Schedule Optimization

Contents

1 Main Text

The TimeTracker platform is an algorithm-enabled tool designed to assist students with work planning and general scheduling. The platform outputs a schedule of times to work on specific assignments based on their priority and student-specific preferences. This schedule serves as an alternative time management tool that each student can use to plan their required work. The provided schedule allows students to focus on their studies instead of creating a daily study plan. In addition to the student focus, TimeTracker provides insights for each department such as a machine-learning powered tool that creates a list of high-performing students that are more likely to succeed in a teaching assistant role or summer opportunity.

The main objective of the TimeTracker home page is to explain the features of the calendar and create easy pathways to log in and sign up. The website design utilizes a dark blue and white color scheme, which was chosen since the colors are not distracting to the users and generally represents positive emotions. The home page also includes a description of the three main competencies of the website: work-time scheduling, student-professor feedback, and algorithmic scheduling. This outlines the main features of the site which should accelerate the learning curve to achieve full understanding of the interface.

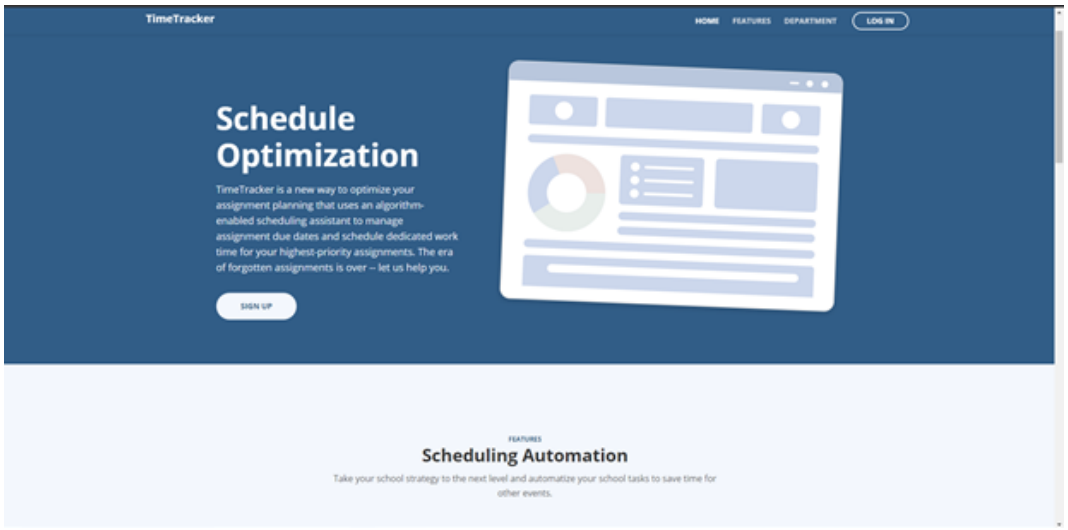

Figure 1: Website Homepage

The stakeholders of this online tool are current students and faculty at Purdue University, West Lafayette. Besides the homepage, each group accesses different portions of the website that serve stakeholder-specific needs.

Students can create and login to their account to view their personalized schedule. Since each student is added to their classes by the professor, the required steps for the student to receive a personal schedule is low. This personal schedule is the first page that appears after sign in and is intentional to streamline the process of each student checking their personalized schedule. The schedule is generated using an algorithm that uses specific student preferences such as preferred sleep time and duration. The student preferences, combined with the assignment information, are used to generate the personalized student calendar.

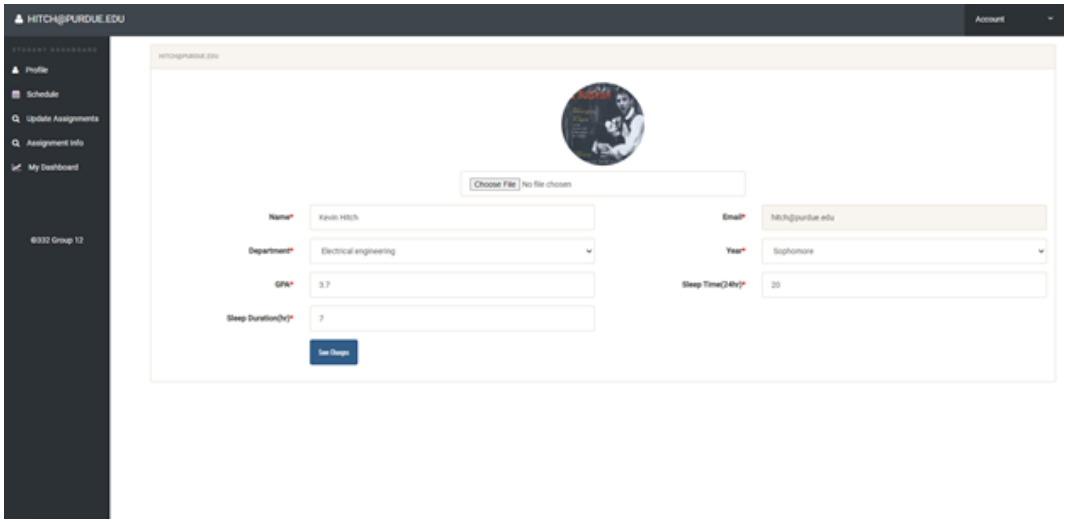

Figure 2: Student Preferences

The student calendar will optimize a work schedule based on due dates with the click of a button. After 'Optimize' is hit, the algorithm will create an ideal schedule based on the student preferences and assignment information. These work times can be easily dragged by the student to a new time if there is an issue with the proposed time. This feature was added to provide flexibility to students when scheduling work times. The schedule displays color-coded assignments by category. Exams are red, homework is black, project work is green, and lab work is blue. The color system provides an easy way to determine the assignment type and provides an additional visual method of separating assignments by category.

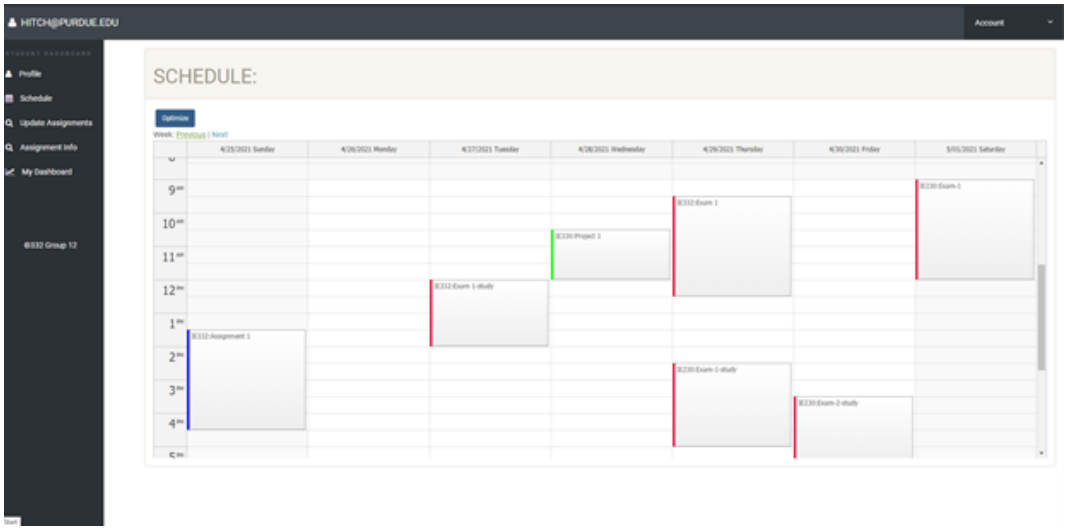

Figure 3: Sample Student Calendar

Additional important features are found in the 'Update Assignment' page. This page provides students a place to change their completion status, edit the actual time required to complete the assignment, or completely remove a particular assignment. The functionality of this allows the scheduling algorithm to remove time slots associated with completed or deleted assignments. The implementation of this feature was added to allow students to track the progress of their current assignments, help professors track the time spent by each student per assignment, and delete completed assignments.

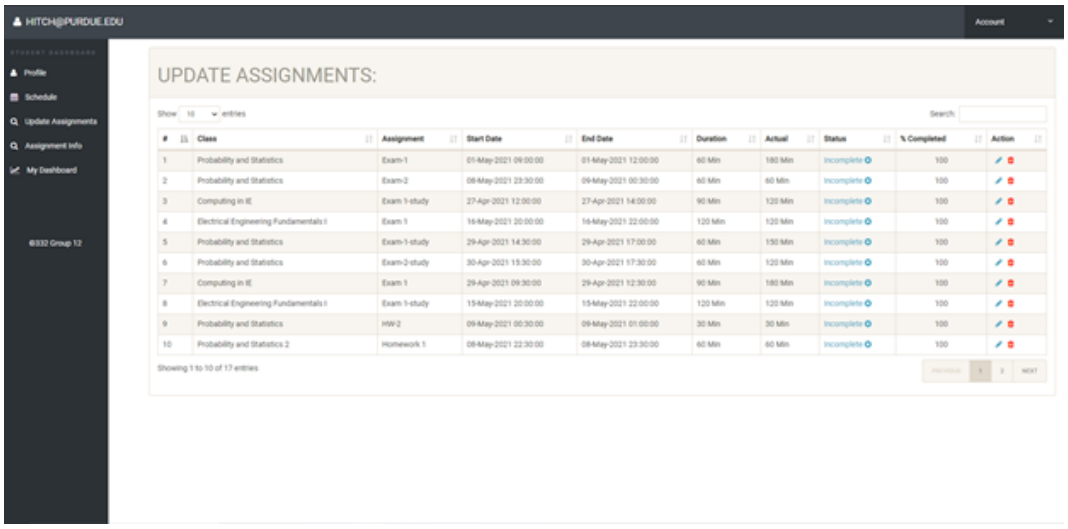

Figure 4: Update Assignments Page

Faculty at Purdue sign into their department page which provides access to web pages for adding a course, class-specific assignments, and students to courses. Assignments are inputted using the class name, assignment title, assignment description, start date, due date, and its estimated completion time. This website format allows the addition of courses as well which minimizes the required time to add any new courses offered at the university. https://www.overleaf.com/project/6082fd18ec8058064382d492

The addition and deleting of assignments allows faculty to input their entire assignment list at the start of the semester and make changes as necessary. The assignments are separated by class. The ease of adding or deleting an assignment provides a quick way for professors to update their assignment list.

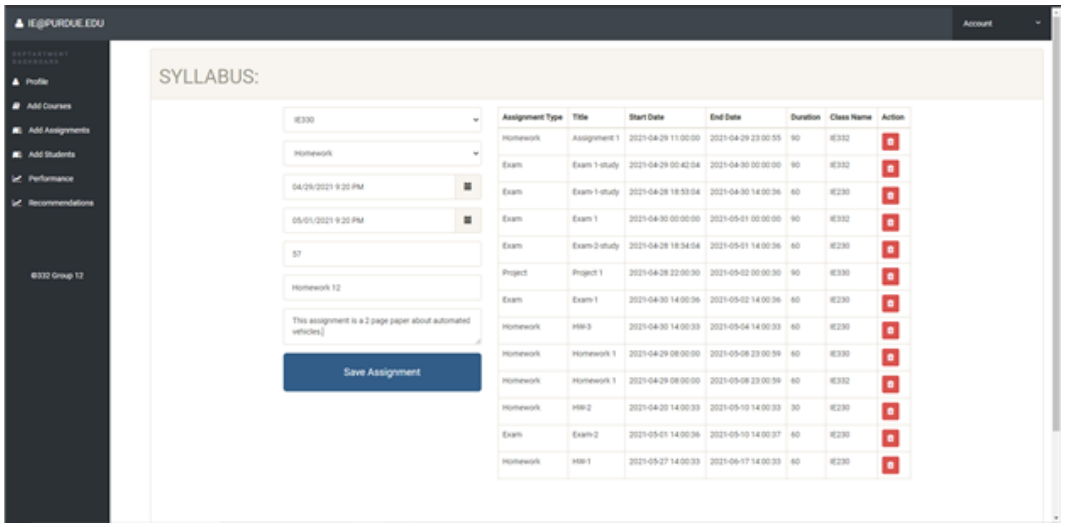

Figure 5: Add Assignments Page

The professors log into the website to add students to the classes they are taking within the Add Students page. After students, are added to each of their classes and at least some assignments are completed, the performance tab can be used to view the average time for a student to complete an assignment. This acts as feedback from the students about assignments. This display provides insights into the differences between perceived assignment length and actual length. The importance of this feature allows each professor to make

informed decisions about their workload provided to each student.

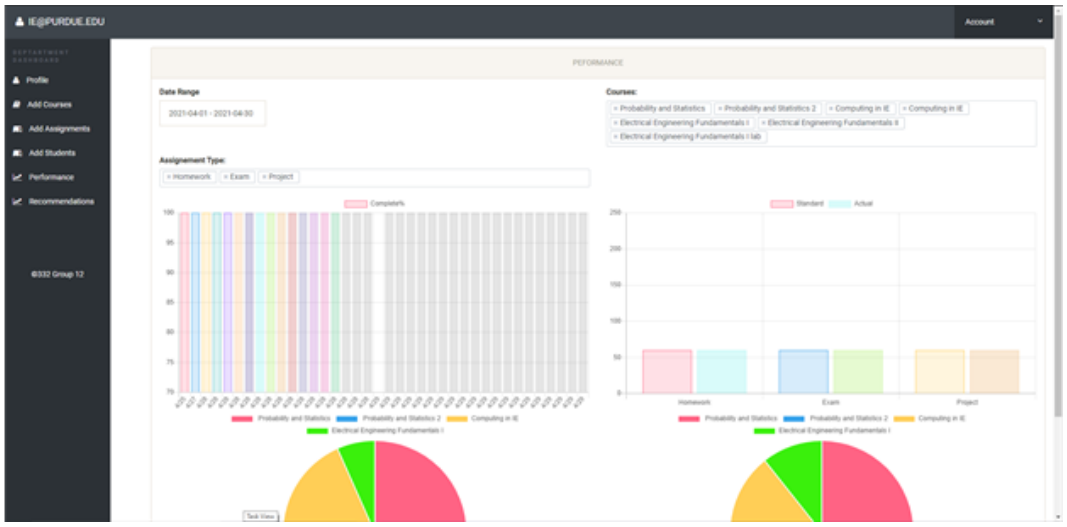

Figure 6: Recommendations Page

The machine learning aspect of TimeTracker utilizes clustering by class grades and the performance factor. The performance factor uses the estimated time of the assignment as a baseline and uses the true time taken on the assignment as a metric. A 'high-performing' student will complete work in a shorter amount of time than the given assignment. This factor affects the student's end score as does their overall grades in each course taken collectively. The clustering also uses GPA as an indicator of high performance.

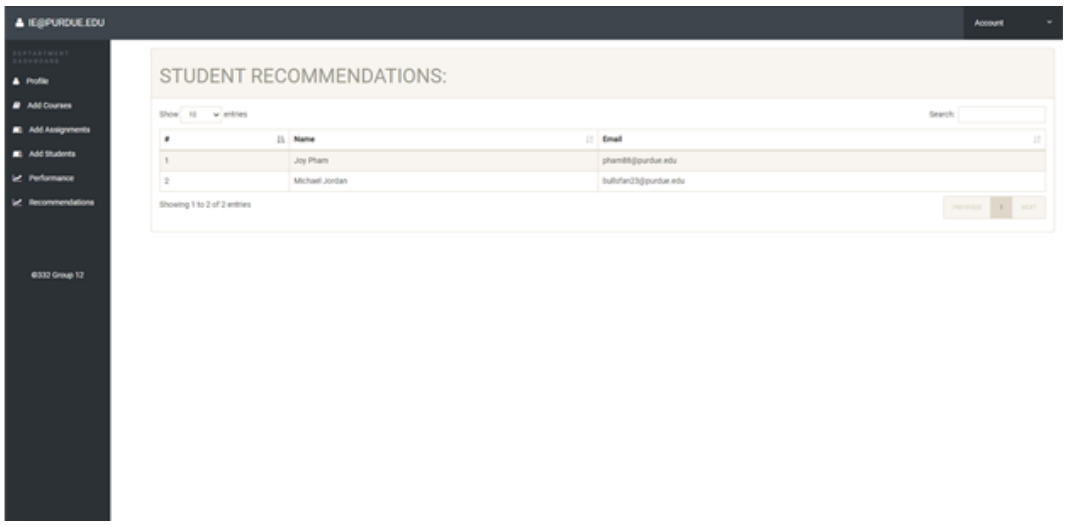

Figure 7: List of 'High Performing' Students using Machine Learning

In summary, TimeTracker provides an alternative approach for students to schedule work for required assignments, reduces the daily planning time for students, and identifies generally high-performing individuals for faculty. These tools provide students with a streamlined approach to scheduling that can provide value to the Purdue student body on a daily basis. In addition to the benefit to the student body, the current format of TimeTracker is simple to navigate. This leads to fast adoption and understanding of the format even though the practicality of the tool is quite high. This solution provides strong value to both stakeholders in a clear and meaningful way.

2 Appendix

2.1 Interface Design

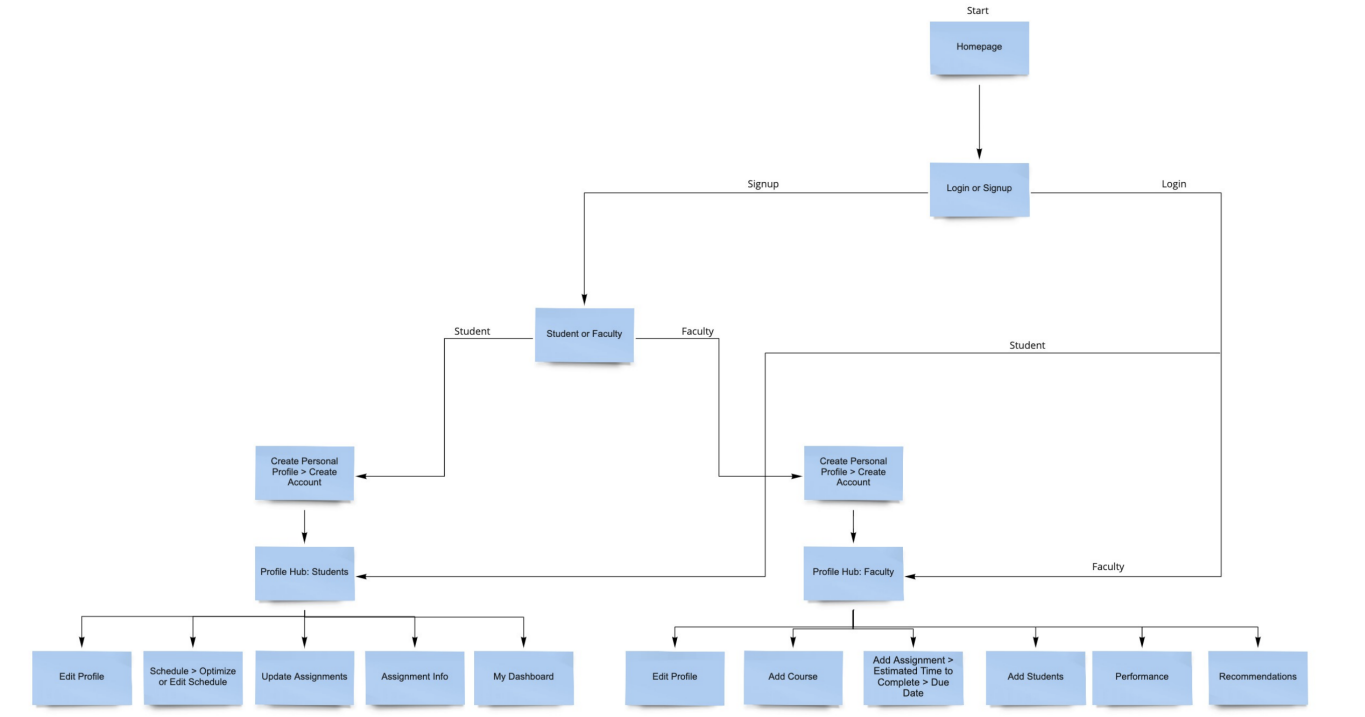

Figure 8: TimeTracker Website Map

TimeTracker's interface design was focused on three main elements: usability, error prevention and overall user satisfaction.

2.1.1 Usability

TimeTracker's interface design is focused on three main elements: usability, error prevention and overall user satisfaction. The usability aspect of this website can be broken down into learn-ability and user efficiency. To increase the users' abilities to navigate TimeTracker with ease, the website was designed to be as simplistic as possible without leaving out any important information. To assist with the learn-ability aspect, the design of this website focuses on grouping relevant information together with clear separation of unlike information based on Christopher Wickens' proximity compatibility principle (Wickens, 1995).

The layout of TimeTracker begins with a homepage, displayed in figure 1, that identifies the purpose of the website and provides users with guidance on their next steps, prompting them to sign up or log in. This conventional design allows users to efficiently utilize the website as if it were a site they used daily. Once the user accesses their account, a dashboard is displayed on the left to allow users to navigate to all the website's key features. This dashboard is always displayed in the same location on the website to allow for users to switch between features regardless of the page in which the user is located.

The calendar feature for students displays their optimal schedule, shown in figure 3, in accordance

with the estimated duration of the department created assignments and student created personal events. Within this schedule interface design, the student user is able to physically place personal events of 30 minute intervals by simply selecting the desired portion of the calendar. This direct manipulation design was created to allow users to easily learn how to create and manage their personal schedules with less errors, more satisfaction and a higher use of recognition memory (Soegaard, 2015).

Another important and underestimated factor in the interface design of TimeTracker is the color scheme which is displayed in figure 1 as well. This color scheme was designed to be user influenced with the primary colors of dark blue, dark grey and white. Blue is the most used color in interface design as it is an overall statistically favorite color that generally represents positive emotions (Potvin, 2019).

2.1.2 Error Prevention

Despite how well-designed an interface may be, humans will make mistakes and other errors, but preventing human error can be achieved by focusing on the interface design. TimeTracker has features in place to help protect from error creation. A specific error prevention method in place is the JavaScript action confirmation requirements. In order to complete, edit, or delete an assignment within the update assignment feature in the student dashboard, the user has to confirm this task or has the option to cancel this action. This same feature is employed on the department profile. This is to ensure mistakes do not happen when editing the courses, assignments or student portion of this website as this could impact many users negatively if no error prevention were to be present.

Another way to alleviate mistakes and frustration is to design our system with "emergency exits" (Langmajer, 2019). We demonstrate this design factor in our log in screen to enable unfamiliar users to switch to their appropriate registration page through many paths. This design practice is also demonstrated throughout both the student and department dashboards located on the left shown in figures 3 and 5.

2.1.3 User Satisfaction

A significant part of user satisfaction is clear task feedback. Humans have a need to be informed and in control of a situation in order to feel secure (Langmajer, 2019). To incorporate system feedback, TimeTracker initially has an icon animation to show you the website is loading. This is practiced to ensure the user knows the website is functional and prevent further interactions during the page load time. Furthermore, other implementations of user feedback is displayed when creating an account, the user is informed the creation was either successful or not. When moving an assignment throughout the user's schedule, the student is informed of a successful or unsuccessful move. A similar set of feedback is shown when simply changing the user's password.

2.1.4 Largest Challenge

The most challenging aspect faced within website design was how we should present the calendar. This task proved to be the most complex and crucial design aspect of the website. As a team, we needed to overcome this challenge as it is one of the main element within our interface functionalities.

How it was overcome

At first, the team outlined what features were necessary and desired to understand how our initial idea of the website schedule function would be outlined. A clearer picture of the desired functionality of the calendar allowed the team to better portray what the design of the calendar would entail. After more than four iterations, the view of the calendar was finalized. The team had decided that we would ideally have a visually pleasing calendar that could also be easily manipulated, similar to current calendar functions that portrayed weekly task assignments.

What we learned

The team learned that design is an iterative process that is made much easier when the functionality of the element is fully understood. This iterative thought process proved to be the information that allowed our team to create final view of the calendar. We initially began with a calendar function within the website that would focus on the week at hand for the students, but quickly found that for this to be of high quality to provide high user satisfaction, we would need to allow students to look into their future and past weeks of their overall semester.

2.1.5 Second Largest Challenge

In order to provide the highest quality of experience, the TimeTracker team found it essential to present a consistent website throughout its functionality and displays. This would include consistent language syntax, presentation and error prevention. Consistency throughout an entire website becomes more challenging the further the project grows. This required the team to continuously ensure the proper and consistent use of documentation and functionalities were being performed.

How it was overcome

As a group, we regularly had multiple team members compare the website's functionality and display methods to ensure proper language syntax, grammar and overall functionality for error prevention dialog boxes, data entry/validation and more. We also employed a high level usability test by having novice users that are not enrolled in this class move through the website to pinpoint any issues that our developing team might not have been able to identify.

What we learned

Through this challenge of interface design, our group learned that the consistency throughout a website can be extremely hard to perfect. This must be a portion of design that is continuously monitored or else it could become an overwhelming challenge to solve toward the end of website development. To perform this aspect of interface design, one must stay on top of reviewing the website for these issues mentioned previously and create syntax and functionality rules for the team in order to maintain proper website consistency.

2.2 Database Design

2.2.1 Entity Relationship Diagram

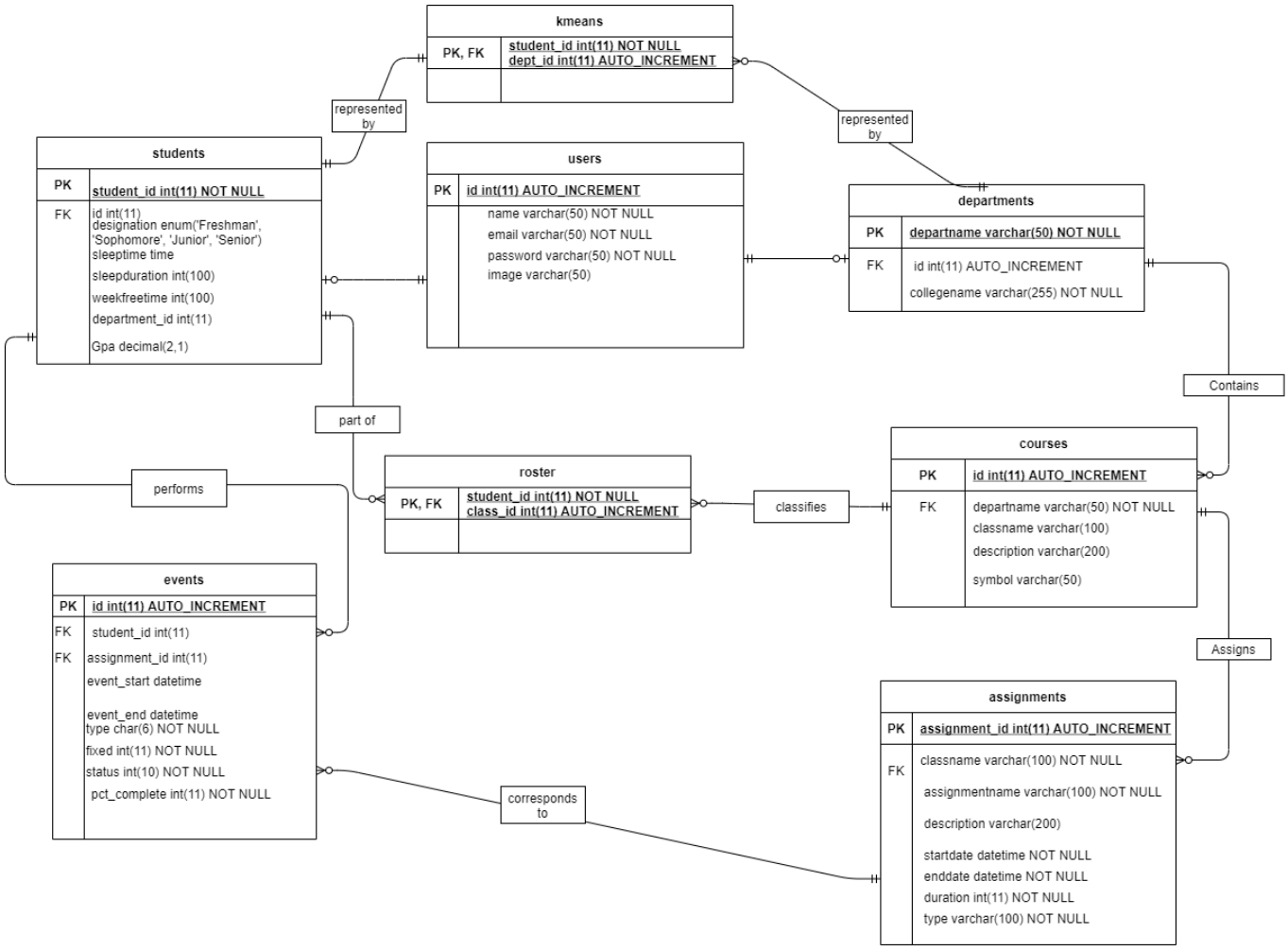

Figure 9: TimeTracker Entity Relationship Diagram

- Each department contains one or more courses. This is a one to many relationship. This is correct because a department must have the ability to have multiple courses within it.
- Each department is represented by one or more kmeans entries. This is a one to many relationship. This is correct because each entry of the kmeans table shows a student and which department they are in. This means there can be multiple entries for each department, but each entry only has one department.
- Each department corresponds to exactly one user. This is an isa relationship. This is correct because each department only has one username to log in.
- Each course is classified by one or more rosters. This is a one to many relationship. This is correct because the roster table shows the students that are paired with each class id. There are multiple students taking each course, but each roster entry is only related to one course.

- Each course assigns one or more assignments. This is a one to many relationship. This is correct because each course can have multiple assignments at a time, but each assignment is only in one course.
- Each assignment corresponds to one or more events. This is a one to many relationship. This is correct because each assignment can be completed in multiple events, however each event only corresponds to one assignment.
- Each student 'performs' one or more events. This is a one to many relationship. This is correct because each student can have multiple events on their calendar, but each event is only for one student.
- Each student is part of one or more rosters. This is a one to many relationship. This is correct because the roster shows the class id for each course the student is taking. There are multiple courses for each student.
- Each student is represented by one kmeans entry. This is a one to one relationship. This is correct because the kmeans table is simply a list of which department each student is a part of.
- Each student corresponds to exactly one user. This is an isa relationship. This is correct because each student must be a user, but not every user is a student.

Each entity relation was carefully examined to ensure there were no issues with data redundancy, no normal form rules broken, and no issues with data integrity. Each column in each table is used for the functionality of the website, machine learning, and/or scheduling algorithm. Each primary and foreign key are used to connect the tables in the database and provide strong data integrity.

2.2.2 Largest Challenge

The top technical challenge faced by our team was ensuring that the relationships between the tables did not violate the database creation rules. We had many issues with our original ERD. Specifically, the Foreign Keys and the relationships between the tables were not connected properly. In our original ERD, we had a few tables that had relationships, but did not have foreign keys to connect them. After speaking with our teaching assistant about the issues, we had a better understanding of where our ERD had errors and how we could correct them.

How it was overcome

To overcome this issue, we spent time looking at each variable in our tables and where it should connect to other tables. This showed us which tables needed a foreign key and what the relationships should be between them. After speaking with out TA about the issues, we had a better understanding of where our ERD had errors and how we could correct them. To test these relationships, we used mock data to think about how the relationships work and if the relationship did not make sense given the data.

What we learned

From this challenge, we learned that the relationships between tables in databases are complex but extremely important. We learned that it is crucial to critically think about how tables relate and what keys must be used.

2.2.3 Second Largest Challenge

Our second technical challenge was connecting the database to the website. We had some issues with ensuring that the data was able to be inserted and selected from the tables. This is extremely important as the functionality of the website would not work without a proper connection.

How it was overcome

To overcome this issue, we examined the code and compared the statements with labs and homework to ensure every line was correct.

What we learned

We learned that one small error in the syntax of PHP and SQL code can cause major issues with the functionality of the code and website as a whole. This small error can result in a large chunk of time lost.

2.3 Optimization Algorithm

2.3.1 Mathematical Model (refer to www/lp-2 for a working linear programming version of the Mathematical Model)

i, j: assignment, $i, j \in \mathbb{N}$
s_i: actual start time for assignment i
e_i: actual end time for assignment i
d_i: recommended duration for the assignment(by department)
M: a large number used to prevent assignment overlapping
p_i: priority of assignment i (higher priority has greater p value)
b_i: release time assignment i (by department)
f_i: deadline of assignment i (by department)
T_i: absolute difference between the estimated duration by the department and the optimized duration by the schedule
$t_{i,j}$: is 1 if assignment i is placed before j in the calendar, is 0 otherwise

minimize

$$\sum_i T_i$$

subject to

Minimum and absolute value constraint

1. $$T_i \geq 0 \qquad \forall i$$

2. $$T_i \geq p_i(d_i - (e_i - s_i)) \qquad \forall i$$

Avoid overlapping constraint t_{ij} is 1

1. $$s_i - e_j \geq -M(t_{i,j}) \qquad \forall i, j$$

2. $$s_j - e_i \geq -M(1 - t_{i,j}) \qquad \forall i$$

Start and end time constraints

1. $$s_i \geq b_i \qquad \forall i$$

2. $$e_i \leq f_i \qquad \forall i$$

Binary constraint

1. $$t_{ij} \in \{0, 1\} \qquad \forall i, j$$

2.3.2 Pseudocode

```
///gets the data
Assignment_id = get(Assignment_id)
Assignment_start = get(Assignment_id)
Assignment_end = get(Assignment_end)
Assignment_duration = get(Assignment_duration)
////////
///define output array
final_start = array[length(Assignment_id)]
final_id = array[length(Assignment_id)]
final_end = array[length(Assignment_id)]
// define demand in which which will show how many intervals are active a specfic section
demand  = array[(Assignment_start - Assignment_end)  /1800]//1800s = 30 min
// calculating where assignments are most active
for(i in demand) {
for(l in Assignment_start){

        if(Assignment_start[l] > demand[i] * 1800 && Assignment_end[l] < demand[i] * 1800){
            demand[i] += 1
        }}
// assigning assignments
max = sum(demand)
for (i in Assignment_start){

    result = findbest(Assignment_start[i]/1800, Assignment_end[i]/1800,
        demand, Assignment_duration /1800)

    // if no space was found for a given duration, lower durations are tried.
    for(l in duration - 1 & sum(result[1:length(result)] > max) {
        result = findbest(Assignment_start[i]/1800, Assignment_end[i]/1800,
            demand, (Assignment_duration -1) /1800)}
    // adds data to output arrays
    final_start = append(result,Assignment_id[i]
    final_id = append(result,result[1]
    final_end = append(result,result[2]}
function findbest(interval_start, interval_end, demand, duration){
    best_int_start = interval_start
    best_int_end = interval_start + duration
    min = sum(demand)
    // finds the interval with lowest demand
    for(i in demand - duration){

       if( sum (demand[i:i + duration]) < min){
            min = sum (demand[i:i + duration])
            best_int_start = i + interval_start
            best_int_end = i + interval_start + duration
       }}
```

```
    // returns the result
    result = array[2]
    result[0] = best_int_start
    result[1] = best_int_end
    return(result)
}}
```

2.3.3 Our Strategy

Our definition of the scheduling problem is to build a schedule that can accept a semester's worth of assignments and build an ideal model for when those assignments should be completed. The algorithm begins by calling a query to collect the incomplete assignment data. This includes: assignment ID, start date, end date, duration, student ID, type, and sleep. Once the algorithm has ended, it deletes all the previous data in events table and inputs the newly optimized data instead.

Our algorithm applies the best strategies to place events/assignments into a student's calendar. We decided to approach the problem with a greedy algorithm, which was chosen for a number of reasons. First, it runs much faster than alternative linear programming approaches. We have to account for potentially hundreds of assignments being sorted at the click of our optimize button, on the student calendar page. Making sure it runs quickly was a priority of ours. The next reason we chose the greedy algorithm was that it is more simplistic in nature. We tried a variety of different algorithms, but found that our final algorithm can handle the most assignments with minimal lost assignments. I will cover why some assignments are lost later on in our challenges and also in our test case section.

Our algorithm applies multiple strategies to place the assignment in the optimal location. It begins by converting the duration, start date, and end date all into seconds, which alleviates the problem of dealing with conversions between minutes and hours. Then, it divides the time between the earliest start date and latest due date into 30 minute interval. The next step is checking which time intervals are the most cluttered and filled with overlapping assignments. The greedy algorithm runs through assignments with the soonest highest priority first, then assigns them based on the demand. The most active areas will be avoided, because we do not want to take that time interval away from the next few assignments, which might need that time slot. Students are allowed to input sleep preferences, including the start and end time. These intervals are reserved for sleep, but line 136 in our R file does allow events to overflow into sleep time if no other times are available. The assignments are valued as integers, which represent the amount of intervals, 30 minutes, needed/recommended to complete it. Once the best intervals are found for the assignment, the algorithm chooses to occupy those and remove them from future assignments.

Our algorithm will only work on assignments that meet our assumptions and constraints. The parameters that the algorithm receives from our database are: assignment ID, start date, end date, duration, and sleep time. These are all used to prioritize the assignments and organize them so the greedy algorithm can run through the soonest assignments first. One constraint is assignments cannot be added if the start date is after the due date. This would potentially break the algorithm, so we made it a constraint and prevented the departments from inputting assignments that would violate this. Another constraint is assignments cannot be uploaded as events if they will not be completed before the deadline. This is to prevent events from appearing after their deadline. We also assume that assignments with the highest priority will run through the greedy algorithm first. This means that assignments that are more important, based on the type, will be made into events before those will lower priorities. We prioritize the types in descending order as: exams, homework, labs, projects, with the reasoning that exams and homework are individual

assignment, while labs and projects are usually assigned in groups. Our last assumption is sleep preferences must be between the hours of 15 - 24. The sleeping duration setting is limited to 1 - 12 hours.

Algorithm file: N1/students/ex.R (lines 1 - 65 and 92 - 227)

Our algorithm has a worst case upper tight bound of $O(n^3)$ as shown by the three nested loops in pseudocode above. However, running three nested loops only occurs under extreme cases as the algorithm can perform a good optimization only using two.

2.3.4 Largest Challenge

We encountered many challenges in the creation process of our algorithm, which led to constant rewriting and taking different approaches to the problems. A big challenge we faced was fitting all the assignments into the events table without exceeding the deadlines. The problem arises when we have a long assignment duration and early deadlines, with lots of assignments. The result was the loss of a few assignments, which would not make it into the schedule on our website. The algorithm does this because it will not create an event with a duration that exceeds the deadline. The resulting assignments that cannot fit will be lost and the student will not be able to see them on their schedule. This a big problem.

How it was overcome and what we learned

We approached this in many ways, but we did not want to dismiss the integrity of the algorithm. What we mean by this is, the algorithm should not place events in the schedule after their due date. If this was allowed, then we would be placing assignments into time slots that are past the due date which is not ideal and would result in late submissions. In a real situation, this would not happen because it would require every 30 minute interval to be filled with an event from the start time till end time. Under testing cases, this could cause problems but does not pose real issues for real cases. Our solution to this was to allow assignments to overflow into sleep time if absolutely necessary. Assignments stay away from occupying sleep time, but can overflow if there is no available space until the deadline.

2.3.5 Second Largest Challenge

The other main challenge we faced was assignments overlapping and occupying the same intervals. This is a problem because there will be overlap in the events table and students cannot do two assignments at the same time. We want to alleviate stress on the student, and having the wrong exam time is a problem.

How it was overcome

The problem was fixed by making sure to apply R functions like floor and ceiling to the start times and end times. An assignment could take 39 minutes for example, but would require 2 intervals. This is done by applying ceiling to the start time and floor to the end time. This rounds the start time up and the end time is rounded down. The allocated time would be 60 minutes, 2 intervals, which helps organize the calendar and provide extra time for the student to complete the assignment if necessary. Breaking up the assignments into intervals worked with our layout of the schedule and also helped solve the problem of overlapping assignments. We ensured that a particular interval cannot be occupied by the 2 or more assignments.

What we learned

Something we learned about is that rounding the times up or down can help with the separation of assignments. It is a useful strategy that allocates the student with, at least, the necessary time to complete the assignment.

2.4 Testing

In this section, testing processes of both the software and the scheduling algorithm are shown in tables to serve a more precise demonstration. Each table below has seven columns, including the test number, the scenario to be tested, the steps to execute the test, the input (data) used to test (if necessary), expected output, the real generated output from the test, and the result of the test (P/F, standing for Pass or Fail). Both successful and failed tests are documented to fully illustrate the processes. Basic criteria of the database connection from the interface and the scheduling algorithm for a "ready" website were ensured to eventually pass the test if failed. On the other hand, algorithm may or may not pass some extreme cases and testing of those scenario may be stopped with the agreement that the algorithm has met the original goal of efficiency. Test case numbers accompanied by an asterisk are tests that were repeated from failed tests.

2.4.1 Testing software procedures

Since a template was found and modified for the website interface, most of the tests done on this part were to ensure the ability to send all data entered by used to the server and saved correctly under the designated tables.

Test #	**Scenario**	**Steps**	**Input (data)**	**Expected output**	**Actual output**	**Result (P/F)**
1	Input assignments are saved to database	1. Login to a department account 2. Navigate to *Add Assignments* tab 3. Select assignment type and enter other attributes	- IE@purdue.edu - 12345678 - IE332 - Homework - 2021-04-28 11:00 AM - 2021-04-30 11:59 PM - 30 - HW1 - Algorithm	Assignments are saved in database under table *assignments* as a row	As expected	P
2	Student enrollment is saved in database after being added by department	1. Navigate to *Add Students* tab in a department account 2. Select course by name and student by username	- muelle78@purdue.edu - IE332	Students' ids are saved with the corresponding course id under table *roster*	As expected	P
3	Modified working duration and time slots in schedule by students is updated in database	1. Login to a student account 2. Hit "Optimize" under tab *Schedule* 3. Drag to change duration of a assignment on schedule 4 Move assignment tab to a different location	- muelle78@purdue.edu - 123 - Drag IE332: HW1 - Check database	Start_event and end_event time under *events* table are changed for the specific student and assignment accordingly	Modification is not saved	F

4*	Modified working duration and time slots in schedule by students is updated in database	1. Implement a built-in function to detect the moved assignment's id and save it (code: www/N1/student/ calender/backend_move) 2. Same as test 3	- muelle78@purdue.edu - 123 - Drag IE332: HW1 - Check database	Start_event and end_event time under *events* table are changed for the specific student and assignment accordingly	As expected	P
5	Creating personal events	1. Click, hold, and drag an empty time slots on the schedule 2. Enter name of the personal event into prompt	- muelle78@purdue.edu - IE332 - Gym	Personal event is saved in database under table *personal_event*	Personal events created by students are not saved	F
6	All type of assignments (homework, project, lab, exam) are saved in database when created	1. Login to a department account 2. Add assignments from each type	- IE@purdue.edu - 12345678 - IE332-Homework - IE332-Project - IE386-Lab - IE330-Exam	Added assignments are saved in database under *assignments*	As expected	P
7	Editing assignment description	1. Login to a department account 2. Add assignments' description with different special characters	- IE@purdue.edu - 12345678	All assignment are saved under *assignments* table	Description with apostrophes will fail the query	F
8	Input enddate of any assignment must be after startdate	1. Login to a department account 2. Add assignments with startdate after end date	- IE@purdue.edu - 12345678 - IE332 - Homework - 2021-04-30 10:00:00 - 2021-04-29 23:59:00	Assignments are not saved to the table after clicking "Save Assignment"	Assignments are saved causing the algorithm to stop functioning	F
9*	Input enddate of any assignment must be after startdate	1. Add input validation (file: www/N1/ department/syllabus) 2. Same as test 8	- IE@purdue.edu - 12345678 - IE332 - Homework - 2021-04-30 10:00:00 - 2021-04-29 23:59:00	Assignments are not saved to the table after clicking "Save Assignment"	As expected	P

Table 1. Website test cases

2.4.2 Testing optimization algorithm procedures

Testing algorithm was executed by both utilizing test script and entering data manually from the interface. Since our algorithm is coded in RStudio, test script was created more easily with the "testthat" package. The tests shown in table 2 are the tests the algorithm has to pass to be deemed usable. After that, table 3 shows the extreme of edge cases our team attempted to make the algorithm pass to have a better understanding of its ability and limitations.

Test #	Scenario	Steps	Input (data)	Expected output	Actual output	Result (P/F)
1	Assignment are placed in the calendar between start date and end date	1. Add an assignment from the department account 2. Login to a student account 3. Clicking "Optimize" under *Schedule* tab	- IE@purdue.edu - 12345678 - muelle78@purdue.edu - IE332	No assignment is placed before its release and after its deadline	As expected	P
2	Assignment tabs on schedule never overlap	1. Add two assignment with same start and end date 2. Add enough assignment to exceeds the real number of time slots within a day	- IE@purdue.edu - 12345678	There is always at most one assignment in the calendar at any given time slot	As expected	P
3	Assignments are not placed in sleeping time	Add assignments which have periods between start date and end date that last over sleeping time	- IE@purdue.edu - 12345678 - IE332 - Homework - 2021-04-28 23:57:00 - 2021-04-30 23:57:00 - 30 - HW1 - HW1	No assignments are placed during sleeping hours	As expected, except for the night after the optimization	Partially P
4	Assignment having estimated duration of completion is longer than period between startdate and end date (test script: www/N1/student/test.R)	Add assignment with duration longer than the period between startdate and end date	- IE330 - Lab - 2021-04-27 11:20 AM - 2021-04-27 11:45 AM - 60	The assignment is automatically disappears when clicking "Save Assignment"	As expected	P
5	No assignment is dropped if there is space for it on the calendar	1. Create enough assignments with similar open period within a short time 2. Check on student's schedule by clicking "Optimize"	- IE@purdue.edu - 12345678 - muelle78@purdue.edu - 123	The number of input assignments from database is equivalent to the number of output events on the schedule unless there are no vacant time slots visible on calendar	Not all assignments from the department were placed into the student's calendar	F
6	No assignment is dropped if there is space for it on the calendar	1. Create enough assignments with similar open period within a short time 2. Check on student's schedule by clicking "Optimize"	- IE@purdue.edu - 12345678 - muelle78@purdue.edu - 123	The number of input assignments from database is equivalent to the number of output events on the schedule unless there are no vacant time slots visible on calendar	As expected	P

Table 2. Algorithm functionality test cases

Test #	Scenario	Steps	Input (data)	Expected output	Actual output	Result (P/F)
1	Assignments that have the same exact start, end date, and type (equivalent priority)	1. Login to a department account 2. Add 2 assignments with similar start date, end date, and type from two different courses in department account	- IE383, Homework, (4-29-2021 10:04 AM), 30, HW1 - IE330, Homework, (4-29-2021 10:04 AM), 30, HW1	Both assignments are placed in the schedule if there are vacant slots	One assignment is automatically dropped	F
2*	Assignments that have the same exact start, end date, and type (equivalent priority)	1. Change algorithm from 1 to 2 2. Same as test 1	- IE383-Homework-(4-29-2021 10:04 AM)-30-HW1 - IE330-Homework-(4-29-2021 10:04 AM)-30-HW1	Both assignments are placed in the schedule if there are vacant slots	As expected	P

Table 3. Algorithm extreme test cases

2.4.3 Largest Challenge

One of the biggest challenges our team encountered when testing the software was determining how to write a test script for algorithm testing. As described above, the algorithm receives input directly from the database and outputs the entire schedule with the exact start and end time for each specific assignment. As it is the nature of a scheduling problem, there are a large amount of possible ways that assignments are placed in the schedule, with the primary constraint that they are between the release date and the deadline. Therefore, identifying an expected output in the test script to be compared with the potential output from running the algorithm is difficult in two ways. First of all, even if the algorithm outputs and the identified output from the test script are both correct solutions, the test script has no ability to recognize that fact. Secondly, given a large amount of assignments with different release dates and deadlines, it is time-consuming and inefficient for our team to figure out a solution for the whole semester to code in the test script for comparison with the actual output from the algorithm.

How it was overcome

With the challenge being presented, our team agreed to brainstorm the extreme cases and input them from the website interface, once it was ready, to test each case manually. We wrote a test script (test.R, all lines) to test a more general aspect, which is the equivalency between the number of input assignments and the number of output events. From this test script, we were able to detect more cases when assignments are dropped from the schedule, and from there, continue to test them manually from the interface.

What we learned

One case that we discovered from the algorithm failing the test script was that the algorithm did not originally take into consideration if start date is entered after output by mistake, so it causes the number of intervals in between to be negative, leading to dropping assignment without any notification. The test helped us implement an error prevention on input start date and end date on the department side.

2.5 Queries, visualization, statistical analyses, and machine learning

The first graph displayed on the left within the student profile demonstrates the percentage of their daily assignments completed filtered by assignment type which are all color coded for better visual representation. This plot is updated when the student updates the 'update assignment' section. The second plot demonstrates the standard or department estimated completion times versus the actual duration of time it took the student complete the assignment. The student can update this actual completion time through a click-and-drag operation on the specific event in the schedule section.

For the department section of TimeTracker, the graphs shown display similar data that represents feedback on how the students in the department are performing with respect to the specific days and / or event type. The left graph represents the percentage of time it took the students to complete their assignments within specific days with respect to type of event. The right graph displays the standard times versus the students' actual times given the type of event.

These plots prove to be extremely useful to professors as they can graphically see the feedback from the student data about their assignments. If a department notices that the students have many assignments in a specific week, they can make adjustments to balance student workload. Also, they can change their estimated times to complete assignments to better reflect the student performance on the specific assignments. These graphs are also useful to students. Students are able to use their workload for a week to change plans and make a better schedule for their time. Knowing the completion percentage of the assignments helps them to stay on track and hold themselves accountable.

Using these plots ensures that the readability of the data is high. They are very simple plots that can be understood quickly based on the denotation and style. Having plots that can be viewed easily and show important, useful data is extremely important as a service to the users.

This queries are used to gather the student's course data and their performance, they can be found on lines 26, 50, 146, and 166 of dashboard.php under www/N1/student.

```
SELECT AVG( duration ) AS duration, AVG( UNIX_TIMESTAMP( event_end ) -
UNIX_TIMESTAMP( event_start ) ) /60 AS actual, a.type, classname FROM
(assignments AS a INNER JOIN events AS e ON a.assignment_id = e.assignment_id)
inner join courses on a.course_id = courses.id WHERE student_id = '$user_id'
GROUP BY a.type order by FIELD(a.type, 'Homework','Exam','Project','Lab')
```

```
SELECT avg(pct_complete) as pct_complete, e.event_start,courses.classname,
a.type  FROM (assignments AS a INNER JOIN events AS e ON a.assignment_id =
e.assignment_id) inner join courses on a.course_id = courses.id WHERE
student_id = 1 GROUP BY year(e.event_start), month(e.event_start),
day(e.event_start)
```

```
SELECT DISTINCT(type) from assignments
```

```
SELECT DISTINCT(classname) from courses
```

Machine Learning:
TimeTracker uses data gathered from the database to decide recommendations for the professors based on using k-means clustering from specific performance parameters from the student data. This is to create a list of 'high-performing' students that could be asked to become a TA, recruited for research or recommended for a summer project.

The database queries used to gather data for K-Means clustering are shown below.

1. This query used to gather students' GPA and their average time to complete assignments, it can be found on line 13 of ML.R under www/N1/department.

```
SELECT events.student_id, Gpa,
AVG((assignments.duration*60)/(UNIX_TIMESTAMP(events.event_end)
                              - UNIX_TIMESTAMP(events.event_start)))
AS PER FROM
((events INNER JOIN assignments
ON assignments.assignment_id = events.assignment_id)
INNER JOIN students ON students.id = events.student_id)
INNER JOIN courses
ON assignments.course_id = courses.id
WHERE departname = <department name>
GROUP BY events.student_id
```

2. This query delete the past mean values. It is coded on line 21 of ML.R under www>N1>department.

```
DELETE FROM kmeans WHERE dept_id = <department id>
```

3. The following query sends the algorithm's result back to the database and the website interface (line 36, www>N1>department>ML.R).

```
INSERT INTO kmeans(dept_id, student_id)
VALUES (", post_table$dept ,",", post_table$std_id ,")")
```

2.5.1 Largest Challenge

Our team initially struggled to determine the type of machine we wanted to implement. We wanted to ensure this would create the most value for our stakeholders through use of data that was already required through our optimization algorithm.

How it was overcome

After several iterations of ideas, we decided to take to take a look at the data that we needed for the creation of our website and scheduling algorithm. When reviewing the data entry our website required, we found that there were similarities within the data that could be aggregated together through the use of k-means. These similarities were found to be useful when grouping student performance. When discussing this topic, our original plan was to simply group students who were in similar classes with similar assignment performance and schedules. However we found that there would be more value added to the website if we were to create a list of recommended students who have performed much higher when compared to others. We decided to account for the actual duration the students took to complete an assignment on average and the grade point average of the individual. This allowed use to cluster students dependant on their academic performances in order to accurately recommend potential teaching assistants or research students.

What we learned

Throughout the brainstorming and implementation stage of our machine learning algorithm we learned several key factors. In order to add substantial value for our stakeholders, we needed to employ a less trivial approach in order to present important recommendations to the departments. In order to accurately implement this unsupervised machine learning approach, our website requires there to be at least ten students in order to cluster, 4 clusters taking in account GPA and overall performance, appropriately as it would not make sense to create 4 clusters of only 5 students. The more data our machine learning algorithm has, the better it will perform the intended action.

2.5.2 Second Largest Challenge

Our team had initial issues attempting to determine the proper information to display on the student dashboard. The student dashboard needs to provide meaningful data and we wanted to ensure that true takeaways could be made.

How it was overcome

We wanted to incorporate a time management aspect of the analysis since the scheduling calendar serves that purpose. As a starting idea, a comparison between estimated assignment completion time and actual assignment completion time was proposed, but this solution did not take into account that student proficiencies can differ by class. The next iteration allowed class filters to be placed on the dashboard, but the addition of a metric to track completion was missing. An additional chart was added to graphically represent assignment completion percentage per day.

What we learned

The key takeaway was that the core functionality of the tool being used should be a starting place when attempting to graphically represent the results of data collection.

3 References

Langmajer, M. (2020, February 22). 10 usability heuristics every designer should know. https://uxdesign.cc/10-usability-heuristics-every-designer-should-know-129b9779ac53.

Potvin, P. (2019, August 26). Tips on colour interface design. https://uxdesign.cc/tips-on-colour-interface-design-bccdc691eb72.

Soegaard, M. (2015). Interaction Styles. The Glossary of Human Computer Interaction. https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/interaction-styles.

Wickens, C. D., amp; Carswell, C. M. (1995). The Proximity Compatibility Principle: Its Psychological Foundation and Relevance to Display Design - Christopher D. Wickens, C. Melody Carswell, 1995. SAGE Journals. https://journals.sagepub.com/doi/10.1518/001872095779049408.

DayPilot.Calendar Class — DayPilot Pro JavaScript API Reference. (2019). DayPilot. https://api.daypilot.org/daypilot-calendar-class/

DayPilot.Calendar.onEventMove — DayPilot Pro JavaScript API Reference. (2019). DayPilot. https://api.daypilot.org/daypilot-calendar-oneventmove/