COS 333 Project Overview Proposal
16 March 2012

Project Title: PAL (Princeton Academic Linker)
Project Manager: Wenley Tong

Group Members:
- Masha Okounkova (mokounko@princeton.edu)
- Prerna Ramachandra (pramacha@princeton.edu)
- Wenley Tong (wenleyt@princeton.edu)

Overview

Princeton students presently have a number of different websites they may need to visit in order to keep track of everything going on in their academic lives, not to mention anything else they may do. Blackboard, ICE, SCG, and (in some departments) Piazza are sites that host useful and sometimes necessary information regarding courses. Needing multiple windows or tabs just to do everything class-related can be annoying, and the interface for Blackboard has many unused features that simply clutter the window. PAL will be a student-end interface to Blackboard and other class-related sites for Princeton undergraduates.

PAL will have a single, updating window for all academic features. This will include access to documents posted in Blackboard, announcements posted by professors, discussion boards, and possibly grades.

Having a common interface to all of these websites which is simple and provides convenient access to the information stored on them will help students better manage their resources and time to store retrieve this information.

Functionality

An interface to improve access to and interaction with Blackboard.

Student User:
- Masha wants to just check in on her classes. Upon logging in to PAL, Masha will be presented with a main page that will show all changes / updates since she last logged in (e.g. new announcements, assignment postings). Masha reads some of them and sees that the assignment and reading list for the week in COS 595 has been posted. She clicks on the COS 595 tab on the left will be a side bar and goes to the course's page. Masha sees the new assignment in the Assignments widget and clicks the link to the document; it starts downloading right away.

- Prerna wants to check the discussion board for ENG 617 to see if anyone has answered her question regarding post-Revolutionary French allegories. She logs on to PAL and sees that there is a new post. After clicking on the ENG 617 tab, she goes to the Discussion Board widget

The main body will contain two columns of widgets. A widget is a representation of a feature of Blackboard (e.g. Assignments, Announcements, Course Materials, Discussion Board). Widgets are resizable and internally scrollable. Preferences for widget layout will stored in both

Faculty User:
- Will continue to use Blackboard as they have in the past. This is definitely a branch that could be explored to expand the system, but is not within the intended scope of the project. Future improvements to the system? Currently let's just focus on student interfaces, since we are most familiar with the issues they face.

Design

- The Miner. This is a back end interface to Blackboard. This gets the ticket from the Content Database, pulls the required information out of Blackboard, and populates the Content Database with class information. This will probably be done with a combination of curl, Node, and possibly some extra Python to hack things together.

- The Content Database. This holds the user's collection of class data; course names, links to documents, announcements, and other data will be held here. This will be held in a session and be refilled every time the user visits PAL. This will probably be implemented with SQLite. This needs to be able to talk to both the Builder and the Miner. It will receive the ticket from the Builder, give the ticket to Miner, accepts class information from the Miner, and gives it all to the Builder. The precise format of this database will depend on which widgets we end up using, which will be influenced by the results of the Faculty survey we sent out.

- The Settings Database. This holds information about the user's personal settings for PAL and some data on past activity. Information stored here will be locations and sizes of widgets, number of links in Assignments when the user last logged in (to allow for alerting the user to changes in content). This will probably be implemented with MySQL. This will have a two tables: one for declaring which widgets the users currently uses, and another for specifying the properties (size, location, type) of the widget. This will accept information from the Processor and give information as requested to the Builder.

- The Builder. This constructs the page that the user will see. Draws on data from the Content Database and the Settings Database to render the HTML/JS page sent to the user's browser. This is the central unit of operation. It redirects the user to CAS if no previous login cookie is detected. It gives new tickets to the Content DB, pulls class information from the Content DB, and sends a rendered HTML/JS page to the user's Browser. This will need something to be able to reflow the page; JS will hopefully be enough. If it isn't, Ajax or even an upgrade to GWT will provide access to prefabricated widget classes.

- The Processor. This receives information sent from the user's interactions with PAL and passes it along to the Relay. It will accept user input from the browser, update information in the Settings DB, and pass along relevant information to the Relay.

- The Relay. This sends information passed from the Processor from the user's interactions all the way back to Blackboard. This is necessary for activities such as submitting assignments, making discussion board posts, writing blog posts. This will be the last module implemented. This will probably be the hardest to implement.

We decided to use a private GitHub repository for version control. Where not otherwise specified, we will use Python with Django to build the website components.

No administrative system is necessary through the browser because almost all the information originates in Blackboard. If we really need to fix something, we can simply go straight to the server and hunt around.

During development, project releases will be hosted and tested on CS servers. If all goes well, it will get moved onto USG servers or wherever things like ICE and Point are hosted.

Milestones

- All team members have working environments to write, test, and run code on their local machines. (Done!).
- Get whitelisted by CAS.
- Acquire a domain. (Preferably pal.tigerapps.org).
- PAL redirects the user to CAS where she logs in. Upon returnign to PAL, she is presented with a page with their ticket number and/or her netID. This represents a fully CAS-interacting system.
- PAL presents the contents of a Content DB that was built with a toy Miner that mines a well-defined website. This proves that the Content DB is up and running.
- PAL has the user/tester log in and presents a textual representation of all of his classes. This represents a fully functioning Miner, fully functioning Content DB, and basic Builder; i.e. the flow of information out of Blackboard is successful.
- PAL can have the user (after logging in through CAS) write some text and have it presented to them the next time they visit. This represents an operating set of interactions between Processor, Settings DB, and Builder.
- PAL is able to pass along assignment submissions and discussion board posts. This represents functioning flow of information from the user all the way back to Blackboard.
- Talk to USG about incorporating ICE as a bubble-wrapped feature of PAL.

Risk + Open Issues

- Sending information back to Blackboard. Some classes will have assignments submitted via Blackboard. Implementing this feature in PAL will require active interaction with Blackboard on top of the passive mining required for everything else.
- Permissibility of Mining. It is possible that the University and/or Blackboard will not allow us to mine sensitive information such as grades or copyrighted materials. If this is the case, then we will either need to drop those features entirely, something that would hurt the functionality of our product; or we will need to find a workaround that will not violate restrictions placed on us.

- Copy-righted features. Related to the previous issue, Blackboard has the capability of only allowing some computers access to video reserves. We do not know how it does this.
- Learning to use all the tools.