This paper was presented at CUMREC '97, The College and University Information Services Conference. It is the intellectual property of the author(s). Permission to print out copies of this paper is granted provided that the copies are not made or distributed for commercial advantage and that the title and authors of the paper appear on the copies. To copy or disseminate otherwise, or to republish in any form, print or electronic, requires written permission from the authors.
STUDENT TRANSACTIONS VIA THE WEB

Cedric Bennett
Director, Applications Support
Stanford University

Margo Kannenberg
Manager, Student Information Systems
Stanford University

INSTITUTION
Stanford University is located in the foothills of the San Francisco peninsula, in Stanford California. It is a mid-sized private institution with an enrollment of approximately 14,000 students (6500 undergraduates and 7500 graduates). The University includes the Schools of Business, Earth Sciences, Education, Engineering, Humanities and Sciences, Law and Medicine. Stanford is supported by over 1400 faculty and 5000 staff, which includes the Office of the Registrar, which first requested a student administrative system.

ABSTRACT
By migrating its student administrative system from the mainframe to the Web, Stanford University provides functionality for students in an easy to learn and use format. This Web-based system allows students to register, apply for housing, see courses and grades, file study lists, update addresses and more.

This paper will describe Stanford's effort in migrating to a Web-based Student Administrative System, specifically covering:

- the background of this effort
- Stanford's architectural approach
- the issues and problems faced, and how Stanford has solved them
- what's next
INTRODUCTION

Stanford University, like other institutions, has embarked on a migration of its major mainframe systems to new client/server technology. This migration effort first began in the late 1980’s with the purchase of a client/server document image system and the subsequent addition of purchased applications for fund raising and financial aid as well as the development of a departmental expense management system and a research administration system. Currently, Stanford is working with Oracle on an implementation of the financial suite of systems.

Since these first efforts, the World Wide Web has emerged as a powerful and appealing technology to utilize in the migration of mainframe systems. Ease of development, cost, platform independence and accessibility are just some of the reasons that using the Web is appealing. Stanford quickly developed several small Web applications and found that they empowered faculty, staff, students and well as others with an interest in Stanford. For example, one small Web application provides information on community housing near the campus; it also supplies a web-based mechanism for prospective landlords to create new listings. Another application allows prospective undergraduate students to request admission applications.

It was in this context, that Stanford then looked to a Web solution to migrate and extend its mainframe student administrative system, Axess.1 This system, running as a part of Stanford’s online mainframe environment, has been providing administrative transaction and information retrieval services to students since early 1992. The migration of the application from mainframe terminals to the more attractive and flexible Web environment would allow students to take care of nearly all of their administrative business via the Web, with a continuation of the immediate updating of, and presentation from, the mainframe student database. In making this migration decision, Stanford was well aware that putting any application onto the web brings its own set of unique challenges. This paper pays particular attention to those challenges.

BACKGROUND OF AXXES

Since the spring of 1992, students at Stanford have been taking care of their own administrative business via the mainframe system, Axess. It is currently being used by all Stanford students as the principal means of registering, submitting study lists, maintaining address data, checking grades, requesting transcripts, applying to graduate, monitoring academic progress, applying for housing and more. It is a line-by-line application meant to provide ubiquitous, training-free services. Stanford students use Axess from their own computers and from computers residing in dorm clusters, academic offices, libraries, etc. They use Axess from off-campus and from Stanford overseas campuses as well.

The current Axess environment has become problematic for Stanford in a variety of ways. The mainframe application is very CPU intensive and at peak-demand times creates much heavier-than-acceptable load on the mainframe which affects other University business usage.2 This mainframe resource requirement also makes it difficult to add additional business functionality to Axess, which has been requested by the offices of the Registrar, Bursar, Financial Aid and others. In addition, because of the affect on the central computing resource, Axess can not be made available for student use during peak times of the day. Moreover, Axess has been developed in a Stanford-proprietary language, and resources capable of maintaining the system are becoming more and more scarce. The interactive prompting, line-by-line modality of its interface has also become much less popular with students.

1 The name, Axess, was chosen to play on the notion of “access” to information and also the annual football rivalry between Stanford University and the University of California at Berkeley in which a perpetual trophy, “The Ax” is awarded to the winning school each year.

2 Over 2,500 business users across the campus could be adversely affected by computer slowdowns during their busiest periods.
A Web solution for *Axess* would allow much of the business logic of the application, as well as all the human-interface presentation function, to be moved to a Web server and be serviced at desktops by Web-browsers such as Netscape, Internet Explorer, and others. This would off-load a great deal of the mainframe use. That, in turn, would allow additional capacity for desired functionality and could be available to students over the entire day. It could also be developed in a relatively short period of time using industry standard tools. Finally, it would replace the less desirable line-by-line presentation mode with a graphical interface that is already very familiar to students.

**Web Development Approach**

Early in 1996, a team of IT staff at Stanford met to write a project proposal for the migration of *Axess* to the Web. An important first step was to determine what the development approach should be for *WebAxess*. Key ingredients to this project were identified:

- replace the line-by-line terminal session with a Web client
- continue to use the mainframe student database as a data server
- develop a web server to manage the presentation and take on some business logic
- restrict the project to the migration of existing *Axess* functions
- use an established project management methodology

Once this approach was agreed upon with key clients, the team addressed its key development challenges: (1) determining the *WebAxess* architecture and services, (2) choosing a development environment, and (3) designing the Web presentation.

*WebAxess architecture and services* — The architecture for *WebAxess* (see diagram at Figure 1) consists of the following components:

- standard web browsers
- an application-specific web server
- network authentication services
- the mainframe student database

The web server provides for secure data transmission and runs on a dedicated **UNIX** host. A Stanford-developed RPC (Remote Procedure Call) protocol was already available to support the passing of data and commands between a variety of servers and the mainframe; this application takes advantage of that RPC service. Network authentication services were upgraded to support Kerberos authentication for web browsers. Current mainframe code is reused (as a server) with little modification and provides all access from, and update to, the student database. Business rule editing and in-context semantic data validation is migrated to the Web server to the extent possible, preventing unnecessary mainframe access. Logging and reference data once kept on the mainframe has been moved to server-based relational databases and is updated and accessed directly from the Web server.

The team envisioned a rich administrative environment to be used both during development and subsequently for ongoing enhancement. This web administrative environment supports both developers and business clients by providing:

- Web page text maintenance (function text, error messages and helps)
- project documentation
- ability to test a variety of students
- flexibility in changing calendar dates for testing
- several levels debugging
- maintenance of *WebAxess* control tables
Development environment — New Web tools are becoming available at an accelerating rate. The team found itself faced with an almost bewildering array of choices that seemed to change each month. In order to avoid the “analysis-paralysis” syndrome, the team decided to time-box the assessment of development tools and concentrate on what was needed for a successful first implementation. The team developed a list of evaluation criteria and then spent only one month examining their top four alternatives. The evaluation criteria included:

- ability to run on a UNIX server
- ability to access one of the standard relational database systems in use at Stanford
- interaction with the mainframe
- support for SSL-based security and the ability to link to Kerberos
- support for the management of state
- high performance
- usability and the support of more than one language

A decision was made to use Oracle’s WebServer, and training was immediately arranged. Working with an Oracle expert, communication between Oracle’s web server and the mainframe was designed.

Web presentation — Although the fundamental decision to use Web browsers as the desktop client had already been made, many decisions regarding the actual interface were still left to be determined. Included among the issues that were still facing the team were: (1) what percent of the student population would have access to a web browser, (2) what alternatives need be provided for those who either had no access to a desktop web browser or could not operate one (because of physical limitations), (3) what web browser capabilities could be reasonably assumed to exist at the desktop, (4) how fast (or slow) would the link be between any particular browser and the servers, and (5) how difficult would the application be to learn.

The development team agreed to a set of presentation guidelines, which helped them as they developed the
application:
- employ simplicity of design
- avoid complicated graphics that degrade performance
- use a consistent design across functions for a coordinated application
- design an intuitive, easy-to-use application

The team evaluated several basic presentation approaches, including the use of frames and decided to limit the client interface to standard HTML Web forms. Radio buttons, select lists, check boxes and input forms would replace the line-by-line prompting of mainframe Axess. Methods were developed to handle helps, errors and specialized notices.

Also, since the application is aware of the particular student using it, the presentation could be tailored toward specific needs based upon that information (e.g., a graduate student would not be asked for information that only relates to undergraduate students). WebAxess could also be controlled by an academic calendar so that certain functions would be active only at appropriate times (e.g., a student could only drop a class within allowable dates set by the registrar). Although the plan was to not modify existing functions during the migration (to help ensure rapid development and deployment), the team nevertheless worked closely with business clients and students to make sure that these initial migrated functions met everyone's needs.

### WEB ISSUES

Prior to the actual design and implementation of the WebAxess application, the development team outlined the issues it would face in the migration of Axess to the World Wide Web:
- security of WebAxess
- privacy of student data
- acceptable performance
- reliability and stability of the application
- accessibility for all students

The team worked with other experts at Stanford to seek advice on issues and to communicate the project's direction and decisions. The migration of Axess to the Web was a major Web development effort, and the team was determined to ensure that decisions made were in concert with the institution's Web development strategy and other Stanford Web development efforts. Further, since it was one of the first major Web applications being developed, it was also in a position to inform those efforts and the overall strategy.

**Security of WebAxess** — Since WebAxess deals with extremely sensitive student transactions and information such as registering, applying for and accepting housing assignments, and viewing grades, the security of this Web application and the privacy of the data are primary issues (privacy is discussed in the section which follows). A careful study of this topic showed that security issues could not be solved by single solution, but rather by a combination of approaches: (1) establishing an authentication and authorization approach, (2) keeping data secure as it travels across the lines, and (3) preventing the misuse of WebAxess.

Stanford addressed each of these security issues:

1. **Authentication/authorization**: A University-developed authentication mechanism (see diagram at Figure 2) is being utilized to enable the use of Kerberos with Web applications and to support single-signon. Students will be authenticated via student IDs and passwords. This authentication service takes advantage of Kerberos, so that ID’s and passwords are passed in an encrypted form. Once an individual’s identity is authenticated, further authorization takes place within WebAxess to make sure that the student can appropriately use the application or specific functions within it.

2. **Data transmission**: A web server that supports a secure socket layer (SSL) is being used. This ensures the bi-directional encryption of all transmission between the browser and the server.

3. **Protecting WebAxess from misuse**: A number of methods have been employed to mitigate against the misuse of WebAxess. They include: monitoring for excessive attempts to enter WebAxess, sending independent email confirmation messages after student transactions, automatically logging users off
after a period of no activity, monitoring for the entrance to WebAxess pages from an unspecified location, and using the application itself to remind students of their roles and responsibilities in preventing system misuse.

Privacy of student data — Although privacy is a very important issue, especially when it comes to student data, much of the problem is addressed by the same mechanisms that are in place to provide security. However, the fact that this application is also used from public workstations adds an additional layer of complexity to protecting privacy. Most Web browsers will store information locally at the desktop to increase performance, and the student who leaves a session without properly logging off, potentially provides access to private student information. Again, a variety of approaches helps to mitigate this issue:

- eliminating the display of name and ID from all WebAxess pages (even if information is left on a screen, it is not associated with any particular student)
- controlling the browser from the server to prevent page caching
- making log-off a prominent selection on each page
- using the system itself to remind students of their roles and responsibilities in protecting private information
- automatically logging users off after a period of no activity and returning the browser to some neutral state such as the display of Stanford’s home page

Acceptable performance — WebAxess must provide an acceptable response time for students. The “layering” of interaction across several different levels of connection during a session helps with this. Time spent on a single browser page filling in fields and checking boxes, etc. will be highly responsive. Editing of input data on the web server is planned to prevent unnecessary mainframe access and return information or errors immediately. Further performance gains are expected due to the relocation of reference and logging data from the mainframe to the Web server.
The mainframe application and database will remain open all day, once the first access is made to it. The team worked with systems staff to assure that student records would remain locked and available during an entire WebAxess session; the overhead of opening student records for each mainframe server call is thus avoided. A simple design approach was adopted, to minimize the time-consuming transmission and rendering of complex graphics. The development of remote procedure calls (RPC) for direct connections between the mainframe and server provides the best possible performance in terms of that access.

Reliability and stability of the application — The development team has worked closely with the operations and systems group from the beginning. Working together, the teams determined the characteristics of a test server environment that would be identical to a production server environment. It was agreed that production and test servers would be separate machines, and a backup approach would be developed, which could include mirroring.

Accessibility to WebAxess — All relevant campus groups were initially contacted to make sure that all students could access this application via the Web. This included residential computing, overseas studies and the disability resource center. It was determined that through cluster access, departmental access and personal access, students would be able to use WebAxess. In order to provide better support for ADA students, it was decided to support a character-only browser as well, using LYNX.

WEB IMPLEMENTATION
Choosing Oracle’s Webs server as a development tool allows the use of PL/SQL as a language for implementation. Since most of the team was relatively new to this language and Web technology in general, they were first given concentrated training. They developed the first WebAxess function as a team, discussing standard approaches, common routines, naming conventions, etc. Once this initial development was complete, they independently and concurrently developed remaining functions.

WEBAXESS FUTURE
The future of WebAxess is bright! Upon completion of this migration project, many new functions are already in the queue for development: these include major/minor declaration, financial aid and bursar displays, and termination of housing. These additional functions will both empower students and relieve central offices of many current responsibilities. As Web technology itself grows, the use of Java to further off-load application code from the server and mainframe can be explored. Eventually, when the student information system itself is replaced, WebAxess can be re-linked to that new system. WebAxess can also be enriched with features such as the use of student preferences, links to additional Stanford information, the use of video, and more.

Another bright outcome: WebAxess is helping to pave the way for Web development in other major applications at Stanford, such the new core financial systems. Many of the issues faced and architectural decisions made will be similar. By careful analysis and documentation, it is expected that WebAxess will prove to be a useful guideline for the utilization of the Web by other applications.