on the website | This system is not implemented in this design, due to the time constraint and the small group size. |

# Appendix 2


*Figure 11) Login page*

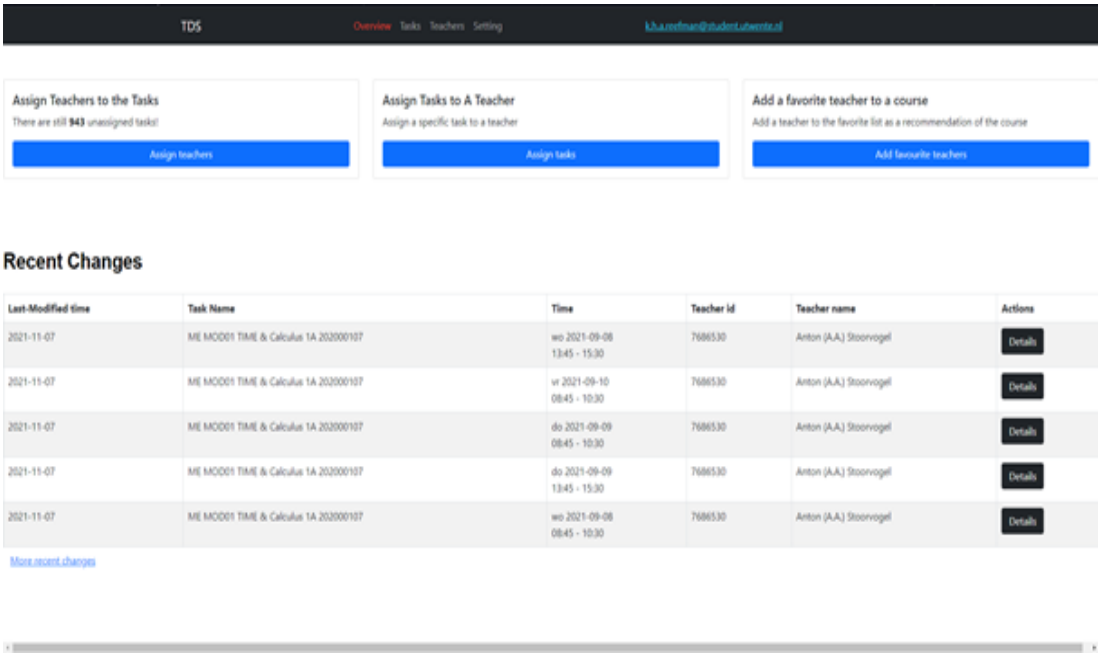


*Figure 12) Main page*

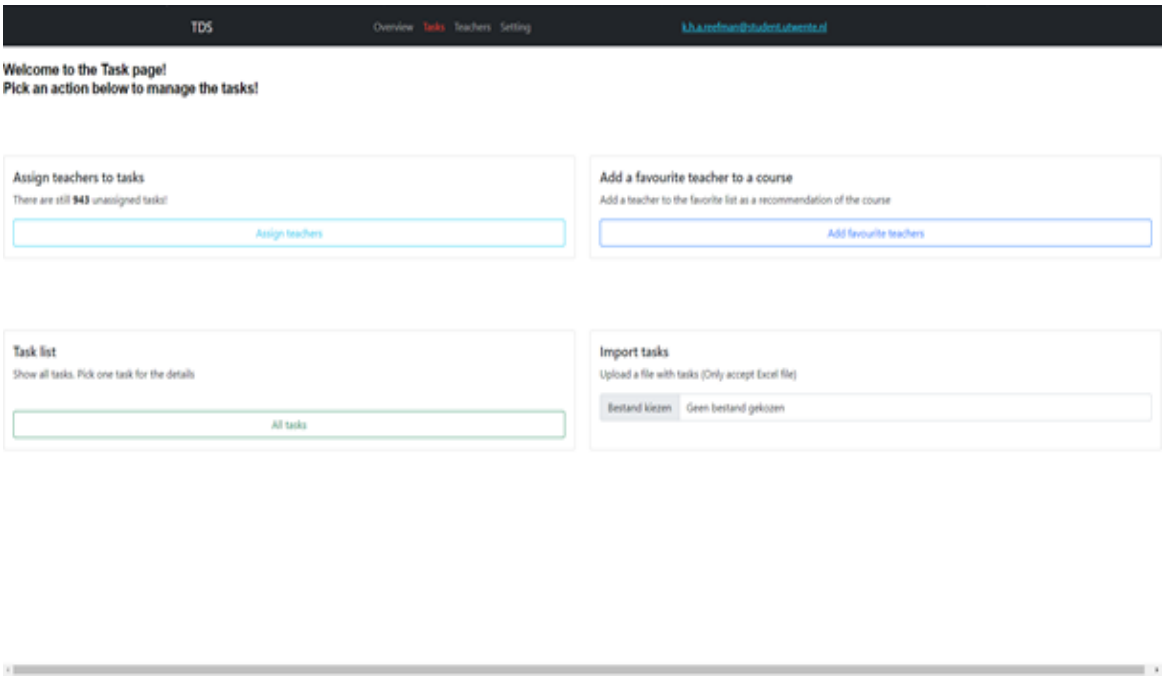


*Figure 13) Task page*

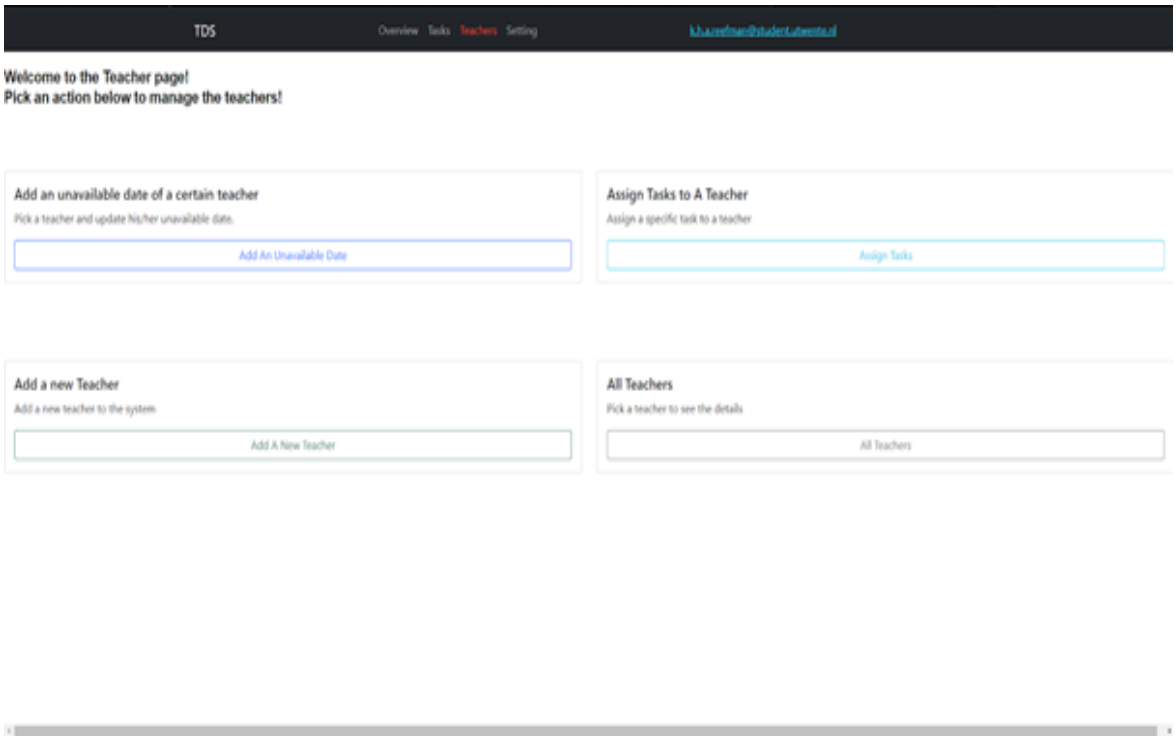


*Figure 14) Teacher page*

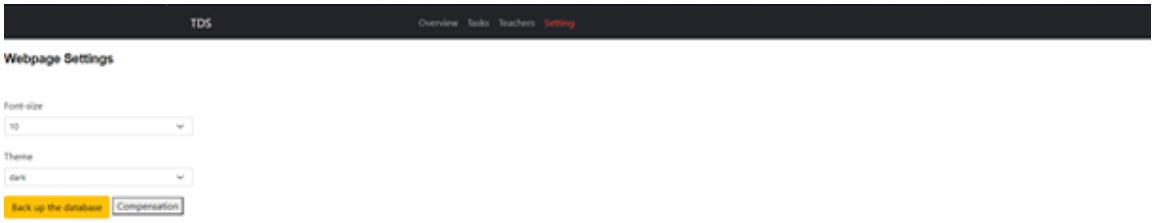


*Figure 15) Settings page*

# Appendix 3

*Table 5) Functionalities that will be tested in the Usability test.*

| # | Test Name | Description | Expected Result | Level |
|---|---|---|---|---|
| 1 | Login with Okta | Identity authentication is the mandatory requirement for the system. | Only the users with credentials can use the system. | H |
| 2 | Assign one or more teachers to one task | The administrator can pick one or more teachers for one task. If there are conflictions, it will show a popup about the conflicting details. | The teachers that are picked can be successfully assigned if they are free at that time bucket, or it will show a popup to inform the administrator. | H |
| 3 | Assign tasks to a teacher in his/her timetable | The administrator can assign tasks to the available time buckets in the timetable of the teachers. | The administrator can assign a task from the available task list and add to the timetable of the teacher. | H |
| 4 | Add one or more teachers to one task favourite list | The user can pick one or many teachers to add them in the favourite_teachers table in the database. | The teachers who are picked will be successfully added in the database,and will display the success alert. | H |
| 5 | Change teacher details | The user can change the teacher data in the teacher detail page like name, available hours.etc. | The teacher can enter the data he wants to change, and after clicking the save button, a success alert will be popped up. | H |
| 6 | Assign or Reassign teachers of one task in the “Task Details” page | The administrator can reassign the teachers for one task on the task details page when he wants to assign a certain task or conflicts happened during assignment. | The assigned teachers can be changed successfully. If the task is unassigned, one or more teachers can be successfully assigned to it. | H |
| 7 | Upload Excel file with tasks to the database | The administrator can upload the Excel file with tasks information to the database. | All tasks will be added to the database after uploading. | H |
| 8 | Add an unavailable date for a teacher | The administrator can add an unavailable date for a teacher. | The teacher will not be assigned any tasks on that day. | H |
| 9 | Add a new teacher | The administrator can add a new teacher to the database. | A new teacher with a different m_number should be added to the database successfully. | H |

| | | | | |
|---|---|---|---|---|
| 10 | Backup the database | The user can back up the database when click the backup button in the setting page | The backup file with the backup date will be stored in the user's local laptop. | L |
| 11 | Logout by clicking on the top email link | The administrator can log out anytime. | The system will log out to the login page and delete cookies. | H |