

Prometheus

'Prometheus was the wisest of all.
His name means "forethought" and
he was able to foretell the future'

Abstract

The paper in front of you is a report of the development of a system called "Prometheus". Prometheus is a system that can help teachers communicate homework to their students and that supports students in planning their homework.

It is believed that the creation of this system is a (partial) solution to a problem that has been recognized years ago in the Dutch education system. Students seem to have problems planning their homework, especially at the University where there are few contact hours. This system can solve this problem by providing functionalities that support planning.

Prometheus is a database application that is accessible over the internet. It is based primarily on PHP and (my)SQL techniques. It can be used by student and teacher (who can also be administrators).

This paper will cover many different aspects of the development, including requirements, functional requirements, organization and planning etc.

Inhoud

1 - Introduction.....	4
2 - Problem	5
2.1 - Problem description	5
2.2 – Goal of the project.....	5
2.3 – Users	5
2.4 – Requirements	6
2.5 – Scenario	7
2.6 – Blackboard	7
2.7 – ‘And we shall call him... Prometheus’	8
Approach	9
3.1 – Method	9
3.2 – Techniques and Tools	9
3.3 – Function Specification	10
3.4 – Planning and organization	12
Results	13
4.1 – E-R model & Realized functionalities	13
4.2 – The Database	15
4.3 – System Architecture	16
4.4 – Graphical User Interface Design	17
4.5 - Security.....	21
Evaluation	22
5.1 – Evaluation of Prometheus	22
5.2 - Extensions	22
5.3 – Evaluation of the project	22
References.....	23
Appendix.....	24

1 - Introduction

In 1998 some changes were made in the Dutch education system. A number of measures were introduced to promote independent studying by students in high school. For example: students were given more control over their own time schedules and there were less checks on the completion of homework. The goal of these measures was to teach the students to work on their own and thereby make it easier to bridge the gap between high school and university. At the university, students are expected to demonstrate a greater deal of independence than they are used to in high school. This causes students to drop out of university [1].

Recent research among students showed that students still feel they do not have enough planning abilities compared to other essential abilities [2]. The percentages in the image below show how much of the students think they master a specific skill “reasonably well to good”. We see that 70% of the students of VWO (high school) with the Second Phase education method think that they master the skill of planning reasonably well to good. This means that 30% does not. We can see in the diagram that this percentage is a lot lower than the percentages of the other abilities (including communication skills, collaboration skills, analytical skills etc.). We can conclude that a lot of students still have problems with planning their homework at the university.

To help students at the university cope with this high degree of independence, tools can be helpful. In this project such a tool is developed. This tool can help students cope with their independence by giving them overviews of homework with associated deadlines and by giving them the possibility to plan their homework in an easy and clarifying way.

FIGUUR 48

OORDELEN STUDENTEN OVER DE BEHEERSING VAN ALGEMENE VAARDIGHEDEN

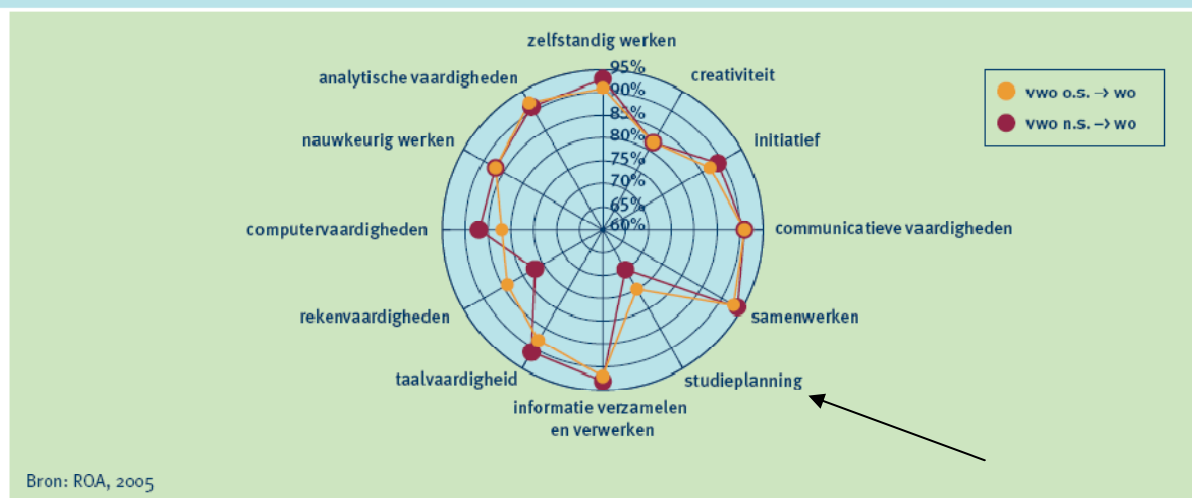


Figure 1 - Judgements from students about their own abilities (includes planning) [2]

2 - Problem

2.1 - Problem description

A lot of students still seem to have problems with planning their homework. In the Dutch education system the government is trying to solve the problem in high schools with the Second Phase. This project targets the students already studying at the university with problems with planning.

Studying at the university is different from going to high school. At the university students have less contact hours than students in high school, but a lot more homework. There is also less control on the completion of homework. Thus, it's the responsibility of the student to plan their homework to fit in the available time. There are a couple of problems that can be expected here. First of all, students can underestimate the amount of homework because they do not have an overview of all the homework assignments. They put off the homework and get in trouble when the deadline approaches. Also, students can find it hard to plan their homework in their head or in another unorganized manner. They may forget assignments or use their time inefficiently.

What adds to these problems is that teachers often do not have a standardized manner of assigning homework. They tell what needs to be done face-to-face, put it on paper and pass that around, put it on a website or put the homework on a digital learning environment. Even within one method, the form can vary greatly.

2.2 - Goal of the project

The goal of this project is to develop a system that addresses the problem described in the problem description, i.e. a system that helps teachers communicate homework to students and helps students to plan those homework assignments. This goal is divided in specific functionalities in the requirements section (2.4).

The system is developed at the University of Amsterdam and will form an extension of the current digital learning environment in use there. This digital learning environment will be described in section 2.6.

2.3 - Users

Three types of users can be distinguished: Students, teachers and administrators. Assistants will get the same rights as teachers and will therefore also be called teachers.

Students: Students can enroll for courses, view their homework and devise a planning for their homework.

Teachers: Teachers can schedule homework assignments for the courses they teach. They also have the possibility to view all the homework assignments of their students. Teacher cannot view the planning of the students.

Administrator: Administrators manage the application. They can add teachers, students and courses to the system. They also have the possibility to view and edit information about teachers, students and courses.

At this stage, only teachers can be administrators (and an administrator is always a teacher). This choice was made to save some time in the implementation phase.

The exact possibilities of the users will be described in the next section.

2.4 – Requirements

This application is designed for students and teachers to help them plan their homework. The system will be an extension to Blackboard. The Prometheus application is a stand-alone application and only extends Blackboard in a way that it offers a possibility that Blackboard does not. Thus enrolling in a Blackboard course will not enroll the student in Prometheus. If Prometheus proves to be successful, the two systems can be integrated. For the time being Prometheus is designed as a stand-alone system however. The system should meet the following requirements.

- *Teachers should be able to see which students are enrolled in the courses that they teach.*
- *Teachers should be able to add homework to the system, with the appropriate deadline and an estimate of the time required. The teacher should only be able to add homework for their own courses. The teacher should also be able to edit the homework. The homework deadline can not occur the created date.*
- *Teachers should have an overview of all the homework they have assigned.*
- *Teachers should have an overview of the homework they assigned per course. This and the previous overview should be able to sort by deadline and by course.*
- *Teachers should have an overview for each course, that tells them which other courses their students have. The homework assignments for these other courses should be displayed next to the own homework assignments, and conflicts with deadlines should be highlighted. This overview should allow a teacher to view homework from other courses, and to edit own homework assignments*
- *Administrators should be able to add courses to the Prometheus system, and assign teachers to these courses. No course should be added if the end date for that course is before the date it is created.*
- *Administrators should be able to add and remove teachers to the system and classify them as also being an administrator or not.*
- *Administrators should be able to add and remove students to the system. However, if an existing student number is used, the system should give an error.*
- *Students should be able to enrol for the available courses. They should not be able to enrol in a course they are already enrolled to.*
- *Students should be able to see for which courses they are enrolled.*
- *Students should have an overview of all the homework in one course. The required time, the deadline and the content of the assignment can be read from this overview.*
- *Students should have an overview of all the homework that they are assigned to do. This means there should be an overview of all homework from the courses the students are enrolled in. This overview can also be sorted by deadline and by course name.*
- *Students should be able to devise a planning (workpackages) for their homework assignments, specifying when they plan to do or to finish the assignments. The students should also be able to add a short personal comment to this planning and to mark the completed homework when it is completed.*
- *The system should allow a student to add a workpackage about a homework assignment, even if the planned date extends the official deadline. This allows the student to plan non-deliverable assignments such as ‘reading’ homework in their own time. The student will be notified by the system though, by changing the personal deadline to a read color if it extends the official deadline.*
- *Students can delete their workpackages whenever they wish (completed or not).*

2.5 – Scenario

In this scenario, the system and its functions will be described to show how it can help students and teachers. Note however that not all functionalities are covered in this scenario.

Henk has just started at the University of Amsterdam studying “Informatiekunde”. He is a bit worried about the workload at the university because he never was really good at planning his homework in high school. He sometimes tried to plan his homework in his head but this never really worked out.

In the first week of the first semester, Henk receives a letter about a system called ‘Prometheus’ along with login information. When he logs in he sees he can enroll for the courses he has to follow. He enrolls for “Basis Informatica”, “Informatie en organisatie” and “Webtechnologie- en talen” and these courses are immediately displayed under “enrolled courses” so Henk knows all went well.

Now Henk clicks on the homework/planning-button. This brings him to the page where all his personal homework is displayed. (This homework is entered into the system by his teachers, see below.) From this overview of homework Henk can start to devise a personal planning. The system provides functionality for creating a planning so Henk doesn’t have to do it in his head anymore.

Henk creates a planning for the following week. He plans to ‘read Chapter 1’ for the course ‘Basis Informatica’ on Tuesday. The required time for this assignment is set on two hours so Henk thinks he can do it right between dinner and his soccer practice. Henk repeats the process of planning the assignments for the next week. As the week progresses, Henk checks back every now and then to see what he plans to do on that particular day. He can mark assignments as being completed and/or delete the assignment from his planning.

The week before the semester starts, Theodore is planning the course he has to teach coming semester. When he is done, he starts Prometheus and logs in. Because Theodore is a teacher he can add homework for the courses he is teaching. He can thus use Prometheus to communicate the homework to the students. When he entered the homework for his courses he can check which students have enrolled for his course already and how much homework these students have. This way he can try to avoid any conflicts with the homework of other courses (and possibly change a homework deadline).

2.6 – Blackboard

At the University of Amsterdam homework is mainly communicated to the students using an electronic learning environment called “Blackboard”. The way teachers assign homework using Blackboard differs greatly. Some teachers put all the homework in a Microsoft Word Document and put that on blackboard, others use the announcements to assign homework, examples are abundant. These differences however are not due to a limitation of Blackboard. Blackboard actually offers a functionality for teachers to assign homework (called ‘tasks’) to students in a standardized manner (figure 2 on the next page).

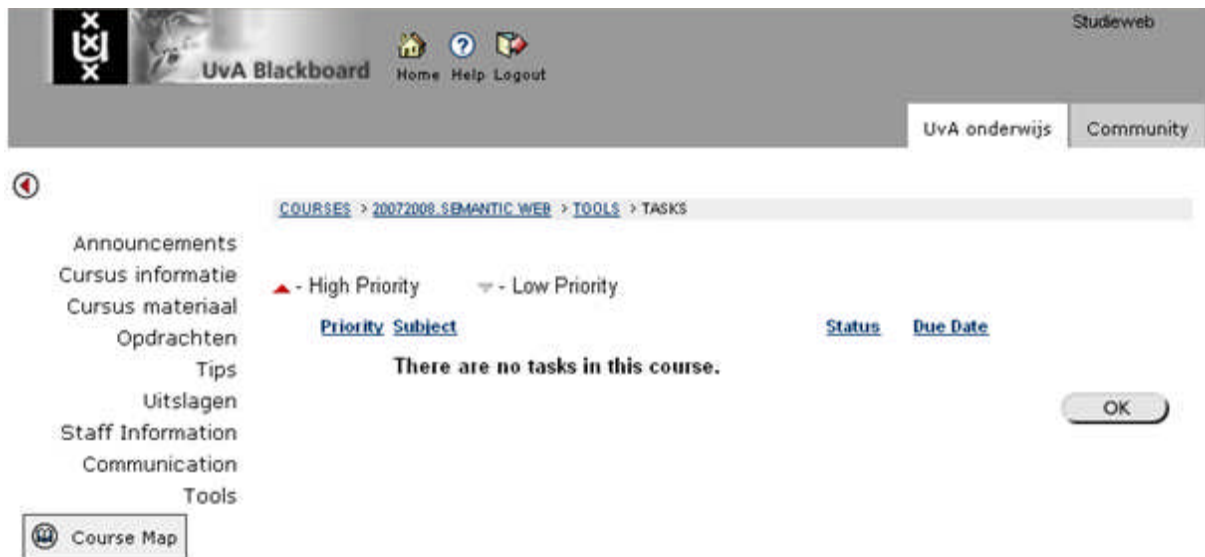


Figure 2 The functionality of Blackboard to assign homework to students

This functionality hardly seems to be used. Then why will the system developed in this project be any different? What is the difference between the Blackboard function of assigning tasks and the system developed in this project?

First of all, the system developed in this project provides an overview of all the homework of a student, not just that of one course. The combination of the homework of all courses into one overview will be more valuable than one distinct overview per course. It will be easier to compare courses or homework assignments and conflicts between different assignments can be found more easily. Also, students only have to look in one place to see all their homework assignments.

Secondly, teachers will be able to add more information about assignments in the system developed in this project than in Blackboard. In Blackboard teachers can add a priority indication, a subject, a status and a due date. The system developed in this project will have more information fields including for example a 'long description' for a detailed description (more than one line) and a field for the (estimated) 'required time'.

Finally and probably most important: the system developed in this project will have the functionality to devise a planning using the overview of the total homework. This way, students can really use the information to make concrete plans instead of an ambiguous plan in their head. It is like having homework already written in their agenda, they only have to give it a place.

The system developed in this project will thus be an extension to Blackboard, able to provide more functionalities than the current Blackboard. (In this phase the system will be designed as a stand-alone system for testing purposes.)

2.7 – 'And we shall call him... Prometheus'

The system is called Prometheus as briefly mentioned in the scenario. Prometheus is a character from the Greek mythology. His name means forethought and he taught people to think ahead. Prometheus also stands for the subjection of nature, taking control. Prometheus didn't adapt to its environment but adapted his environment to him. He tried to understand how things worked so he could use them for his own goals. (P. Pekelharing, lecture 2 Honoursmodule Ethics, 10 oktober, 2007.)

Approach

3.1 – Method

The method used for the various parts of this project will be explained here. Some choices that were made during the project will be explained here as well.

The first step of the project was to make an analysis of the problem. The two group members used their own experience with homework and planning as frame of reference. Through conversation and brainstorming the problem became clear.

During these brainstorming sessions the user requirements were defined as well. A few scenarios were imagined and the user requirements were recorded along the way. The user requirements were thus created and tested by scenarios.

During this phase, the system was designed in a general sense as well. The intention was to create a database application that was accessible over the internet. The choice was made for a relational database application because that is the most ideal form of storage for the kind of information used in the system. Relational databases provide the possibility to connect information (hence: *relational* database) and create new information without having to store it explicitly. It should be accessible over the internet because students have to be able to access the system from home, school or anywhere else. (More on the architecture of the system in section 4.3.)

From that point on, the implementing phase began. This was really a trial and error process. On the one hand trial and error on the implementation side (e.g. code, queries), on the other hand on the interaction side with the University of Amsterdam (design, show design, re-design with feedback).

The documentation was during each phase of the project but especially when the application was finished. This way there was no need to rewrite parts because of changes to the system. (To illustrate this point, there were about 5 different versions of the Graphical User Interface. It would be inefficient to change the documentation with each version, therefore only the last version is documented.)

3.2 – Techniques and Tools

The following techniques and tools were used during the development of this project:

Techniques:

(x)HTML	Used to structure the elements on the website.
CSS	Used to define the layout of the (elements in) the website.
PHP	Used to create parts of the websites dynamically <i>using</i> information from the database. PHP was also used to process input from html-forms and to keep variables during the entire visit of the user to the website.
MySQL	Used (within PHP) to communicate with the database.
SQL	Used (within PHP) to write queries that can be applied to the database using MySQL.

Tools:

PHPmyAdmin	Used to create and manage the database
Online editor	Used to write all the code. It is a very basic editor. It only colors some parts of the code. The advantage of this editor was that the creators could work on the server directly and didn't have to upload their files repeatedly. (A screenshot of the editor can be found in the appendix, appendix 2.)
Photoshop	Used to create the bars at the top and bottom of the interface and the image in the log in screen (also used in a different form as a watermark throughout the system).

All these techniques were chosen because they were perceived as being the most commonly used techniques for this kind of project. The tools were chosen by availability. Because the project is a learning project, it might be noteworthy that the creators only had (some) prior experience with (x)HTML, CSS, Photoshop and with SQL-queries. Figure 3.1 shows where the different techniques and tools were used.

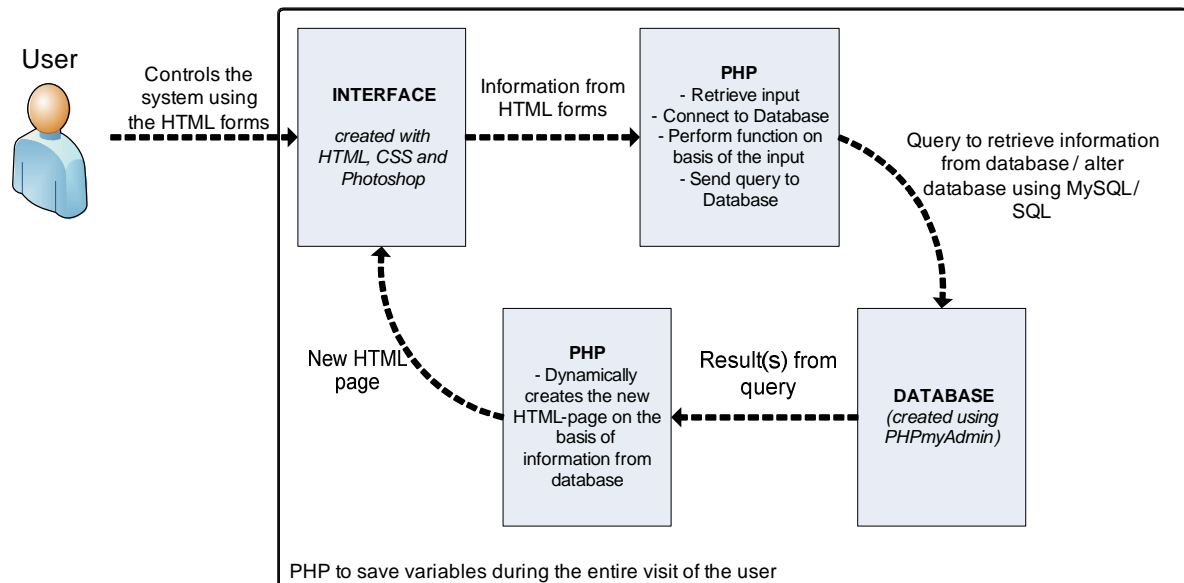


Figure 3.1 Techniques used in Prometheus

For the calendar we used an open source PHP calendar class (<http://www.cascade.org.uk/software/php/calendar/>). It was modified to work with our system. The calendar class that was used only offered the basic functions, it could only display the current month or year, with the appropriate dates. However, the class had to be fully understood and modified for use with the Prometheus project. This required a great deal of time and programming during this project.

It was beyond the scope of this project, to build our own php calendar. Moreover, the calendar we used from this open source website had no connection to a database when we used it. The modifications that took place allowed this calendar to communicate with the database properly, which is the very core of this project.

3.3 – Function Specification

- Show courses for/from a person. The interface which relies on the php code, should be able to call a 'select' statement, which retrieves the course number, course name and start & end date. If the user is an administrator, this person should be able to delete information from this table with a SQL Delete function. If the user is a student or a teacher, this person should be able to view a more specific description about this table information
- Enroll to courses. The Gui should allow to call an Insert SQL statement, once a student decides to enroll to a course. Students should not be able to enroll to a course twice.
- Show homework for/from a person. The PHP code should be able to call a 'select' statement, which retrieves the homework number, homework description, deadline and the required amount of time. If the user is a teacher, this person should be able to edit or delete information from this table using an SQL Update statement. If the user is a student, this person should be able to view a more specific description about this table information

- Show homework from a course. The PHP code should be able to call a 'select' statement, which displays the homework number, homework description, deadline and the required amount of time which apply for the selected course. If the user is a teacher, this person should be able to edit or delete information from this table using an SQL Update statement. If the user is a student, this person should be able to view a more specific description about this table information
- Create Workpackage. The GUI should allow the user to select any of the homework assignments and let the user create a personal deadline and a personal comment for it. The GUI then calls an Insert statement. This function will only be available for the user group 'students'.
- Show Workpackage from a person. This piece of code should allow the user group, which are only students in this case, to view information from their personal workpackages; personal deadline, planning and completed. This function uses a Select SQL Query to find all workpackages for a certain user. Users can also update the attribute 'completed'. This function calls a SQL statement which updates the database.
- Update profile. This function applies to teachers and students who wish to update their profile. This code should be designed to update the database with an 'update' statement. When the user wishes to update his/her password, it has to be entered twice, and the second user input should match the first.
- Create homework. This function allows the user group 'teachers' to create information which will be added to the database. The user has to insert the description, deadline and required time. A long description is optional. When the created deadline occurs before the current date, the system should warn the user. Otherwise, the code should call an Insert statement.
- Homework Overview. This function is designed to retrieve the essential information about a user's courses, and other courses and deadlines linked to it. The system should retrieve the proper data from the tables, color them appropriately, and allow the user to use this information to view, or edit it, depending on the authorization. This function is only available for teachers.
- Create teacher. A php code designed to call an insert statement allows the user to create a teacher, defining the teachers last name, first name and whether this person should be an administrator or not. A teacher number should automatically be generated by the system. This function is only available for users who have an 'administrator' tag.
- Create Student. This code is almost the same as the code above, and also available to administrators only. However, this time the administrator cannot define the student as an administrator. Also, a student number has to be created. This is because students often already have a student number, created by the university administration. The system should warn the administrator if the created student number is already in use.
- Create course. This PHP code has to allow the user to create a course name, a start date and an end date, and to assign one or more teachers to it. This code then calls an insert statement which inserts the course into the databases, and assigns the teacher(s) to this course properly. This function will only be available to the administrator group.
- Sort table. This code uses the SQL function 'order by' to refresh the table and to sort the requested attributes. This function is available to all users.

3.4 – Planning and organization

The general planning of the project can be seen in the table below.

Week	Activity	
1	Overall planning, defining topic, initial ER modeling	
2	User requirements, scenario, ER-model, learn PHP / MySQL	
3	Learn PHP / MySQL, implement first functionalities of students, setting up database	
4	Implementation: Log-in, student functionalities, expanding database	
5	Implementation: Student functionalities, admin functionalities Documentation	
6	Implementation: Teacher functionalities, admin functionalities Documentation	
7	Implementation, documentation and testing	
8	Presentation / Demo	

Each week the two group members met several times for a meeting, once or twice each week with someone from the University of Amsterdam to discuss the progress of the project. Outside of those meetings, the group member communicated through e-mail and Microsoft Live Messenger (instant messaging). Both group members participated equally in the different forms of tasks (programming, modeling, documenting etc.). This was because of the nature of the project: a learning project.

Results

4.1 – E-R model & Realized functionalities

In this section the conceptual structure of the database will be described using an Entity-Relationship-Diagram. The choices in this ERD shall also be explained in this section.

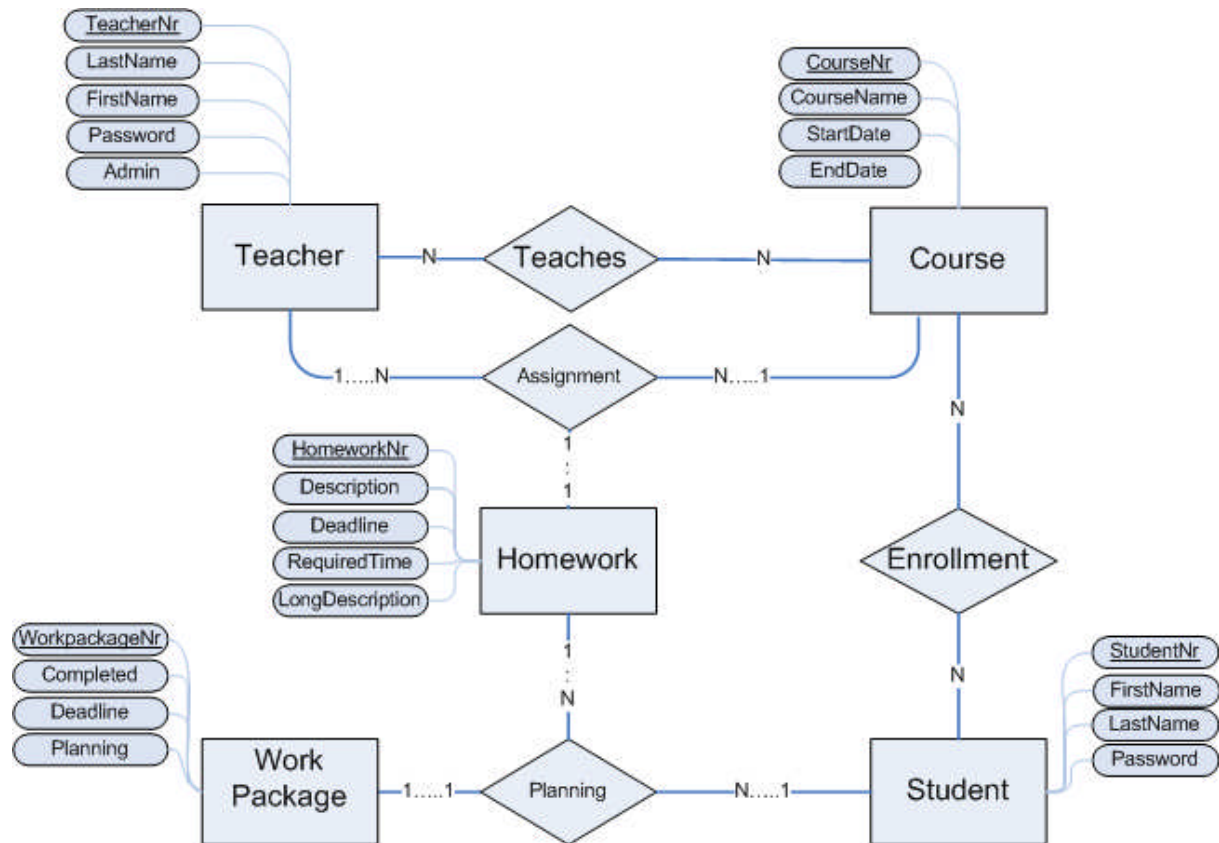


Figure 4.1 ER-diagram

The following choices have been made during the construction of this ERD.

- First of all, multiple teachers should be able to teach the same course. For example, the course 'statistics' is taught by a lecturer and an assistant (they will both be registered as 'teachers').
- A Teacher can teach multiple courses.
- A teacher can create multiple homework assignments for one course, and can create homework for every course he or she teaches. This means that a teacher can also create homework for multiple courses. A homework assignment can only belong to one course and can only have been created by one teacher.
- The primary key in the 'assignment' relationship is 'HomeworkNr'. This relationship contains three attributes: TeacherNr, CourseNr & HomeworkNr. and links the three entities (Teacher, Course & Homework) together. The only unique key in this relationship is 'HomeworkNr', because the same teachernumbers and coursenumbers will occur multiple times in this table. In this application, only a single assignment can be linked to a homework object. This is because otherwise homework assignments wouldn't be as specific as they should be. For example, a homework object 'read chapter 1' could be used for multiple assignments. What Prometheus needs however, are more specific homework objects which have a required time and a longer description. Although 'read chapter 1' can be re-used, the required time

and the longer description cannot. It would be possible to merge the 'homework' and 'assignment' table, since they both contain the same primary key. This design flaw was recognised by the designers and has been a point of discussion even during the implementation phase. The separation of this data could indeed support the function we declined: reusing homework objects for multiple assignments. Since that is not the case, these tables could best be merged. However, by the time the designers agreed on this point, the implementation phase had already started and many files relied on the original database design. Since the separation of the tables 'assignment' and 'homework' functions equally well as one table containing both 'assignment' and 'homework' data, the designers chose to leave the tables as they are. This design mistake has been recognised, but will not affect the functionality of the system. Although 'Teacher' and 'Course' are linked together twice, without the 'teaches' relationship the database could never know if a teacher teaches a course, if there isn't a homework assignment that links the teacher to the course. On the other hand, without the 'teacher' or 'course' attribute in the 'assignment' relationship, the database can't tell which homework belongs to which teacher or course. This double relationship cannot be avoided and will also appear between the 'student' and 'course' entities. Teachers can only create homework for their own course. Students can only create workpackages for homework assignments that they have, which implies they can only create workpackages for courses they are enrolled to.

- A student can be enrolled in multiple courses, and a course can be followed by multiple students.
- A student can create a workpackage for every homework assignment he or she receives. Using the 'planning' relationship, the entities 'Student', 'Workpackage' & 'Homework' are linked together. For this connection, the 'Planning' relationship contains the attributes 'StudentNr', 'HomeworkNr' & 'WorkpackageNr'. 'WorkpackageNr' will be the primary key, because this relationship will contain multiple workpackages about the same homework assignment (and thus the same homeworknumbers and studentnumbers). A workpackage can only be made by a single student, while a student can create multiple workpackages. A student has multiple homework assignments, and a homework assignment is intended for multiple students.
- Student and course is linked together twice, but the same argument as described above applies here; without the 'enrollment' relationship, the database can't tell if a student is enrolled in a course when that course has no homework assignments.

4.2 – The Database

In this section we will give an overview of the database. In the diagram below you can see the tables and the attributes. The lines depict a direct connection between two tables. (More detailed table schema can be found in the appendix (appendix 1)).

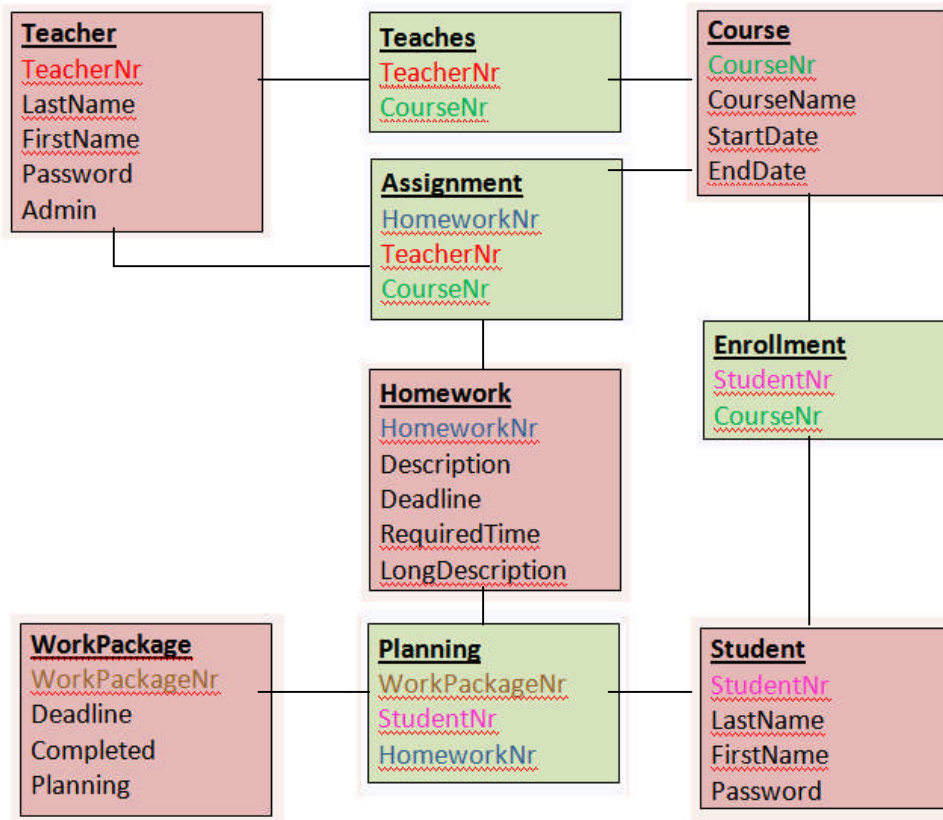


Figure 4.2. Database schema

The database was designed using the ER-model. The explanation of the ER-diagram is therefore for a large part also the explanation of the structure of the database. Attributes that may be unclear are explained here:

Teacher.Admin	A Boolean value that determines whether a teacher is also an administrator or not.
Homework.Description	Short description of the assignment, for example "Read chapter 5"
Homework.RequiredTime	An estimate of the teacher how long the completion of the homework will take
Homework.LongDescription	This attribute provides the possibility to provide a longer description of the homework assignment. For example: <i>"Read chapter 5 of 'Databases for Dummies'. Pay special attention to the section 'ER-diagrams' because next week you'll have to make on yourself. Also don't forget to bring the chapter to the next lecture"</i>
WorkPackage.Deadline	The personal deadline of the student for an assignment (not the teachers deadline!)
WorkPackage.Completed	A Boolean value that determines whether the assignment is completed or not
WorkPackage.Planning	This attribute provides the possibility to enter a subtask. If the assignment is for example "Read chapter 1 to 3" the subtask might be "Read chapter 1".

4.3 – System Architecture

The system developed in this project consists of three components: the graphical user interface (GUI), the storage and query manager and the database. In this project a client/server model was used to arrange these three components.

The client is the computer of the user. With his computer, the user can connect to the system from all over the world over the internet. When he connects to the system, the GUI is transported to the internet browser of the machine of the user. The processing power of the client is used to generate the GUI on the screen.

Whenever the user uses a functionality of the GUI, a request is sent to the storage and query manager on the server to provide that functionality. This storage and query manager will honor the request if possible by contacting the database.

The database is also located on the server. A query on the database will produce a result which is sent back to the storage and query manager.

When the storage and query manager receives the results from the database, it will put it in the right form and sent the information back to the GUI of the client. All the processes of the storage and query manager and the database use the processing power of the server.

When the GUI of the client receives the new information it will display it and the entire process can start again.

This approach has a couple of advantages. First of all, not much is asked of the clients. The client machines only have to display the information and send requests. The server does most of the difficult processing. This way, almost any user with a system that has a internet browser can use the system. The second advantage of the client / server architecture is safety. A user cannot access or change the database directly. This way, the creator of the system can determine exactly what a user can do/see and what not.

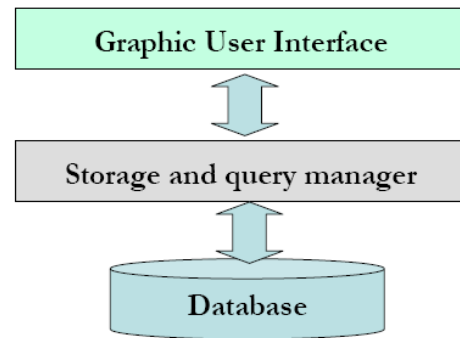


Figure 4.3. System architecture

4.4 – Graphical User Interface Design

In this section the graphical user interface (GUI) will be discussed.

Log in screen

All users start at the same log in page. This first page is designed to be visually attracting and recognizable. The man in the painting is Prometheus. The same picture will be embedded as a watermark throughout the entire system. The users have to enter their username and their password in order to log in. The password won't be visible when it is entered.

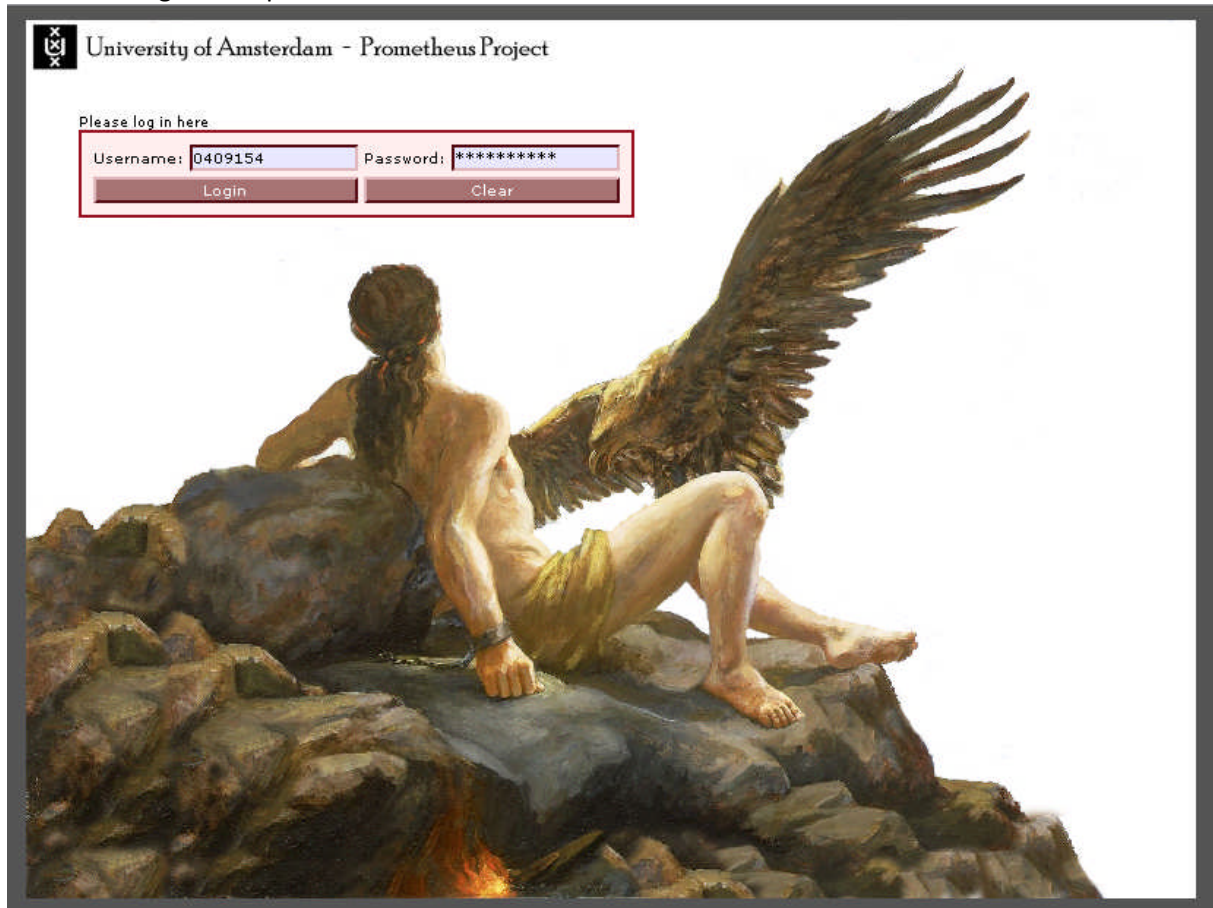
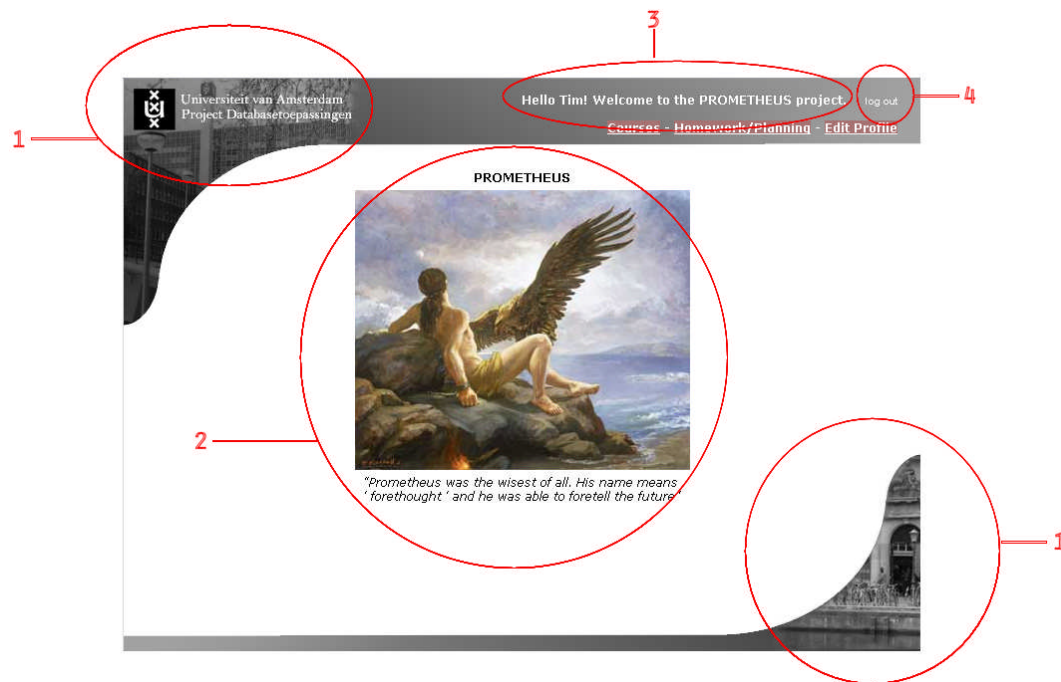


Figure 4.4 Log in screen

When a user logs in, the system determines what kind of user it is (student, teacher or teacher/administrator) and will send the user to the right page. (Photoshop 7.0 was used to adapt the picture of Prometheus.)

Aesthetic GUI-components



1: On the top and the bottom of the screen there are bars. These bars are meant to associate the system with the University of Amsterdam and to make the system more visually pleasing. There are two pictures in the bars, one of the 'Roeterseiland' complex and one of the 'Oudemanhuispoort' complex, the two largest complexes of the University of Amsterdam. The top bar also contains the logo of the University of Amsterdam, the text "University of Amsterdam" and the title of the course in which the system was developed. (Photoshop 7.0 was used to create these bars and picture from the University of Amsterdam website.)

2: All users are welcomed with a picture of Prometheus and a quote about him. This is to create an 'academic' and inspiring feel and to keep reminding users why the system is called Prometheus. (Actually, the website of the University of Amsterdam also has some references to ancient Greek characters.) Again, this picture is also meant to make the system more visually pleasing. The same picture is also embedded as a watermark throughout the entire system for the same reasons.

3 and 4 will be mentioned in the next subsection: Menu.

Menu

The menu is always available at the top of the screen.



At the top there is a welcoming message including the name of the user (3 in figure 4.5). This will provide feedback about a successful log in (and about who is logged in). Next to the welcoming message, a user can also log out pressing the log out button (4 in figure 4.5).

On the left of the screen teachers and administrators have a little information field which shows if the user is on an admin page or on a teacher page. This is there because some users have two roles. The buttons are made red and with a rollover effect to make them look like buttons. Teacher who are also administrators have an extra button in their menu with capital letters. This button is different from the other buttons and thus displayed different (with capital letters). This button lets the user switch between the teacher page and the admin page.

Main pages

All the information that appears in the center of the screen has the same design. It has a color-scheme designed to fit the entire site design but also to make the information more clear (and attractive). Each page has a watermark embedded in it.

Each table has a title so that the user knows what the meaning of the table is. The buttons in the tables have a slightly different layout so that they will be recognized as buttons. What the button is for is displayed on the button and in the top row of the table.

Enrolled Courses					
Coursenr	Coursename	Startdate	Enddate	Show homework per course	More Info
1	Conceptueel Modelleren	2007-09-03	2007-10-31	<-- Show Homework	More Info
3	Databases IK	2007-09-05	2007-10-31	<-- Show Homework	More Info

Available Courses				
Coursenr	Coursename	Startdate	Enddate	Enroll
2	Organisatiekunde	2007-09-02	2007-10-31	<-- Enroll
1	Conceptueel Modelleren	2007-09-03	2007-10-31	<-- Enroll
3	Databases IK	2007-09-05	2007-10-31	<-- Enroll

Figure 4.7 Example of table layout

The homework/planning page of the students is probably the most important page of the entire system. This page also has more interactional options (next to the buttons) and a calendar.

Homework					
#	Course name	Assignment	Date	Required time	More info
1	Conceptueel Modelleren	Opdracht 1 blackboard	2007-10-10	03:30:00	<-- More info
5	Conceptueel Modelleren	Opdracht 2 blackboard	2007-10-18	03:55:00	<-- More info
10	Conceptueel Modelleren	dubbele deadline	2007-10-18	05:10:00	<-- More info
11	Conceptueel Modelleren	driedubbele deadline	2007-10-18	03:00:00	<-- More info
3	Databases IK	Read chapter 5	2007-10-28	02:00:00	<-- More info

October 2007						
S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Add assignment to planning			
Select Assignment	(Sub)task	Personal Deadline	Completed
Databases IK[3] : Read chapter 5		2007 10 21	No
insert		Clear	

Planning						
Course	Task	(Sub)task	Personal Deadline	Completed	Edit	Delete
Conceptueel Modelleren	Opdracht 1 blackboard		2007-10-08	Yes	Edit	Delete
Conceptueel Modelleren	Opdracht 2 blackboard		2007-10-18	No	Edit	Delete
Databases IK	Read chapter 5	Test	2007-12-17	No	Edit	Delete
Databases IK	Read chapter 5	Test 2	2007-10-29	No	Edit	Delete

Figure 4.8 Homework/planning page

The interactional possibilities are for a large part self-explanatory and very common over the entire internet. It is assumed that a regular user will have experience with elements like drop-down boxes and text input fields.

Some tables in the system can be sorted. The columns that can be sorted can be recognized by a slightly different color from the other (dark red instead of black). They all have rollover effects to make them look more like buttons.

Some deadlines in the planning table are red. This means that the personal deadline is after the deadline set by the teacher.

The calendar also uses colors to add information to the display. Orange is the current day. Light blue means one deadline and dark blue means more deadlines on one day. Users can click on these dates to get more information.

The calendar on the page in the screenshot below uses different color codes. They are explained on the page.

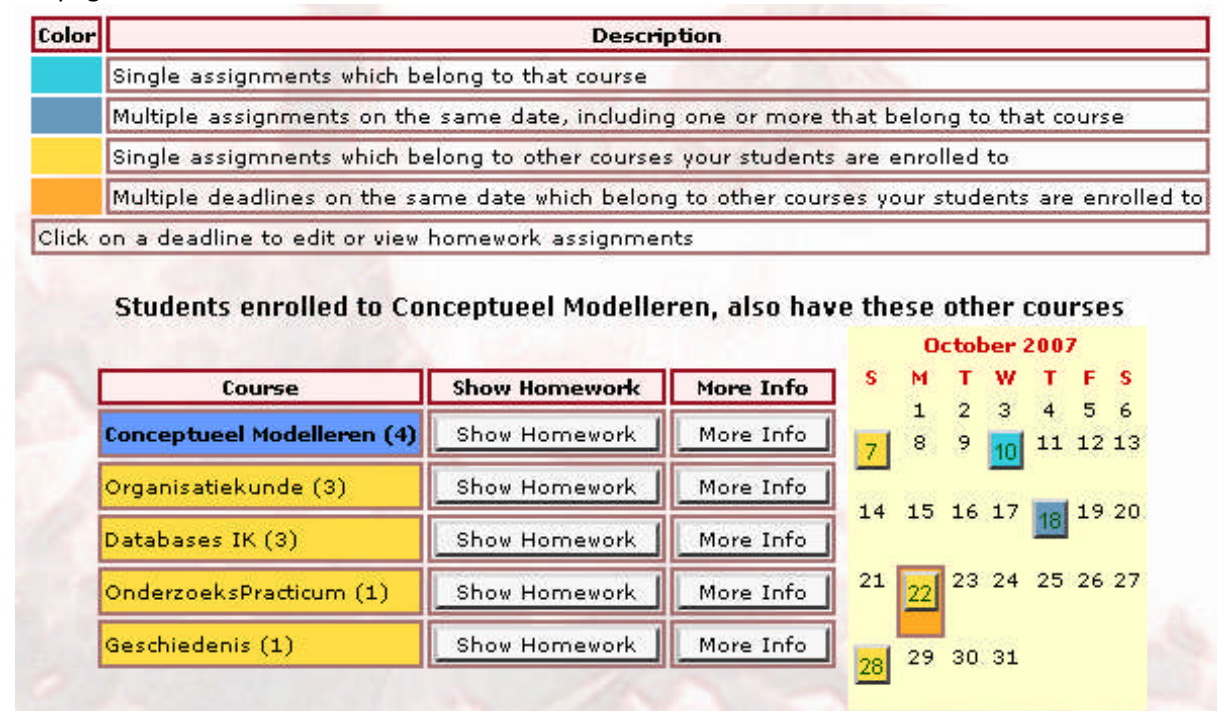


Figure 4.9 Homework overview page

In this section the graphical user interface was presented. However, there was only a possibility to show a few pages. It is assumed that most of the information speaks for itself; therefore the GUI was not described in more detail (especially the GUI design of the main pages). For a better understanding of the GUI, please take a look at the real system.

4.5 - Security

In this paragraph the security measures are discussed.

Before a user can do anything, he has to log in. The system checks the username and password and sends the user to the administrator page, the teacher page, the student page or back to the log in page in case of a mismatch between username and password. Therefore, only users that are registered by an administrator are able to use the system and manipulate (some) data in the database. A user can try to type the URL of a file that the system uses into the address bar of the browser directly to surpass the log in system. A user that will do that will receive an error message and will be sent back to the log in page. A PHP session is used to do this. A Boolean variable is kept through the entire visit of the user, if this variable has the value 'false' than a user will be sent back to the log in page on any action. The variable is set to 'true' if the user enters the right log in information. The variable cannot be set to 'true' manually.

The passwords used in the system are encrypted using md5-encryption. Because of this the passwords are not stored in their original form in the database but in an encrypted, unrecognizable form. This provides privacy for the students (because the administrator cannot see their passwords) and security in case of a corruption of the database.

Users can also log out. When a user logs out the PHP session is destroyed. This means that all information about the current session will be deleted and will be set back to default. (Default for the "access" value mentioned above is false!) If an user works on a public computer and logs out but leaves the browser open, nobody will be able to re-enter the system using the "back" button of the browser. The system will provide an error message and send them back to the log in page.

Evaluation

5.1 – Evaluation of Prometheus

Prometheus was designed and implemented in about six weeks (and not fulltime!). Therefore, it is not surprising that a lot can be improved in order for it to be used in a real life situation. In this subsection only a few improvements will be mentioned that are necessary if the University of Amsterdam wants to use the system.

- More hierarchy in the information structure and information display. Prometheus displays all the courses in one list, not according to study, faculty or something like that. The same goes for teachers and students.
- There was a design flaw in the ER-diagram (not combining 'homework' and 'assignment' into one table) and therefore in the database. This can be improved. If the University of Amsterdam wants to use the system, the current design will not be optimal. In the current version of the system the design flaw doesn't cause any inconvenience, in a system with thousands of entries the additional disk space and processing power needed because of this design flaw can be expected to be significant.
- Prometheus is designed for Firefox. It also works in Internet Explorer, but not as good as in Firefox. Some things can be improved here. This problem is probably caused by the different handling of CSS by the different browsers.

The improvements mentioned above do not mean that there is anything wrong with Prometheus. The functions it allows do work. There were no big problems with the final version of the system. Of course there are things that can be added or improved if there was more time for the project. (E.g. expansion of the homework overview function of teachers using the RequiredTime attribute.)

There was no time for usability tests.

5.2 - Extensions

A logical extension of Prometheus would be an interconnection possibility with Blackboard. This way the information already available in Blackboard can be used by Prometheus (enrollment, course information etc.). Prometheus could then also be integrated in Blackboard entirely so that students only have to log in once and have everything available in one browser. Prometheus could then just be a tab in the Blackboard system.

5.3 – Evaluation of the project

The biggest problem in the project was time. This project used to be fulltime, but now both group members had to attend three other courses while doing this project. Considering the amount of work of the project (including learning PHP, MySQL etc.) two people were too few. We would like to have had more time to improve the system but especially to improve the report and the (preparation for the) presentation.

A more specific problem we encountered during the project was that sometimes the two teachers contradicted each other. In some cases we could just inform both teachers about each other's opinion, but not in all cases. For example with the functional specification we didn't have time to consult both teachers.

References

1. Ministerie van Onderwijs, Cultuur en Wetenschap (OCW). *Dossier Tweede Fase: doel tweede fase*. 2006. Available at: <http://www.minocw.nl/tweedefase/497/Doel-tweede-fase.html>
2. Tweede Fase Adviespunt. *Zeven jaar Tweede Fase, een balans*. 2005. Available at: <http://www.tweedefase-loket.nl/doc/evaluatie/balans.pdf>

Appendix

Appendix 1 - Database Table Schema's

Table structure for table Assignment

Field	Type
HomeworkNr	int(4)
TeacherNr	int(7)
CourseNr	int(4)

Table structure for table Planning

Field	Type
WorkPackageNr	int(4)
StudentNr	int(7)
HomeworkNr	int(4)

Table structure for table Course

Field	Type
CourseNr	int(4)
CourseName	varchar(25)
StartDate	date
EndDate	date

Table structure for table Student

Field	Type
StudentNr	int(7)
LastName	varchar(25)
FirstName	varchar(20)
Password	varchar(32)

Table structure for table Enrollment

Field	Type
StudentNr	int(7)
CourseNr	int(7)

Table structure for table Teacher

Field	Type
TeacherNr	int(7)
LastName	varchar(25)
FirstName	varchar(20)
Password	varchar(32)
Admin	tinyint(1)

Table structure for table Homework

Field	Type
HomeworkNr	int(4)
Description	varchar(25)
Deadline	date
RequiredTime	time
LongDescription	longtext

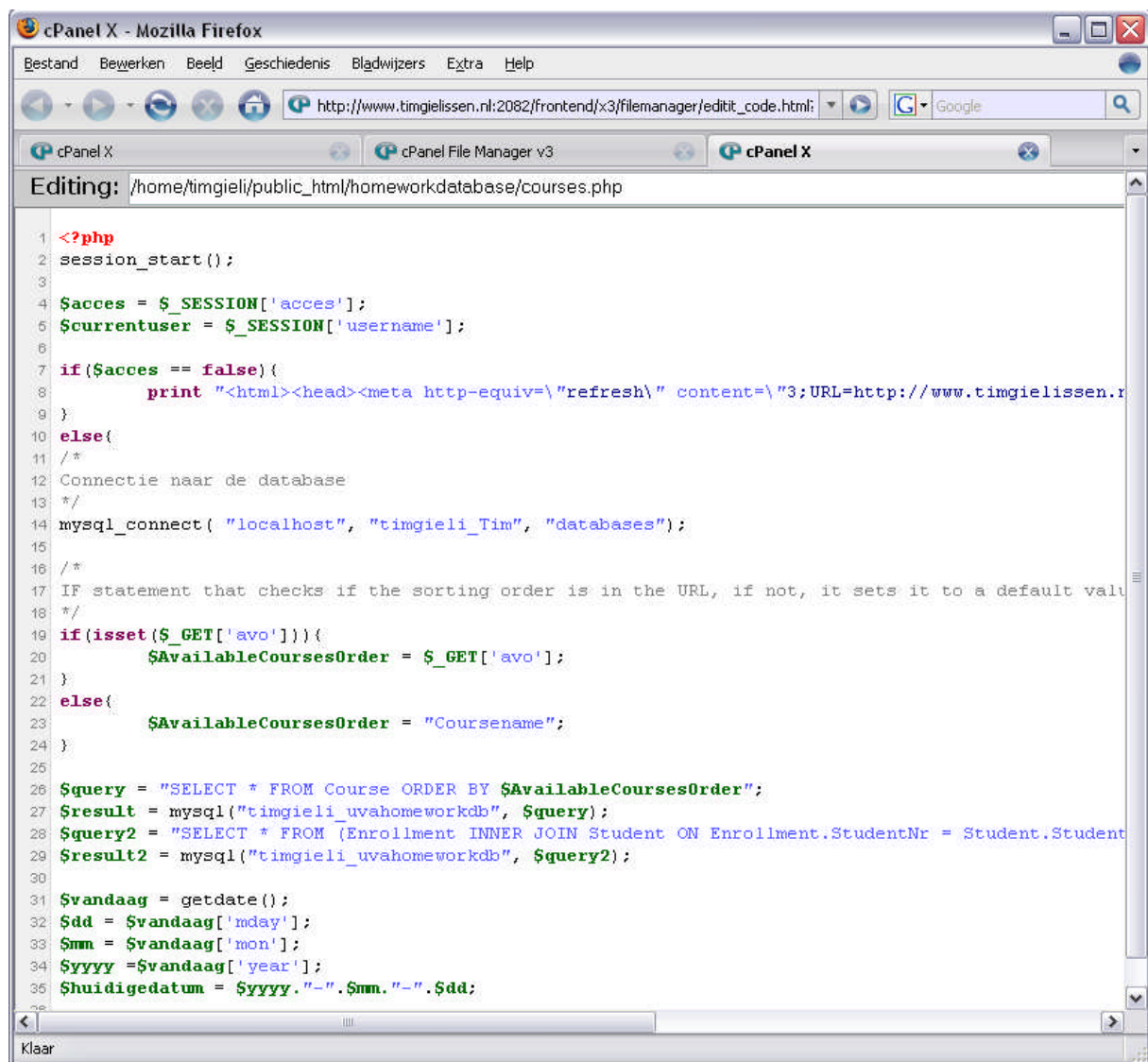
Table structure for table Teaches

Field	Type
TeacherNr	int(4)
CourseNr	int(4)

Table structure for table WorkPackage

Field	Type
WorkPackageNr	int(4)
Deadline	date
Completed	tinyint(1)
Planning	text

Appendix 2 – The editor used to write the code



```
1 <?php
2 session_start();
3
4 $accses = $_SESSION['accses'];
5 $currentuser = $_SESSION['username'];
6
7 if($accses == false){
8     print "<html><head><meta http-equiv=\"refresh\" content=\"3;URL=http://www.timgielissen.r
9 }
10 else{
11     /*
12     Connectie naar de database
13     */
14     mysql_connect( "localhost", "timgieli_Tim", "databases");
15
16     /*
17     IF statement that checks if the sorting order is in the URL, if not, it sets it to a default value
18     */
19     if(isset($_GET['avo'])){
20         $AvailableCoursesOrder = $_GET['avo'];
21     }
22     else{
23         $AvailableCoursesOrder = "Coursename";
24     }
25
26     $query = "SELECT * FROM Course ORDER BY $AvailableCoursesOrder";
27     $result = mysql("timgieli_uvahomeworkdb", $query);
28     $query2 = "SELECT * FROM (Enrollment INNER JOIN Student ON Enrollment.StudentNr = Student.Student
29     $result2 = mysql("timgieli_uvahomeworkdb", $query2);
30
31     $vandaag = getdate();
32     $dd = $vandaag['mday'];
33     $mm = $vandaag['mon'];
34     $yyyy = $vandaag['year'];
35     $huidigedatum = $yyyy."-".$mm."-".$dd;
```

In the screenshot you can see the editor that was used during the project. It only colors pieces of code it recognizes, it doesn't suggest or auto-complete code. It is part of the cPanelX software installed on the webserver.