ClassTrak

John Geewax (jgeewax@seas.upenn.edu)
Michael Gottlieb (gottlies@seas.upenn.edu)

Faculty Advisor: Dr. Zachary Ives
Abstract:

ClassTrak is a web application providing students a collaborative environment in which to manage their coursework and maintain contact with other students while in school. ClassTrak is primarily aimed toward college students but could be extended to provide services for high school students.

In an educational environment with so much freedom and personal responsibility, an application such as ClassTrak seeks to give students a better way of managing their time. One of the primary problems of managing college coursework is that the burden of posting information and reminders is entirely on the professors or students individually. ClassTrak intends to alleviate this burden by allowing students to post course data collaboratively, which will be verified and monitored by other students. By allowing users to post documents and information relevant to their classes, professors are no longer solely responsible for posting information on websites or systems such as Blackboard. This information includes homework due dates, exam dates, required readings, and any other events or deadlines.

Students will keep track of this information through a calendar and to-do list system. The calendar display will show the student all of the events pertaining to that day, week, or month. These events include classes, meetings, exams, due dates, school-wide events, and any personal events added to the personal section of the user’s calendar. The to-do list is just a running list of upcoming due dates and reminders. The events that appear in this system can be school-wide, class-wide, or personal. These distinctions will restrict what appears on an individual’s calendar and what events get added automatically. Information will retain its integrity through community posts, edits, and confidence ratings.

Related Work:

Facebook (http://www.facebook.com)

Facebook’s original design allowed users to post their classes and also showed them which of their friends were in classes with them. As they have now removed that feature, the uniformity of the system has moved specifically towards socializing and away from academics. This has left a void in the realm of social networking tailored to academics. The component we intend to use from this body of work is simply the ability to communicate with classmates through the website. Facebook’s development platform has led to the creation of several apps aimed at connecting students. One example, Courses 2.0, gives students the ability to display their schedule. The benefits of this application are allowing students to find other students in the same class to share textbooks and form study groups. These applications do not have the depth of ClassTrak and tend to be more aligned with Facebook’s social networking agenda than helping students keep track of their studies.

Blackboard (http://www.blackboard.com)

Blackboard is currently one of the main systems used by professors to communicate with their students. It currently lacks a clean and simple interface, which we intend to design. It also places the burden of updating course information on the professors and teaching assistants. In ClassTrak the task of adding information is distributed over all users and the data integrity is also managed by all of the users. Blackboard goes beyond the scope of ClassTrak in some features. Things like grade reporting sections and electronic assignment submissions are not part of the functionality of ClassTrak. ClassTrak does not make distinctions between Professor and Student users so some functions like those previously mentioned will be impossible to implement and are not present in the initial release of ClassTrak.

MediaWiki (http://www.mediawiki.org)

MediaWiki is much broader than our software which holds a structure that is stricter than the “free text markup” stored for a wiki. However, we followed the editing model proposed by MediaWiki because it has been a very successful tool for moderation of popular wiki websites such as Wikipedia (http://www.wikipedia.org). A possible pitfall in terms of globally writable data is that the data is easily corrupted. MediaWiki seems to have created many methods of preventing this which we implemented or specified. Practices like flagging disputed or supposedly false information is part of ClassTrak to maintain data integrity. A confidence rating system for information and users helps protect the community from errors and intentional data corruption.

Schedulizer (http://www.schedulizer.com)

Schedulizer is an online application for planning students’ course schedules, although it is currently only available for a few schools. Schedulizer allows students to add classes to a possible schedule
as well as add personal events. It will take these into account when suggesting a schedule for the student to use. These personal events are only recurring events. ClassTrak builds a schedule for students based on the classes they enroll in. It allows for the creation of recurring and non-recurring events.

**BackPack by 37 Signals** ([http://www.backpackit.com](http://www.backpackit.com))

BackPack is an online application for managing to-do lists. You can create multiple “pages”, which can consist of list items, notes, and dividers. The UI in the system is very smooth and sleek. Item addition and editing is done in-line using smooth operations that do not require re-loading the whole page. BackPack features a lot of the functionality and the sleek UI feel that is to be a part of ClassTrak. However, ClassTrak distinguishes itself by having the work of building that list diffused among everyone in a user’s school and classes.

**CampFire by 37 Signals** ([http://www.campfirenow.com](http://www.campfirenow.com))

Campfire is a Javascript based web chat application. The ClassTrak specification includes a similar chat client to facilitate communication between all students and especially group project members. By logging all chats for the users, it is possible to keep all discussions for later review, as well as giving the people who may not be available for chat a chance to review the “minutes” of the discussion. Another related feature of ClassTrak is class forums. Users can post topics and replies on the forum to communicate with their classmates.

**BaseCamp by 37 Signals** ([http://www.basecamphq.com](http://www.basecamphq.com))

Basecamp is an online project management suite which allows users to add projects and input deadlines. Students can use ClassTrak as a project management suite for collaborating on group projects providing the ability to add deadlines, milestones, upload files, forum topics, and comments. We also mimic this type of dashboard display for a course page, treating a course as a collaborative “project” among the students enrolled. BaseCamp contains a real-time chat client for team chatting, which is part of the ClassTrak specification.

**Zoho Online Suite** ([http://www.zoho.com](http://www.zoho.com))

Zoho has a suite very similar to that of 37 signals (outlined above), and as a result we integrated some of the smaller nuances from this suite, and looked to it for design ideas in the user interface. It includes a document editor, spreadsheets, project management tools, presentations, a planner, chat client, wikis, and mail. Features related to scheduling, organization, and group project management exist in ClassTrak and we look to Zoho for inspiration on tools and UI. Features like a document editor and spreadsheet editor are beyond the scope of ClassTrak and are not be part of the application.

**Google Shared Docs** ([http://docs.google.com](http://docs.google.com))

Google has a nice interactive office suite, and although we do not have a WYSIWYG text editor like Google Docs, we allow users to upload files and share them between users. We look to the privileges provided by a standard Google doc (view, collaborate, publish) for uploaded and collaborative files.

**Google Calendar** ([http://calendar.google.com](http://calendar.google.com))

Although Schedulizer (see above) is our starting point for the calendar and scheduling components we integrated, the ClassTrak specification contains plans to provide a collaborative calendar with public and private events which will allow other users on the site to view public calendars. This will make it possible for those users to determine when their friends are busy, or to see an “overview” calendar for a course, judging how busy students are with devoting their time to the course. Google Calendar has a very clean and easy to navigate UI. This was an inspiration for our calendar layouts and UI.

**College Ruled:** ([http://www.collegeruled.com](http://www.collegeruled.com))

CollegeRuled is a simple site that allows it users to create a schedule, message boards, and organizers for class. Students have the ability to create a weekly schedule, create an organizer page for each class, and create a class forum for communicating with other students. Users also have the ability to schedule “Activities” which are non-class recurring events that lack organizers and forums. CollegeRuled is overly simplistic in its approach. Its calendar only displays recurring events and has no regard for the actual date therefore one-time events and deadlines are not possible. The class organizers do have the ability to list assignments with due dates attached, but there are no reminder systems. Its GUI also suffers from bugs and design flaws. If a class or activity is scheduled at an odd time like 2AM the whole 24 calendar gets dumped on to the page creating a need for the user to scroll up and down the whole page. When re-organizing in the organizer the drag element of the GUI is buggy and causes odd behavior in the rest of the page elements. CollegeRuled does contain the functionality to sync with Facebook which is a good idea and may be part of ClassTrak in the future.
Overall Design Goals:

Our overall goal was to complete a specification for ClassTrak and create a working prototype of the core functional elements. The core functional elements are items like to-do lists and calendars as well as functionality like collaborative scheduling. Core functional elements do not include features like exporting vCards for syncing with other calendar applications or the built-in chat client. These features are part of the ClassTrak specification, but are not present in the prototype. Our ultimate goal is to make ClassTrak available to the public and to design it in such a way that students can depend on it and will use it frequently.

To make an application that students would use frequently we had to address certain common problems with scheduling applications and web based applications. These problems include requiring large amounts of time to set up and add items to a schedule and ugly, clumsy, and frustrating interfaces. ClassTrak’s collaborative scheduling functionality means that students can spend less time building their schedules and more time getting things done. By diffusing the job of building schedules we have made the process more efficient freeing up time for less tedious tasks while leaving an individual well organized. By looking to examples of good interfaces like the 37 Signals suites, Zoho suite, Google applications, and Facebook we can create an interface that students will use and find intuitive. These suites and applications are extremely popular and have found ways to make web interfaces beautiful and easy to navigate. The UI of the prototype is a rough design of the look of ClassTrak. Some of the more complicated aspects of the UI like asynchronous element updating are specified, but not present in the prototype.

The choice to implement the Core functionality of ClassTrak rather than as much functionality as possible was in some sense made for us. We chose to write this application in PHP. This was because it is a powerful web language that we are familiar with. In writing the application we found we needed to create a large amount of utility libraries just to get abstractions into the system. This could have been done for us if we had made the choice to work in another language on top of some web framework. During the second semester of this project we became familiar with Python and the Django framework. This would have been an ideal choice to implement in from the beginning. Had we made that choice we may have had more time for implementation of the public elements of the system. We considered re-writing the entire application in Python with Django, but we were worried about getting minimal functionality working quickly enough. The choice we made was to continue working in PHP and make a working prototype “proof of concept” for the project. This actually turned out to be a good thing. Creating this prototype and working on a specification has helped us flesh out the details. We now see that Python would have been a better choice and can re-write the application with a better vision of the final product in site.

Technical Approach:

ClassTrak will work through the contributions and maintenance of its users. Anyone will be free to create a user account, although we limit the initial testing to only users with email addresses ending in .edu. These accounts are password protected and require e-mail verification for creation. Users have the ability to create a short profile, enroll in classes at any school, and create To-Do lists and other content. By not restricting users to their own school, we take into account the possibility of cross registration (e.g. Harvard students can cross register and take classes at MIT) and semesters or years abroad. Students have the ability to post events and reminders for themselves, particular classes, as well as for entire schools. This allows information like due dates, holidays, and personal reminders to be posted and maintained while distributing the
burden of these data-entry tasks among the users.

An important consequence of allowing this kind of mass data editing is loss of integrity. We put several measures in place to prevent this. Classes will be imported from different schools so that false classes will not be created. A class page must be started by a user enrolling in the class (i.e., there will not be a class page for CSE 400 if nobody has enrolled in it). All of the users who enroll are given the privilege of posting and editing information and events pertaining to the course. Using a system partially inspired by Digg, postings, comments, and other information added by users are susceptible to ratings. (Digg, 2008) This way when any posted content acquires a sufficiently negative rating, it is hidden from the view of others. As an additional incentive, the user who posts content that gets buried will suffer the consequences in the form of his or her future posted content being automatically rated slightly lower. The purpose of this is to prevent “repeat offenders” from spamming the course pages. Users will also be able to report posted content, which can result in action taken against the poster.

As with most database driven applications, we divided the task of structuring the application into tiers (see diagram). It is important to note that the standard database application model is usually a three tiered structure, and we added an extra layer called “Form Processing.”

Outline of each tier:

Database Access:

The database access tier consists of two things: the architecture of the database itself, and the abstraction layer we use to interact with the database. Since we used PHP and MySQL, we used a customized database abstraction layer that has very few functions which we wrote as part of the project. (Schlossnagle, 2004) In order to keep the project easily maintainable, we use a common database object that provides a simple interface to each child object. (Fowler) The database uses an archive-friendly design for most of the information, which does not delete data but simply adds a new row labeled as “more recent”. These new rows become the only visible piece of data above the database access component. (Sweat, 2005)

In the case that a user has “deleted” rows of data, rather than having to rely on the MySQL query logs to restore the database, we can actually revert to any point in time simply by deleting the more recent (defaced) entries in the table. This way of maintaining data integrity is possible through a separate information retrieval library that we built on top of the database access component.

Aggregation and Analysis:

The aggregation and analysis level is responsible for most of the “work” done by the software. This level is responsible for determining the distance between two dates, how a name should be formatted, what items to show in a to-do list, what events to push to an RSS feed, which to show in the calendar display, which pages are available for editing, calculating ratings of posted items, allocating disk space and maintaining pointers to uploaded files, and normalizing the raw data of which courses are available to users among other things. This layer consists of about 30-40 modules, each with their own classes and methods for manipulating and handling data. For example, there is a hierarchy of Users, the top level containing a “checkPassword” method common to all the child classes. This type of method is considered to be a part of the analysis layer.

Form Processing:

The Form Processing layer -- which in most applications is combined with the analysis layer -- is what makes this application structure different from the common web application. The form processing layer is closely coupled with the User Interface layer as they need to interact with each other constantly, yet we decided that
the modularity of the software would be compromised if we were to leave this layer out.

In any web application, the HTML forms are submitted, and then the data needs to be sent down to the analysis layer, which in turn sends data to the Database layer. Unless the data is checked each time on its path from the top layer all the way down, error messages do not propagate upwards as is necessary to inform the user should something go wrong.

The simple solution is to check the data at each stage, but in a rapidly changing application, this means duplicating code on several different layers. As a result, we have created a general form processing module which takes as arguments the following: a template file for the form being displayed, functions for validating the form submitted using the global PHP variables $_GET, $_POST, and $_FILES, and the logic to be redirected upon successes and failures. We then create a specific class for each form to be submitted and instantiate this subclass in order to process the specific form. As a return value, we get strictly boolean values to say whether the form succeeded or not. Additionally, the module holds an error message to be displayed to the user on failure, which will state whether the error was invalid data (caught by the validation function) or another error (not caught by the validator). The error message is displayed inside the template file.

This plan of dedicating an entire tier to validating user data is aimed at preventing the many types of security vulnerabilities, as well as centralizing the code required to normalize “tainted” data for a particular action. (Shiftlett, 2005) This method, which we believe is unique to our application, compartmentalizes the functionality of the application one step further, making our application simple to update and extend with new functionality.

User Types:

In the preliminary design ideas, we first wanted to try to regulate how users registered to use the website, as well as to have different user account types for students, teachers’ assistants, and professors. We found that the only problem with this structure is the lack of a way to validate any of this data. Although on a small scale (say, Penn Engineering) we could manually verify who was a professor and who was a TA, on a large scale – as we plan to have the scope of this application move beyond Penn Engineering – there is no reasonably efficient way of validating whether a user is truly a professor of the course. As a result, we came up with the structure seen here.

We define a standard object class which is common for any object being stored in the database. Below that we have the standard User object which is an abstract class to be extended by the various types of users.

The reason we have the administrator user at the lowest tier is simply due to the fact that the administrator should have everything the standard user has, as well as extra permissions for editing and deleting posted content. This also creates the ability for us to add new types of users, at which point the administrative user seen here becomes a “standard administrator”.

Courses:

As the courses are a main component of the project, it’s important to note how they interplay with the other pieces of the application. We designed courses as belonging to a particular school, both being database objects. The main sub-components of database objects. The main sub-components of courses will be assignments and events, keeping in mind that an assignment contains an abstraction of a “due date event” as a piece of the assignment. This is done to maintain uniformity when we develop the calendar
Assignments:

Assignments are at the core of the application in that they will be aggregated to form To-do Lists, as well as constituting the majority of the collaborative content on the website. Assignments could be something like “CIS 400 Paper Revision 1” which would be added under the CIS 400 course. Users can add files, notes, comments, questions on the bulletin board provided for the assignment, and edit/update the information about the assignment such as the due date or requirements. The due date is considered an “event” by the data structure so that we aggregate only events and their parent objects on the various calendars and to-do lists throughout the website.

We specified a special sub-type of assignment called a “group project” which can encompass any assignment where the assignment is intended to be done with other students. In this case, the assignment has many of the same features, except it assumes more of a “course” type role in that students “enroll” in the group project (just as they do for a course) with other students. At that point, the assignment becomes aggregated only among those students who are enrolled rather than aggregated for all of the students enrolled in the course. This is simply an abstraction of adding “blinders” between each of the pages, and will serve to give groups the isolation from other groups while providing the group members a smaller, yet still collaborative, environment to work on their project. Just with any other assignment they can upload files, change information with respect to their particular assignment, add milestones they intend to complete, have a personal bulletin board, etc. This would not remove the functionality of a course-wide bulletin board, chat room, or the collaborative ability for the users to update the due date of the project, which is applicable to each of the groups.

Bulletin Boards:

Bulletin Board functionality is part of the ClassTrak specification using some system like PHPBB, which will be available for each assignment as well as generally for courses and schools. (phpbb, 2007) We first planned to integrate PHPBB itself, but after realizing that we will need to duplicate the actual bulletin board repeatedly – much like Facebook provides a bulletin board for each of its groups – we realized that we will be better off designing our own minimal but functional approach. The goal of the bulletin board is to have it more complete than Facebook’s version by making it more clear who is replying to which posts, but not encompassing all of the features of PHPBB such as extensive formatting. We also specify a Digg-style system for hiding posts that people rate as poor comments or spam. (Digg, 2008)

To-do Lists:

One of the main components is the standard To-do list, which is similar to the one provided by 37 signals in its simplicity. Keeping with the theme of extensibility, we designed the list and its composing items as pointers only throughout the database so that we are able to combine items of different types and allow multiple copies of items in different lists. An example would be a personal to-do list with items “buy milk” and “study for CIS 400” at which point we also would like to add the item “CIS 380 Project 1” from the CIS 380 course to our personal To-do list. By storing lists as sets of pointers we can accommodate this.

We also need to keep in mind that the item “CIS 380 Project 1” can be marked as “done” by one person, but that is that person’s “private” copy of the To-do item. Basically we want the private copy to belong to its respective owner, but we don’t want to lose on the collaborative aspect of the website by removing the ability for other students to update the due-date. By storing these items as pointers from a global copy to a private copy, we maintain the collaborative aspect of the website and accommodate this necessary functionality.

Client side functionality using Javascript to ease the burden of refreshing the entire page is part of the UI specification. (Spolsky, 2001, p. 91-93) By reducing round-trip requests through the XMLHttpRequest object available in Javascript we can not only save on bandwidth, but improve usability with less “flashing” and “refreshing” of the page.

Calendar:

There are four calendar views within ClassTrak, each with a distinct primary use. They are as follows:
Year View:
The Year View displays each month of the year as a relatively small box with just enough room for numbers to act as links for each day. Users can click on a month name or day number to move to the corresponding view.

Month View:
The month view displays one month in a larger form than the way months are displayed in the Year View. Each day has a clickable number link that takes the user to the day view. Each calendar spot gives a small preview of the day similar to the month view in Microsoft Outlook or Google Calendar. This preview is made up of class events, school events, and personal events.

Week View:
The week view presents columns for each day of the week starting at Sunday and ending at Saturday. Events appear as boxes spanning over the time period they are scheduled to occur for. Times appear to the left of the columns of the days. This sort of layout is used by many other calendar systems including Google Calendar and Microsoft Outlook. This trend appears to be a result of the fact that this layout is actually very good. It allows users to see most of what is going on during their week and provides an easy way to detect conflicts. The week view also appears within a div that is scrollable. This allows 24 hours to be listed without needing a really long layout on the page because the window is only about 10hrs big.

Day View:
The day view simply lists the events scheduled for that day in order of occurrence. In the listings notes about the events can appear such as location or status.

Technical Challenges:
The Technical Challenges we faced during production of the ClassTrak prototype were extensive. We chose to work in php because it is a powerful web programming language that is open and that we were familiar with. This proved to be a bad decision. In attempting to create a working implementation of ClassTrak we found ourselves writing a large amount of libraries to abstract away tedious details of the system. This meant large amounts of time writing framework libraries and less time focusing on some of the more interesting parts of the system.

The libraries are powerful and solid. They include unit tests and well designed class hierarchies. This took a good amount of time to develop. Once we had completed the majority of the libraries we realized that we were behind schedule for some of the more public areas of the system like UI and form processing. This meant that certain features had to be cut from the final prototype or nothing would get finished. Our choice to try developing the front-end and back-end in parallel was not particularly successful. We ended up with a lack of focus and jumped back and forth too often, which was terribly inefficient.

Towards the end of the development of the ClassTrak prototype we became familiar with Python and the Django web framework. We found that building an application on top of Django is much simpler. Many of the libraries and much of the code we completed in php is built into Django. Had we been able to develop in Python using Django from the beginning we may have completed more of the specified functionality. This was the largest lesson we learned from this project. Researching other programming languages and toolkits is more important when choosing a development language than familiarity or comfort. Our future plans include re-writing and fully implementing ClassTrak in Python using Django. This version will be the one to be released.

The most interesting design oriented challenges we faced were data integrity and motivation to post. These problems share some common causes. In discussing the likelihood of students to use a system such as this the general answer was “yes if the information is right.” The two situations affecting the integrity of the data are a lack of information entering the system and accidental or malicious input of false data. We believe that there is enough incentive to post. In a “pay it forward” sense individuals will post information in the hopes that some of their work will be done for them. This system has proven to work for
other applications like social news sites. Sites like Digg have many more readers than they do posters, but there is incentive to post to keep the system afloat so that even posters can get the benefits of readers. (Digg, 2008) We believe this is the case with ClassTrak as well. Accidental data entry can be corrected with global editing, burial ratings, and flagging. This lets users know to take a closer look at this one date and verify its integrity. This will be the rarer case so the burden on the user will not be significant. Some students we spoke to suggested that a malicious student would post knowingly false information to throw off his/her classmates. This situation is alleviated by burial ratings, user ratings, and reporting users to administrators. These cases should be the exceptions to the rule and would not cripple the project.

Although we faced considerable technical and design challenges we were successful in fleshing out most of the details of the ClassTrak system including all of those specified above. We have specified most of the final functionality and our prototype is useful as a proof of concept. Though this prototype lacks the non-core functionality, it has proven to us that the basic idea of collaborative scheduling can be extremely useful and that a system to do this is a feasible engineering feat given the right tools and enough time.

**Conclusion:**

The result of the project is a prototype of the ClassTrak system described in this paper. We focused on the core functionality that could help us test the idea and believe it to be a feasible and useful product. An additional product of our labor is a powerful library of php classes that can act as a framework for other web applications. Actually implementing parts of the ClassTrak system helped us complete and perfect the design. At this point we feel a re-write of the system using a different language with a suitable framework would result in a much better implementation that would both be better to use and easier to write. The choice to design the UI and back-end in parallel turned out to slow us down. In re-writing this we would first write out the usage and test cases. Then write the application from one end to the other. Defining the test and usage cases would give us that idea of what our end product should be and force us to think of the cleanest way to create the final product rather than beginning by writing pieces that would organically grow into a final product that may not be exactly what was specified.

**References:**

**Bibliography**


http://www.basecamphq.com

BaseCamp is a currently operating web application that helps groups collaborate on projects. It is a member of the 37 Signals suite and features the same smooth UI style that is common to all of the 37 Signals products. The product is for anyone. Their business model has a tiered pricing system for their services. ClassTrak when made public will be free. We looked at BaseCamp as a reference for designing ClassTrak’s group functionality and UI.


http://www.backpackkit.com

BackPack is a small-business group management. It resembles BaseCamp in its pricing, functionality, and UI feel. We believe ClassTrak is a service that should be free to students, but BackPack is another good reference for designing the look of ClassTrak and some of the group features.


http://www.campfirenow.com
CampFire is another currently active 37 Signals product. It is a chat client style application for project groups. Group chat and other forms of group communication are part of the ClassTrak specification and CampFire is a really good resource to look at for ideas on how best to implement this.


Zoho is an online suite that includes a large number of products for purposes like document editing, chatting, spreadsheets, etc… Zoho has many products that are beyond the scope of ClassTrak, but it is another well designed web application that is currently used by a large amount of people. It is a good example of well designed web applications and software that allows individuals to collaborate although in a different way than ClassTrak.


Blackboard is a system for web access to class material that is very closely tied and adapted to individual universities. Students log-in using their university id and users are distinguished as Professors, Teaching Assistants, and students. Blackboard allows for electronic submission of homework and secure distribution of grades. We primarily used Blackboard as a contrast to the design of our system. Blackboard has a poorly designed interface and is extremely hard to use. We took these lessons into account when designing ClassTrak’s UI and overall design.


CollegeRuled is a fairly new scheduling utility that is geared towards students. It lacks collaborative scheduling and has a frustrating UI. CollegeRuled focuses more on functionality like interfacing with Facebook and on pushing their FixTunes software through the site. We believe it is not a system good example of what a web based scheduling application for students should do. We used CollegeRuled to contrast UI choices with some of the better implementations from 37 Signals and Zoho.


Digg is a social news website where news stories from other locations on the web are posted by users. Users can also leave comments on the articles. Users have the ability to “digg” articles which boosts the popularity counter of the article making it appear in more locations on the site. Comments can also be “dugg” or buried. This burial system is the primary inspiration for the ClassTrak burial system that regulates data integrity. Digg’s rating system has been extremely successful in providing a good experience for their users.


Facebook is one of the largest social networking sites in the world. It started as a site restricted to college students, but has opened registration to anyone. Facebook also has a development platform which allows for the creation of mini web apps that sit inside an individual’s profile on the site. Some apps related to college classes and scheduling have been created, but nothing is on the scale of ClassTrak. We primarily looked to Facebook for its clean UI and ClassTrak may include a Facebook synchronization
component in the future, however this is not currently part of the specification or the prototype.


Martin Fowler is a software developer and consultant, currently working at ThoughtWorks. The idea presented in this article is now considered the standard way of managing database entries in web applications, and used in the most of the modern web frameworks such as Ruby on Rails and Django for Python. The work in the article is more of a short summary of the design principle, which we worked to implement much like other frameworks. As far as development, we found that the method itself was excellent, but implementing it from scratch was "reinventing the wheel" and we could've progressed further had we used another toolkit.


Google Calendar is one of the most popular calendar applications along with iCal and Outlook. However of those it is the only web based application. Google Calendar is a large inspiration to the look of the ClassTrak Calendar.


Google Docs is an online WYSIWYG document editor. It has limited functionality compared to non-web based document editors, but it has a powerful collaboration feature. Google Shared Docs allows multiple users to do editing. Changes in the document are resolved in a version control system. This kind of resolution is an important part of the ClassTrak system because any student can edit the shared data and the changes must be saved and propagated in a similar fashion.


PHPBB stands for PHP Bulletin Board. It is an extremely popular bulletin board system. It is written in PHP. It is used everywhere from blogs to news sites to school forums. We used it as a reference for the type of forum and class communication system to be included in ClassTrak. PHPBB is extremely powerful and contains way more functionality than we would actually need, but was a valuable reference.


This is a fairly current straightforward reference on PHP programming including some of the more advanced techniques. Sams puts out a large amount of books on learning different programming languages that are usually really good. This was one we looked at as a reference for PHP since that is what we wrote the ClassTrak prototype in. The audience of the book is PHP programmers and it was much easier to read since we were already familiar with PHP.


This reference was useful in designing ClassTrak and keeping it free from security vulnerabilities like cross-site scripting attacks and SQL injection. O'Reilly
is another publisher who puts out a large amount of great references on computer programming topics.


Schedulizer is a web application that allows students to plan a course schedule. It is limited to a small number of schools and only serves the purpose of planning a schedule. It does not do schedule maintenance. Users have the ability to mark schedules public which makes them viewable to other users. This functionality is something we decided was beyond the scope of ClassTrak in a way. ClassTrak will generate a calendar and to-do lists when a user enrolls in a particular class. It does not handle planning a schedule any different than creating a schedule. Schedulizer was a good example of what other systems are available to help college kids with scheduling problems.


Joel Spolsky is a respected software developer, previously working at Microsoft and currently working at Fog Creek Software, a software company he started. He is well known for his writing and opinions on software design that were considered somewhat unorthodox but have lately become more accepted in the software community. He is also the author of the book Joel on Software and writes a blog of the same name. As this source is rather dated and the technology behind designing interfaces has changed dramatically, we pulled from this source more of the design principles, aiming for simplicity and a minimum number of clicks as the book suggests. Our more recent user feedback suggests that this book was a good source of information in its principles.


This article introduced us to the “archive friendly” model of storing data. The basic idea is that instead of storing one record and having to revert the database back upon error we constantly make new entries that are labeled “more recent” and those are the ones used by default. We did not fully implement this into the prototype because we later found that it would not be necessary for ClassTrak. This model is nice for tracking changes between data, and we do not necessarily want to do that for everything in ClassTrak. The re-write we will be doing will not use this model at all.


The WikiMediaFoundation is the parent company of popular wiki sites like Wikipedia and Wikibooks. Wikipedia is an important example of a site that allows global editing, but manages to maintain a respectable level of data integrity. Since ClassTrak faces a lot of the same problems that Wikipedia and other wiki sites face we felt that the WikiMediaFoundation products are in some ways a proof that ClassTrak could reliably provide a service to college students. The data integrity practices of the wiki sites are a good resource for the policies of ClassTrak.