Faculty Information System

James Curry, Aaron Flocke, Geetha Krishnasamy and Leena Paulose

Abstract

The Faculty Information System is a web-based application designed to track current and prospective faculty information for the School of Computer Science and Information Systems (CSIS) at Pace University. This application provides a user-friendly web interface for prospective faculty to add, manage and submit their curriculum vitae. The application stores details of all faculty vitae in a searchable database to be reviewed by authorized personnel and matched to a suitable position. Current faculty will use the same interface to update their academic achievements and credentials. Microsoft SQL Server is used for the backend database and Cold Fusion and Fusebox for developing the web interface. This paper includes a detailed description of the application and the methodology used to implement it. The paper concludes with a hand-off to the client with a recommended exit strategy listing future enhancements.

1. Project Description

The main objective of this project is to provide a manageable, efficient, state-of-the-art information management tool to use during the hiring process for CSIS. It will help manage the hiring activities among the various campuses (geographically separated) by eliminating the hassles of unwanted paperwork and ‘snail mail’. Furthermore, it provides the prospective applicant a fast and reliable means of managing and submitting his or her vitae from the comfort of their home.

Candidates using the Faculty Information System for the first time will register with a unique username and password to create an account. Once registered, they can log into the system from anywhere using a web browser and complete the necessary information required to submit their vitae. The vitae data is stored in a database until it is ready for submission. The candidate is eligible to submit his or her resume when the required information, Contact Information, Academic Degrees Earned, Work History and Teaching Preferences are provided. Once the candidate submits their vitae, the system will generate a confirmation and the vitae is flagged to be reviewed by the Academic Chairs and representatives from the Dean’s office. Once a candidate submits their vitae, they will not be restricted from modifying their vitae. If the candidate wishes to modify their vitae, they can withdraw the submission and then make the modifications and re-submit.

The Academic Chair and representatives from the Dean’s office can also log into the same system. This user group is provided with additional privileges and appropriate links are displayed upon login. They can sort the submitted resumes in a variety of ways – by department content area i.e. Computer Science, Information Systems etc, and/or by course content area i.e. Security, Database, Programming etc. Each selection will query the database and provide a general overview of the candidate with hyperlinks to summary information as well as a complete overview of the candidate. This will allow the reviewers to survey the candidate, and, when interested, view that candidate’s information in greater detail. When the viewer has made a decision, he can email the candidate expressing interest. All contacted candidates will be flagged to let the other viewers know that that candidate is being considered. After the review process is completed and the candidate has accepted a position with, their status will be changed from ‘applicant’ to ‘faculty’.

2. Methodology

Our goal to develop a robust, user-friendly and efficient database driven; web-based application required a structured and systematic approach. In the traditional Point-to-Point programming methodology, forms, screens, database objects, etc., become intertwined into a complex ‘web’ of functionality. The resulting spaghetti code, as shown in Figure 1, is often not documented and is definitely expensive to maintain over time. ‘Fusebox’ is a standard framework for building complex web applications and its methodology, ‘FliP,’ Fusebox Life Cycle Process, introduces an elegant new way of managing the software development process in a few simple steps [4]. It is compatible with just any web application server and supports multiple web languages. Code modularity and reusability being its key features, the Fusebox methodology worked well for the design and implementation of the ‘Faculty Information System’ with Cold Fusion as the language of choice.
The Fusebox framework consists of a series of files known as ‘core files’ that allow you to easily achieve nested layouts, inherited variables and settings, and breaks your application into more manageable pieces called ‘circuits.’ A typical Fusebox system consists of several fuseactions and fuses but a single Fusebox controller [1]. ‘Fuseactions’ define what the user wants to do and controls the flow of the application. A ‘fuse’ is the code in PHP, CFML, etc., used to perform a single task. It is reusable. The Fusebox controller calls the appropriate fuse depending on the fuseaction being performed. It is known as the glue of the application [4]. A typical Fusebox architecture is shown in Figure 3.

As the first step, the system was wireframed. ‘Wireframing’ is a way to quickly model the proposed actions that will be performed by the application [5]. In web-speak, a wireframe is a skeletal rendering of every click-through possibility on your site - a text-only "action," "decision" or "experience" model. Its purpose is to maintain the flow of your specific logical and business functions by identifying all the entry and exit points your users will experience on every page of your site. Figure 2 shows the wireframe model of the first few actions performed by the Faculty Information System. The goal to ensure that the entire application is laid out before any coding begins is accomplished by greatly improving the client-developer communication. The wireframe model is the blueprint of the web application under development.

### Quick Map of "FacultyInfoSystem"

<table>
<thead>
<tr>
<th>Page</th>
<th>Exits and Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homepage</td>
<td>register/faculty</td>
</tr>
<tr>
<td>First Time Applicant Page</td>
<td>forgetPassword loginValidate</td>
</tr>
<tr>
<td></td>
<td>accountValidate resetPage</td>
</tr>
<tr>
<td>Forgot Password</td>
<td>validate</td>
</tr>
<tr>
<td>Validate E-Mail Address</td>
<td>passwordSuccess passwordFailure</td>
</tr>
<tr>
<td>Login Validate</td>
<td>loginFailed applicant admin user</td>
</tr>
</tbody>
</table>

Fig 2: Wireframe Model
Wireframing was followed by the development of a ‘**HTML prototype**,’ which portrays the ‘look and feel’ of the system. Here the text-based wireframed application is converted into a statically linked html application. These html pages are just artistic sketches of the actual system with dummy data. It depicts the navigational layout, color, graphics, etc.

The next step includes the database design and the code implementation. This project being an addition to an existing application at Pace University, the database design only involved the addition of the required tables to the existing databases on the **SQL Server**, while preserving the existing structure and relationships. Coding starts off with dividing the system into fusebox circuits and identification of fuseactions. The *Faculty Information System* is divided into two distinct circuits – the Applicant circuit and the Faculty circuit. The reusable fuses were then determined and coded in Cold Fusion. Separate files are maintained for all the required SQL queries. The universal file in the Fusebox methodology integrates and glues the application together.

**Cold Fusion** is a Web application development tool that allows developers to build dynamic data-driven applications for use on the Internet and intranets. With Cold Fusion, you can develop everything from intranet database applications and groupware to E-commerce applications [3]. Cold Fusion employs a high-level server-side markup language called CFML (Cold Fusion Markup Language) to interact with relational databases, Crystal Reports, SMTP and POP servers, and a whole host of other network technologies [6].

CFML is a markup language similar in structure to HTML in that all of the language constructs are contained within tags. CFML enjoys a relationship with HTML similar to that of other embedded scripting languages such as JavaScript or VBscript. HTML is used for layout control while the CFML handles the brunt of the work such as database queries, arithmetic and string functions, and other server side processing tasks. Files created with CFML are saved as Cold Fusion templates and are designated by a ".cfm" extension [2].
3. Relevance in the Context of Other Work

In the current ‘Internet Era’, many Schools and Universities are moving towards online resumes. Their main goal is to make the hiring process simple, efficient, quick and especially user-friendly. Although there are a few systems similar to the ‘Faculty Information System’, the ones available differ in some important features.

Some of the systems worth mentioning are provided by MIT (Staffing Services) [7], Cornell [8], Columbia [9], California State University (Northridge Alumni Association) [10] etc. The MIT’s Staffing Service is similar to our system in that a prospective candidate, current faculty and Administrative personnel can log on to the system. The similarity ends here. Most of the systems provide a list of available positions for which the candidate can apply. In our Faculty System, the candidates submit their resume and will be considered when a suitable position becomes available. The Academic Chair and the other authorized personnel decide which candidate is suitable for which position. This way, the same candidate could be reviewed by different Departments, which would increase their chance of being hired.

All the other systems require the candidate to attach his resume and cover letter in text form. These text files could be lengthy and going through these documents will hinder the speed of the hiring process. The Faculty System eliminates the use of any text documents. By using this approach, the viewers of the applications has the freedom to sort the applications is a variety of ways.

One interesting feature in the MIT’s Staffing Service is that even a current faculty can use the service to apply for an available position. Our system does not have provisions do to that. The current faculty at Pace can only update his information.

4. Use Case Scenario

Any individual using the Faculty Information System for the first time should register with the system as shown in Figure 4. After the registration process, the user will have his unique username and password which he can use to login. Once registered, the applicant is guided through several steps.

![Fig 4: The Faculty Information System Homepage and Login screen](image)
During the course of these steps, the applicant is required to provide information regarding his Contact Information, Degrees Earned, Work History and Teaching Preferences as shown in Figure 5. At this point, the applicant is eligible to submit his application. He may do so, or continue to provide optional information like Patents, Publications, Grants, and Awards etc. The ‘Required Information’ and ‘Optional Information’ are provided in the left navigation bar as shown in Figure 5.

The user interfaces to add, delete or edit information are similar to the one in Figure 6. If an applicant is not ready to submit his application, he has the option to just save it. When an applicant submits his application, his status is changed from ‘inactive’ to ‘active’. All active applications are available to be viewed by reviewers. Once an application has been submitted, the applicant has to withdraw the application (thereby making it inactive) before he can make any changes.
In addition to the applicants, there are 2 user groups who access the Faculty Information System – faculty and reviewer. The faculty at CSIS can use the same application to update their credentials. The ‘Reviewer’ is a select group of faculty who has the privilege to review the submitted applications and is therefore provided with three additional links in the left navigation bar as shown in Figure 7.

Fig 7: Candidate Search Options

Reviewers can search for candidates who meet their requirements of Course Type, Campus, Employment type and Teaching level. Figure 8 displays the output of a successful search operation. Every detail (regarding the candidate) that the reviewer might need is displayed in this screen. The reviewer can either contact the promising candidates or save their vitae in a folder for future review.

Fig 8: Candidate Search Results
5. Recommendations for the Future

While this is technically a complete system, the opportunities for expanding are almost limitless. Here are a few additional features we recommend to be added in the next version.

Some useful additions to this system might be to

- Log and document all electronic communications between an applicant and the reviewer(s). Although this facility has its merits, it has its limitations too. It would take up a significant amount of space on the server. In addition it could give rise to concerns regarding confidentiality and security.

- The current system has been designed and implemented as per the requirements of the School of CSIS. By expanding the system further, it can be made suitable to cater to the needs of the remaining academic schools within Pace University.

- When the system has been upgraded and ready to be used by all the departments/divisions within Pace, it could be integrated into the Blackboard System (or whichever portal system).

6. References


