May 7, 2009

School of Information Final Project

Spring 2009

Final Project Writeup

Michael Lee & Seung-Hyun Rhee

Advisor: Robert Glushko, PhD

School of Information

Table of Contents

1. Introduction

This report summarizes a 15-month project at the UC Berkeley School of Information (iSchool) carried out to satisfy requirements for two graduate courses, Document Engineering and Information Systems and Service Design (ISSD), and to meet a final project graduation requirement. The project originally examined the some of the problems with the course catalog and registration systems provided by UC Berkeley. As actual students and users of these systems, we had firsthand experience dealing with the difficulty of performing typical academic student services such as looking up courses, checking registration times, and registering for courses. To validate our hypothesis that academic student services were lacking, we had numerous interviews with current students and staff members, and conducted a survey of iSchool students to determine their opinions about the existing services and applications.

For the final project, we extend these initial observations and suggest a more ambitious idea - creating an application platform. The application platform is a place where any user would be able to develop student/course-related applications by drawing data from legacy systems and sharing their code with other users. The rationale behind this idea come from the observation of the ephemeral nature of many student projects developed on campus that start off strong but fail to gain wide adoption in the long run. While the concepts and analysis were solid in these past projects, there were always limitations on reflecting what users really wanted since each team had to scope their project into a manageable chunk. With a top-down approach for understanding users' requirement, designers may overlook points which should have been considered. Therefore, we realized that this approach may not be the best way to improve the inefficiency in the current legacy systems. We needed to think of a new way to improve the

system as a whole instead of trying to solve specific problems which may or may not be relevant or useful for everyone using the system. With our proposed idea, we hope that the system design will provide new ways of thinking about and improving or creating new campus applications. With this new system, users will be able to collaborate and build on each others' work since they would be working with the same underlying architecture to access the data needed to create their applications. We hope this will lead to wide adoption of the system which will make it sustainable and allow for continuous innovation. We will call the application platform, "Courseland."

This report will give detailed information about the issues and design decisions made in the creation of Courseland. The upcoming section will define the problem space of the current campus systems. The following section will define Courseland and list out its benefits. Next, the challenges in designing Courseland will be discussed. Finally, we will describe the design of Courseland and some examples of applications which are built using its API.

2. Problem Statement

2.1 A Lack of Useful Applications

Staff and students that were interviewed pointed out that the current academic affair information service systems do not provide additional applications that would be useful. Example applications that were requested often included a course recommender and/or a course visualizer. In one of our interviews, a master's program academic adviser stated that students were constantly asking her and fellow students for recommended courses that could fulfill their elective requirements. Consequently, students developed a "low-tech" solution consisting of a spreadsheet with a list of recommended courses that was updated and passed

around at the end of each semester. In our student interviews, many students complained about the lack of a calendar visualization to help them plan out their semester while getting ready to schedule for courses. Student solutions to this problem ranged from using online calendaring programs (e.g. Google calendar) to drawing schedules out on paper to help them visualize their potential weekly schedule(s). In brief, while there are a remarkable number of information systems on campus, users still need useful services which legacy systems do not support.

2.2 Dissatisfaction with Current Legacy System Applications

According to our student survey, 60% of iSchool students are either unsatisfied or very unsatisfied with currently available academic information services. These legacy systems include Telebears, Bearfacts, CARS, Course Catalog, etc. These systems do provide users with essential functions such as course registration, grade checking, and tuition payment. However, users continually report dissatisfaction with these systems due to two main reasons. First, these systems are not integrated,, which in turn requires users to log in to each of them separately. This is seen as tedious and often causes confusion since a user has to remember where they have to log in to do a specific task. For example, a student has to log in to Bearsfacts to check their grades, but has to log in to Telebears to register or modify courses. Second, these systems have overlapping and somewhat inconsistent functionality.. Related to the first problem, a student has to log in to multiple systems to do seemingly related tasks. For example, students have to log in to Bearfacts to check their registration time. However, to register, they have to log into Telebears. Furthermore, in order to register for a course, a student must have the CCN (course control number) which is found on the course catalog (or departmental website). As a result of these issues, most of the students we interviewed

expressed their inconvenience with the legacy systems. Furthermore, these problems are not only experienced by students, but also extends to staff. One undergraduate adviser we interviewed stated that the single-most useful feature that would benefit her would be to have single-sign-on that would allow her to access everything in one place.

2.3 Inability to Influence Change and Creation

While there have been several attempts to remedy the usability and functional problems with the current legacy systems, the great majority of these projects have not been successfully adopted or integrated for common use. Even though significant effort and resources are spent by teams to develop new tools by attempting to understand the preferences and incentives of users, many of these projects are not successfully disseminated or incorporated into existing systems as shown by past iSchool and other campus projects. In other words, a top-down approach to system development or improvement has resulted in limitations of wide adoption by the broader user community.

We analyze reasons for the failure of wide adoption in previous projects. First, many of these past projects are very usability-oriented. They did thorough usability analysis by analyzing user needs through interviews and surveys, creating intuitive user-interfaces, and iterating through and testing prototypes. However, many of these projects were superficial fixes to problems, and they did not think seriously about how to interoperate their with legacy systems - the source of data. Too much bias focusing on usability without looking too much at the source of data leads to no access to richer information, which would make applications more functional. In addition, using these new solutions typically required the design and creation of a separate database. If developers are less motivated to provide their personal

information for systems, those systems miss opportunities to capitalize on information which could have been acquired from currently existing systems. Therefore, usability-oriented projects did not necessarily meet users' satisfaction.

Another reason for failure for wide adoption could be the other extreme of the spectrum from the first example. Some projects focused heavily on data interoperability with existing systems. However, highly technology-oriented projects may overlook a user-friendly interface and therefore may be difficult or cumbersome for use.

For example, SylViA, is a syllabus management application developed at the iSchool and recipient of the prestigious Chen Award. Though the application was very powerful, only a handful of faculty members chose to use it, with other faculty members citing several reasons for not using it including: requiring an additional investment in time to write the syllabus into the right format, and it being technically difficult to transcribe. In this case, the lack of a user-friendly interface to easily create syllabi led to poor adoption.

The final problem we identified is that typically, team members of worthwhile projects either finish with their course requirements or graduate without having anyone to continue supporting or working on their project. For example, the developers of SylViA have graduated and therefore cannot continue to improve SylViA even though there were obvious ways they could improve the program and gain wider support. Consequently, the discontinuity of system maintenance also contributes to the failure of systems adoption.

2.4 Our Suggestion: Courseland

In brief, previous methods of using a top-down approach attempting to address the lack of

useful applications in system development have illustrated the difficulty in the successfuly adoption of student projects. Instead, we suggest a bottom-up approach for building campus applications. In other words, providing users the power to develop and improve applications by providing a common data source can help facilitate wider usage and create a broad user community.

Therefore, we suggest Courseland!

3. Why Courseland?

3.1 Definition

What exactly is Courseland? We define it as: *an application platform serving course and student data, allowing for the development of new or improved academic service applications by users.*

3.2 Benefits

However, why is Courseland necessary? Courseland is characterized by a bottom-up approach to application development. It allows users to create course and student-related applications on their own by providing student and course information from one location using a common set of instructions. This is enabled by democratizing data and source code for consumption. In other words, a certain level of access to student and course data, which was previously unavailable, allows users to review and remix the data, leading the improvement of existing applications, or creation of new applications altogether.

Judith Estrin classified innovation into: breakthrough innovation, orthogonal innovation and

incremental innovation. Breakthrough innovation is the creation of new things that have never existed before. Orthogonal innovation is organizing existing elements in creative ways and creating new things. Incremental innovation is improving existing products and services with better input.

Above all, Courseland supports incremental innovation. The initial Courseland apps will support schedule planning and registration, the most difficult things in the current legacy architecture. For example, putting additional data into a calendar, a calendar application can provide campus schedule with richer information.

Courseland enables orthogonal innovation. Users can create applications through new ways of organizing student and course data. For example, there is Lazy student's scheduler, which is targeted at students who like to take courses in close proximity to each other. If developers can acquire information on course name, CCN, and building location, it can be laid out on Google Map, which in turn effectively shows distance among campus buildings for courses. Another application example is Frugal student's book finder, which is for students who would like to take courses and spend the minimum dollar cost for text and readings. This can be done either by looking for courses using the books from previously enrolled courses or looking for courses with no readings or cheaper readings. If developers can acquire information on course name, readings (ISBN), it can connect to book-retailers such as Amazon to show how much a particular semester schedule will cost.

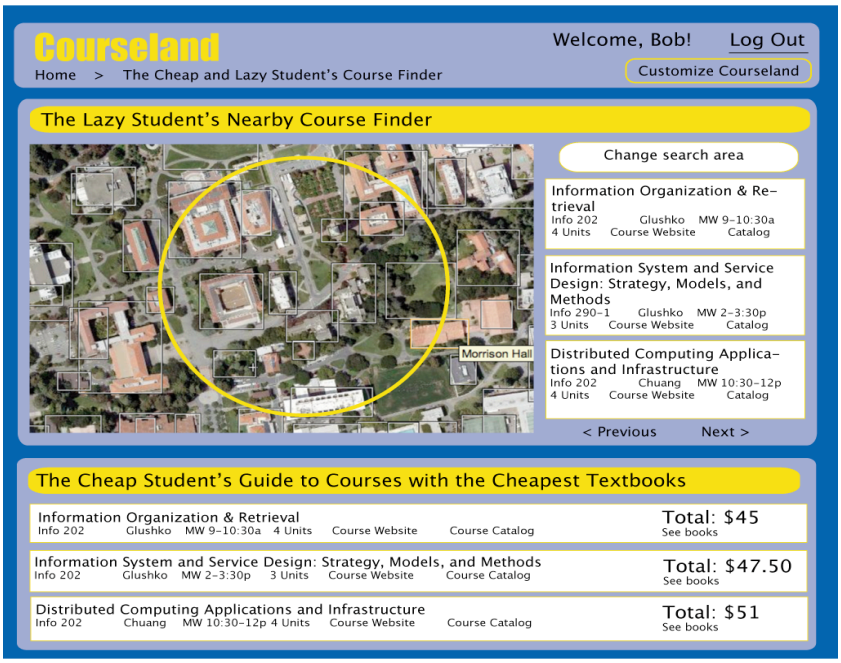

Figure 1. *Book Price Comparison and Nearby Course Finder*

Most significantly, these examples illustrate possible applications that can be built on Courseland using the data it provides. Courseland is not limited to these ideas, but instead allows users to create any kind if application using the data Courseland provides.

4. Challenges for Courseland

What issues need to be resolved in designing and building Courseland? We found three main relevant issues: technical, motivational and legal issues.

4.1 Technical: Dispersed, Heterogeneous Systems

Courseland should be able to provide necessary data easily from one location so that users can have a common platform where they can develop applications. However, current legacy systems are dispered and have hetereogenous data that may or may not be shared or synchronized across systems. To address this, we considered two approaches for Courseland

to provide s single point of data access for developers. 1) Courseland examines the data structure of all the existing systems and consolidate their data structure into Courseland. Courseland would then become interoperable with all the systems and would be able to pull necessary data from any of them. Or 2), all the existing systems are integrated into a single place first. Courseland would be able to get access to data using this consolidated data repository.

For the first approach, we could analyze distinctive data structures (e.g. investigating data field name, format, etc.) and reflect the analysis into our single, consolidated data structure. We conducted this type of data analysis in our Spring 2008 Document Engineering course which looked at data from Telebears, Bearfacts, Management of Technology course list and iSchool course list (please see Appendix A for this work). Once our data structure becomes consolidated and interoperable with other systems, standardized documents can be to share data across different systems. However, if there is any change in any of the existing systems but this change is not shared with Courseland, data interoperability would break.

On the other hand, the second approach is beyond our control. We cannot put all the campus systems together in order to communicate with Courseland. However, it is obvious that if there is a single place where student and course data are standardized and shared each other, Courseland's single data access for users will become dramatically simple.

Nevertheless, we take the second approach. Currently, there is a campus-wide consortium, Kuali, that is integrating all the campus systems into a single space. Kuali is standardizing separate data structure in each of those systems and consolidating them so that all the academic affairs are stored, managed and processed. Since Kuali will replace Bearfacts,

Telebears, CARS, etc., it is desirable for interface directly to Kuali, rather than choosing the first approach. As a result, Courseland will be interoperable with Kuali. More detailed information addressing interoperability issues will appear in Section 5.

4.2 Motivational: Interest in Creating Applications

For Courseland to be successful, it must encourage development of new applications. Our survey in Appendix C showed that 34% of iSchool students would be willing to create applications using an easy-to understand API. Given that iSchool students are technically savvy and lead users for advanced web applications, 34% is a promising number.

More specifically, Eric von Hippel analyzed three main motivations of open source software developers: 1) user's direct need, 2) enjoyment of work itself and 3) enhanced reputation. Once one of them is met, developers are willing to participate in open source software development. According to our survey for iSchool students, 34% of respondents said they wanted to see better applications; 46% of respondents said they spend time outside of work/ studies for development; 29% of respondents said they were motivated by being identified as the author of a particular piece of software. Therefore, specific motivations also support the critical mass of developers for Courseland.

4.3 Legal Issues: Limitation on Data Access by Privacy Issues

Even though Courseland provides access to student and course data for users, some of the information in Kuali is restricted and/or protected by law. Information that can lead to the identification of any single, specific students would not be allowed. Home address, e-mail, and grades are examples of identifying information that is protected. However, the owner of

the information is always entitled to request and receive their own information.

This restriction is less applicable for aggregated information. For example, retrieving a list of the top classes taken by iSchool students would be okay since the data would not be able to be used to identify a specific student. Simply, Courseland cannot identify a specific, individual user, but can use aggregated data.

More specifically, we give a guideline on scope of information accessible based on the privacy issue.

Table 1 - *Scope of Data Accessibility*

	Students	**Course**
Individual	Definition – any information that can be used to identify a specific individual Example – student name, grade, courses, departments, etc. Accessibility – any individual student information is not allowed to acquire from Kuali unless it is by the owner of that information.	Definition – any information on an individual course with identifiable keys available through course catalog or public-facing registrar website Example – Professor Bob Glushko's INFO 202 in Spring 2008, a course at Monday 9am at South Hall 202 Accessibility – any individual course information can be shared

Aggregate	Definition – any information that shows a certain chosen group's quantitative or qualitative characteristics Example – geographic distribution of iSchool class 2009, average grade of INFO 203 in Spring 2008 Accessibility – All aggregated data is allowed as long as no individual can be identified.	Definition – any information that shows collection of individual courses with common characteristics Example – most popular courses of iSchool's first-year students in Spring 2009 Accessibility – any aggregate course information is allowed to be retrieved from Kuali

Some applications mix student and course data at the same time. Any applications with individual student information are not permitted to be created. The other combinations of individual course data, aggregate student data and aggregate course data are legally allowed. Consequently, this guideline will in turn define types of applications which can be built on Courseland. For example, if a developer wants to create an application (i.e. mentor finder) which finds a mentor who has taken interesting courses to junior students, he cannot access to data on course history of individual students. So, the mentor finder is unable to search for individual mentors. On the other hand, if a developer builds an application (i.e. course recommender) where a student is recommended to other courses to take based on their similarity to other students with similar course schedules, the course recommender would be

allowed to gather data on aggregate student information to gauge similarity among students. Therefore, this guideline is useful for developers to judge the feasibility of their applications in a legal aspect.

5. Courseland Design

Once these issues are resolved, our next step is designing Courseland. Our design issues include how to interoperate with Kuali and how to provide data access for users. Another issue is to resolve why users should would use Courseland instead of accessing Kuali directly. To put it simply, the answer to these issues is to include a default application in Courseland where users can generate their own data - which adds value to the existing data - allowing the creation of a broader ranges of applications.

5.1 Interoperability with Kuali

Obviously, Courseland should be able to retrieve desirable results by intergrating with Kuali. The interoperability issue is a key point in designing Courseland since it needs to establish a streamlined data flow between Kuali and the user.

How does Courseland communicate with Kuali? We had several meetings with Kuali developers and were provided with their database schemas on students and courses. (Please see Appendix B. for the complete list of data schema of Kuali). Courseland accesses available information directly from Kuali, so it does not create its own, duplicate data of Kuali's database. Likewise, Courseland delivers the data it retrieves directly to the users. Therefore, the interoperability between the systems and standardized output for consumption become important issues. These issue will be discussed in the next section.

5.2 Single Information Gateway through API

How does Courseland interface with Kuali and deliver the appropriate data to developers? The answer is to provide an Application Programmer Interface (API) for developers. Developers can use a specific, well-documented API to run functions and retrieve data in a standardized format. For example, if users want to acquire all of iSchool's course data, they could use the following command as defined by the API: getCoursesByDepartment("iSchool"). Another example includes searching for courses in a certain building. Then, users could use the following command as defined by the API: getCoursesByBuilding ("Wheeler Hall").

As with any large database, Kauli is built with specific rules, associations, data types, and table names. The Courseland API serves to abstract away the specific commands necessary to access particular data within Kuali. by providing a documented list of commands that will return the appropriate data. Calling the API function will run the necessary query and logic, interfacing with Kuali, and then in turn transform the returned data into a standardized XML message that can be used by the developer. Figure ?? shows the simple exchange of XML messages for API getCoursesByDepartment("iSchool").

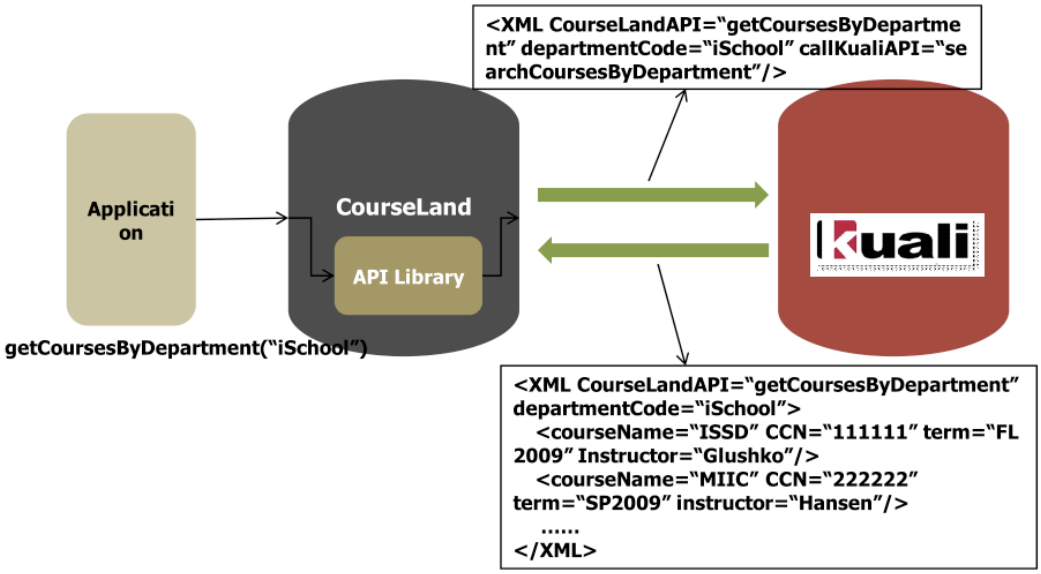

Figure 2. *Exchange of XML Messages between Courseland and Kuali*

Appendix C will show our API collections which include each function and its XML message for sending and receiving data.

On the other hand, as we have discussed, all the data is not allowed to be retrieved. For example, individual student records are not permitted for users to search. Based on our Chapter 4.3 guideline, the scope of data available through the Courseland API is definitely (and purposely) limited. In other words, our API does not support retrieving individual student records, but enables accessing aggregate student data and individual and aggregate course data, as defined in Section 4.3. The list of our API in Appendix C also reflects this legal limitation.

5.3 Courseland Default Application: Profile Management

While Courseland is an application platform, it also includes a default application for two two specific reasons. First, during one of our meetings with Kuali developers, they raised an important question: what incentives would Courseland provide that would motivate users to use it over Kuali? Our answer came from what value we could add to Kuali's existing data so that developers could create more valuable applications. Adding extra information that is unavailable on Kuali to our own databases (e.g. job interests, work experience, etc.) would make Courseland more attractive to use. Therefore, our default application would allow users to input additional personal data on top of existing Kuali data, adding value to the overall system.

Secondly, our default application widens the scope of available data. As discussed earlier, Kuali's data access is limited by issues of privacy and data protection. Information that could lead to the identification of a specific individual is not allowed. However, voluntarily submitted, user-generated data would not have such strict restrictions. Therefore, the gathering of user-generated information through the default application enlarges the scope of information available by developers.

As a result of the two examples illustrated above, Courseland will provide its own default application. We call it "Profile Management," because it manages each student's profile data such as student name, major, courses taken, address, etc. In addition, Profile Management will have additional information that is unavailable in Kuali such as: academic interests, work experience, job interests.

Figure 3 presents the abstract Courseland Architecture, based on our discussion in Chapter 5.

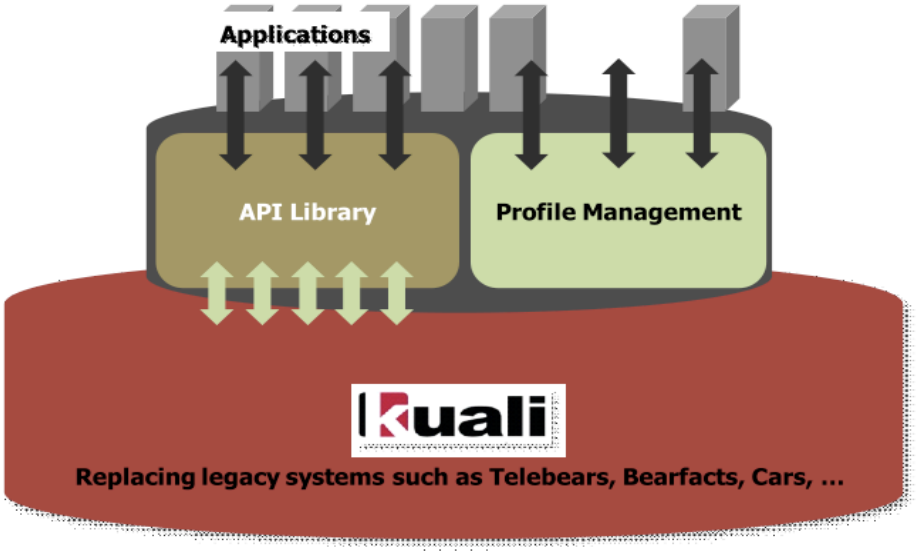

Figure 3. *Abstract Courseland Architecture*

6. Application Examples

From here, we will show some examples of applications built on Courseland. Through those examples, we will also demonstrate how to leverage data using the "Profile Management" application and how to access Kuali's data through our API.

1) Profile Management:

The Profile Management application will add additional value to the information being provided by Kuali. Students will be able to insert their own data, enriching the information available to developers. Additional data the students could provide include: Academic interests, Job experience, and Job interests. These data fields can be extended in future iterations to add additional value if needed.

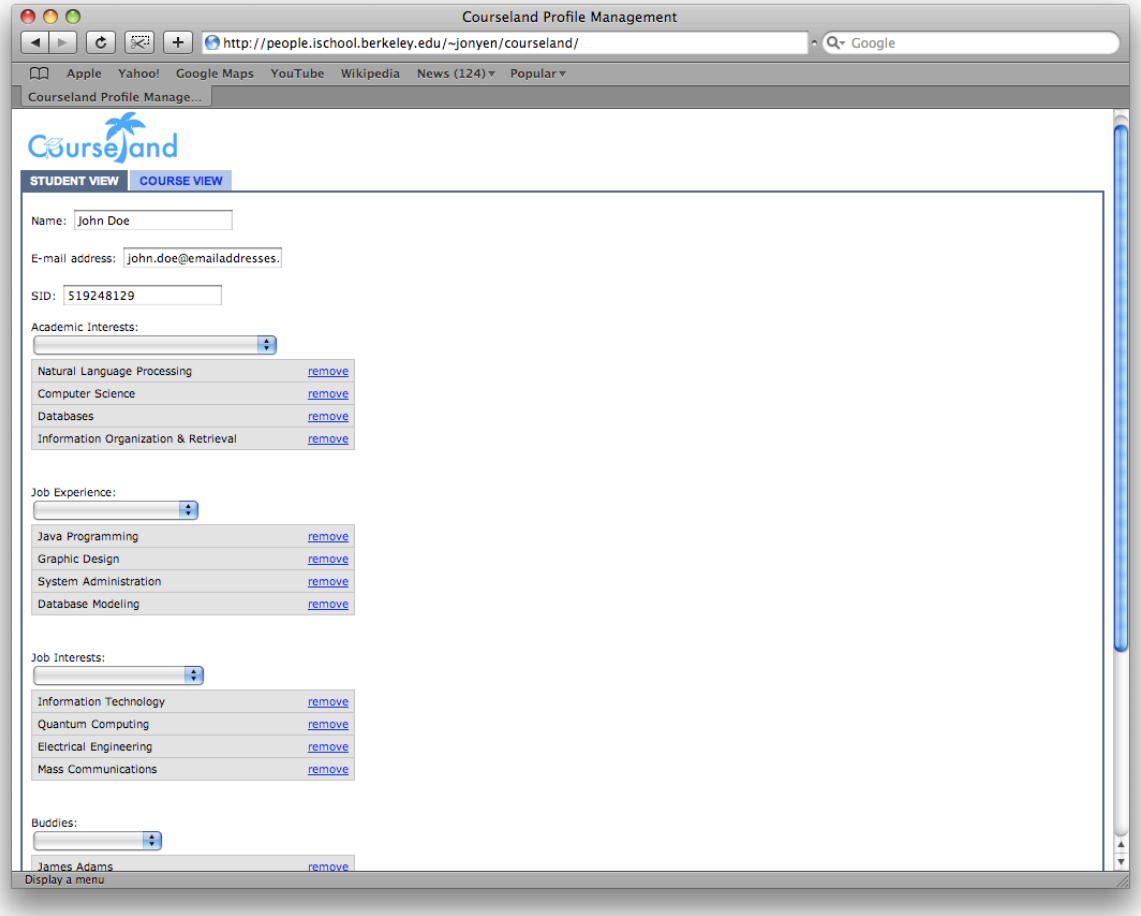

Figure 4. *Screenshot of the Student Profile Management Prototype*

2) Course Recommender:

The course recommender is envisioned to act much like the popular Amazon.com's book recommender. Through this kind of collaborative filtering, either through actual taken courses or reported academic interests, students looking for classes would be better informed of their choices using this application.

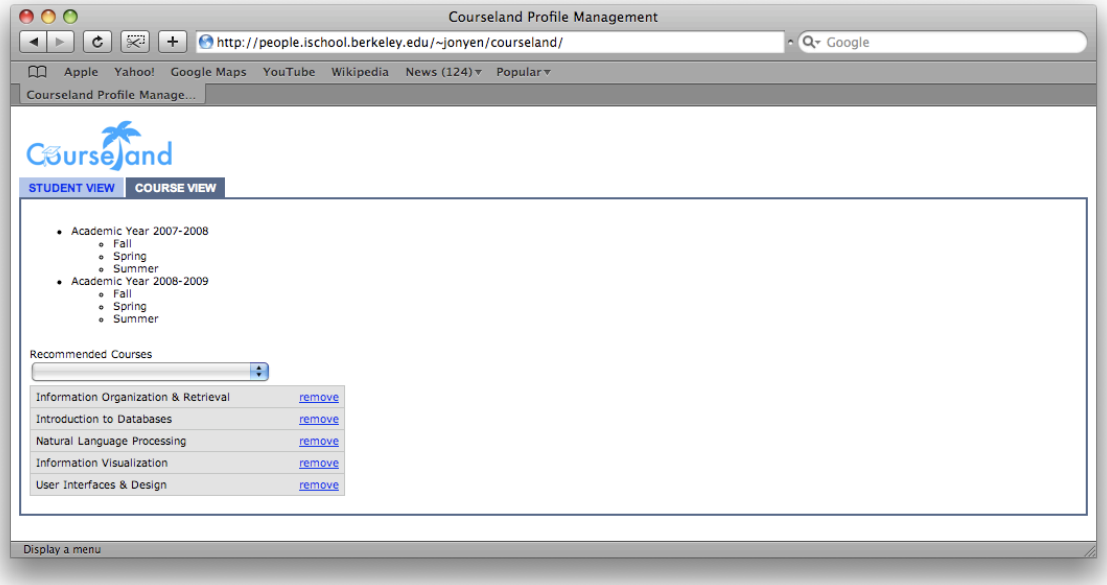

Figure 5. *Screenshot of the Course Recommender Prototype*

3) Course Plotter

The course plotter is a visualization of the location of student's courses using a mashup of the Google Maps API and the Courseland API.

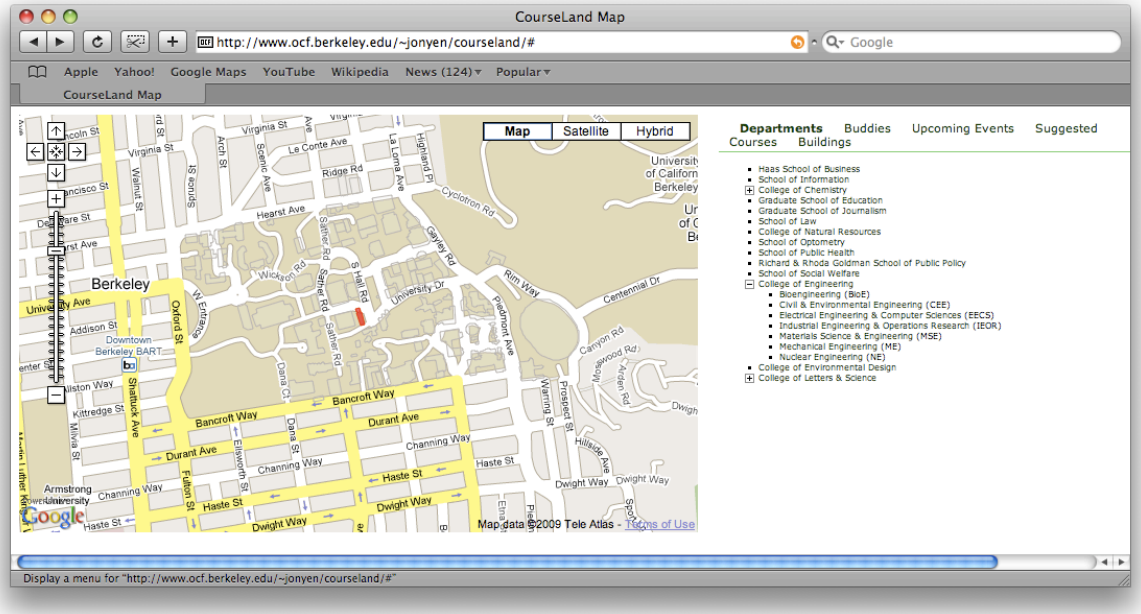

Figure 6. *Screenshot of the Current Course Plotting Prototype*

7. References

[1] Judy Estrin, Closing the Innovation Gap: Reigniting the Spark of Creativity in a Global Economy, McGraw-Hill, 2008

[2] Kuali Student, http://student.kuali.org

[3] Karim Lakhani and Eric von Hippel, How Open Source Software Works: Free User-to-User Assistance,' *Research Policy,* Vol. 32, No. 6, 2003, pp. 923-943

[4] SylViA Project, http://groups.ischool.berkeley.edu/sylvia/

8. Acknowledgements

First and foremost, we would like to thank our academic advisor, Bob Glushko, for all of his help, dedication, and insight. Next, we would like to thank the Kuali Student development team for support and willingness to meet with us regularly.

A big thanks also goes to Bears Breaking Boundaries for the consideration and financial support of our project - particularly Tom Kalil and Annie Yeh.

Finally, legal advice was received from Tomoki Ishihara, and development was provided by Jonathan Yen.

9. Appendix

A. Data Consolidation Work

In Spring 2008 Document Engineering, we have attempted to analyze course catalog and graduate requirement check in existing systems and then consolidate them into a standardized data structure. This efforts facilitated understanding the complexity of standaridization in existing systems and led to the decision of direct interoperability with Kuali.

1) Data Analysis in Course Catalog and Graduate Requirment Check

1.1 iSchool website				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Differentiation	CrossListing	Type of course: graduate/ special topics/ cross listed MOT/ seminar/ undergraduate	string	
Course number	CourseNumber		string	[department] + [1-199 undergrad; 200-299 masters]
Course name	CourseName		string	
Course description	CourseDescription		string	
Semester	Semester		string	
Year	Year		int	
Instructor(s)	Instructor		string	
Time	DayOfWeek	days of the week	string	
Time	Time	time	string	
Location	Location		string	
CCN	CourseControlNumber		int	5-digits
Course website	CourseWebsite		URI	
1.2 General catalog				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Course	CourseNumber		string	[department] + [1-199 undergrad; 200-299 masters]
Course Title	CourseName		string	
Location	Location		string	
Location	DayOfWeek			
Location	Time			
Instuctor	Instructor		string	
Status/Last Changed	LastChanged		string	
CCN	CourseControlNumber		int	5-digits
Units/Credit	Units		int	range of numbers
Final exam group	FinalExamGroup		string	
Restrictions	RegistrationRestriction		string	
Note	GeneralCatalogNote		string	

Enrollment on when	CurrentEnrollmentStatus		string	
1.3 MOT Website				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Differentiation	CrossListing	Type of course: core/ related	string	
Course name	CourseName		string	
Course number	CourseNumber		int	
Units of Credit	Units		int	
Time	DayOfWeek	days of the week	string	
Time	Time	time		
Location	Location		string	
CCN	CourseControlNumber		int	5-digits
Note	MotWebsiteNote		string	
2.1 Master's Handbook				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Degree requirements	DegreeRequirementDescription		string	
Length of the MIMS Program	LengthDescription		string	
iSchool course requirement description	CourseRequirementDescription		string	
iSchool course requirement course title	RequirementCourseTitle		string	
iSchool course requirement course number	RequirementCourseNumber		int	
Plan A description	PlanADescription		string	
Plan B description	PlanBDescription		string	
Technical requirement description	TechnicalRequirementDescription		string	
Technical requirement course title	TechnicalRequirementCourseTitle		string	
Technical requirement course number	TechnicalRequirementCourseNumber		int	
Project management / Systems analysis requirement description	ProjectManagementRequirementDescrip		string	
Project management / Systems analysis course number	ProjectManagementRequirementCourse		int	

Project management / Systems analysis course title	ProjectManagementRequirementCourse		string	
Transfer of units	TransferOfUnitsDescription		string	
Grade requirements	GradeRequirementDescription		string	
Successful completion of required courses	SuccessfulCompletionDescription		string	
Filing for the degree	FilingForTheDegreeDescription		string	
Faculty counselors	FacultyCounselorDescription		string	
Graduate advisers	GraduateAdvisorDescription		string	
3.1 Advising Form				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Semester	Semester	Semester when form is being filled out	string	Only "Fall"
Year	Year		int	Two-digit Year
Name	StudentName		string	
SID	StudentID		int	8 digits
Course Title	CourseFullName	Full course name, Course number + course title	string	Should conform to official name
CCN	CourseControlNumber		int	5 digits
Units	Units		int	> 0
Signature (Student)	SignatureStudent		string	
Signature Date (Student)	SignatureStudentDate		date	
Advisor Name	FacultyAdvisorName		string	
Signature (Advisor)	SignatureFacultyAdvisor		string	
Signature Date (Advisor)	SignatureFacultyAdvisorDate		date	
Advisor's Notes	FacultyAdvisorNotes	Freeform box for faculty advisor to add notes	string	
Instructions	InstructionClause	Additional instructions	string	
3.2 MIMS Checklist				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Student Name	StudentName		string	
Date	SubmissionDate		date	
Advisor's Initials	FacultyAdvisorInitial		string	optional

Date Initialized	FacultyAdvisorInitialDate		date	optional
Checkbox - Core Coursework	Checkbox		boolean	
Course Name - Core Coursework	CourseFullName	Full course name, Course number + course title	string	
Term - Core Coursework	Term		string	
Grade - Core Coursework	Grade		string	
Units - Core Coursework	Units		int	
Waived	Waived	Was I206 waived?	boolean	
Checkbox - Technical Requirement	Checkbox		boolean	
Course Name - Technical Requirement	CourseFullName	Full course name, Course number + course title	string	
Term - Technical Requirement	Term		string	
Grade - Technical Requirement	Grade		string	
Units - Technical Requirement	Units		int	
Course Title (EECS) - Technical Requirement	CourseFullName	Full course name, Course number + course title	int	
Section (EECS) - Technical Requirement	SectionNumber		int	
Term (EECS) - Technical Requirement	Term		string	
Grade (EECS) - Technical Requirement	Grade		string	
Units (EECS) - Technical Requirement	Units		int	
Checkbox - Project Management Requirements	Checkbox		boolean	
Course Name - Project Management Requirements	CourseFullName	Full course name, Course number + course title	string	
Term - Project Management Requirements	Term		string	
Grade - Project Management Requirements	Grade		string	
Units - Project Management Requirements	Units		int	

Course Title (Alternative) - Project Management Requirements	CourseFullName	Full course name, Course number + course title	int	
Section (Alternative) - Project Management Requirements	SectionNumber		int	
Term (Alternative) - Project Management Requirements	Term		string	
Grade (Alternative) - Project Management Requirements	Grade		string	
Units (Alternative) - Project Management Requirements	Units		int	
Checkbox - Final Project	Checkbox		boolean	
Course Name - Final Project	CourseFullName	Full course name, Course number + course title	string	
Term - Final Project	Term		string	
Grade - Final Project	Grade		string	
Units - Final Project	Units		int	
Course Title - 290	CourseFullName	Full course name, Course number + course title	string	
Section - 290	SectionNumber		int	
Term - 290	Term		string	
Grade - 290	Grade		string	
Units - 290	Units		int	
Course Number - Outside	CourseNumber		int	
Course Title - Outside	CourseName		string	
Section - Outside	SectionNumber		int	
Term - Outside	Term		string	
Grade - Outside	Grade		string	
Units - Outside	Units		int	
Fall Unit Count - 1st Year	SumYearOneFallUnits		int	
Spring Unit Count - 1st Year	SumYearOneSpringUnits		int	
Fall Unit Count - 2nd Year	SumYearTwoFallUnits		int	
Spring Unit Count - 2nd Year	SumYearTwoSpringUnits		int	
Outside Unit Count	SumOutsideUnits		int	
S/U Unit Count	SumSatifactoryUnits		int	
297 Unit Count	Sum297Units		int	
299 Unit Count	Sum299Units		int	

3.3 ACMD				
Physical Name	**Logical Name**	**Definition**	**Datatype**	**Restriction**
Plan I Checkbox - Check Appropriate Box	Checkbox		boolean	
Plan II Checkbox - Check Appropriate Box	Checkbox		boolean	
Degree	DegreeTitle			
Major	DegreeMajor			
Checkbox - Approved Degree	Checkbox	Approved concurrent degree?	boolean	
Degree Program - Approved Degree	ConcurrentDegreeName	If so, what is the title?	string	
SID	StudentID		int	
Name	StudentName		string	
E-mail	StudentEmail		string	
Address	StudentAddress		string	
Phone Number	StudentPhoneNumber		string	
Semester (Internal) - Completed Course List	Semester		string	
Course (Internal) - Completed Course List	CourseFullName	Full course name, Course number + course title	string	
Unit (Internal) - Completed Course List	Units		int	
Grade (Internal) - Completed Course List	Grade		string	
Semester (External) - Completed Course List	Semester		string	"Fall" or "Spring"
Course (External) - Completed Course List	CourseFullName	Full course name, Course number + course title	string	
Unit (External) - Completed Course List	Units		int	
Grade (External) - Completed Course List	Grade		string	
Student Signature	SignatureStudent		string	
Student Signature Date	SignatureStudentDate		date	

Head Advisor Signature	SignatureHeadAdvisor		string	
Head Advisor Signature Date	SignatureHeadAdvisorDate		date	
Residency - Graduate Division Only	GradDivisionResidency		string	
GPA - Graduate Division Only	GradDivisionGPA		float	
Approval Date - Graduate Division Only	GradDivisionApprovalDate		date	
Approved By - Graduate Division Only	GradDivisionApprovalStaffName		string	
Filing Fee - Graduate Division Only	GradDivisionFilingFee		float	
Expiration Date - Graduate Division Only	GradDivisionExpirationDate		date	
Final Exam - Graduate Division Only	GradDivisionFinalExam		string	
Certificate Issued Date - Graduate Division Only	GradDivisionCertificateIssueDate		date	

2) Data Consolidation in Course Catalog and Graduate Requirement Check

Data	*Source*	*Type*	*Desc*	*Changes*	*Reasons*
<Course Catalog>					
CourseNumber	1.1, 1.2, 1.3	string	[department] + [1-199 undergrad; 200-299 masters]		
CourseName	1.1, 1.2, 1.3	string			
Instructor	1.1, 1.2, 1.3	string			
Location	1.1, 1.2, 1.3	string			
CCN	1.1, 1.2, 1.3	int			
Units	1.1, 1.2, 1.3	int			
~~DayofWeek~~	1.1, 1.2, 1.3			Removed/Renamed	
DayofInstruction	1.1, 1.2, 1.3	string		New; Renamed from "DaysofWeek"	
Time	1.1, 1.2, 1.3	time			
~~ISchoolDifferentiation~~	1.1			Merged into "Alternate CourseName"	This field was actually talking about dept. crosslisting. A new field has been made for it.
~~MotDifferentiation~~	1.3			Merged into "Alternate CourseName"	This field was actually talking about dept. crosslisting. A new field has been made for it.
AlternateCourseName	1.1, 1.3	string		New	
CourseDescription	1.1	string			
~~LinkToCatalogDescription~~	1.2			Removed	Links to a catalog description will not be needed in our product (i.e. it is the catalog)
~~GeneralCatalogNote~~	1.2			Merged into "AdditionalNote"	Additional notes were consolidated into a new field
~~MotWebsiteNote~~	1.3			Merged into "AdditionalNote"	Additional notes were consolidated into a new field
AdditionalNote	1.2, 1.3	string			
CourseWebsite	1.2	URI			
~~Semester~~	1.1			Merged into "Term"	Merged into a single term because these are always used together
~~Year~~	1.1			Merged into "Term"	Merged into a single term because these are always used together
Term	1.1	string		New	
~~LastChanged~~	1.2			Removed	Not necessary for our system
~~FinalExamGroup~~	1.2			Removed	Not necessary for our system
~~RegistrationRestriction~~	1.2			Removed/Renamed	Field name has been changed
CourseRestriction	1.2	string		New; Renamed from "CourseRestriction"	
~~CurrentEnrollmentStatus~~	1.2			Removed/Renamed	Field name has been changed
AvailableSeats	1.2	string		New; Renamed from "CurrentEnrollment"	
Fullfillment		string		New	
<Master's Handbook>					
~~DegreeRequirementDescription~~	1.2	string		Removed; description is redundant	
~~LengthDescription~~	1.2	string		Removed; description is redundant	
~~CourseRequirementDescription~~	1.2	string		Removed; description is redundant	
~~RequirementCourseTitle~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~RequirementCourseNumber~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~PlanADescription~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~PlanBDescription~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~TechnicalRequirementDescription~~	1.2	string		Removed; description is redundant	
~~TechnicalRequirementCourseTitle~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~TechnicalRequirementCourseNumber~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~ProjectManagementRequirementDesc~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~ProjectManagementRequirementCour~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~ProjectManagementRequirementCour~~	1.2	string		Removed; Merged into "Fullfillment_ISchool"	
~~TransferOfUnitsDescription~~	1.2	string		Removed; description is redundant	
~~GradeRequirementDescription~~	1.2	string		Removed; description is redundant	
~~SuccessfulCompletionDescription~~	1.2	string		Removed; description is redundant	
~~FilingForTheDegreeDescription~~	1.2	string		Removed; description is redundant	
~~FacultyCounselorDescription~~	1.2	string		Removed; description is redundant	
~~GraduateAdvisorDescription~~	1.2	string		Removed; description is redundant	

Fullfillment_ISchool	1.2	string	List or courses/requirements needed to graduate from ISchool	New	
<Course List Part>					
~~Year~~	~~3.1~~			Merged into 'Term'	Merged into a single term because these are always used together
~~Semester~~	~~3.1~~			Merged into 'Term'	Merged into a single term because these are always used together
~~Checkbox~~	3.2 (Core Coursework, Technical Requirement, Project Management Requirement, Final Project)			Removed	Not necessary for our system
Term	3.2 (Core Coursework, Technical Requirement, EECS Technical Requirement, Project Management Requirement, Alternative Project Management Requirement, Final Project, 290, Outside), 3.3 (Internal, External)	string			
~~CourseFullName~~	3.1, 3.2 (Core Coursework, Technical Requirement, EECS Technical Requirement, Project Management Requirement, Alternative Project Management Requirement, Final Project, 290, Outside), 3.3 (Internal, External)			Instead of CourseFullName, we are using CourseName and CourseNumber	Separated into two terms - course number and course name
SectionNumber	3.2 (EECS Technical Requirement, Alternative Project Management Requirement, 290, Outside)	string			
CourseNumber	3.2 (Outside)	string			
CourseName	3.2 (Outside)	string			
CCN	3.1	int			

Units	3.1, 3.2 (Core Coursework, Technical Requirement, EECS Technical Requirement, Project Management Requirement, Alternative Project Management Requirement, Final Project, 290, Outside), 3.3 (Internal, External)	int			
Grade	3.2 (Core Coursework, Technical Requirement, EECS Technical Requirement, Project Management Requirement, Alternative Project Management Requirement, Final Project, 290, Outside), 3.3 (Internal, External)	string			
~~Waived~~	3.2 (Core Coursework)			Removed	Not necessary for our system
<Course Summary Part>					
SumYearOneFallUnits	3.2	int			
SumYearOneSpringUnits	3.2	int			
SumYearTwoFallUnits	3.2	int			
SumYearTwoSpringFallUnits	3.2	int			
SumOutsideUnits	3.2	int			
SumSatisfactoryUnits	3.2	int			
Sum297Units	3.2	int			
Sum299Units	3.2	int			
<Student Part>					
~~ConcurrentDegreeName~~	3.3			Merged into 'DegreeTitle'	Merged into a single term
DegreeTitle	3.3	string	Using Code Sets (e.g. Arts, Science, Engineering)		
DegreeMajor	3.3	string			
Checkbox - Approved Degree	3.3	boolean	Approved concurrent degree?		
Plan I Checkbox - Check Appropriate Box	3.3	boolean			
Plan II Checkbox - Check Appropriate Box	3.3	boolean			
~~StudentName~~	3.1, 3.2, 3.3			Separated into StudentFirstName, StudentMiddleName and StudentLastName	Split into more granular level
StudentFirstName		string		New	
StudentMiddleName		string		New	

StudentLastName		string		New	
SID	3.1, 3.3	int			
StudentEmail	3.3	string			
StudentAddress	3.3	string			
StudentPhoneNumber	3.3	string			
~~Signature (Student)~~	3.1, 3.3 (SignatureStudent)			Removed and changed to confirmation checkbox	Not necessary for our system
Signature Date (Student)	3.1, 3.2 (SubmissionDate), 3.3 (SignatureStudentDate)			Automatically generated	
<Advisor Part>					
Advisor name	3.1	string			
~~FacultyAdvisorInitial~~	3.2			Removed	Redundant procedure
~~FacultyAdvisorInitialDate~~	3.2			Removed	Redundant procedure
~~Signature (Advisor)~~	3.1, 3.3 (SignatureHeadAdvisor)			Removed and changed to confirmation checkbox	Not necessary for our system
Signature Date (Advisor)	3.1, 3.3 (SignatureHeadAdvisorDate)			Automatically generated	
~~Adviser's Note~~	3.1			Merged into 'ExceptionNote'	Changed to semantically clear term
ExceptionNote		string		New	
~~Instructions~~	3.1			Removed	Not necessary for our system
GradDivisionResidency	3.3	string			
GradDivisionGPA	3.3	float			
GradDivisionApprovalDate	3.3	date			
GradDivisionApprovalStaffName	3.3	string			
GradDivisionFilingFee	3.3	string			
GradDivisionExpirationDate	3.3	date			
GradDivisionFinalExam	3.3	string			
GradDivisionCertificateIssueDate	3.3	string			

B. Current list of data schema of Kuali

Kuali is currently working on their data schema for courses across multiple universities. They have a concept of a “learning unit” which can be recursively imbedded in itself to add complexity. This documentatiom can be found on the internal Kuali Student website. Currently, Kauli has provided with an interim database (SQL) table layout structure that is being used for the UC Berkeley course/student data. The provided tables are not all-inclusive of Kuali’s current data schema but a portion that is relevant to our project. Kuali will continue to develop using this table layout structure, so future versions should be compatible with Courseland with minimal modifications to to API.

CLASSMEET VIEW

Name Null Type

TERM_YR NUMBER(4)

TERM_CD VARCHAR2(1)

COURSE_CNTL_NUM NUMBER(5)

BLDG_CD VARCHAR2(12)

ROOM_PREFIX_NUM VARCHAR2(2)

ROOM_NUM VARCHAR2(4)

ROOM_SUF_NUM	VARCHAR2(2)
WEEK_GRP	VARCHAR2(7)
START_TIME	VARCHAR2(4)
START_TIME_AM_PM_FLAG	VARCHAR2(1)
END_TIME	VARCHAR2(4)
END_TIME_AM_PM_FLAG	VARCHAR2(1)
INSTR_SHORT_NAME	VARCHAR2(18)
EMPLOYEE_ID	VARCHAR2(9)
INSTR_FUNC	VARCHAR2(1)
PRINT_CD	VARCHAR2(1)
MULTI_ENTRY_CD	VARCHAR2(1)
LOAD_DATE	DATE

STUDENT

Name	Null	Type
STU_ID	NOT NULL	NUMBER(8)
ACAD_REG_BLK_TERM_GRP		CHAR(6)
ADMIN_REG_BLK_TERM_GRP		CHAR(6)
ADMIT_BASIS_CD		CHAR(1)
ADMIT_DATE		CHAR(8)
ADVSTG_QTR_UC_PASSED		NUMBER(4,1)
ADVSTG_TOT_PASSED		NUMBER(4,1)
ADVSTG_TOT_UC_PASSED		NUMBER(4,1)

AMER_CULT_REQ_CD	CHAR(1)
AMER_CULT_SATISF_DATE	CHAR(4)
AMER_HIST_REQ_CD	CHAR(1)
AMER_HIST_SATISF_DATE	CHAR(4)
AMER_INST_REQ_CD	CHAR(1)
AMER_INST_SATISF_DATE	CHAR(4)
ANTICIP_GRAD_DATE	DATE
AP_TOT_CRED_UNIT	NUMBER(4,1)
BIOGRAPH_RELEASE_FLAG	CHAR(1)
BIRTH_DATE	DATE
BIRTH_PLACE	CHAR(25)
CITZN_CNTRY_CODE	CHAR(3)
COLL_CHNG_FLAG	CHAR(1)
CORP_EDUC_LEVEL	CHAR(1)
CUM_GPA	NUMBER(4,3)
CURR_TRANSCR_BLK_FLAG	CHAR(1)
DEAN_LIST	CHAR(1)
DISABL_GRP	CHAR(7)
DISCLOS_FLAG	CHAR(1)
DISCLOS_ETHNIC_FLAG	CHAR(1)
DR_CANDID_DATE	CHAR(4)
DR_CANDID_TERM_NUM	CHAR(2)
ENGL_COMPOSITN_REQ_CD	CHAR(1)
ETHNIC_CD	CHAR(1)

FEE_RESID_CD	CHAR(1)
FIN_REG_BLK_TERM_GRP	CHAR(6)
FIRST_REG_SUB_TERM	CHAR(1)
FIRST_REG_TERM_CD	CHAR(1)
FIRST_REG_TERM_YR	NUMBER(4)
HOME_LOC_CD	CHAR(3)
HOME_LOC_CITY	CHAR(25)
HS_CD	NUMBER(6)
HS_GRAD_DATE	CHAR(4)
INTEND_COLL_CD	CHAR(2)
INTEND_MAJ_CD	CHAR(3)
INTL_CON_PHYS_PRES_CD	CHAR(1)
INTL_SEVIS_STATUS_CD	CHAR(3)
LAST_ATTEND_YR	NUMBER(4)
LAST_SCHOOL_CD	NUMBER(6)
LGR_QTR_ATTEMPT_UNIT	NUMBER(4,1)
LGR_QTR_GRDPT	NUMBER(5,1)
LGR_TOT_ATTEMPT_UNIT	NUMBER(4,1)
LGR_TOT_GRDPT	NUMBER(5,1)
LGR_TOT_PASSED_UNIT	NUMBER(4,1)
MAJ_CHNG_FLAG	CHAR(1)
NEW_SCHOOL_TAB_FLAG	CHAR(1)
PNP_QTR_ATTEMPT_UNIT	NUMBER(4,1)
PNP_TOT_ATTEMPT_UNIT	NUMBER(4,1)

PNP_TOT_PASSED_UNIT		NUMBER(4,1)
PROFESSNL_ENTRY_TERM		CHAR(3)
PROFESSNL_LEVEL		CHAR(1)
RCD_CHNG_DATE		DATE
RCD_CHNG_USERID		CHAR(8)
READMIT_FEE_STATUS_CD		CHAR(1)
REG_BLK_TERM_GRP		CHAR(6)
REG_SPECIAL_PGM_GRP		CHAR(10)
RESID_CHNG_TERM		CHAR(3)
ROLE_CD		NOT NULL CHAR(1)
SEX		CHAR(1)
SIR_FEE_STATUS		CHAR(1)
SSN		NUMBER(9)
STATUS_PRIOR_WITHDRAW		CHAR(1)
STU_NAME		CHAR(35)
STU_NAME_CHNG_FLAG		CHAR(1)
SUBJ_A_EXAM_RESULT_CD		CHAR(1)
SUBJ_A_REQ_CD		CHAR(1)
SUBJ_A_SATISF_DATE		CHAR(4)
TOT_INCOMPL_UNIT		NUMBER(3,1)
US_CITZN_CD		CHAR(1)
UG_GRAD_FLAG	NOT NULL	CHAR(1)
VISA_CD		CHAR(2)
ALEV_TOT_CRED_UNIT		NUMBER(4,1)

ABIT_TOT_CRED_UNIT	NUMBER(4,1)
IBAC_TOT_CRED_UNIT	NUMBER(4,1)
CHNG_DATE	DATE
CHNG_USERID	CHAR(8)
DNLD_DATE	DATE
LOAD_DATE	DATE

CLASS VIEW

Name	Null	Type
TERM_YR		NUMBER(4)
TERM_CD		VARCHAR2(1)
CRSESECT_SUB_TERM_CD		VARCHAR2(1)
COURSE_CNTL_NUM		NUMBER(5)
DEPT_CD		VARCHAR2(7)
COURSE_NUM		NUMBER(3)
COURSE_SUF_1_NUM		VARCHAR2(1)
COURSE_PREFIX_NUM		VARCHAR2(1)
COURSE_SUF_2_NUM		VARCHAR2(1)
PRIM_SEC_FLAG		VARCHAR2(1)
SECT_NUM		VARCHAR2(3)
INSTRUCTN_FORMAT_DET		VARCHAR2(3)
MULTI_ENTRY_CD		VARCHAR2(1)
COURSE_OPTION		VARCHAR2(2)

COURSE_TITLE	VARCHAR2(19)
ENROLL_LIMIT	NUMBER(5)
ENROLL_COUNT	NUMBER(5)
WAITLIST_COUNT	NUMBER(5)
CREDIT_CD	VARCHAR2(2)
LOWER_RANGE_UNIT	NUMBER(3,1)
UPPER_RANGE_UNIT	NUMBER(3,1)
VAR_UNIT_CD	VARCHAR2(1)
FIXED_UNIT	NUMBER(3,1)
CANCEL_FLAG	VARCHAR2(1)
SEC_SECT_REQ_CD	VARCHAR2(1)
LOAD_DATE	DATE

C.iSchool Survey

We conducted a survey for iSchool students in order to analyze their motivation as application developers. Our main questions include seeing three main motivations: user's direct need, enjoyment of development and reputation. Our results show their motivation is strong enough to create useful applications.

Courseland

Survey Status: Active Launched: 4/26/2009 11:10 PM Closed: *N/A*

Email Invites: 0	Visits: 41	Partials: 0	Screen Outs: 0	Over Quota: 0	Completes: 35 *(Does not include blank responses)*

General Background Questions

1. What is your current student status?

Undergraduate (including exchange students)	0	0%
Graduate (Masters, PhD)	35	100%
Visiting Scholar	0	0%
Alumni	0	0%
Other, please specify	0	0%

2. What is your current major or home department?

View 35 Responses

3. Have you ever developed a web application using a web-language* (e.g. PHP, PERL, CGI, Pyhton, JAVA, etc.)? (*This does not include designing a webpage using only HTML or CSS).

Yes	28	80%
No	7	20%
Total	35	100%

4. Have you ever been paid as a developer/programmer?

Yes	22	63%
No	13	37%
Total	35	100%

5. Rate your level of expertise in programming on the following scale:

No programming experience	0	0%
Have taken a few introductory programming courses	2	6%
Worked on several small course or work-related programming projects	16	46%
Worked on larger projects; possibly a member of a	5	14%

development team			
Junior level programmer for an institution		8	23%
Senior level prorammer or software team leader for an institution		4	11%
	Total	35	100%

Satisfaction with Online Academic Services

6. How would you rate your satisfaction with the functionality provided by: TELEBEARS?

Very unsatisfied		4	11%
Unsatisfied		20	57%
Neutral		7	20%
Satisfied		4	11%
Very satisfied		0	0%
	Total	35	100%

7. How would you rate your satisfaction with the functionality provided by: BEARFACTS?

Very unsatisfied		4	11%
Unsatisfied		14	40%
Neutral		14	40%
Satisfied		3	9%

Very satisfied		0	0%
	Total	35	100%

8. How would you rate your satisfaction with the functionality provided by: CARS (billing & payment services)?

Very unsatisfied		3	9%
Unsatisfied		20	57%
Neutral		10	29%
Satisfied		2	6%
Very satisfied		0	0%
	Total	35	100%

9. How would you rate your satisfaction with the functionality provided by: COURSE CATALOG?

Very unsatisfied		7	20%
Unsatisfied		18	51%
Neutral		8	23%
Satisfied		2	6%
Very satisfied		0	0%
	Total	35	100%

10. How would you rate your satisfaction with the functionality provided by all the academic services overall?

Very unsatisfied		1	3%
Unsatisfied		20	57%
Neutral		9	26%
Satisfied		5	14%

Very satisfied		0	0%
	Total	35	100%

Software Development Questions

11. Assuming that you could have access to all the relevant student affairs data through an easy-to-use API, how willing would you be to create applications for yourself and others?

Very unwilling		0	0%
Unwilling		13	37%
Neutral		10	29%
Willing		12	34%
Very willing		0	0%
	Total	35	100%

12. How many hours per week do you program outside of paid work and/or coursework?

0 hours / week		19	54%
1-3 hours / week		14	40%
4-6 hours / week		2	6%
7-9 hours / week		0	0%
10 or more hours / week		0	0%
	Total	35	100%

13. How important is it for you to be identifiable as the author of (unpaid/volunteer) parts/works of code?

Very unimportant		4	11%
Unimportant		8	23%
Neutral		13	37%
Important		8	23%
Very important		2	6%
	Total	35	100%

14. As a developer for an open source platform, do you have any specific motivations or expected payoff resulting from your contributions?

View 15 Responses

That's it! Thank you for taking our survey! We appreciate your time!

15. If you would like to be considered for our $25 Amazon gift certificate drawings, please enter your e-mail address below.

View 26 Responses

D. Courseland API Documentation

· getStudentData(sid)

Returns the student data as an object, or returns null if sid does not exist.

Ex. Usage

```
<?php
      $student = getStudentData(241);
      print_r($student);
            /*
                  Object
                  (
                  [sid] => 241
                  [name] => "Littlest Bear"
                  [major] => "Information Management & Systems"
                  [cum gpa] => 3.9
                  ...
                  )
            */
?>
```

· getCourseData(cid)

Returns the course data as an object, or returns null if cid does not exist.

Ex. Usage

```
<?php
      $course = getCourseData(101);
      print_r($course);
            /*
                  Object
                  (
                  [cid] => 101
                  [title] => "Introduction to Information Management"
                  [ccn] => 12345
                  [instructor] => "Claude Shannon"
                  ...
                  )
            */
?>
```

· addTagToStudent(sid,tag);

Adds a tag to the corresponding student. This will update the tags array in the student data object.

Ex. Usage

```
<?php
      addTagToStudent(241, “mascot”);
      $student = getStudentData(241);
      print_r($student.tags);
            /*
                  Array
                  (
                  [0] => “bear”
                  [1] => “small”
                  [2] => “mascot”
                  )
            */
?>
```

· addTagToCourse(cid,lat,lng);

Adds a tag to the corresponding course. This will update the tags array in the course data object.

Ex. Usage

```
<?php
      addTagToCourse(101, “fun”);
      $course = getCourseData(101);
      print_r($course.tags);
            /*
                  Array
                  (
                  [0] => “information”
                  [1] => “management”
                  [2] => “cool”
                  [3] => “fun”
                  )
            */
?>
```

· addGeoTagToCourse(cid,tag);

Adds a geotag to the corresponding course. This updates the geotag field in the course data object.

Ex. Usage

```
<?php
      // The following code sets course location to center of world map
      addGeoTagToCourse(101, “0.0, 0.0”);
?>
```

```
CONSTANTS:
ACAD_INTERESTS  // Academic interests
JOB_EXPERIENCES   // Job experiences
JOB_INTERESTS   // Job interests
FRIENDS   // Friends

getInfo(sid, type)
Returns an array of the data for the sid.

Ex. Usage

<?php
    $array = getInfo(5312351, ACAD_INTERESTS);
    print_r($array);
        /*
            Array
            (
            [0] => “Natural Language Processing”;
            [1] => “Databases”;
            [2] => “Information Organization & Retrieval”;
            )
        */
?>
```