TAM 2.0

BEP Report

D. Böhm
G.K.G. de Gans
D.T. Franken
M.B. Musters
B.T. van Kooten

Technische Universiteit Delft

TUDelft

TAM 2.0
BEP Report

by

D. Böhm
G.K.G. de Gans
D.T. Franken
M.B. Musters
B.T. van Kooten

in partial fulfillment of the requirements for the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Thursday, July 2, 2020 at 11:00 AM.

Project duration:	April 20, 2020 – July 2, 2020	
Client:	Stefan Hugtenburg, MSc	Distributed Systems Group
Coach:	Dr. Lydia Chen	Associate Professor
Bachelor Project Coordinators:	Thomas Overklift, MSc	Teaching Team
	Ir. Otto Visser	Distributed Systems Group
	Dr. Huijuan Wang	Department of Intelligent Systems

TU Delft

Preface

For our Bachelor End Project, we worked on improving the Teaching Assistant Management (TAM) platform. Our main aim in improving this platform was to alleviate the tasks that have to be performed by the TA coordinator, who previously had to perform heaps of manual labor. In the past 10 weeks we have extended the original TAM with many features which either automate work, or make it such that the work can be done more easily.

Special thanks go to our coach Lydia Chen. Her tips and tricks have ensured that we focused on the appropriate things, and could create the best product possible.

We would also like to thank Stefan Hugtenburg, our client. During our weekly meetings we received invaluable feedback from him. Mostly as a client, but also as a second coach.

During the project, we worked together with the LabraCORE team to integrate the LabraCORE product. We like to thank them for extending LabraCORE with the features that were required for us to make TAM function. We also want to thank the team who worked on TAM previously, as they have given us a great baseline to start our project with.

D. Böhm
G.K.G. de Gans
D.T. Franken
M.B. Musters
B.T. van Kooten
Delft, June 2020

Summary

Most courses in the Computer Science Bachelor at the Delft University of Technology make use of lab sessions. During these lab sessions students can ask questions about course material and get feedback on their assignment. Moreover, their knowledge about assignments can be orally tested. In order to properly help the students, teaching assistants, or TAs, are selected to assist the lecturer during the lab sessions. With the number of students in the Bachelor quickly growing, the process of manually recruiting students to become a TA and assigning the TAs to lab sessions is becoming very time consuming and almost impossible.

During a Bachelor End Project in 2018 four students (van Deursen et al., 2018) created the Teaching Assistant Management (TAM) platform. This project aimed to ease the process of recruiting and scheduling TAs. All parties involved in the process of appointing TAs can use TAM to provide their input. Lecturers can register their courses on TAM and students are able to indicate their interest and availability to help with different courses.
However, the first version of TAM missed a number of important features. For example, student availability data had to be extracted manually and teachers still had to email their TA selection to the coordinator.

This project aims to continue and improve TAM with these missing features. In order to achieve this goal TAM 2.0 has been developed.

TAM 2.0 consists of three components: a MySQL database, a back end written using Spring and Java containing the business logic, and a front end website created using Vue to provide an interface to its users. TAM 2.0 also integrated LabraCORE. LabraCORE provides user and course information to several platforms and stores it conveniently in one central place.

During the development of TAM 2.0, a few problems have been encountered. The biggest problem was that more time needed to be spent on integrating with LabraCORE than anticipated. All in all, only two minor features were not implemented. One of them turned out to be redundant. All important features that will improve the workflow of TA management have successfully been implemented.

Contents

1 Introduction

In order to select Teaching Assistants (TAs) for courses, the Bachelor and Masters of Computer Science and Engineering currently use the Teaching Assistant Management platform (TAM). This platform lets students indicate whether or not they are interested in TAing a specific course. This ensures all student preferences are kept in a single place.

Since the TAs are an invaluable asset to the quality of courses, a lot of time is spend finding the correct students for the job. However, this currently takes up an enormous amount of manual labor for the TA coordinator. Most of this work comes in the form of looking through multiple spreadsheets, keeping track of everything from student preferences to whether or not a student has completed their TA training. After evaluating the current workflow with the TA coordinator, the most important bottlenecks were found in the current workflow. The aim of this project was to extend the current version of TAM with features that would either automate or alleviate the work required for these bottlenecks.

This report gives an overview of all the additions that were made to TAM during our Bachelor End Project. Chapter 2 provides an analysis of the current system and workflow. In this analysis the main bottlenecks in the system are identified. In chapter 3 the conceptual changes that are made to the existing system are discussed. Chapter 4 discusses our team process during the Bachelor End Project. The final result of the product is discussed in chapter 5. This includes all the information about the system architecture, the newly implemented features, as well as feedback received from user testing and the Software Improvement Group (SIG). Chapter 6 discusses the ethical implications that come with using our project in a live environment. In chapter 7 a conclusion is provided on the entire project. Finally, in chapter 8, we touch upon the possibilities of extending TAM, as well as reflecting upon the past 10 weeks working on this project.

2 Research

The original description of this project (see appendix B) provided us with an idea of what the project was about and what was expected of us. However, more information was required before we could start working on it, so we had to research the problem, the current approach taken to address the problem, and the wishes of our client. This chapter outlines the results of our research phase by providing a definition of the problem, a thorough analysis of the problem, and the requirements we have derived from it.

2.1. Problem definition

Teaching assistants (TAs) are recruited with the use of TAM, a teaching assistant management tool. The TA coordinator enters courses that need teaching assistants into TAM. Students then use TAM to indicate that they want to become a teaching assistant for a specific course. Once the deadline for expressing interest has passed, the TA coordinator provides teachers with lists of students interested in TAing for their courses. Subsequently, teachers make a selection from the TA candidates, and make their selection known to the TA coordinator. The TA coordinator then gets the final say on which students are recruited as teaching assistants, and handles contracts, TA training, and more.

Now the problem is that the majority of the tasks above take place outside of TAM. Of the tasks mentioned above, only adding courses and indicating TA preferences are performed using TAM. The other tasks all have to be done manually with the use of database queries, spreadsheets and emails. As a result, the TA coordinator's tasks are laborious and time consuming.

A second problem becomes evident from this as well, namely that not all actors involved in the recruitment process currently use TAM. If teachers could directly use the TA platform, this would eliminate some of the work carried out by the TA coordinator, and give teachers more control over the process.

Lastly, TAM is currently being used in isolation from other tools employed by Delft University of Technology's Computer Science and Engineering program, such as CPM and Queue. As a result of not sharing any data, information such as user details and course information need to be replicated. This can already be seen in the outline of the recruitment process above, where the TA coordinator has to enter course information for every course that requires teaching assistants. This problem also goes the other way, since teaching assistants need to be manually added to the other tools as well.

2.2. Problem analysis

In this section, we analyze the entire process in its current state (i.e. using TAM 1.0). For reference, we have documented the functionality and high-level architecture of TAM 1.0 in appendix C. There are three explicit types of users in TAM 1.0, namely students, course managers, and verifiers. Users with a verifier role were intended to verify changes in TA training or English test status of students, but this ended up being a task carried out by the TA coordinator.

Selection procedure

Two months prior to the start of the semester the TA coordinator adds all courses of that semester which need TAs to TAM. Subsequently, the coordinator will send a message to all students to inform them that they can sign up for these courses by registering their course preferences in TAM. A deadline is set, before which students can sign up for courses and indicate their interest by choosing from three options: interested, not interested, or interested and TAd this course before (see figure C.3).

When the student deadline has passed, a list of interested students is sent to the teachers of each of the courses which includes the name, student number, email address, and indicated interest of each student. The teachers can then select students from this list as TAs. Another deadline is set for the teachers to make their selection. If a student has previously shown to be unsuited for the job, this will be indicated in the list by the TA coordinator. This, however, is a rare occurrence. Before the teachers' deadline, teachers send the list of selected TAs back to the coordinator.

Contracts

From all the emails the coordinator receives from the teachers, he/she creates contracts for all of the TAs. The contracts differ based on the pay scale (which is based on acquired ECTS) and the number of hours a TA is available. All of this information is collected in one spreadsheet with the following columns: name, course, job code, name online approver, email, registered by, max hours, pay scale, start date, end date, batch and remarks. The spreadsheet with this information will then be sent to FlexDelft.

Training and socials

Every student must complete a TA training in order to become a TA. Currently all students that already followed such a training are in a single spreadsheet managed by the coordinator. There are two types of TA training: a 'general' TA training as well as a project-specific training. Every new TA that has not yet followed the required training should be notified by the coordinator and asked to submit their availability using a survey. This list is then sent to the trainers allowing them to make a training schedule. After fulfilling the training, students are added to the spreadsheet that is used to keep track of all students who followed the training.

TAs also need to sign a code of conduct in which they declare that they will handle the extra information they might gather by being a TA honorably. The students that have signed this code of conduct are currently registered in a spreadsheet that is managed by the coordinator. The coordinator needs to manually look for and inform students that still need to sign the code of conduct.

When a student is a TA for the first time, they will receive a TA polo. Students have to enter their desired polo gender and size into TAM on their profile page. The coordinator then gathers this information in a spreadsheet and makes sure there are enough TA shirts in the required sizes and genders.

At the end of each quarter there is a TA drinks event. The coordinator manually gathers the names of all students who TAd during the quarter and sends an invite to all of them. At this social event, badges are handed out by the coordinator to students for each of the courses they TAd for. The coordinator manually keeps track of which students have received their badges using a spreadsheet. This is not done very strictly, and badges are therefore handed out according to a trust-based system.

Scheduling

The scheduling is currently performed by the coordinator. Students can fill out their availability in TAM, which the coordinator will then export as a SQL dump. This SQL dump displays each individual day and time slot that a student is available. Based on this information, the coordinator manually assigns all TAs to specific labs, and shares the schedule with the students.

Special cases

There are a couple of special cases in which the procedure differs slightly from the described procedure. These will be addressed here.

The first case are shared labs, which are labs that are shared by multiple courses. For these labs it is preferable to have TAs that can assist with more than one course. In selecting TAs for those courses with shared labs, the coordinator informs the teachers about which students can help with multiple courses. The teachers can then choose specific students as their TAs to make sure a larger number of TAs can help with as many courses as possible during these labs. Currently the lists of interested TAs of these courses are manually compared.

Second of all, if students work more hours than specified in their contracts, their contracts must be updated to allow them to request payment for the extra hours. In this scenario, students need to inform the coordinator. Subsequently, the coordinator will inform FlexDelft to make sure the additional hours are approved.

Finally, when a teaching assistant does not assist during a lab session, but, for instance, helps checking exams instead, the student needs to be paid from a different financial envelope. The student needs to email their name, pay scale and hours to the coordinator, who will subsequently approve their request and make sure that he TA gets paid correctly.

Process Diagram

Three figures can be found below which illustrate the TA process as seen from three different viewpoints: that of a student, of a teacher, and that of the TA coordinator. Figure 2.1 describes the standard process students follow, starting from the point of indicating interest in becoming a TA, to the end of the quarter. Figure 2.2 describes the standard process teachers follow to recruit TAs for their course. Figure 2.3 describes the standard process the TA coordinator follows to manage TAs for all courses. This diagram only shows the main process and not the special cases, since those are not always applicable or necessary.

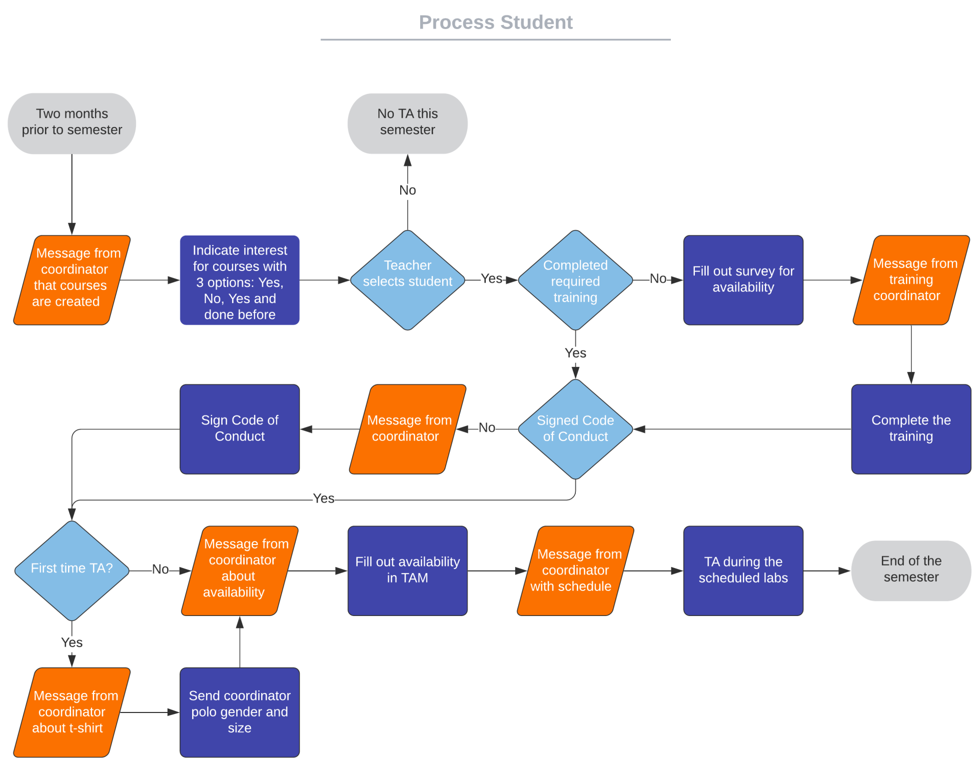

Process Student
Two months prior to semester
Message from coordinator that courses are created
Indicate interest for courses with 3 options: Yes, No, Yes and done before
Teacher selects student
No
No TA this semester
Yes
Completed required training
No
Fill out survey for availability
Message from training coordinator
Complete the training
Yes
Signed Code of Conduct
No
Message from coordinator
Sign Code of Conduct
Yes
First time TA?
No
Message from coordinator about availability
Fill out availability in TAM
Message from coordinator with schedule
TA during the scheduled labs
End of the semester
Yes
Message from coordinator about t-shirt
Send coordinator polo gender and size

Figure 2.1: The process of a student becoming a teaching assistant.

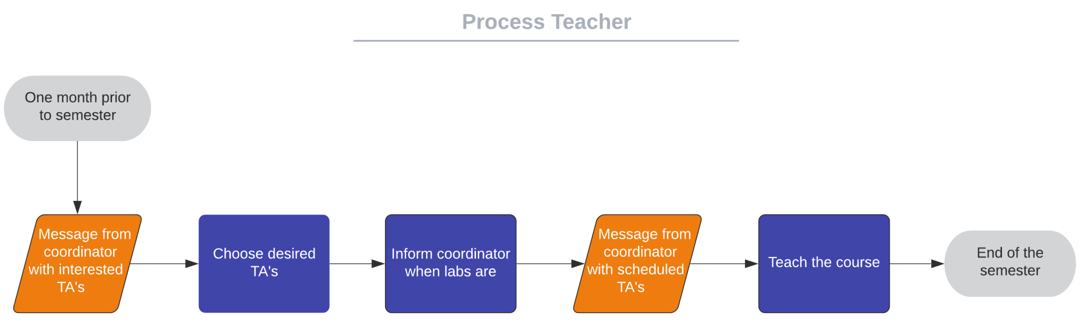

Process Teacher
One month prior to semester
Message from coordinator with interested TA's
Choose desired TA's
Inform coordinator when labs are
Message from coordinator with scheduled TA's
Teach the course
End of the semester

Figure 2.2: The process of a teacher selecting teaching assistants for their course.

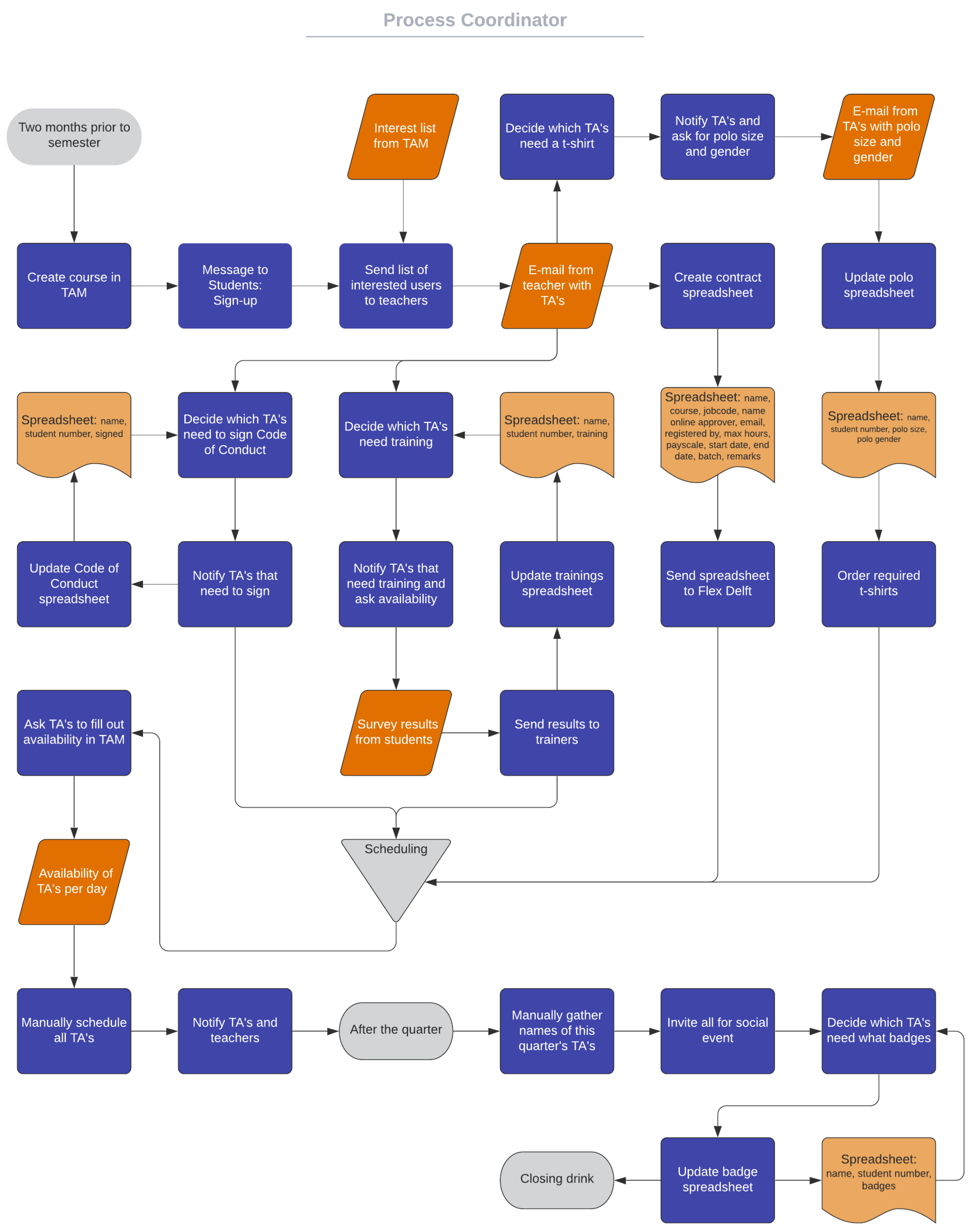

Process Coordinator
Two months prior to semester
Interest list from TAM
Decide which TA's need a t-shirt
Notify TA's and ask for polo size and gender
E-mail from TA's with polo size and gender
Create course in TAM
Message to Students: Sign-up
Send list of interested users to teachers
E-mail from teacher with TA's
Create contract spreadsheet
Update polo spreadsheet
Spreadsheet: name, student number, signed
Decide which TA's need to sign Code of Conduct
Decide which TA's need training
Spreadsheet: name, student number, training
Spreadsheet: name, course, jobcode, name online approver, email, registered by, max hours, payscale, start date, end date, batch, remarks
Spreadsheet: name, student number, polo size, polo gender
Update Code of Conduct spreadsheet
Notify TA's that need to sign
Notify TA's that need training and ask availability
Update trainings spreadsheet
Send spreadsheet to Flex Delft
Order required t-shirts
Ask TA's to fill out availability in TAM
Survey results from students
Send results to trainers
Availability of TA's per day
Scheduling
Manually schedule all TA's
Notify TA's and teachers
After the quarter
Manually gather names of this quarter's TA's
Invite all for social event
Decide which TA's need what badges
Closing drink
Update badge spreadsheet
Spreadsheet: name, student number, badges

Figure 2.3: The management process of the TA coordinator.

Identified issues

Now that we have seen the entire process of TA recruitment and management, we consider various problematic aspects of the process as areas upon which TAM could improve. Finally we discuss whether it would be preferable to use TAM 1.0 as a starting point, or to start over from scratch.

Preferences manually extracted
By manually executing SQL queries and running a Python script, the TA coordinator extracts the course preferences (i.e. which student would like to be a TA for which course) from the TAM platform. This approach is unintuitive and requires a lot of manual labor.

Preferences manually distributed
Then, the teachers of each course have to manually be emailed a list of students interested in being a TA for that course. This should not necessarily be automated, but it could be. Alternatively, the lists of interested TAs are not emailed to the teachers, but can instead be made available to them on TAM.

No UI lab timetable support
The back end has support for timetables (with TA availabilities) for course labs, but the user interface has no support for it. The problem is hence that this feature cannot actually be used in practice.

Third preference option unnecessary
For every course that TAs can apply for, there are three options: “Yes” (I am interested in being a TA for this course), “No”, and “Yes, and I’ve been TA for this course before”. The last option may be redundant if the system could detect this from previous data.

Contract information
Once the teachers have made a selection of TAs from the list of interested students, they are emailed back to the coordinator so that the necessary work contracts can be made for the TAs. For every contract, the pay scale and working hours must be determined. Additionally, it must be determined whether selected students have yet to follow a TA training. Currently, this information is gathered manually in a spreadsheet, but this could be automated if teachers select TAs using the TAM platform.

Contracts
To create contracts for FlexDelft, the following information must be determined for each contract: name, course description, job code, name of online approver, email, signed in by, max hours, pay scale, start date, end date, flex batch, and remarks. Now, this information is manually gathered in a spreadsheet, but ideally this would be automatically available in TAM.
The contracts are emailed manually to FlexDelft. For now this is preferred since money is involved and contracts can be manually checked for errors, but in the future it may be possible to automate this as well.

TA drink invites
At the end of each quarter, a TA drink is organized for TAs that were a TA during that quarter. To invite the right TAs, a record must be kept of which TAs worked during that quarter. Now, this record is kept manually in a spreadsheet, but preferably TAM could invite the right TAs automatically.

TA training
To organize a TA training, it must be known how many students need training and when they are available. In the current system, a Google Forms survey is used to obtain the availability of each student. It is possible to track which students have passed their training in TAM, but in practice this requires too much time and a spreadsheet is used instead.

Badges
During the TA drinks, TAs are given badges for the courses they TAd. However, TAs should not be able to receive a badge more than once, which is why a record is kept of who received what badges. In addition, a record must be kept of the badge inventory. Now, this is done manually in a spreadsheet, but being able to do this in TAM is preferred.

Help with examining or extra labs
If a TA helped with examining or helped during an extra lab session, the TA has to write an email concerning their extra work. Now, this information has to be collected from all these emails so that

the extra work can be paid and administered all at once. Ideally this process could be improved by including it as part of TAM, instead of doing everything by email.

Detecting TAs for shared labs

A number of courses have shared labs. An attempt is made to schedule students for these labs who are a TA for more than one of the courses sharing the lab. By doing this, less TAs are needed overall. Lecturers currently need to be asked to compare their TA lists to find such TAs. An improvement over the current version of TAM would be if it could detect such TAs automatically.

Code of conduct

Every TA has to sign a code of conduct. TAM should be able to store a record that keeps track of which students have signed the code of conduct in TAM.

Labrador/LabraCORE integration

For now, it is not possible to use information from other TU Delft platforms. It would be very helpful if, for example, TAM could instantly make TAs available in Queue, and feedback from Queue could be sent to TAM. Labrador/LabraCORE integration could make this possible. The Labrador project is a recent undertaking which allows participating platforms to share data which is stored in a centralized database accessible by an API. To ease communication between systems, a definitive model has been (currently: is being) designed of the university and all parts that belong to it, e.g. courses, assignments, and users.

Improving TAM

The final issue we had to consider was whether it would be more beneficial to take TAM 1.0 as a starting point or to start from scratch. One point of consideration was that the existing platform already contained a lot of functionality that would have had to be implemented again if an entirely new version of TAM were to be created, but on the other hand it was also known that TAM 1.0 would require a lot of rewriting and redesigning to accommodate the integration with LabraCORE. By doing some more research into the tools used by TAM 1.0 we learned that the system is quite modular and was something that could be worked on in parallel. In addition to this, we believed it would be too much of a waste to start entirely anew, since we anticipated that we would end up having less time to implement new features. Hence, we decided to use TAM 1.0 as our starting point.

2.3. Proposed solution

After having questioned our client and having decided to continue from TAM 1.0, we have derived the following initial set of requirements, which we have discussed and refined with our client. The requirements are assigned to one of four priority classes using the MoSCoW method: must have, should have, could have, and won't have. The priorities of the requirements are based on the wishes of our client.

Functional requirements

Must Have:

- RM01 Teachers can view a list of interested TAs for each of their courses.
- RM02 Teachers can select from those lists the TAs that they want for each course.
- RM03 TA coordinators can export a spreadsheet with all the TAs that have been selected, the corresponding courses, their pay scales, amount of hours, etc. to be sent to FlexDelft.
- RM04 TA coordinators can get an overview of TAs that have not yet finished their TA training.
- RM05 TA coordinators can import a list of TAs that finished their TA training.
- RM06 TA coordinators can get an overview of all TAs that have not yet signed the Code of Conduct.

- RM07 TA coordinators can import a list of TAs that newly signed the Code of Conduct.
- RM08 TAs can send a request to teachers for payment for overtime or extra work.
- RM09 Teachers can view and accept or reject requests from RM08 and forward accepted requests to TA coordinators to view.

Should have:

- RS10 Automatically send notifications to teachers when they can start selecting TAs.
- RS11 Teachers and TA coordinators should be able to get an overview of the availability of TAs.
- RS12 TA coordinators can view which students TAd during the last quarter and for what courses, in order to arrange TA drinks and badges.
- RS13 TA coordinators can see which TA has received what badge.
- RS14 TA coordinators can see a list of new TAs for TA shirt management.
- RS15 TA coordinators can attach notes to a TA that teachers can also see.
- RS16 TAs are able to change their ECTS total, which determines their pay scale. TA coordinators should be notified to accept or reject a change in pay scale for a TA.

Could have:

- RC17 Import TA feedback from Queue into TAM (LabraCORE).
- RC18 Export TAs from TAM into Queue (LabraCORE).
- RC19 Detecting overlap in TA course lists for shared labs.

Won't have:

- RW20 Scheduling the TA training for the TAs.

Non-functional requirements

- The expected number of users is 1000 per year.
- Peak usage is expected for 2 weeks a year (up to 500 users), but increased latency is acceptable.
- The project should have clear documentation for installation, usage, and development purposes.
- It should be documented what requirements exist in order for TAM to be updated for use for multiple study programs.
- The back end should use Java version 11.

Requirements update

During the first few weeks of the project we have had multiple meetings with our original client and a new client from the LabraCORE development team. From these meetings we concluded that integration with LabraCORE is much more important than originally thought. Therefore we have decided to add two more must haves. The first must have is the integration of LabraCORE. This change requires us to make changes to our database and rewrite existing parts of the code. To streamline future database changes and migrations we added a second must have: add Liquibase to the project setup.

3 Design

In general our design is still the same as the design of the previous version of TAM, since we built further upon the existing version. This is mainly because we believe that improving the current version of TAM allows us to satisfy the client more, by implementing features that save a lot of time. The design of the back end and front end have therefore not been changed drastically, i.e. we kept using Java Spring for the back end and Vue and Bootstrap for the front end. For a more detailed overview see 5.1. We did need to change the design of the database because of the requirement of having TAM interacting with LabraCORE, which is why this chapter will further elaborate on the design changes in the database.

3.1. Database

Because of the requirement to have TAM connected to the LabraCORE API, a number of database changes have to be made. Parts of the original database system are no longer required, and other parts are moved to LabraCORE. Figure 3.1 gives an overview of the schema we designed prior to the software development phase. Later in the project this schema has changed, the final version can be found in 5.17. In this section we will discuss why we have made certain changes.

3.1.1. LabraCORE

LabraCORE is an initiative to centralize all common data needed for systems that are used by the Computer Science department (and later for the entire faculty and other faculties). The rationale behind this is that a lot of these systems use the same data, e.g. information about students or courses. By centralizing this in one database, the systems are able to share data more easily and they can be maintained more conveniently.

3.1.2. Changed tables

A number of features are not presently being used, and are also not desired to be in TAM in the future. Therefore we have decided to remove the tables associated with these features from our database. Furthermore, a lot of information that is being used by TAM is also relevant to other systems used within the department of Computer Science (e.g. Queue, CPM). To be able to share this information, data previously stored by the TAM database has been moved to LabraCORE and is obtained through its API.

CoursePreference

This table was used to allow verifiers to give every student a rank stating how much they want this student to be a TA for their course. This information was then used to create 'the best' TA assignment using an algorithm. This way of assignment has not been used and is no longer desired, and therefore this table is no longer necessary.

Schedule

The Schedule table was used to store the schedules of course and lab assignments of TAs. Just like with the CoursePreference table, this function was used for automated scheduling of TAs. TAM is currently not used for the scheduling task, so it no longer needs to be included.

Lab / LabSession / LabAssignment
The information about labs and lab assignments is used by multiple systems as well. This caused the tables Lab and LabSession to be moved to LabraCORE. The LabAssignment table was also used for the scheduling feature of TAM, which is no longer desired to have in TAM.

User / Profile / Prerequisites
The most relevant user information (netid, name, email), will be taken from LabraCORE. This information is used by many systems, so it would be redundant to store this separately for every system available. There is however some TAM specific information that needs to be saved, e.g. completion of TA training, polo preferences, et cetera. Some of this information was saved in the Prerequisites table, but this has been merged into one table called Profile.

CourseEdition / CourseAssignment
All systems rely on information about the courses (course_code, teacher, quarter), and TAM is no exception. Since TAM fills in parts of this information, namely who is a TA for what course (CourseAssignment table), it is important that other systems can also access this information. LabraCORE also provides information about who manages a course.

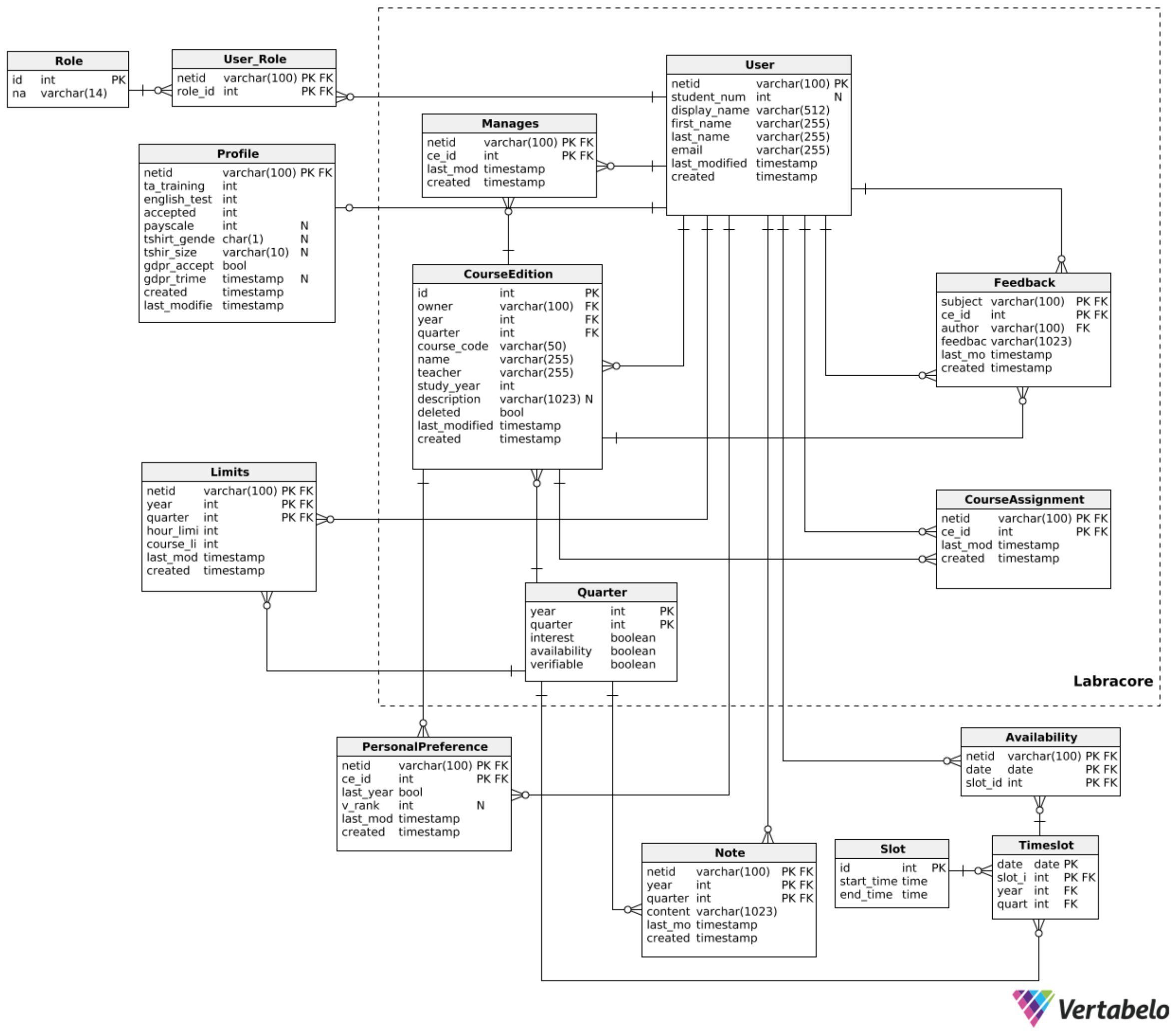


Figure 3.1: The initial database schema design of TAM 2.0

4 Process

This chapter describes the process of the project. First, a summary of how various parts of the product and report were organized and handled by our group is given. After that, an in-depth timeline is provided which details how this report and the end product came to be.

4.1. Internal organization

In the sections below, we will detail which solutions we came up with and why we used certain tools for various technical aspects of the project.

4.1.1. MoSCoW

Almost immediately after the start of the project it was decided to set up and make use of a list of requirements in a MoSCoW format (Must have, Should have, Could have, Won't have). The initial requirements were decided upon after the first meeting with the client, and remained largely the same during the rest of the project, the biggest change being the addition of LabraCORE integration as a must have requirement.

4.1.2. Communication methods

Messaging

After a first round of emails were sent around using TU Delft student mail addresses, our group shared contact details and came together in a WhatsApp group which we have used throughout the project for planning and general messaging. WhatsApp was chosen because almost everyone has it installed on their smartphone nowadays.

Later, a Discord server was set up which was used for more detailed messages about programming issues and advice. This server also included a channel which we shared with three LabraCORE/LabraDoor developers for intercommunication. Discord is easy to set up and use, and permissions for e.g. the shared LabraCORE channel are easily managed.

For questions and other short discussions with the client outside of meetings, A private channel was made in the BEP Mattermost server.

Meetings

For meetings with the client and/or the coach, the Jitsi meet instance hosted by the TU Delft[1] was used. Typically during each meeting, the next meeting was scheduled and calendar invites were sent out to each participant.

For internal meetings, we started out also using Jitsi, but switched over to Discord because we did not value the video features of Jitsi as much during team meetings, and the voice quality and noise filtering of Discord were much better in our opinion.

[1]TU Delft Jitsi meet instance at `https://jitsi.ewi.tudelft.nl/`

4.1.3. Sprints

In the first week, we made a time plan in which we concluded that we would be able to finish the project optimally by making use of two-week sprints. During these sprints, people either worked on one large feature, or two smaller features, with each feature being worked on by one or two people, depending on the size. Each feature mapped to roughly one requirement from our MoSCoW list.

4.1.4. Version control

To save all of our code and keep track of changes, we made use of Git. Specifically, the TU Delft hosted GitLab instance[2]. Not only is Git a useful version control tool but the existing TAM 1.0 code-base is hosted on this same GitLab instance, so it was easy to create a fork and work from there.

For each feature, a separate branch was created from the 'dev' branch and worked on until completion. A merge request would then be created which was to be reviewed by preferably each team member. All merge requests relating to new features had a minimum of 2 approvals needed, while smaller minor fixes required only one approval.

4.1.5. Meetings

Regular meetings

Because meeting in-person was not an option during the ongoing measures against the COVID-19 pandemic, we had at least one online meeting every week, usually more, where we all came together in a conference call.
Each week, we had one team-only meeting. Every two weeks, we had a meeting with our coach, and also about every two weeks we had a meeting with our client. There were also two extra meetings with (part of) the LabraCORE team.

Midterm meetings

At the start of the project, as suggested by our coach, we scheduled two midterm meetings. One at the half-way point of the project, and one at the three-quarters point. These meetings would include our full group, our coach from TU Delft, as well as our client.

During these meetings we would give a short presentation on the progress we have made thus far, how we envisioned the rest of the project, and included a live demo of newly completed features. Afterwards we would receive feedback from both our client and our coach. After both left, the team would stay in the conference call a little longer to reflect on the presentation and discuss what steps to take next.

4.2. Timeline

The Start

Our group came together very last-minute at just two days before the start of Q4. Our group consists of five people who had, up until then, not found a group yet. Unfortunately, the company that proposed the project we chose initially informed us that the project had been canceled, and we had to switch to an internal TU Delft project.

On Wednesday of the first week, we had our first live meeting with the client to discuss what the project exactly comprised, and on Friday that week, we had the first meeting with our coach. Thus, it was not until the start of the second week that we were fully set up and ready to start working on the source code.

The Middle

We were well on track implementing some of the new features that our client requested but not all we needed to do was in our planning at that time. There was one thing the client was not sure about because he did not write the original project description himself. One part of the project we had not paid much attention to, looming in the background: LabraCORE.

[2]TU Delft GitLab instance at `https://gitlab.ewi.tudelft.nl/`

When we initially inquired the Labrador team about their platform, they had given us access to several repositories on the GitLab instance. LabraCORE, LabraDoor, and Librador, all with little to no documentation. Because it still was not clear what we were to do with it, we scheduled a meeting with both our client and the Labrador team during which everything became clear. Unfortunately, this meeting didn't take place until week 3.

LabraCORE integration was going to be a lot more work than planned and a part of our team went to work immediately. The rest of the team continued working on other features.

The End

The subteam working on integrating LabraCORE had to rework all existing features and the features we had made until that point to work with data from LabraCORE. Not only did it take some time to get used to the way that LabraCORE handles data, but quite some needed features did not exist in LabraCORE at all. For each newly requested LabraCORE endpoint it obviously took some time from request to code to review to merge to deploy until we were able to use it.

The last few weeks everyone has mostly been updating our new features to work with the new LabraCORE back end, and we put the report on the back-burner for a while. And so, by the time of writing this timeline, all features have been merged, and we are done coding.

5 Final product

This chapter gives an overview of the final product that is delivered. First, the system architecture is addressed. This section centers around figure 5.1 which gives an overview of how the components of TAM 2.0 and LabraCORE work together. Next, the implemented features are described in detail with screenshots of what they look like to the user. Following that, the final model of the database is explained and compared to the initial design. After that, the results of the user tests that have been conducted are discussed. Next, the feedback from the Software Improvement Group (SIG) is discussed and what part of the feedback has been processed. In the last section of this chapter the choice to not have a User Manual is explained.

5.1. System architecture

This section addresses the system architecture of TAM 2.0. The structure of the front end and back end remain largely unchanged since the first version of TAM. Please refer to chapter 5 of the report by van Deursen et al. for a detailed description [1]. New in TAM 2.0 is support for LabraCORE. In figure 5.1 it is visualized how the front end, back end, and LabraCORE work together.

Back end

Since most of the architecture is inherited from TAM 1.0 we will only discuss the front and back end structure of our project. Our back end consists of a system using Spring Controllers, Services, and Repositories. These are created for all the different types data used in TAM, e.g. Users and Courses. The repositories are used to retrieve and send data to our data sources, which are both our own database and LabraCORE. The services are used to handle the in and outgoing data, as well as checking if the data exists. Finally, the controllers are used to create end points which can be accessed by our front end.

Front end

The front end is built using a JavaScript framework called Vue. Vue allows us to serve the entirety of TAM as a single-page application, which is composed of numerous components. Each page, such as the profile page available to students, is encapsulated in its own component. This structure makes it very easy to develop different components in parallel, without interfering and causing merge conflicts. It also allows us to manage permissions for every page, since access requirements can be set up per component.

LabraCORE

Now that TAM has to be able to share data with other systems, parts of the data it needs should be stored by LabraCORE. To this end, TAM uses a library called LabraDoor. LabraDoor configures and manages authentication, allowing users to sign in using Single-Sign On (SSO). In addition, LabraDoor enables TAM to use LabraCORE's API through an API client, which simplifies the tasks of retrieving and storing data in LabraCORE.

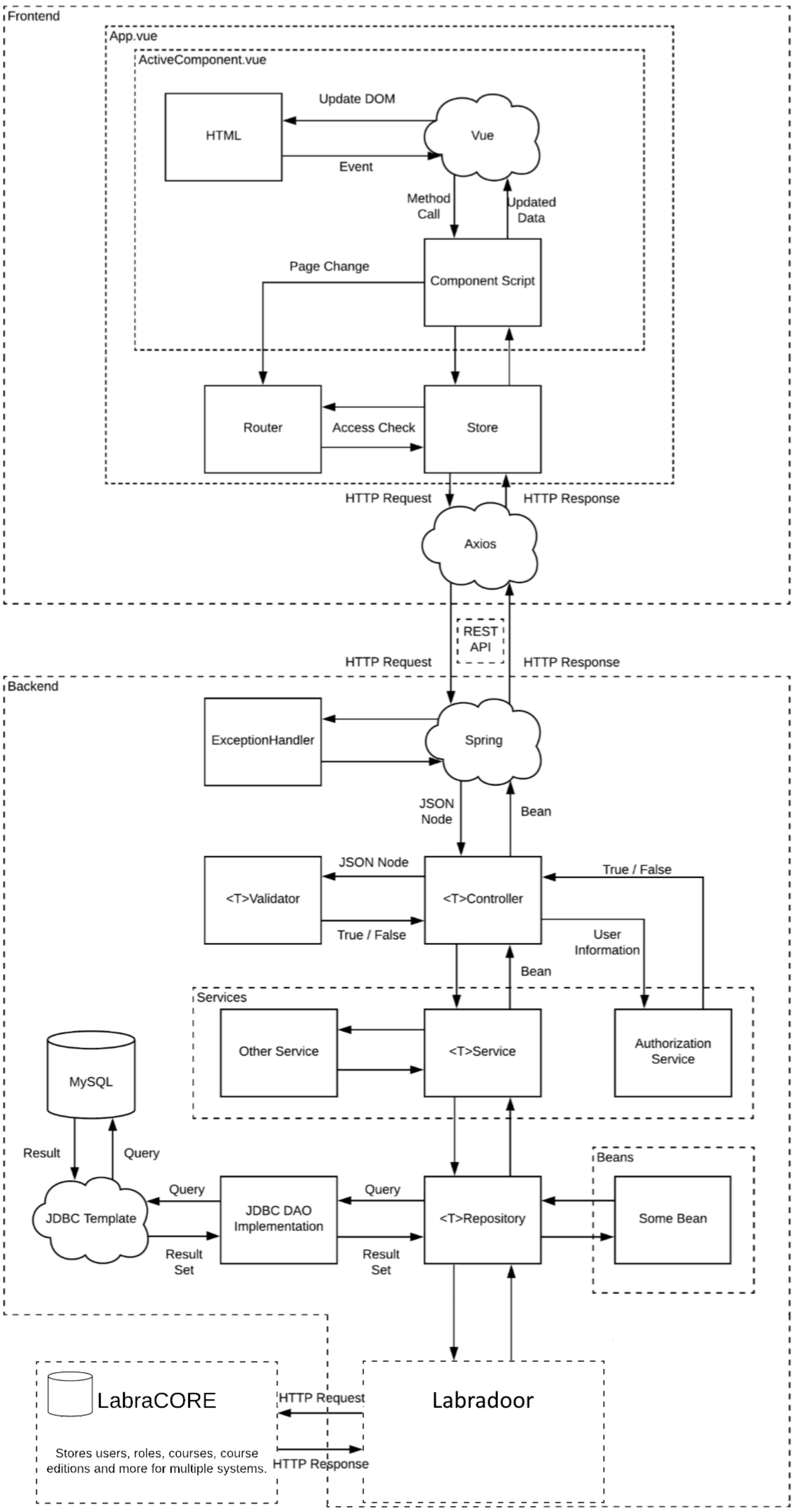


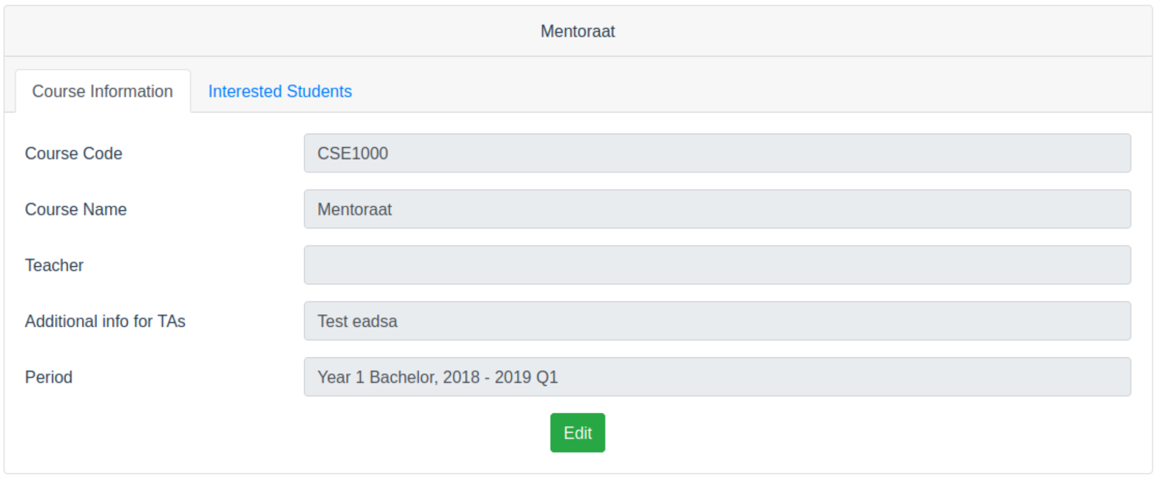
Figure 5.1: Detailed structure of the front- and back end. This figure was taken from the TAM 1.0 report by van Deursen et al. [1] and adapted to resemble the current situation.

5.2. Features

In this section we describe and illustrate the new features that we have added.

Teachers: View and select interested students

The first new feature addresses requirements RM01: “Teachers can view a list of interested TAs for each of their courses” and RM02: “Teachers can select from those lists the TAs that they want for each course.”. The course page has been updated to include a new section which allows teachers to display a list of students who are interested in TAing for their course. Teachers can navigate between the course information page and the interested students overview by clicking on the respective tab. The new tab provides teachers with three methods of selecting TAs. They can opt to pick TAs one by one, select all interested students at once, or they can provide input containing student numbers to select only a subset of all interested students. The updated course page and interested students page can be seen in figure 5.2 and figure 5.3 respectively.



Course Code	CSE1000
Course Name	Mentoraat
Teacher	
Additional info for TAs	Test eadsa
Period	Year 1 Bachelor, 2018 - 2019 Q1

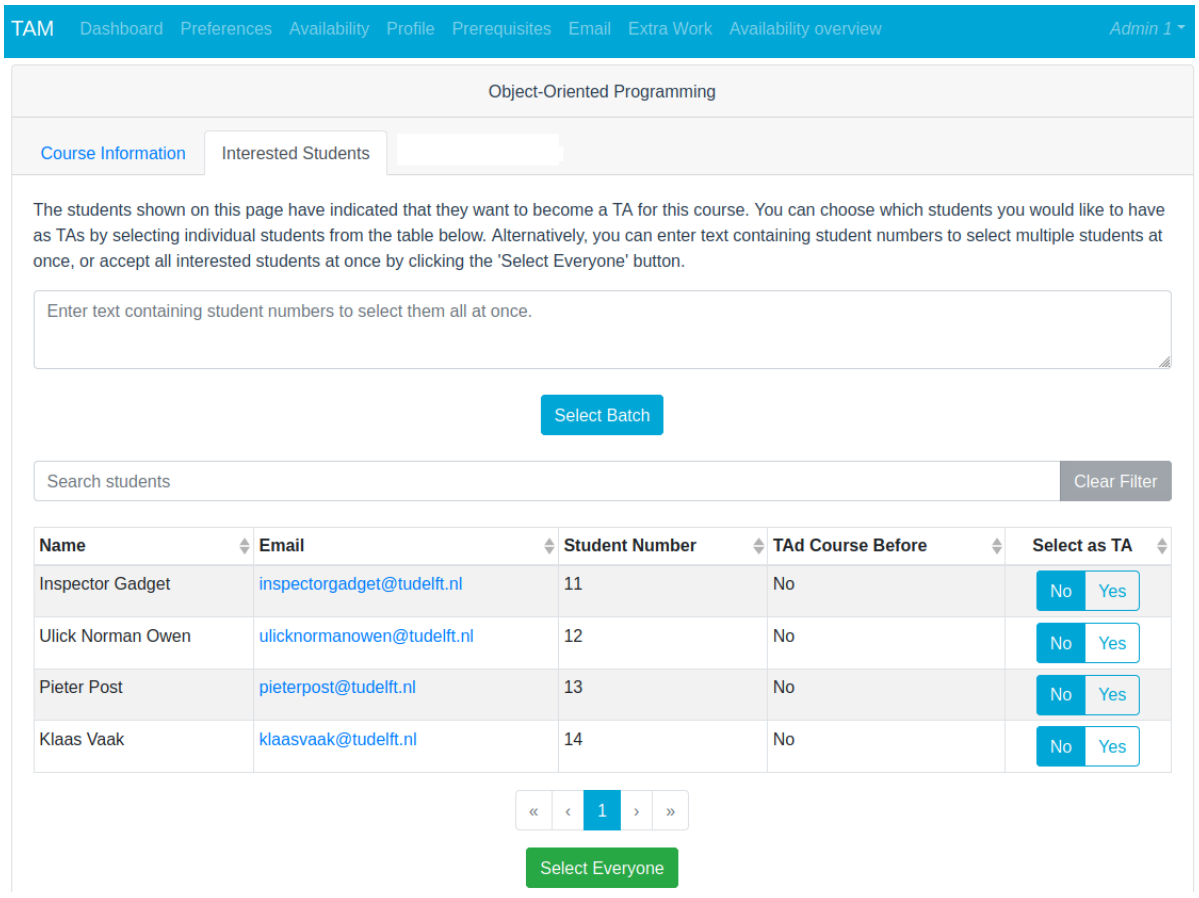
Figure 5.2: The updated course information page. Note that labs have been removed from this page, and that there are now two tabs to choose from.

TA Coordinator: Accept TAs

This feature is an extension to the feature that allows teachers to select TAs. Another tab has been added to the course information page which is only available to the TA coordinator (figure 5.4). Initially, teachers could directly add students to their course as TAs, but this meant that the TA coordinator lost control and insight into the selection process. Hence, students selected by teachers now first must be accepted as TAs by the coordinator. This page has similar input methods to the teacher’s TA selection tab, allowing the coordinator to approve one, multiple, or all TAs at once. Once TAs have been accepted by the coordinator, they will be registered as TAs in LabraCORE’s database, meaning that other platforms can retrieve the newly added TAs.

Prerequisites manager

This feature can be directly linked to requirements RM04 and RM06. Currently the coordinator has a separate spreadsheet of all TAs that successfully finished the required TA training and another spreadsheet in which there is a notion of which TAs signed the Code of Conduct. This information however is needed for other functions within TAM as well. Therefore a dedicated manager to keep track of these prerequisites for TAs is needed. In TAM 1.0 there is an overview for the coordinator of all TAs and some information, e.g. TA training, English test and pay scale (See Appendix C figure C.9). The reason this function was not used for the TA training however is partly because the coordinator did not notice its existence. In addition to that, updating the status is not convenient. The status either had to be updated manually per student, or with an import in which every column had to be filled out. The coordinator found it therefore more convenient to keep track of a separate spreadsheet.



Object-Oriented Programming

The students shown on this page have indicated that they want to become a TA for this course. You can choose which students you would like to have as TAs by selecting individual students from the table below. Alternatively, you can enter text containing student numbers to select multiple students at once, or accept all interested students at once by clicking the 'Select Everyone' button.

Enter text containing student numbers to select them all at once.

Search students

Name	Email	Student Number	TAd Course Before	Select as TA
Inspector Gadget	inspectorgadget@tudelft.nl	11	No	No Yes
Ulick Norman Owen	ulicknormanowen@tudelft.nl	12	No	No Yes
Pieter Post	pieterpost@tudelft.nl	13	No	No Yes
Klaas Vaak	klaasvaak@tudelft.nl	14	No	No Yes

Figure 5.3: The new interested students page, which allows teachers to view and select students as TAs for their courses.

In Tam 2.0 we have extended upon the current overview. The Code of Conduct column is now added, which keeps track on which TAs signed the Code of Conduct. Since the status for the TA training was already implemented, this requirement is also met. In figure 5.5 it can be seen what the overview looks like now. Part of the reason this feature was not really used yet was the inconvenient way of updating it. This is something that will be addressed with the next feature.

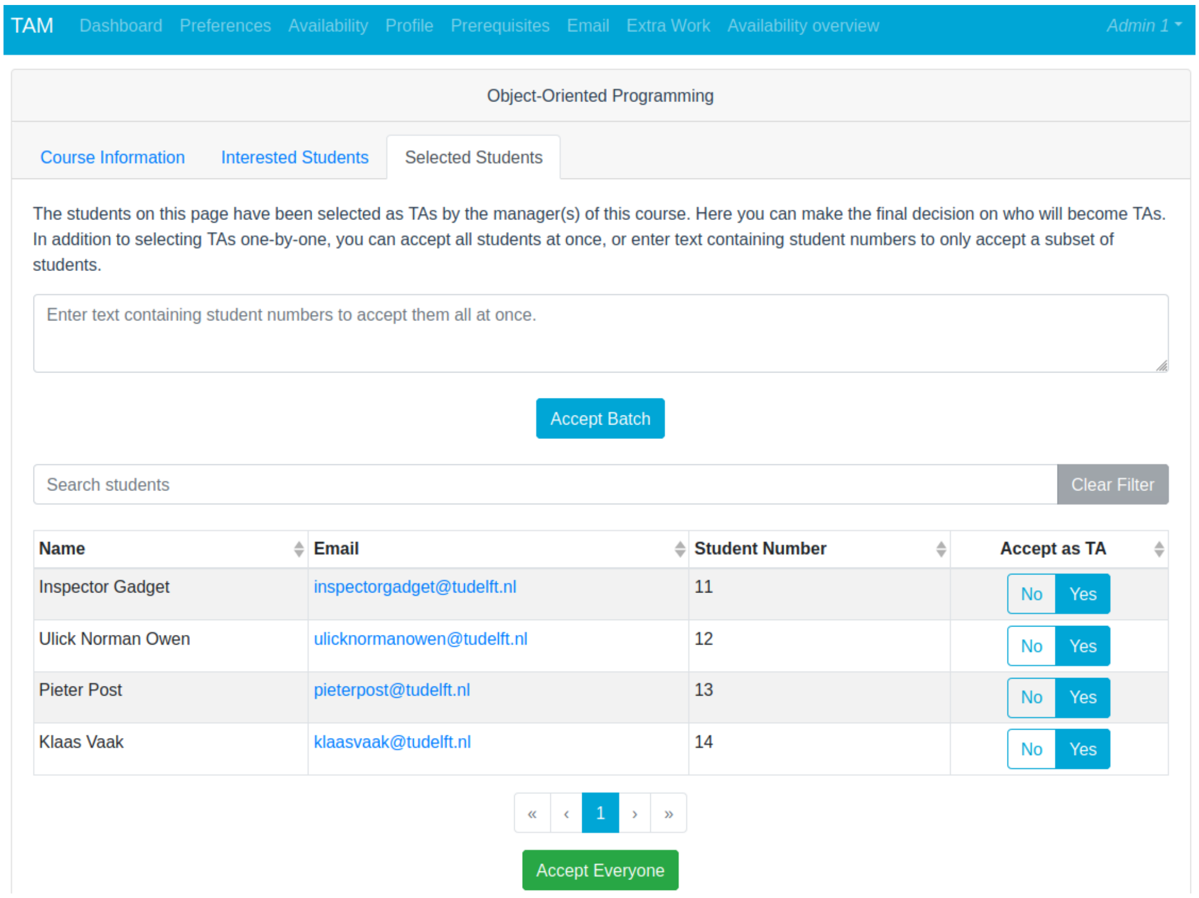
Partial batch import

The following section will describe the partial batch import feature, which can be linked to requirements RM05 and RM07. The feature of importing a batch of students into the prerequisites manager was already present. Uploading a CSV file with the correct header and all information about the TAs in the file, resulted in an addition of the new TAs to the table and an updated version of the TAs that were already present. If, however, a column was not filled out in the CSV file, TAM would also update those empty columns. This resulted in loss of data, since the fields that were not filled out in the file would be reset to a $NULL$ or 0 value.

In the newly implemented feature this is not the case anymore. If a field was left empty in the CSV file, TAM will not update its value and keep the old value. Even when the header is not complete, e.g. there is only a $netid$ and $ta\_training$ column in the uploaded file, TAM will be able to process the data, without manipulating data that should not be manipulated. The $netid$ column will always be necessary, since this is used as an identifier for the users.

Contract exporter

With this feature, a Coordinator can easily export the details of all students that have been accepted as a TA for all courses in a given period. This corresponds to requirement RM03.



The students on this page have been selected as TAs by the manager(s) of this course. Here you can make the final decision on who will become TAs. In addition to selecting TAs one-by-one, you can accept all students at once, or enter text containing student numbers to only accept a subset of students.

Enter text containing student numbers to accept them all at once.

Name	Email	Student Number	Accept as TA
Inspector Gadget	inspectorgadget@tudelft.nl	11	No Yes
Ulick Norman Owen	ulicknormanowen@tudelft.nl	12	No Yes
Pieter Post	pieterpost@tudelft.nl	13	No Yes
Klaas Vaak	klaasvaak@tudelft.nl	14	No Yes

Figure 5.4: The TA coordinator can use this screen to have the final say on who becomes a TA.

To export contracts, the Coordinator first selects the quarter from which to export TAs. Upon selecting the quarter, the start and end date of the contract are automatically filled in according to the week numbers in the TU Delft Academic Calendar[1], but can of course be manually adjusted.

Per course type (Bachelor or Master course), a job code can be filled in and then accepted TAs to be exported can be selected. Because the accepting procedure may not always be instantaneous, it is possible to select a subset of TAs for export. There are select-all checkboxes for each course, and for entire course types.

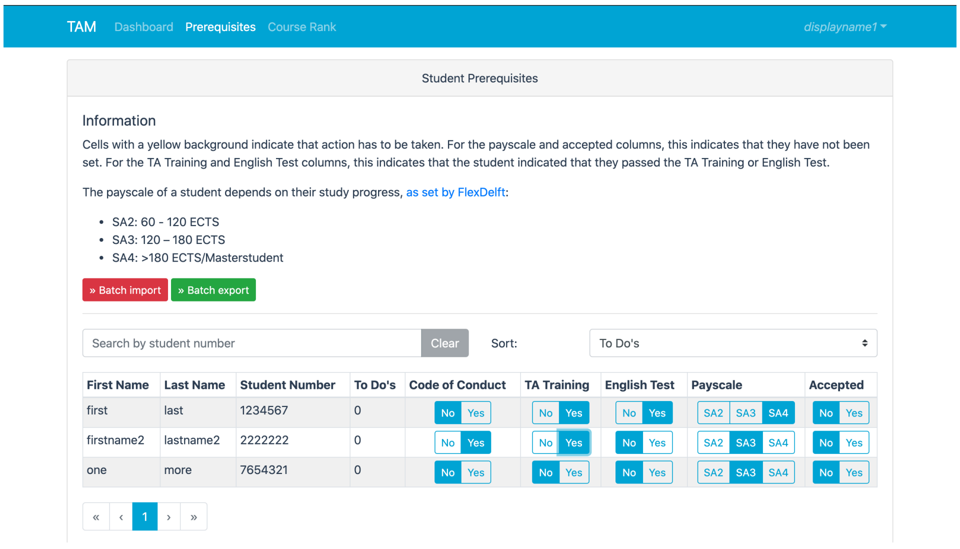
A screen capture of what it may look like is included as figure 5.7.

Availability overview

This section will describe the availability overview feature, which can be linked to requirement RS11. The feature for students to fill out their availability was already present in TAM, but a convenient way for the coordinator to get a clear overview of these availabilities was absent. This resulted in the coordinator having to extract the availabilities by manually executing SQL queries and running a python script on the output.

With the newly implemented Availability Overview feature this is not necessary anymore. To get an overview of which students are available for a weekly lab (and thus might be suitable to be TA for this lab) the coordinator selects the course and quarter of the lab and the weekday and timeslot the lab will take place. The result is a list with students who are interested in the course and are at least once

[1]TU Delft Academic calendar: `https://www.tudelft.nl/en/student/education/academic-calendar/`



Student Prerequisites

Information

Cells with a yellow background indicate that action has to be taken. For the payscale and accepted columns, this indicates that they have not been set. For the TA Training and English Test columns, this indicates that the student indicated that they passed the TA Training or English Test.

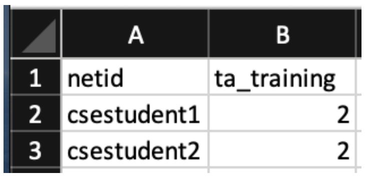
The payscale of a student depends on their study progress, as set by FlexDelft:

- SA2: 60 - 120 ECTS
- SA3: 120 – 180 ECTS
- SA4: >180 ECTS/Masterstudent

Sort: To Do's

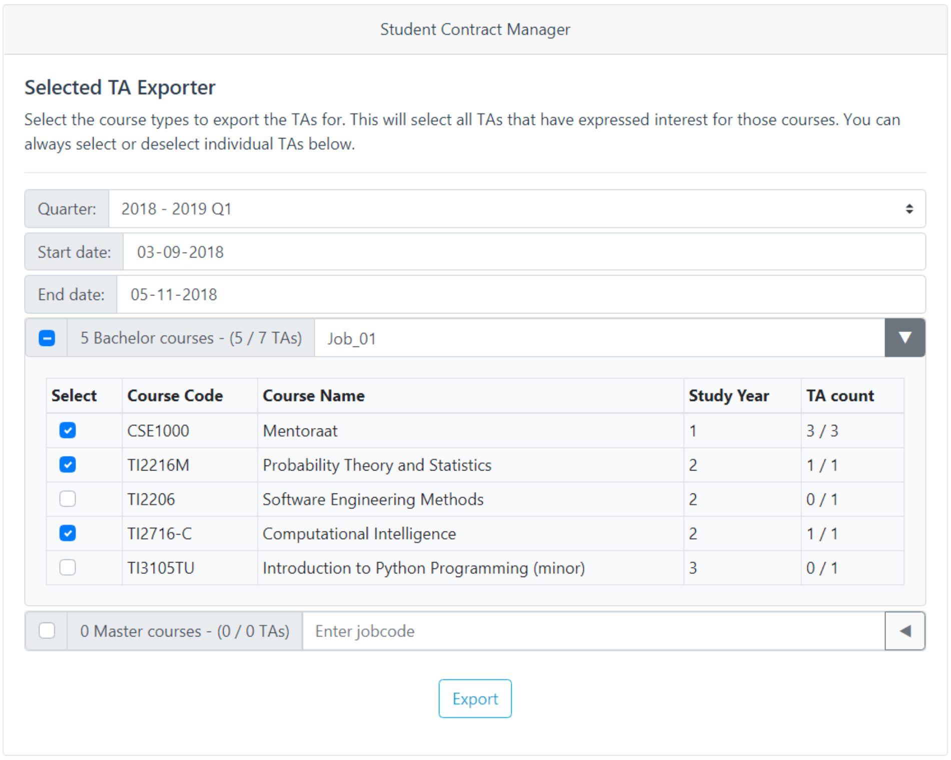
First Name	Last Name	Student Number	To Do's	Code of Conduct	TA Training	English Test	Payscale	Accepted
first	last	1234567	0	[✓ No] Yes	No [✓ Yes]	No [✓ Yes]	SA2 SA3 [✓ SA4]	[✓ No] Yes
firstname2	lastname2	2222222	0	No [✓ Yes]	No [✓ Yes]	[✓ No] Yes	SA2 [✓ SA3] SA4	[✓ No] Yes
one	more	7654321	0	[✓ No] Yes	[✓ No] Yes	[✓ No] Yes	SA2 [✓ SA3] SA4	[✓ No] Yes

Figure 5.5: The coordinator is now able to verify and set all student prerequisites using this page.



	A	B
1	netid	ta_training
2	csestudent1	2
3	csestudent2	2

Figure 5.6: An example of a partially filled out file that can be processed by TAM 2.0.



Student Contract Manager

Selected TA Exporter

Select the course types to export the TAs for. This will select all TAs that have expressed interest for those courses. You can always select or deselect individual TAs below.

Quarter: 2018 - 2019 Q1

Start date: 03-09-2018

End date: 05-11-2018

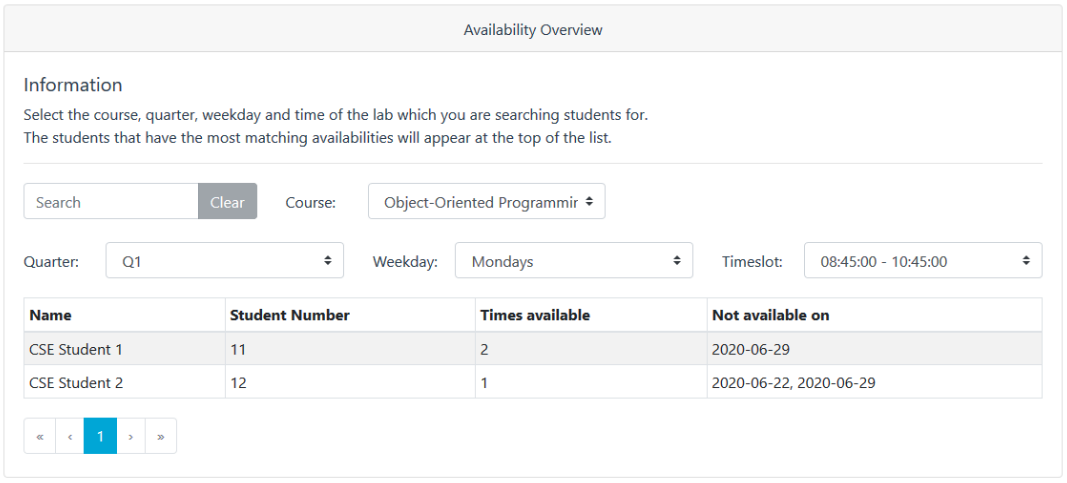
5 Bachelor courses - (5 / 7 TAs) Job_01

Select	Course Code	Course Name	Study Year	TA count
☑	CSE1000	Mentoraat	1	3 / 3
☑	TI2216M	Probability Theory and Statistics	2	1 / 1
☐	TI2206	Software Engineering Methods	2	0 / 1
☑	TI2716-C	Computational Intelligence	2	1 / 1
☐	TI3105TU	Introduction to Python Programming (minor)	3	0 / 1

0 Master courses - (0 / 0 TAs) Enter jobcode

Figure 5.7: The Contract Exporter. (see chapter 5.2)

available during the weekly timeslot. For every student his/her name is shown, his/her student number, how many times they are available for the selected weekly timeslot and lastly a list of dates they are



Availability Overview

Information

Select the course, quarter, weekday and time of the lab which you are searching students for.
The students that have the most matching availabilities will appear at the top of the list.

Course: Object-Oriented Programmir

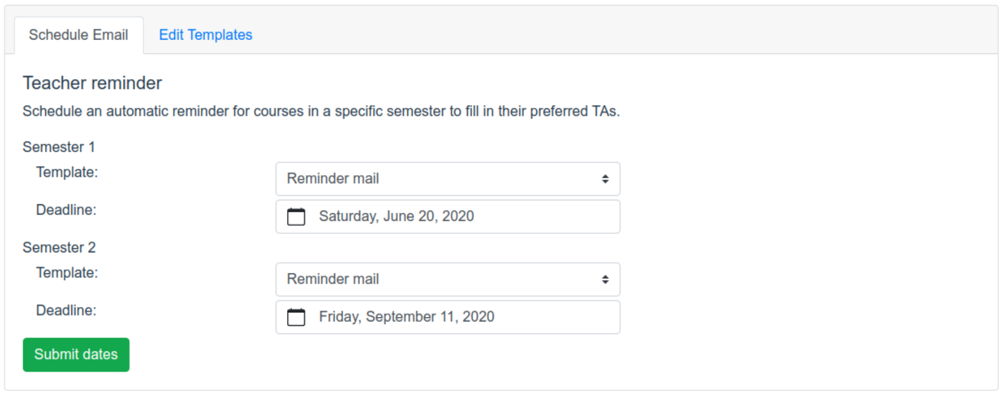
Quarter: Q1

Weekday: Mondays

Timeslot: 08:45:00 - 10:45:00

Name	Student Number	Times available	Not available on
CSE Student 1	11	2	2020-06-29
CSE Student 2	12	1	2020-06-22, 2020-06-29

Figure 5.8: The result of the request to show students that are interested in OOP and are available on Mondays between 8:45 and 10:45 during the first quarter.



Teacher reminder

Schedule an automatic reminder for courses in a specific semester to fill in their preferred TAs.

Semester 1

Template: Reminder mail

Deadline: Saturday, June 20, 2020

Semester 2

Template: Reminder mail

Deadline: Friday, September 11, 2020

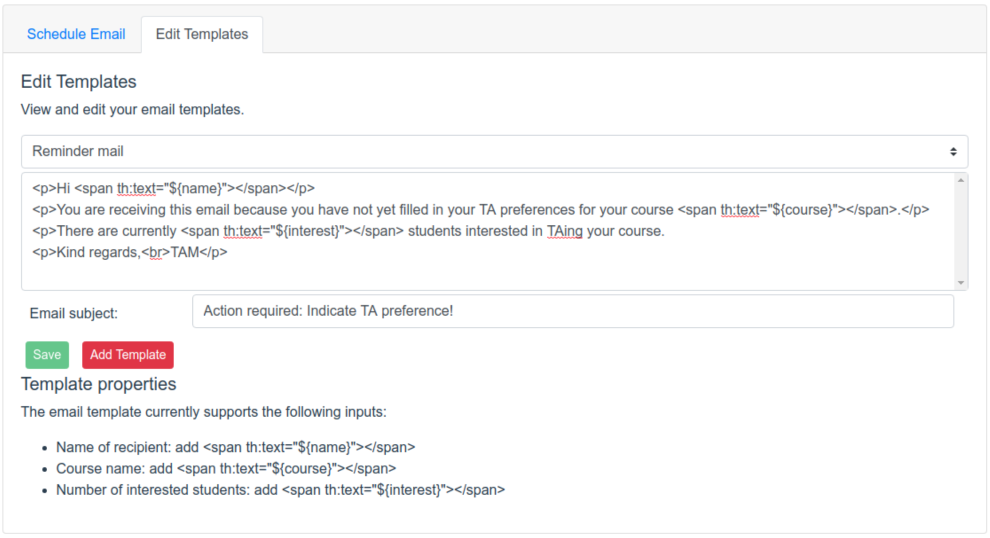
Figure 5.9: The view in which the TA coordinator sets the deadlines. For each semester the coordinator can set an email template and a deadline.

not available. If the student is available every week for the selected timeslot the last column shows ”Always available”. The list is sorted so that students with the highest availability are on top.

To see a specific student’s availability, the (part of a) name or student number can be entered in the search bar and only that student will be displayed. A screen capture of what it may look like is included as figure 5.8.

Automatic emails and Templates

This feature is linked to requirement RS10. With this feature the TA coordinator can schedule emails that will be send to teachers who have not filled in their TA preference yet. This way the TA coordinator does not have to check for every course individually if the tasks of the teacher have been performed already. What the email scheduler looks like can be seen in figure 5.9.
In order to personalize the emails to the needs of a specific course, TAM now also has an email template editor. Here the user can create HTML emails. Besides standard HTML attributes, there are also multiple TAM specific keywords that can be inserted in the template. These keywords will then be converted to the name of the course, the name of the teacher, or the number of interested students.



Edit Templates

View and edit your email templates.

Reminder mail

```
<p>Hi <span th:text="${name}"></span></p>
<p>You are receiving this email because you have not yet filled in your TA preferences for your course <span th:text="${course}"></span>.</p>
<p>There are currently <span th:text="${interest}"></span> students interested in TAing your course.
<p>Kind regards,<br>TAM</p>
```

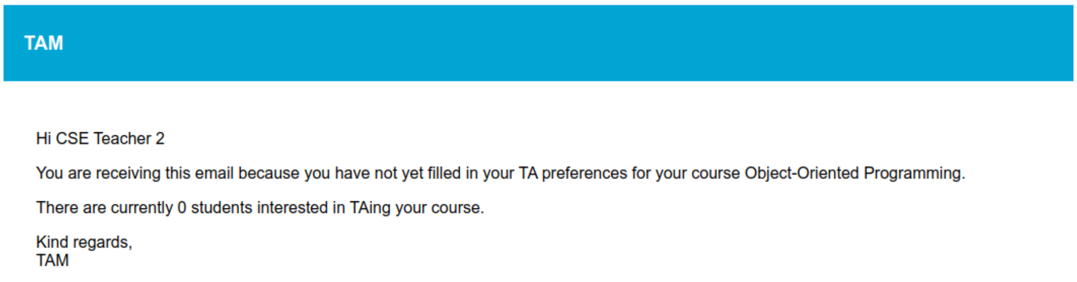
Email subject: Action required: Indicate TA preference!

Template properties

The email template currently supports the following inputs:

- Name of recipient: add <span th:text="${name}"></span>
- Course name: add <span th:text="${course}"></span>
- Number of interested students: add <span th:text="${interest}"></span>

Figure 5.10: The view used to edit email templates. Instructions about the variables are included at the bottom of the page.



Hi CSE Teacher 2

You are receiving this email because you have not yet filled in your TA preferences for your course Object-Oriented Programming.

There are currently 0 students interested in TAing your course.

Kind regards,
TAM

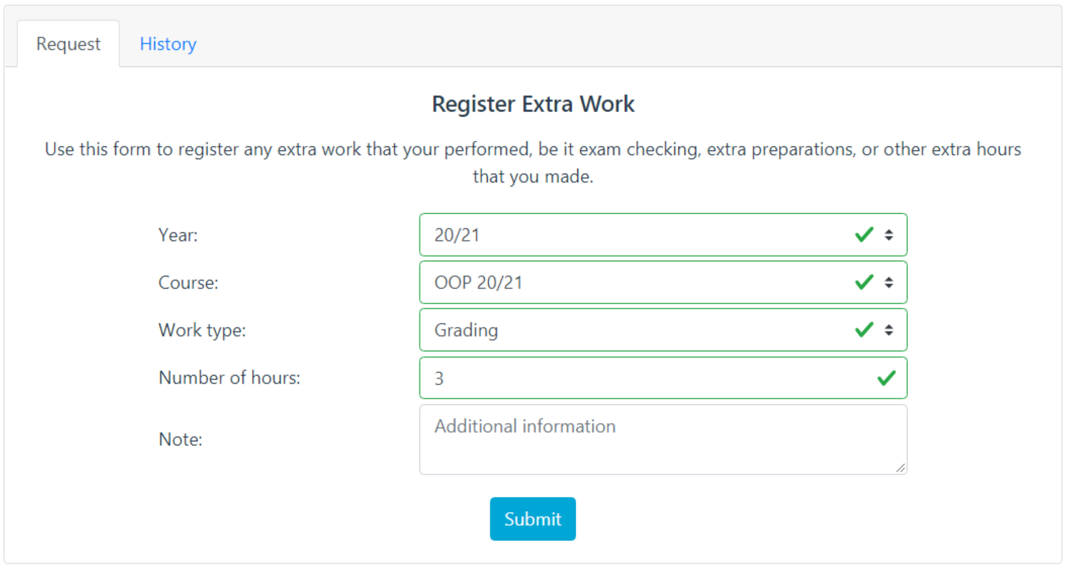
Figure 5.11: The resulting email using the created email template. The name, course name, and student interest are correctly filled in.

An example of the template editor in action can be found at figure 5.10. Using this template ensures the email clearly conveys the necessary information needed to take action. An example of an email send by the system can be found in figure 5.11.

Extra work and Overtime

A new feature has been made that allows students (TAs) to be able to request payment for any extra work they have performed outside of their regular TA contract. This corresponds to requirements RM08 and RM09. In the navigation bar, a new button titled 'Extra Work' can be found. This page is accessible to Students, Teachers, and Coordinators. For each of these groups, a different view is exposed.

For students, there is a form with which it is possible to submit extra work to the relevant teacher, and to view any past submissions they've made and their status (see figure 5.12).
For Teachers, there is a list of incoming requests from students, and it is possible to either accept or reject each request (See figure 5.13). On denial, an optional reasoning can be entered, which is visible to the student.



Request History

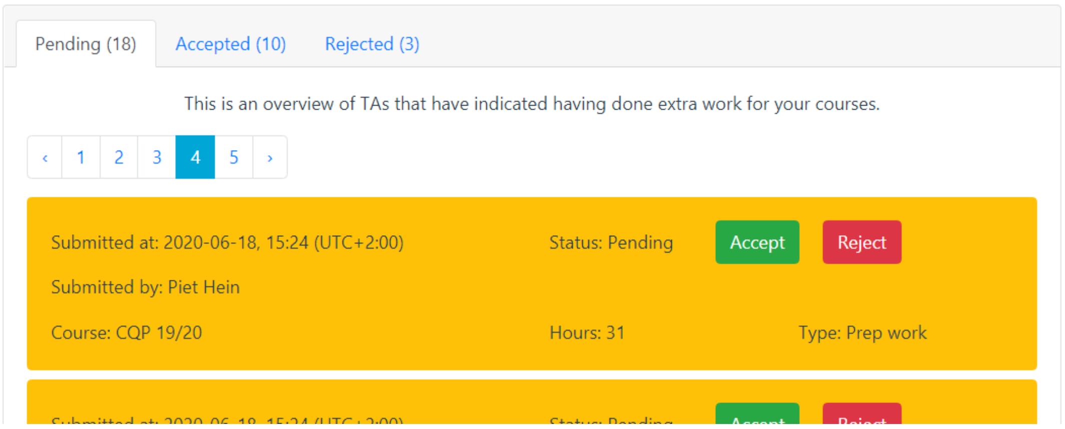
Register Extra Work

Use this form to register any extra work that your performed, be it exam checking, extra preparations, or other extra hours that you made.

Year:	20/21
Course:	OOP 20/21
Work type:	Grading
Number of hours:	3
Note:	Additional information

Submit

Figure 5.12: A form for students to submit extra hours.



Pending (18) Accepted (10) Rejected (3)

This is an overview of TAs that have indicated having done extra work for your courses.

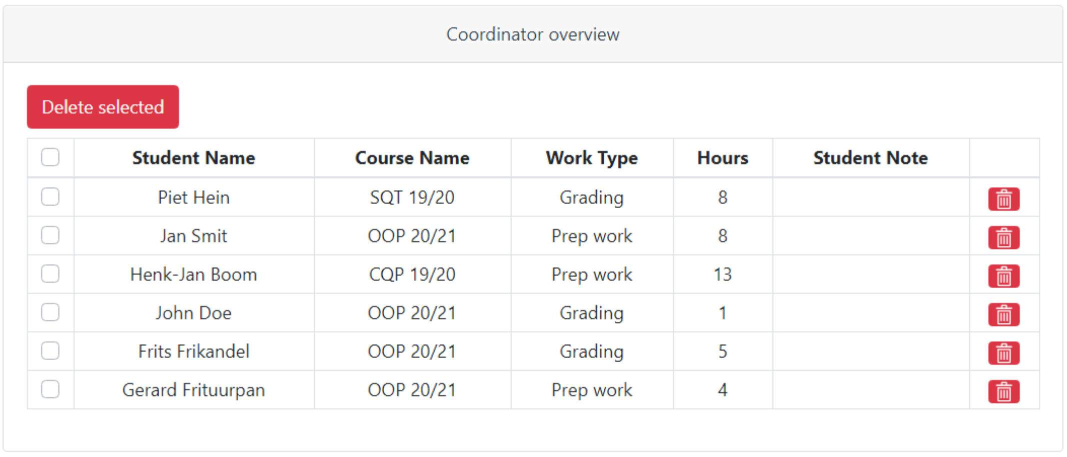
‹ 1 2 3 4 5 ›

Submitted at: 2020-06-18, 15:24 (UTC+2:00) Status: Pending Accept Reject

Submitted by: Piet Hein

Course: CQP 19/20 Hours: 31 Type: Prep work

Figure 5.13: A teacher's overview of submitted extra work


Coordinator overview

☐	Student Name	Course Name	Work Type	Hours	Student Note	
☐	Piet Hein	SQT 19/20	Grading	8		🗑
☐	Jan Smit	OOP 20/21	Prep work	8		🗑
☐	Henk-Jan Boom	CQP 19/20	Prep work	13		🗑
☐	John Doe	OOP 20/21	Grading	1		🗑
☐	Frits Frikandel	OOP 20/21	Grading	5		🗑
☐	Gerard Frituurpan	OOP 20/21	Prep work	4		🗑

Figure 5.14: A coordinator's overview of accepted extra work

For coordinators, there is a compact list of all extra work submissions that have been accepted by teachers (see figure 5.14). This way the coordinator can write up a contract for each student and then remove the request from their list once it has been processed.

Inventory manager

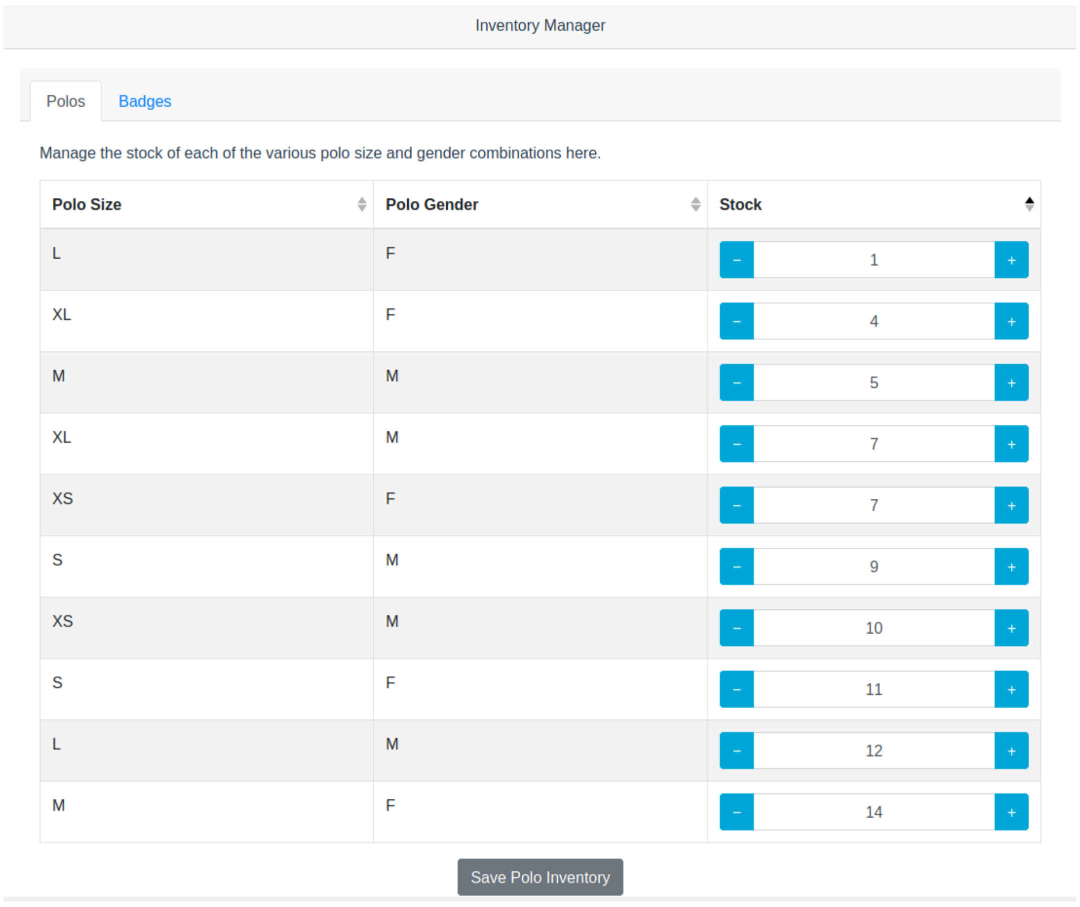
One of the last features we added is the inventory manager. This page allows the TA coordinator to keep track of the stock of the various TA polos and badges. On the pages seen in figures 5.15 and 5.16, the TA coordinator can see and update the stock of each polo and badge.
In addition, the coordinator can use this page to hand out badges and polos, as well as to see who has received what items.

5.3. Database

Several changes have been made to the database in comparison with the database design in 3.1. This was due to new features that needed their own table, developments made within LabraCORE and having a better understanding of the purpose of the data that is in the database. In this section we will highlight those changes and elaborate on our choices to change the structure of the database. An overview of the schema can be found in 5.17. For simplicity we only included the tables and columns of the LabraCORE database that are relevant for TAM.

LabraCORE

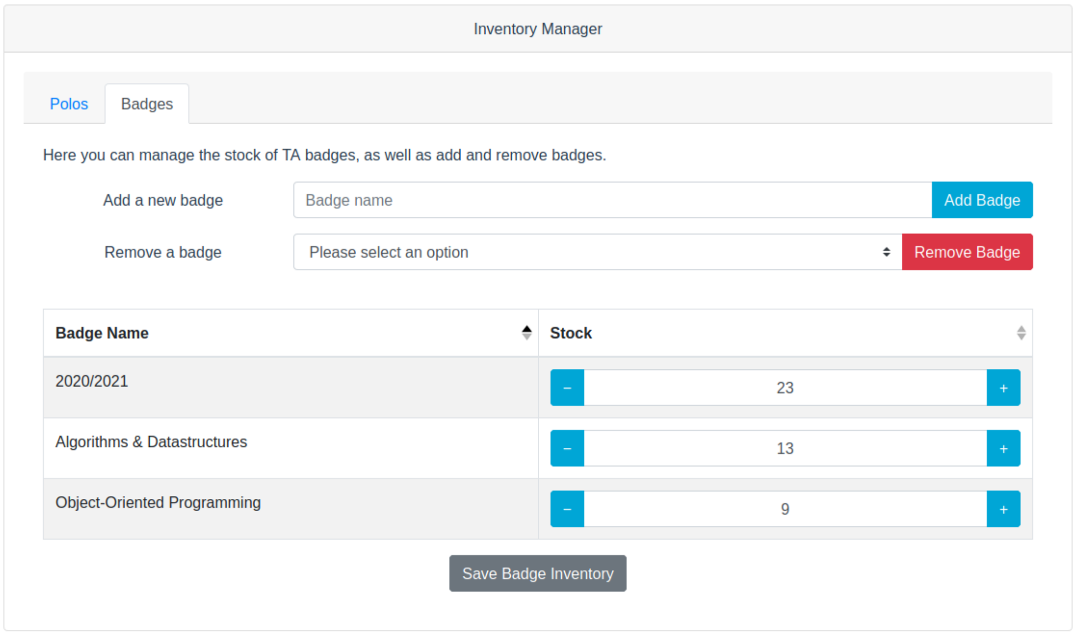
During the timespan of this project, LabraCORE was still under development. This resulted in some changes on the LabraCORE side, which caused for example an edition to have a start and end date instead of a year and quarter. Besides this, working on LabraCORE integration with TAM caused us to get a better understanding of how exactly data is stored in LabraCORE, e.g. CourseEdition is actually called Edition, which is linked to a separate table Course.
A major difference is also the primary key of the User/Person table, namely a bigint instead of the string of a netid. This had major consequences for the TAM database, since we needed to switch our person identifier from the netid to the LabraCORE id. This is the reason all tables related to a user now make use of the bigint user_id and not the string netid.
Another difference in the initial design and the final design is the absence of the Manages and CourseAssignment tables. In LabraCORE a course edition has an object named roles. This object includes the user id's of all teachers, TAs, students and other course related roles and this object is what is used in TAM to retrieve and update course related roles.
The final difference we want to address is the absence of the Feedback table in the final design. This table will in the future still be in LabraCORE, but is not included in this overview. The feature that is related to this table has not been implemented in TAM during this period in consultation with the client, and therefore TAM currently does not use this table.



Manage the stock of each of the various polo size and gender combinations here.

Polo Size	Polo Gender	Stock
L	F	1
XL	F	4
M	M	5
XL	M	7
XS	F	7
S	M	9
XS	M	10
S	F	11
L	M	12
M	F	14

Figure 5.15: Here the stock of each polo can be managed.



Here you can manage the stock of TA badges, as well as add and remove badges.

Badge Name	Stock
2020/2021	23
Algorithms & Datastructures	13
Object-Oriented Programming	9

Figure 5.16: The coordinator can add and remove badges on this page, in addition to managing the badge inventory.

Timestamps

In the initial design all rows in almost all tables had a last_modified and created field with the corresponding timestamps. We found that these timestamps were not used and were redundant for the TAM database, with the only exception being the timestamp in the GDPR table. For both simplicity as well as efficiency reasons all other timestamps were removed from the database.

Roles

As can be seen in 5.17 the table User_Role is not included in the final design. This is due to a change in the way roles are saved and used in TAM. In the former version of TAM every person would have a role in the TAM database, but, with the arrival of the LabraCORE database and its mention of roles, this was no longer necessary. Student roles could be retrieved through the roles object in Edition, just like the Teacher roles. The only TAM related roles that are not included in LabraCORE are the TA Coordinator, Verifier and TAM Admin. Only the users that need one of those three roles will be saved in the TAM database. With the Admin role having access to all pages and endpoints, the Coordinator role only to those pages relevant for the coordinator and verifier, and the Verifier role just to the pages relevant for a verifier. By saving just these roles a massive decrease in data could be achieved.

Additions

Several tables are added in comparison with the initial design, due to the addition of features to TAM. Most of these tables are self explanatory, e.g. ExtraWork and EmailTemplate, and will not be discussed further. However, there are some tables we want to elaborate on. First of all, the addition of the Polo table. With the introduction of the inventory feature, polo specifications needed to be used in more than one table. To make sure that a clear overview remains of what polos are present in the database, we chose to move this to a separate table and make foreign key constraints for tables that uses this data. Lastly, the addition of the CourseEdition table. In the current version of TAM it is possible for teachers to give a short description about TA'ing this course, e.g. "This is a semester course". These remarks are solely used in TAM and will therefore not be stored in LabraCORE, which is why a new table needed to be added.

GDPR

Accepting the General Data Protection Regulation (GDPR) is necessary for every user to be allowed to use TAM. In the previous version of TAM this was stored in the User table with a boolean and timestamp. We found this slightly inconvenient, since this boolean must never be false. Not having accepted the GDPR should immediately result in no access to TAM to make sure all regulations are obeyed. For this reason we made a separate table for all users that accepted the GDPR, storing their user id and the timestamp of the moment they accepted it. Without a mention in this table, a user will not get access to TAM.

Timeslots

The timeslot and availability tables have received an overhaul, starting with a change in the naming of the Slot / Timeslot / DateSlot tables. In the initial version of TAM the Slot table would store timeslots that are available during a random day, e.g. 8:45:00 - 10:45:00. The Timeslot table would then store dates with timeslots that, which were used among other things to allow users to submit availability. We found the naming of these tables slightly awkward, which is why we changed the name of Slot to Timeslot, since it stores timeslots, and the name of Timeslot to DateSlot, since it stores a date with a timeslot. Furthermore, we have greatly reduced the amount of data redundancy, resulting in the data being simpler to use, requiring less storage and allowing the availability page to use less bandwidth to load available timeslots and store availability.

5.4. User tests

TA with platform experience

One of the users we got to test our platform was a TA who used TAM previously to give their course preferences. Before accessing the website, the user was given a list of tasks to perform. Instructions on where to perform these tasks were omitted intentionally.

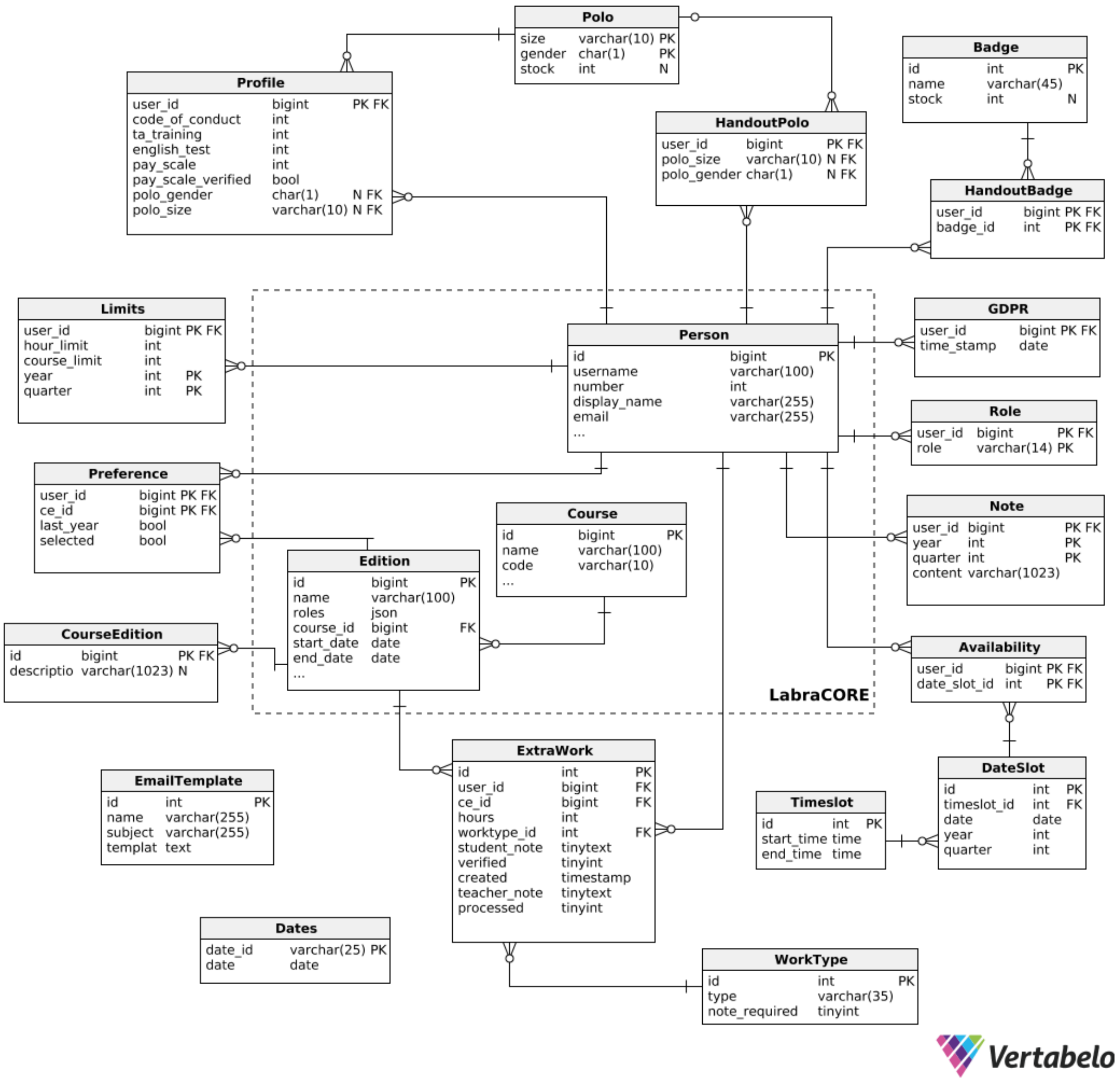


Figure 5.17: Final database design schema of TAM 2.0

- Indicate your preference for one or multiple courses.
- Request extra hours for preparing a lab course.
- Indicate the timeslots you are available to TA.

The user indicated afterwards that filling in both their preferences and availability were easy to do, since these are already present in the live version of TAM. Therefore the user has already filled in these preferences before. For requesting extra work, the user took slightly longer to find the correct page to perform the task. However, after getting to the correct page, the user indicated that the interface was incredibly clear and intuitive to fill in.

Students without platform experience

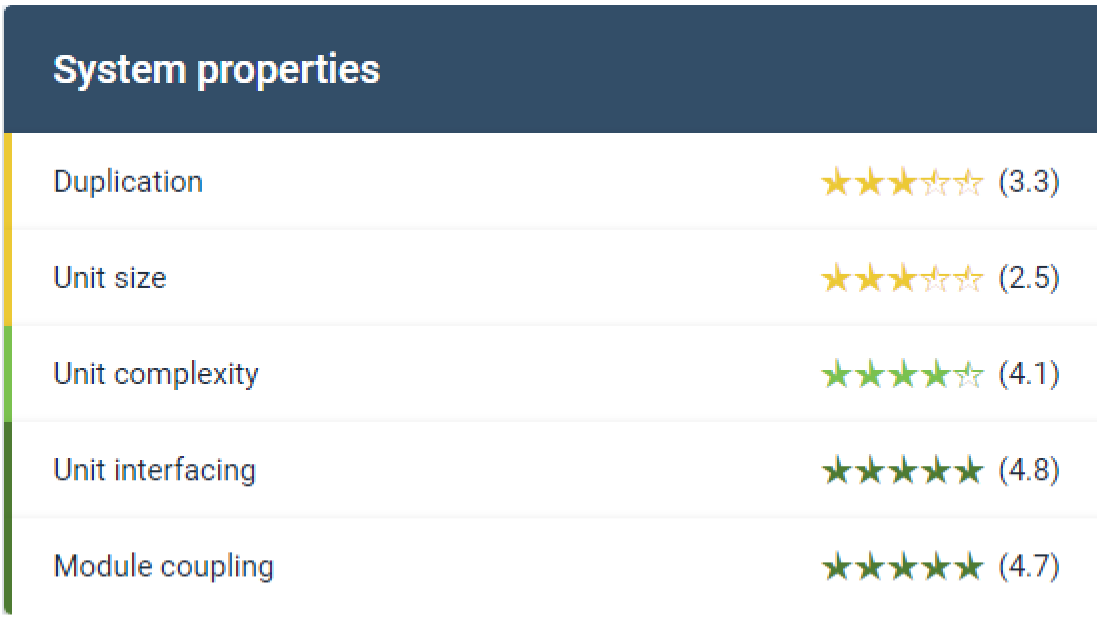
Two other users we have tested did not have any experience with TAM or signing up to be a TA in any way. We wanted to test our version of TAM with these users as well, since it can give us an impression of how user-friendly TAM really is. The users were only told they have received an email from the TA coordinator with TAM being mentioned as the platform to indicate TA preferences. The tasks they had to perform were:

- Indicate your preference for one or multiple courses.
- Indicate your polo preference.
- Update your pay scale.
- Indicate the timeslots you are available to TA.
- Request extra hours for preparing a lab course.

The users had no difficulty navigating through TAM and they managed to fulfill all tasks without further help. There were two minor things they found somewhat confusing: the polo preference button on the dashboard and the copy and save button when updating the availability. The first was indeed not yet updated. The button used to redirect to the polo page, but this became the Profile page. The latter was intended since it provides an extra use case. However, it is arguable whether or not this extra use case is necessary, since it might cause confusion.

5.5. SIG

During the last stages of the project, two opportunities were given to let the code be checked by the Software Improvement Group[2] (SIG) to get feedback on how to improve the code. SIG does not actually execute the code but reviews the code in aspects like maintainability, module coupling and component balance. In figure 5.18 an overview is given on the different scores we received in the tested categories.


System properties

Duplication	★★★☆☆ (3.3)
Unit size	★★★☆☆ (2.5)
Unit complexity	★★★★☆ (4.1)
Unit interfacing	★★★★★ (4.8)
Module coupling	★★★★★ (4.7)

Figure 5.18: Overview of scores received from SIG.

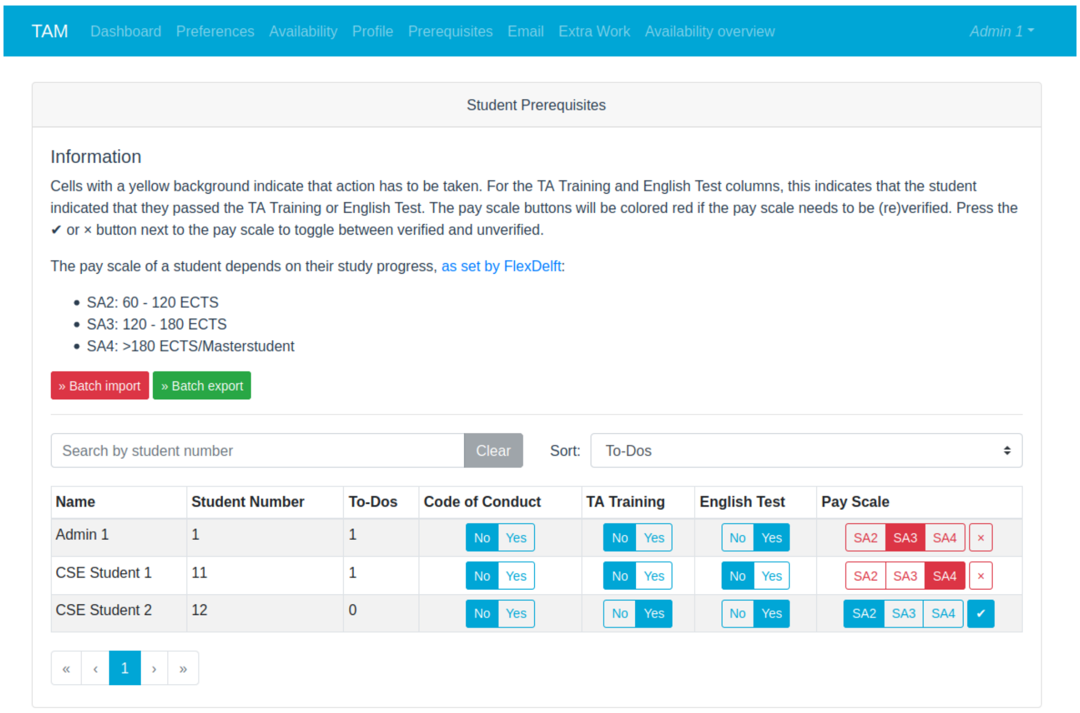
Overall our feedback was very positive. There were only two categories which we did not score well on: Duplication and Unit size. We analyzed the files marked by SIG and came to the following conclusions. Unit size only marked files containing test code and data files. Since these files are inherently long, and do not benefit from being split up, we decided that we would not change these files. With regards to duplication we found again that most of this originated in test files. However, we did decide that it was beneficial to minimize duplicated code.

5.6. User manual

When designing new features, we have always aimed to make the most intuitive system possible. Besides this, we have provided a detailed explanation of how to use specific parts of our system on

[2] `https://www.softwareimprovementgroup.com/`.

their respective web pages. An example of this can be found in figure 5.19. We believe that the combination of these two things means that it is not necessary to write a user guide, as this should not be needed in order to use TAM.



Student Prerequisites

Information

Cells with a yellow background indicate that action has to be taken. For the TA Training and English Test columns, this indicates that the student indicated that they passed the TA Training or English Test. The pay scale buttons will be colored red if the pay scale needs to be (re)verified. Press the ✔ or × button next to the pay scale to toggle between verified and unverified.

The pay scale of a student depends on their study progress, as set by FlexDelft:

- SA2: 60 - 120 ECTS
- SA3: 120 - 180 ECTS
- SA4: >180 ECTS/Masterstudent

» Batch import » Batch export

Search by student number Clear Sort: To-Dos

Name	Student Number	To-Dos	Code of Conduct	TA Training	English Test	Pay Scale
Admin 1	1	1	No Yes	No Yes	No Yes	SA2 SA3 SA4 ×
CSE Student 1	11	1	No Yes	No Yes	No Yes	SA2 SA3 SA4 ×
CSE Student 2	12	0	No Yes	No Yes	No Yes	SA2 SA3 SA4 ✔

« ‹ 1 › »

Figure 5.19: This page clearly states all the information needed to use the system at the top of the page.

6 Ethical implications

During the development of TAM 2.0 we have made a number of decisions that have ethical implications with regard to storing data. In this chapter we will discuss these implications, and how we have reduced their effect. Firstly, we will talk about personal data stored in TAM, secondly, we will discuss the integration with LabraCORE. Finally, we will discuss the changes we made to our GDPR policy.

6.1. Personal data

In order for TAM to be used to its full potential, a student has to provide the system with some personal information. This information ranges from polo information to the pay scale the student is in. A student also has to provide their availability of certain time slots. In order to comply with the General Data Protection Regulation (GDPR), TAM only starts collecting this data from our users when they approve us to do so. The system will only collect the minimal amount of data needed for the proper functioning of our platform. This still means that there is certain types of data that is only needed for certain roles within our system. To limit the amount of data a particular user can view, we have designed our system around specific roles. For example, a teacher is able to see basic information (name, email, student number) of students that are interested in teaching their courses. However, this teacher is not able to see more detailed information, like the pay scale of these students. To ensure this division in access levels, our system checks for the required level of authentication on every request made. While well intended users of our system will not encounter situations where they request data not intended for them, these strict authentication requirements ensure that users with bad intentions will be denied access.

6.2. LabraCORE

In order to create a more robust structure between different TU Delft systems, the TU Delft has created LabraCORE. LabraCORE is a central point to store all information about students and courses. Since TAM is now also integrated with LabraCORE, this means that some of the data used will come from LabraCORE, and some data will also be shared with LabraCORE. To reduce the ethical implications of this integration, TAM has taken a few measures. Firstly, none of the data stored in TAM can be accessed outside of its own system. This greatly limits the risk of our data falling in the wrong hands. Secondly, only essential data is shared with LabraCORE. Information like a specific student being assigned as a TA is required for different TU Delft systems, and is therefore necessary to share. But whether or not a student is interested in being a TA for a course is not relevant out side of TAM. Therefore we will not share it.

6.3. GDPR

When evaluating the way the previous version of TAM regulate the GDPR, we found that some data was stored before the user gave permission. When logging in for the first time, when the user is asked to accept the GDPR, on the back end a database entry is already created. While it is possible to delete the data, the user has to manually do this. This means that there will undoubtedly be entries in the database that have not accepted the GDPR. This practice obviously violates the rules of the GDPR, since data is stored of users without permission. Besides this, the data was also stored in a very inconvenient way. The User table used to contain a GDPR boolean and timestamp. However, since TAM should not be storing data when the GDPR is not accepted, the boolean must never be false. Because of this reason, we have created a separate GDPR table. This table stores the user id and a timestamp representing the moment when the GDPR is accepted. Users are not able to access TAM when this table does not mention them. The addition of this table ensure that TAM does not contain any data of users that have not given consent.

7 Conclusion

With the previous version of TAM, managing TAs already got way more convenient for the TA coordinator. After analyzing the current process of the coordinator, teachers and students, and analyzing the current version of TAM we came to the conclusion that there still was a lot to improve to ease the lives of all stakeholders. By implementing new features, tasks that could previously take hours are now finished by simply clicking a button. Not only does this save a tremendous amount of time for especially the coordinator, it also improves the correctness of data, since it is no longer stored in separate spreadsheets on the coordinator's machine. All data is now centralized in the TAM database.

Although the integration of LabraCORE in TAM did not come easily, it now makes sure that TAM is more maintainable and that TAM can more easily interact with other system within the Computer Science department. This integration also required a complete redesign of the database. The new design results in a database that is easier to understand and that stores less duplicate and redundant data.

Having a system that needs to access personal information of its users, TAM needs to obey to the GDPR. We have taken all measures possible to make sure we only collect the data necessary for TAM to function. Furthermore, data is only accessible by users that actually need that data. The integration with LabraCORE added extra data sharing, but we minimized the amount of data that is sent to LabraCORE and LabraCORE does not have access to the TAM database to prevent data leaks. The improved GDPR handling makes sure that users are aware of their rights and the fact that TAM will be storing their personal information.

In the process of improving TAM, we as a team learned a lot about software development and teamwork. Doing this project in a period of quarantine caused us to completely rediscover our work methods, but by keeping in contact through all kinds of platforms we were able to still collaborate in this project. Staying in contact with both our coach and client weekly, made sure that we stayed on track and kept on meeting our clients desires and requirements. We are proud to say that the COVID-19 crisis did not prevent us from finalizing this bachelor end project and solving all of the major problems we identified during our problem analysis. There are still ways to improve TAM in the future, but with this new version of TAM the biggest burden has been lifted of the shoulders of the client.

8 Discussion

Now that our Bachelor End Project is nearing completion, we can look back on how the project progressed, the decisions we have had to make, and what difficulties we have encountered along the way. As such, we will start this chapter with a reflection of our project, which will be followed by potential future work where we suggest a couple of features and changes that may improve the usability and usefulness of the TAM platform.

8.1. Reflection

Communication and meetings

Throughout the project we have made extensive use of programs such as Discord to discuss features, have meetings and assist each other. We held meetings regularly within our group (once or twice a week), and scheduled meetings with our coach and client about every one or two weeks. Deadlines have been communicated clearly, and responses were given in a timely fashion. Since all group members, our coach and client were kept in the loop, it was easy to measure our progress.

LabraCORE

The importance of LabraCORE became known to us a few weeks into the project. As a result we updated our set of requirements, and to accommodate for this our planning had to be adjusted as well. LabraCORE was deployed, and we started using an online instance to rely on in our development environment. We managed to keep up with our original planning for a couple of weeks, until we unfortunately faced an issue with LabraCORE while we were busy rewriting all old and new code depending on data that now had to be taken from the LabraCORE database. The issue we were facing was a bug in LabraCORE that prevented us from authenticating, which in turn prevented us from properly testing our changes. This, combined with a lack of documentation at the time on the LabraCORE side, stalled our progress on migrating to LabraCORE, since we deemed it too risky to use the system based on our assumptions. Once the authentication issue had been resolved, we switched to running a local version of LabraCORE to have a more stable environment to work with. In hindsight, we should have used a local version from the very beginning to make it easier for us to transition to LabraCORE and test changes. We managed to finish the majority of the migration, with the exception of data that was missing from LabraCORE's model.

In particular, TAM uses the notion of quarters to present and keep track of all kinds of data (e.g. course preferences, availability). The LabraCORE model did not include any data allowing us for instance to determine to what quarter a course belongs. Initially, after proposing to add a year and quarter to LabraCORE's representation of course editions - which was met with a positive response - an issue was made requesting this information to be added. Then we received a response, which questioned the use of quarters, which is a good point to consider when keeping different study programs in mind. After some more discussion, it was decided that a start and end date would be added to LabraCORE's course editions. Two weeks later, we could finally use this bit of information, but due to the discussion and uncertainty over what kind of data we could expect, we could only work on TAM based on assumptions in the meantime. As a side effect of this change being introduced extremely late in the project, we have chosen to let the coordinator decide the dates during which courses should be displayed for the first and second semesters, which we are not particularly fond of.

8.2. Future work

In this section we discuss potential improvements and extensions that could be made to TAM in the future.

Email accepted students

In the future, it should be possible to use the current email system to notify students when they are accepted as a TA. This email could also provide the student with information regarding the TA training if they still have to complete the training.

Display current email deadlines

Currently it is impossible to see when TAM will send the reminder email to teachers who have not filled in their TA preferences. It would be helpful to the coordinator if he/she has an overview of the scheduled deadlines, so they can always check whether they are scheduled correctly.

Schedule deadlines years ahead

Currently only the upcoming semester can have deadlines set for the reminder email to teachers. When the template used by the TA coordinator has been perfected over multiple iterations, it is useful if he/she can schedule this email for the coming years so they do not have to worry about it anymore for a substantial amount of time.

Scale TAM for use by multiple study programs

There are two approaches that can be taken to allow TAM to be used by multiple study programs and faculties. The first approach is to simply create and run different instances of TAM. The other approach is to make TAM aware of the fact that there are multiple study programs. This requires fields and filters to be added, such that only information relevant to a specific program is loaded and stored. However, two obstacles exist which must be addressed either way. The first one being LabraCORE. Since TAM is now dependent on LabraCORE for a large part of the information it uses, it is required that LabraCORE's model is compatible with that of different study programs, and that data of other programs is stored in the LabraCORE database. The other obstacle is TAM's usage of quarters. Since the majority of courses in the Computer Science and Engineering program span a quarter, the original version of TAM relied heavily upon this information. Quite late during our project we discussed this with members of the LabraCORE team, but no alternative to quarters was available yet. A potential solution would be to base everything on start and end dates, or on duration of a course expressed either in days or partition of the year (e.g. quarter, semester, octal).

A
Info sheet

The info sheet can be found on the next page.

General information

Project title: Teaching Assistant Management platform (TAM) 2.0
Client organization: Educational Innovation Projects (EIP), TU Delft
Date of the final presentation: July 2, 2020

Description

The main challenges of this project were integrating LabraCORE, an initiative to centralize data shared by TU Delft systems being worked on by the EIP group, into TAM, while maintaining and extending its functionality by simultaneously adding new features to the system to improve its usability.
During our research phase, we analyzed the TA recruitment process and the first version of TAM, and learned that some parts of the existing system were not in use but that it was a good base to continue work on.
We organized our project into sprints of two weeks and held weekly meetings with our group members. In addition, we planned progress meetings with our client and coach about every two weeks during which we discussed the status of the project and demonstrated new features.
With our project we have integrated LabraCORE into TAM, and extended its functionality to allow much more to be done through TAM, reducing the amount of manual work to be performed by the TA coordinator. As the LabraCORE integration took more time than initially planned, we had to let go of two minor features.
In our report we propose a couple of areas where TAM could be improved, and we suggest additional features. Finally, TAM 2.0 is intended to be used at the start of quarter 1 of academic year 2020.

Members of the project team

Name	Interests	Contributions
Dennis Böhm	AI, Machine learning, Cybersecurity	Back-end development, database design, LabraCORE integration
Dylan Franken	Algorithm design, Computer graphics, Cybersecurity, Programming	Front-end and back-end development, database design, project configuration
Govert de Gans	Front-end development, Embedded programming	Front-end and back-end additions
Bram van Kooten	Full stack web development	Front-end and back-end development
Bas Musters	Front-end development, Embedded systems, Robotics	Front-end development

All team members contributed to writing the different iterations of our reports and preparing presentations and demos for both the midterm meetings, which gauged progress of the project, and the final presentation session.

Key persons

Client:	Stefan Hugtenburg, MSc	CSE Teaching Team, TU Delft
Coach:	Dr. Lydia Chen	Distributed Systems, TU Delft
Contact:	Dennis Böhm	`denni.bohm@gmail.com`

The final report for this project can be found at `https://repository.tudelft.nl/`.

B Original project description

Implement a new version of TAM as part of the Labrador Project

We need a new or improved version of TAM, our TA Management system; one with an improved data model that can connect to our Labracore infrastructure and thus becomes an integral part of Labrador (as Queue and CPM 2.0). It needs to be as flexible as it is now, but we should also learn from the flaws.

There are many challenges in this project:

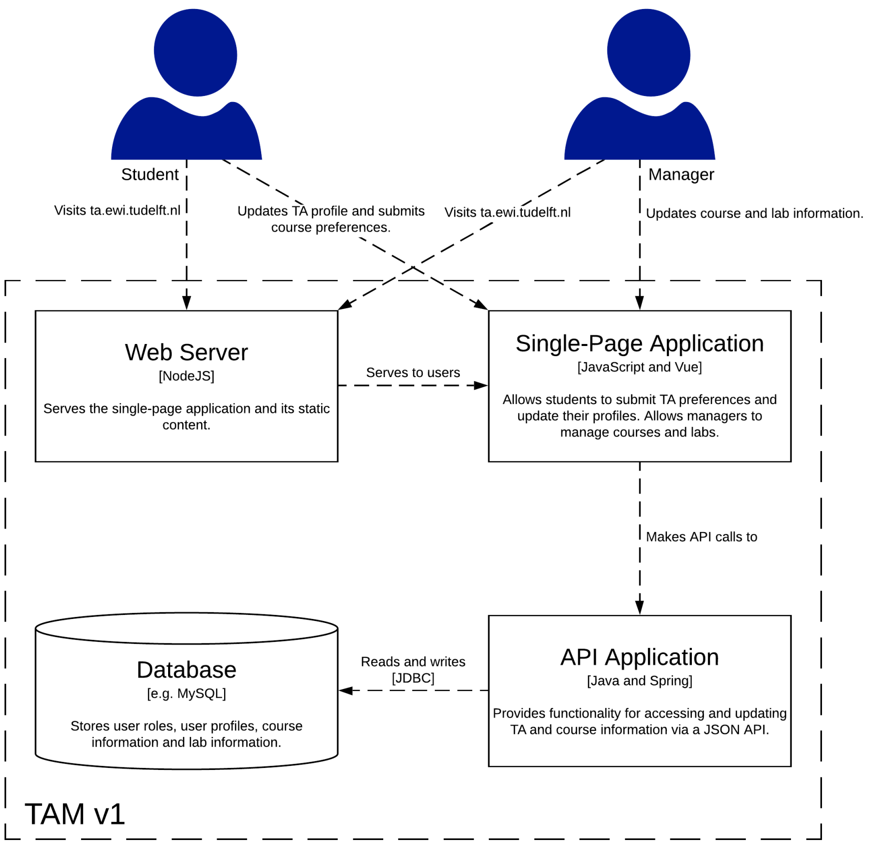
- scalability (100s of TAs for many courses (and several programmes!))
- flexibility
  - a structure with programmes, courses, TAs, Teachers, Admins, Timetables, ...
  - selection of TAs, keeping track of a TAs history, their training status, ...
  - functionality for TAs to indicate their preferences and availability
- portability; integration with Labracore to export/import TAs to/from existing services like CPM, Queue and Gitlab.

Since this project will be used by (developers of) the TU it is important that the project has proper documentation and that design choices are clearly documented. This way the project can still be extended and maintained after you've graduated.

The group will need to figure out what is required for TAM 2.0 and determine what can be achieved as an MVP in 10 weeks. Your project will be used in Q1 of academic year 2020/2021 if all goes well and the code will probably be released under a AGPL 3 license.

C
TAM 1.0

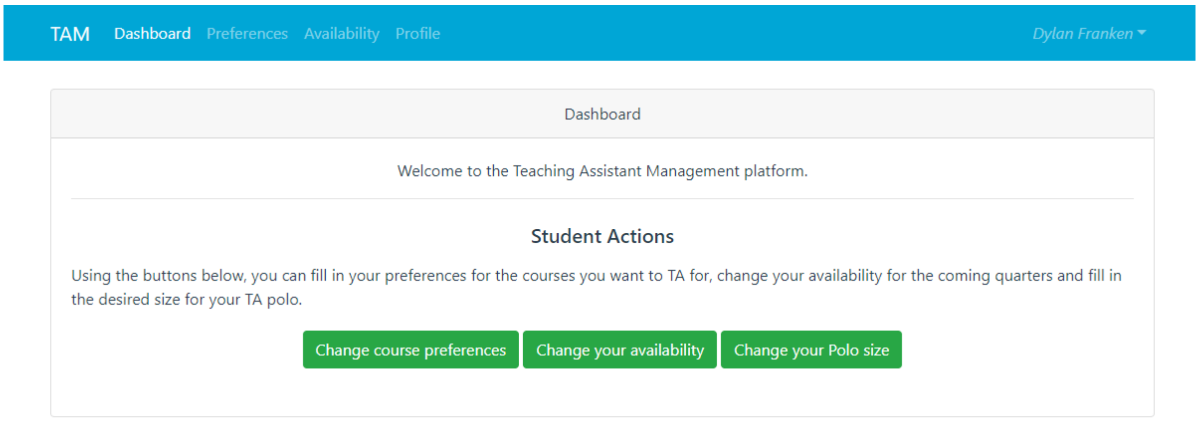
This appendix documents the features that were originally present in TAM 1.0 by providing screenshots and descriptions of the pages available to various types of users. In addition, we provide a high-level overview of the architecture in figure C.1.



Student
Manager
Visits ta.ewi.tudelft.nl
Updates TA profile and submits course preferences.
Visits ta.ewi.tudelft.nl
Updates course and lab information.
Web Server
[NodeJS]
Serves the single-page application and its static content.
Serves to users
Single-Page Application
[JavaScript and Vue]
Allows students to submit TA preferences and update their profiles. Allows managers to manage courses and labs.
Makes API calls to
Database
[e.g. MySQL]
Stores user roles, user profiles, course information and lab information.
Reads and writes
[JDBC]
API Application
[Java and Spring]
Provides functionality for accessing and updating TA and course information via a JSON API.
TAM v1

Figure C.1: A high-level overview of TAM 1.0. Note that the single-page application is instead served by Spring in the production environment.

C.1. Student view



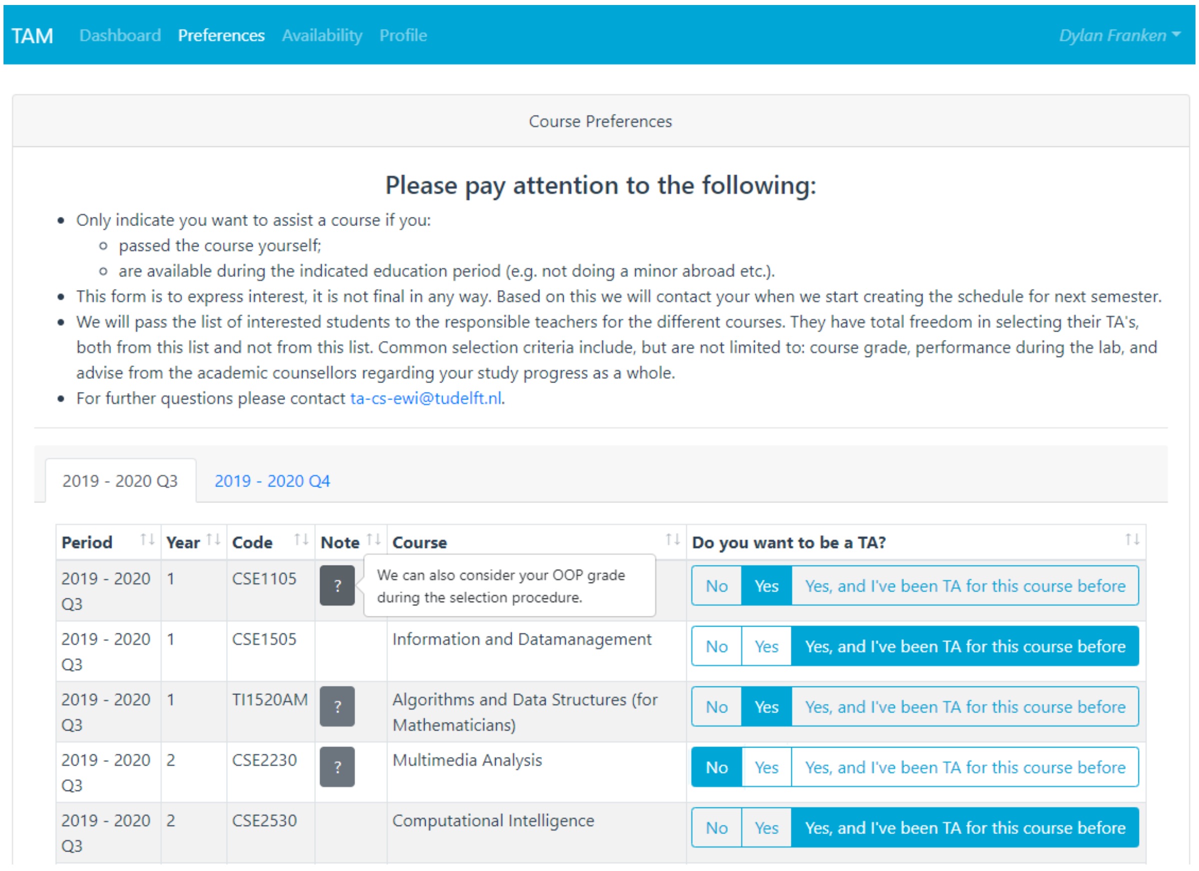
Dashboard

Welcome to the Teaching Assistant Management platform.

Student Actions

Using the buttons below, you can fill in your preferences for the courses you want to TA for, change your availability for the coming quarters and fill in the desired size for your TA polo.

Figure C.2: This is the dashboard page for students. It briefly explains what a student can do, and links to the other pages.



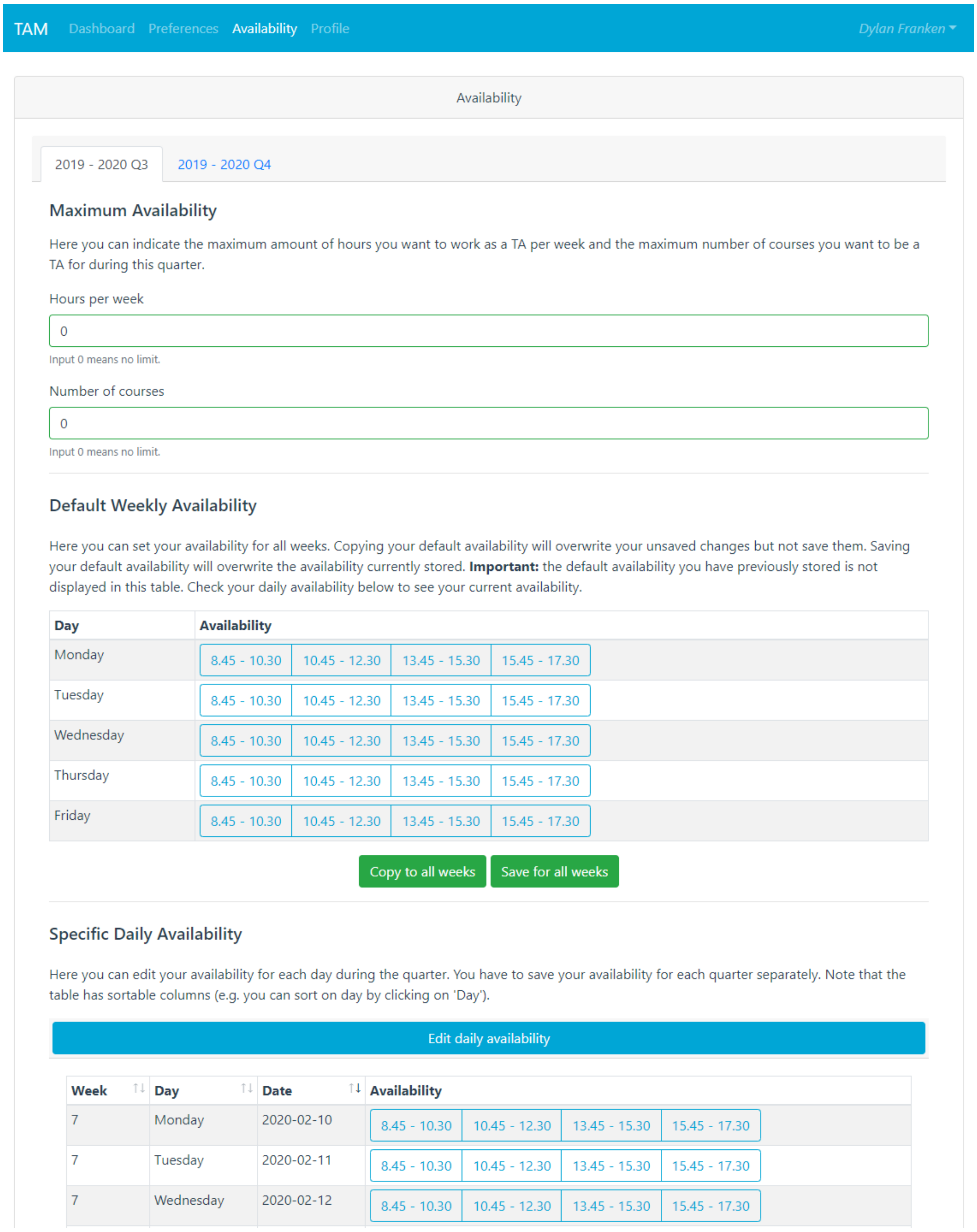
Course Preferences

Please pay attention to the following:

- Only indicate you want to assist a course if you:
  - passed the course yourself;
  - are available during the indicated education period (e.g. not doing a minor abroad etc.).
- This form is to express interest, it is not final in any way. Based on this we will contact your when we start creating the schedule for next semester.
- We will pass the list of interested students to the responsible teachers for the different courses. They have total freedom in selecting their TA's, both from this list and not from this list. Common selection criteria include, but are not limited to: course grade, performance during the lab, and advise from the academic counsellors regarding your study progress as a whole.
- For further questions please contact ta-cs-ewi@tudelft.nl.

Period	Year	Code	Note	Course	Do you want to be a TA?
2019 - 2020 Q3	1	CSE1105	? We can also consider your OOP grade during the selection procedure.		No [✓ Yes] Yes, and I've been TA for this course before
2019 - 2020 Q3	1	CSE1505		Information and Datamanagement	No Yes [✓ Yes, and I've been TA for this course before]
2019 - 2020 Q3	1	TI1520AM	?	Algorithms and Data Structures (for Mathematicians)	No [✓ Yes] Yes, and I've been TA for this course before
2019 - 2020 Q3	2	CSE2230	?	Multimedia Analysis	[✓ No] Yes Yes, and I've been TA for this course before
2019 - 2020 Q3	2	CSE2530		Computational Intelligence	No Yes [✓ Yes, and I've been TA for this course before]

Figure C.3: The course preferences page allows students to indicate that they are interested in being a TA for a course, and whether they have any experience TAing for that course. If a course manager has attached a note with additional information to a course, students will be able to read them here.



Availability

2019 - 2020 Q3 | 2019 - 2020 Q4

Maximum Availability

Here you can indicate the maximum amount of hours you want to work as a TA per week and the maximum number of courses you want to be a TA for during this quarter.

Hours per week

0

Input 0 means no limit.

Number of courses

0

Input 0 means no limit.

Default Weekly Availability

Here you can set your availability for all weeks. Copying your default availability will overwrite your unsaved changes but not save them. Saving your default availability will overwrite the availability currently stored. **Important:** the default availability you have previously stored is not displayed in this table. Check your daily availability below to see your current availability.

Day	Availability			
Monday	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30
Tuesday	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30
Wednesday	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30
Thursday	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30
Friday	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30

Copy to all weeks | Save for all weeks

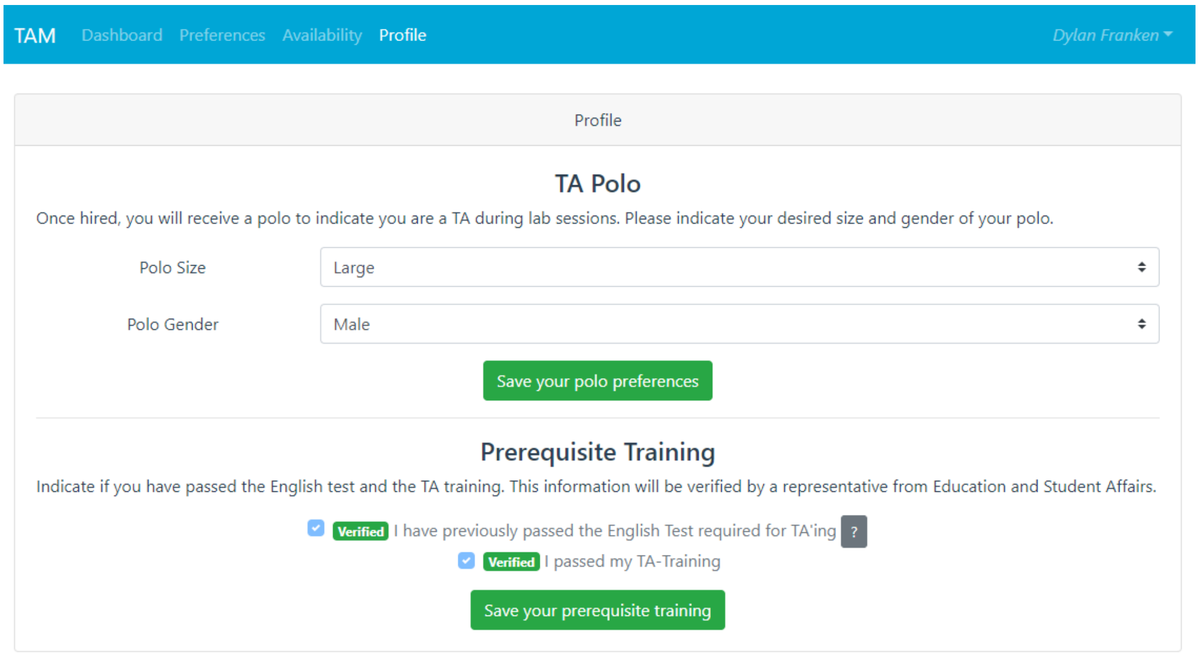
Specific Daily Availability

Here you can edit your availability for each day during the quarter. You have to save your availability for each quarter separately. Note that the table has sortable columns (e.g. you can sort on day by clicking on 'Day').

Edit daily availability

Week	Day	Date	Availability			
7	Monday	2020-02-10	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30
7	Tuesday	2020-02-11	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30
7	Wednesday	2020-02-12	8.45 - 10.30	10.45 - 12.30	13.45 - 15.30	15.45 - 17.30

Figure C.4: Students can indicate their availability by selecting timeslots during which they can TA from a fixed list of four timeslots. In addition, students can enter the maximum number of hours they want to work per week, and the maximum number of courses they would like to TA at a time.



TAM Dashboard Preferences Availability Profile Dylan Franken

Profile

TA Polo

Once hired, you will receive a polo to indicate you are a TA during lab sessions. Please indicate your desired size and gender of your polo.

Polo Size	Large
Polo Gender	Male

Save your polo preferences

Prerequisite Training

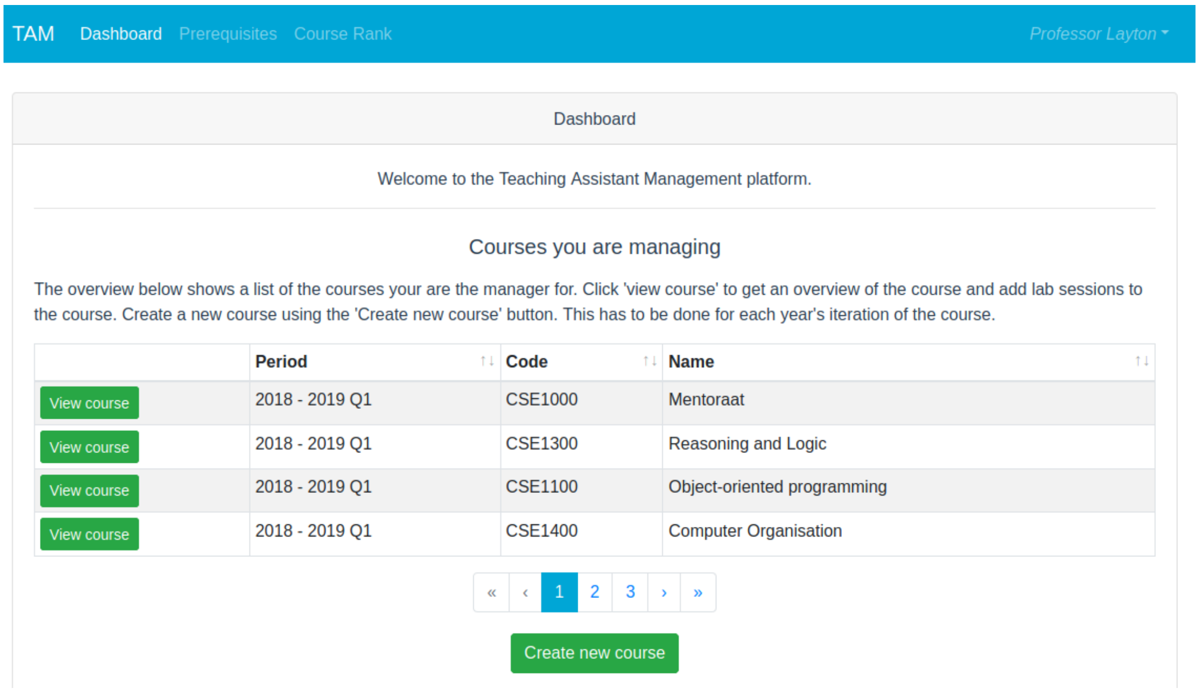
Indicate if you have passed the English test and the TA training. This information will be verified by a representative from Education and Student Affairs.

✓ Verified I have previously passed the English Test required for TA'ing ?
✓ Verified I passed my TA-Training

Save your prerequisite training

Figure C.5: On this page, students can enter what polo they want to have, and whether they have completed the prerequisite TA training and English test.

C.2. Course manager view



TAM Dashboard Prerequisites Course Rank Professor Layton

Dashboard

Welcome to the Teaching Assistant Management platform.

Courses you are managing

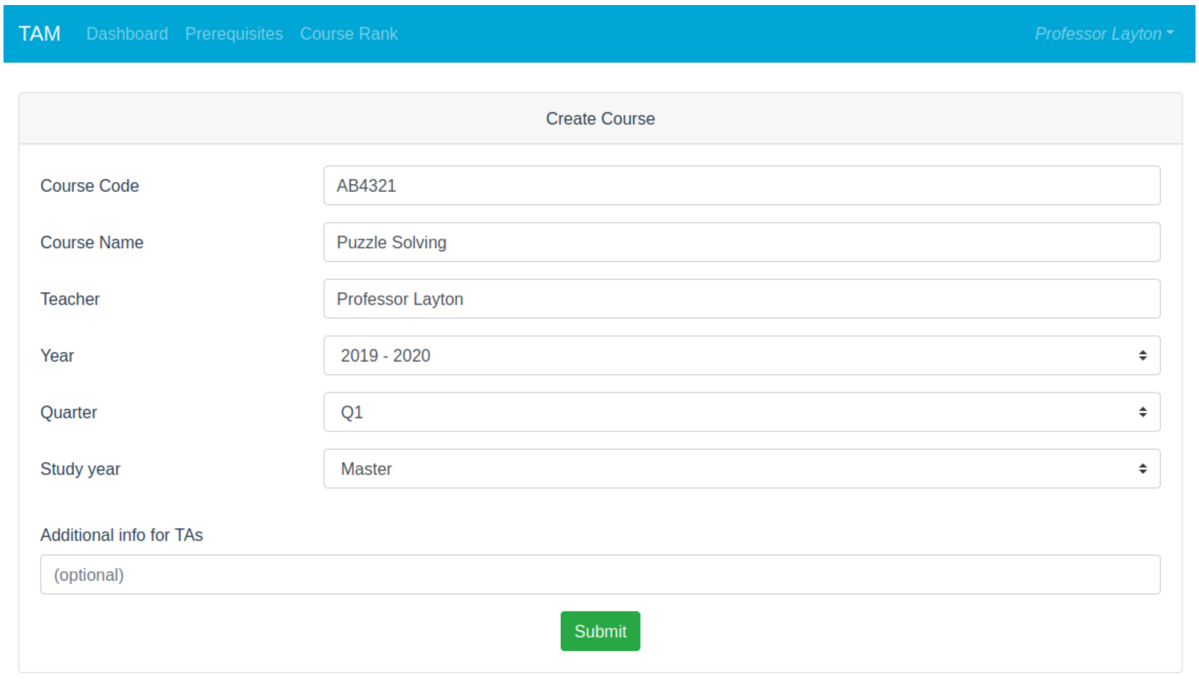
The overview below shows a list of the courses your are the manager for. Click 'view course' to get an overview of the course and add lab sessions to the course. Create a new course using the 'Create new course' button. This has to be done for each year's iteration of the course.

	Period	Code	Name
View course	2018 - 2019 Q1	CSE1000	Mentoraat
View course	2018 - 2019 Q1	CSE1300	Reasoning and Logic
View course	2018 - 2019 Q1	CSE1100	Object-oriented programming
View course	2018 - 2019 Q1	CSE1400	Computer Organisation

« ‹ 1 2 3 › »

Create new course

Figure C.6: This is the dashboard for course managers. They are presented with a list of courses they manage. Managers can add a new course, or edit ones they manage.



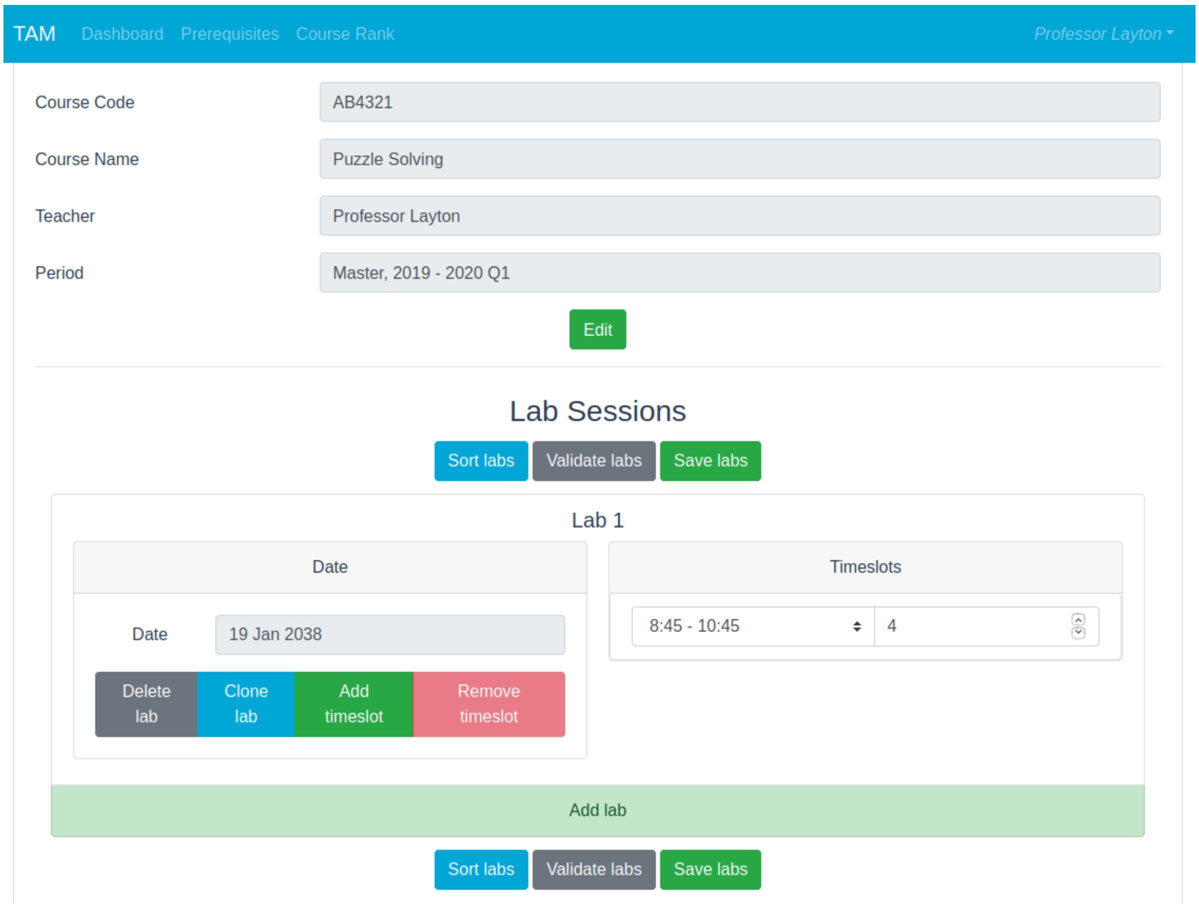
Create Course

Course Code	AB4321
Course Name	Puzzle Solving
Teacher	Professor Layton
Year	2019 - 2020
Quarter	Q1
Study year	Master

Additional info for TAs

(optional)

Figure C.7: This screen is used to create new courses, but also to edit existing courses. The additional info text is displayed to students on the page in figure C.3.



Course Code	AB4321
Course Name	Puzzle Solving
Teacher	Professor Layton
Period	Master, 2019 - 2020 Q1

Lab Sessions

Lab 1

Date

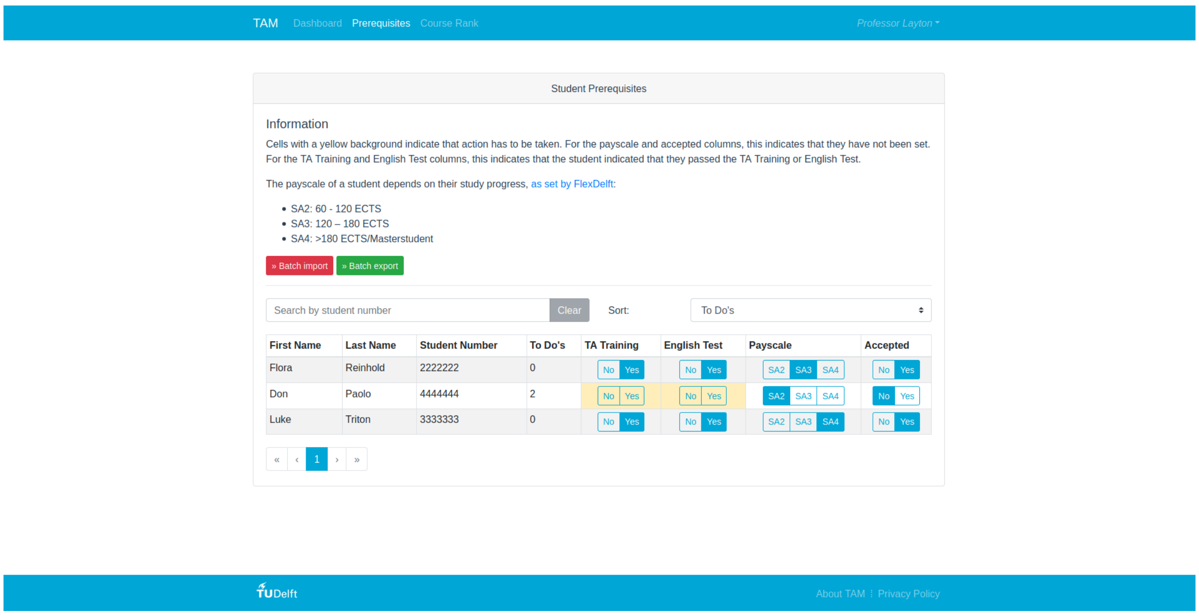
Date 19 Jan 2038

Timeslots

8:45 - 10:45 4

Figure C.8: Course managers could add lab sessions to their course here.

C.3. Verifier view



Student Prerequisites

Information

Cells with a yellow background indicate that action has to be taken. For the payscale and accepted columns, this indicates that they have not been set. For the TA Training and English Test columns, this indicates that the student indicated that they passed the TA Training or English Test.

The payscale of a student depends on their study progress, as set by FlexDelft:

- SA2: 60 - 120 ECTS
- SA3: 120 – 180 ECTS
- SA4: >180 ECTS/Masterstudent

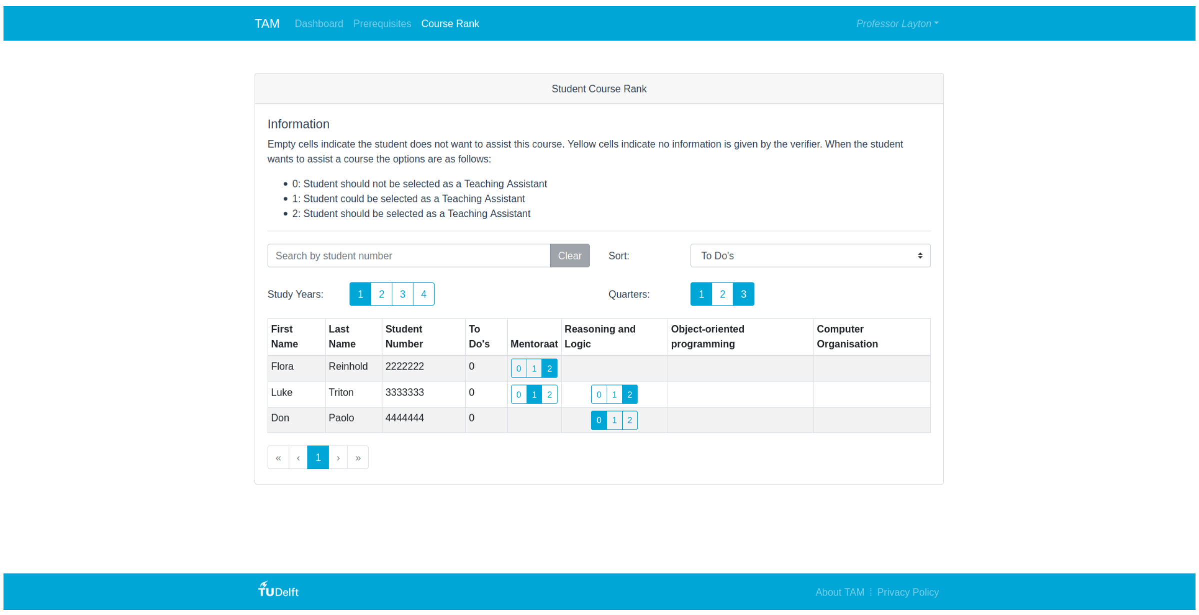
» Batch import » Batch export

Search by student number | Clear | Sort: To Do's

First Name	Last Name	Student Number	To Do's	TA Training	English Test	Payscale	Accepted
Flora	Reinhold	2222222	0	No [✓ Yes]	No [✓ Yes]	SA2 [✓ SA3] SA4	No [✓ Yes]
Don	Paolo	4444444	2	No Yes	No Yes	[✓ SA2] SA3 SA4	[✓ No] Yes
Luke	Triton	3333333	0	No [✓ Yes]	No [✓ Yes]	SA2 SA3 [✓ SA4]	No [✓ Yes]

« ‹ 1 › »

Figure C.9: This page lists students along with the completion statuses of their TA trainings and English tests. Cells with a yellow background have been set to ”completed” by the student, but still need to be verified. Verifiers should also set the pay scale of each student here.



Student Course Rank

Information

Empty cells indicate the student does not want to assist this course. Yellow cells indicate no information is given by the verifier. When the student wants to assist a course the options are as follows:

- 0: Student should not be selected as a Teaching Assistant
- 1: Student could be selected as a Teaching Assistant
- 2: Student should be selected as a Teaching Assistant

Search by student number | Clear | Sort: To Do's

Study Years: [✓ 1] 2 3 4 Quarters: [✓ 1] 2 [✓ 3]

First Name	Last Name	Student Number	To Do's	Mentoraat	Reasoning and Logic	Object-oriented programming	Computer Organisation
Flora	Reinhold	2222222	0	0 1 [✓ 2]			
Luke	Triton	3333333	0	0 [✓ 1] 2	0 1 [✓ 2]		
Don	Paolo	4444444	0		[✓ 0] 1 2		

« ‹ 1 › »

Figure C.10: The course rank page allows verifiers to assign a value to a student for each course they are interested in. The values were to be used by a scheduler, with 0 meaning that a student should not TA for a course, 1 that a student could TA for a course, and 2 meaning that a student is a preferable TA candidate.

Bibliography

[1] Max van Deursen, Max Pigmans, Geert Habben Jansen, and Ruben Keulemans. Teaching assistant management platform: Automating the recruitment and scheduling of teaching assistants, Jul 2018. URL `http://resolver.tudelft.nl/uuid:c8939ef4-40d8-49a9-9a2b-47a9d7ef7e24`.