Assignment Organization through Synchronized Calendars

Ilya Krasnovsky - ilyak@princeton.edu
Andrea Malleo - amalleo@princeton.edu
Nicholas Maselli - nmaselli@princeton.edu
Gina Hong - ghong@princeton.edu

Project Manager - Andrea Malleo

Section 1: Overview

The issue this project addresses is coordination of problem set collaboration. The goal of the project is to design a web application that will allow Princeton University students to view the assignment due dates for all of their classes on a visually appealing user interface. This interface displays time blocks proposed by other students during which to work on the assignment together. It allows the student to join an already existing time block or propose a new block of time to work on the current assignment throughout the week for other students to see. The application provides an informal and easy to use class-wide method of communication that encourages networking and collaboration with other students the user may not yet know and at the mutual convenience of everyone involved. This application sets out to accomplish two primary objectives: automatic organization of assignment due dates between multiple classes and convenient facilitation of peer collaboration with the hopes of increasing subject material understanding and problem set success.

Section 2: Requirements and Target Audiences

This application endeavors to solve difficulties in personal scheduling and peer to peer communication. It does this by providing an automatic visual system to coherently display the due dates of all of the user's assignments as well as a shared interface to organize problem set solving sessions. It is geared toward University students taking classes that have semi-regular problem sets that allow for group collaboration. Our application will require such students to enter our website by logging in through Princeton’s CAS login. This gives access to a calendar with a tab for each class, each displaying due dates for corresponding problem sets. These calendars are shared among every member of the class that has signed up for the application, and thus provide a common scheduling GUI for classmates to organize and coordinate problem solving sessions.

This application will significantly benefit students that may not already have an existing friend base or group of people they work with because it will allow the students contact other students who have similar work schedules to them. Our application will also help students that are generally more disorganized or who have multiple classes with due dates scattered throughout the week, because the due dates will be clearly listed on their calendars automatically when they log in.

Currently, there are no known systems or applications that allow students to view when other students plan to work on assignments. There are calendar applications that exist but none of them automatically place the due date for Princeton course’s on the calendar.

Section 3: Functionality

Imagine that you are a freshman at Princeton University. You know very few people in your classes and your organizational skills may have room for improvement. After the first or second week of the semester you have likely been given due dates for assignments in each of the 4 or 5 classes you are in. As these are Princeton classes, the material is much more challenging than that of the classes taken in high school. You find yourself with too many classes and due dates to worry about and you may be struggling to complete some of the early assignments. A student advisor, RCA, or email encourages you to sign up for our application which requires you to sign in with your user netid and password. Immediately you can see all of your due dates for your assignments on a nice visual screen. Since many freshmen have been encouraged to sign into this application, you already see several proposed time blocks for today, tomorrow, and the following day on which to work on the assignment.

It is Monday at 4:00pm and you have a MAT 201 problem set due Friday at 5:00pm. You see there are many people who plan to work on the assignment from 4:30pm to 7pm this upcoming Wednesday, but since you have sports practice or another extracurricular activity at that time, you cannot make it. You decide to click on the Wednesday 7:30pm to 10:00pm time slot and indicate that you can work on the assignment even though it appears nobody else has chosen that slot yet. You then head off to your extracurricular. After returning from your activity you get an email in your inbox sent from a student in your class with another student cc'd asking where you would like to meet to work on the assignment *on* Wednesday at 7:30pm. All three students in that group email indicated that they could work on the assignment at 7:30 pm on Wednesday and one of them sent an email to the others to determine the meeting location. You note to yourself that next time, you will likely be the one to send the email. With this application, your organization level has significantly improved and your difficulties finding problem set buddies have been solved with a simple application and a few seconds of pointing and clicking on a calendar.

This application will not only be for freshmen. Imagine you are a junior or senior in an eating club. You have many friends, are organized, and are doing well at Princeton. You have friend groups with which you typically do assignments, but sometimes they are not available or in all of your classes. After logging into this application once, you can conveniently see all of your classes' due dates. To avoid procrastinating, you click on a time you plan to work on this week's COS 333 assignment as well as this week's MAE 433 assignment. Of course, COS 333 assignments do not allow collaboration. Nobody will email you wanting to work on it, but you *a*ppreciate having the allotted work time on your schedule. Someone does email you about 30 minutes later asking if you would be willing to work on the MAE 433 problem set. You have never met this student before and happily agree to work on the problem set with them at the specified time.

In summary this application has a target audience that encompasses all students at Princeton University taking at least one collaborative problem set oriented class. Students in every year and probably every major are candidates for using this application.

Section 4: Design

The three main components of our project are the front end user interface, the back end storage system, and the middleware that includes logic to connect action in the GUI with the contents of the storage system, as well as the Blackboard scraper. The currently envisioned implementation languages, platforms, and/or services for each of the aforementioned components are presented in the table below.

Component	Language(s)	Platforms/Services	Implementation Details
Front End/Calendars	HTML 5 / CSS 3 JavaScript	Firebase Hosting	Google Calendar API
Business Logic / BB scraper	Python	N/A	Use of urllib and mechanize packages
Back end/Data Storage	Python	Firebase REST API (Python interface)	User account management (if permanent) likely done by Firebase

After the user signs in, the blackboard scraper discovers the classes the user is currently taking and any new assignments posted for each one. It returns this data (classes and new assignments) to the middleware logic. For each of the classes, any new assignments are sent to the back end storage to update the assignment list for the class. Additionally, the name of the user is added to the list of students in the class if it is not already there. Then this same list of classes is fed to the front end.

The front end will have a left side panel displaying the classes and a center/right section with a calendar. While the left panel gets its data passed up from the middleware, the center/right speaks directly to the storage system. The calendar will retrieve for each particular class the next assignment's due date as well as the due dates for the previous assignments in the class for display. On each day of the calendar, the user can see colored blocks marking problem set solving times, as well as the number of students who have agreed to that time. Additionally the blocks will have some manner of displaying the email addresses of the all of the students to facilitate communication. All of this information- time blocks, student count, and student information, is retrieved from the back end content storage. Clicking and dragging across several hours in the day will highlight the time block a specified color and initiate a new time block to be stored. This action triggers a method that sends this information (new student count of one, with their email, associated with this new time block) back to the data storage for this class for later retrieval.

Likely we will be using Firebase for content storage. Data will be organized by date and time order for easy access. The current plan is to implement the Blackboard scraper and middleware logic in python. For at least the first stage of production, every user logging in creates a temporary session, and no data is stored at the student level. The plan is to implement the front end design with HTML 5 and CSS 3 with JavaScript being used to handle anything dynamic that is not covered by the Google Calendar API**.** If we are able to get access rights to the Google Calendar API, this is our first choice.

Section 5

- March 8th: Meeting with Professor Moretti to discuss our application
- March 15th: Complete design document
- March 21st:
    - Develop a Project Status Website
        - Simple HTML 5/CSS 3 similar style to the COS 333 website

    - Obtain a project manager TA
    - Write an initial “elevator speech”
- March 28th: Complete initial prototype
    - Blackboard Scraper that gathers class information for students that are not us (proof of concept for CAS login)
    - Content storage system that contains class names from blackboard scrapes for a few student accounts
    - A web app containing a panel that lists the classes of the student who logged in
- April 4th:
    - Blackboard Scraper can recognize an assignment has been posted since the last time user scraped
    - Content storage system contains all past and present due dates for class
    - Calendar shows up on the web app
- April 11th:
    - Users who log in are added to the student lists for all of the classes they are in
    - Users can click on calendar and mark proposed study times that are saved to storage
    - Saved study times show up on other students calendars
- April 18th: Complete Alpha version of project
    - All of the above features in working order
- April 25th: Complete Beta version of project
    - Cleaned up interface of web app
    - email notification functionality?
    - Potential saving of student information instead of temporary sessions
- May 2-5: Demonstration and presentation
- May 12th:
    - Final documentation, product guide, and final report due
    - Submit the final code and URL for testing

Section 6: Risks and Outcomes

The first possible issue we may run into concerns the effective scraping of Princeton’s blackboard site. This is critical to the success of the project because it is the crucial source of data for both setting up class specific user calendars and new assignment posting dates. There is a possibility for issues in identifying links to problem sets, all named differently for each class, as well as gleaning the due dates of each problem set. The second issue can be remedied by allowing for user input or verification of the due date on the calendar interface. The first issue can hopefully be remedied by the fact that each link on a blackboard page has identical html structure that can thus be exploited for its predictability.

A second set of possible issues surrounds our access to specific user's blackboard sites. This can cause an issue where we are unable to determine which classes a student is enrolled in when they log into blackboard. Our program needs to automatically be able to save data for classes a student is in and assignments that a class has, so being able to determine which classes a student is critical. This issue could potentially be fixed by scraping TigerHub instead, though if security access is the barrier, this will prove unhelpful. We take faith that previous 333 assignments' successful harnessing of the CAS login system indicates our ability to do so as well.

Placing the data in a back end database and loading it in a front end user interface is unlikely to produce any problems. Learning a database language is not a huge hinderance so we are less concerned about problems arising in these two areas of our project. The main issue we may struggle with is being able to obtain the data we need to complete this project which is the job of the blackboard scraper.