Web-based Employment Application & Processing Support System

A Manuscript
Submitted to
the Department of Computer Science
and the Faculty of the
University of Wisconsin-La Crosse
La Crosse, Wisconsin

By
Sujan Shakya
in Partial Fulfillment of the
Requirements for the Degree of
Master of Software Engineering
May, 2008
Web-based Employment Application & Processing Support System

By Sujan Shakya

We recommend acceptance of this manuscript in partial fulfillment of this candidate’s requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

__________________________________________  ____________________________
Dr. Thomas Gendreau                                             Date
Examination Committee Chairperson

__________________________________________  ____________________________
Dr. Kasi Periyasamy                                             Date:
Examination Committee Member

__________________________________________  ____________________________
Dr. David Riley                                             Date:
Examination Committee Member
ABSTRACT


Web-based Employment Application & Processing Support System (WEAPSS) is an online common platform for both applicants and departments/administration for the pool search vacancies. It will replace the traditional "paper application" process for part-time positions for instructional academic staff with a new employment jobsite and an online employment application tracking system. This web application will automate the entire hiring process, including the position requisition approval process, employment and application processing, affirmative action (recruitment profile) and personnel action processing.

This manuscript, in particular describes development of the WEAPSS including the challenges, issues that arose during its development and what counter measures were taken to deal with it.
I would like to express my sincere thanks to my project advisors Dr. Thomas Gendreau and Dr. Kasi Periyasamy for their insightful comments, outstanding advice, and exceptional guidance. I would like to thank the project sponsor Jennifer B. Wilson and project manager Joy J. Gutknecht who initiated this project and provided the support for this project. I would like to express my gratitude to the Computer Science Department and the University of Wisconsin-La Crosse for providing the computing environment for my project. Finally, I want to thank my family for their understanding and support over the course of this degree.
TABLE OF CONTENTS

ABSTRACT....................................................................................................................................... iii
ACKNOWLEDGEMENTS.............................................................................................................. iv
TABLE OF CONTENTS................................................................................................................ v
LIST OF FIGURES....................................................................................................................... vi
GLOSSARY...................................................................................................................................... vii
1. BACKGROUND INFORMATION. ............................................................................................ 1
2. A BRIEF INTRODUCTION TO SOFTWARE LIFE CYCLE MODELS.. ...................................... 6
3. DEVELOPMENT OF THE WEAPSS .................................................................................... 10
  3.1. COLLECTING SOFTWARE REQUIREMENT................................................................. 10
  3.2 USER CLASSIFICATION AND CHARACTERISTICS....................................................... 12
  3.3 USER INTERFACE DESIGN............................................................................................ 18
4. DESIGN..................................................................................................................................... 19
  4.1. HIGH-LEVEL ARCHITECTURAL DESIGN................................................................. 21
  4.2. DATABASE DESIGN OF WEPASS............................................................................... 22
  4.3. ARCHITECTURE OF WEAPSS.................................................................................. 24
  4.4. TESTING.................................................................................................................... 31
  4.5. DEPLOYING WEAPSS.............................................................................................. 33
  4.6. SECURITY.................................................................................................................. 34
5. LIMITATIONS....................................................................................................................... 35
6. CONTINUING WORK.......................................................................................................... 36
7. CONCLUSION...................................................................................................................... 37
8. BIBLIOGRAPHY................................................................................................................. 38
APPENDIX A: SAMPLE WEAPSS SCREEN SNAPSHOTS......................................................... 39
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Waterfall Model</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Incremental Software Development Life Cycle</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Use Case Diagram for Administrator user</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Use Case Diagram for Applicants</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Use Case diagram for Administrative users</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Model II Architecture</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>High-Level Architecture of WEAPSS</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Entity Relationship diagram of WEAPSS</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>Class Diagram of WEAPSS</td>
<td>26</td>
</tr>
<tr>
<td>10</td>
<td>Portion of Control Flow Diagram of struts-config.xml.</td>
<td>27</td>
</tr>
</tbody>
</table>
GLOSSARY

AA&D
Assistant to the Chancellor for Affirmative Action and Diversity

AAO
Affirmative Action Officer

CSS
Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in a markup language. It’s most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document.

Deployment profile
Deployment profile is a project component that manages the deployment of an application. A deployment profile lists the source files, deployment descriptors (as needed), and other auxiliary files that will be included in a deployment package.

EAR (Enterprise Archive)
An Enterprise Archive, or EAR, is a file format used by Java EE for packaging one or more modules into a single archive so that the deployment of various modules onto an application server happens simultaneously and coherently. It also contains XML files called deployment descriptors which describe how to deploy the modules on an application server
**Front Controller pattern**

The Front Controller pattern defines a single component that is responsible for processing application requests. A front controller centralizes functions such as view selection, security, and templates, and applies them consistently across all pages or views. Consequently, when the behavior of these functions needs to change, only the controller and its helper classes of the application needs to be changed [15].

**HR**
Office of Human Resources

**HTML (Hypertext Markup Language)**
A markup language designed for creating web pages and other information to view in a web browser.

**JSP**
Java Server Pages is a Java technology that is used to develop dynamic web pages. Java Server Pages (JSP) is comprised of HTML tags with embedded Java code. A JSP compiler is used to generate a Servlet from the JSP page.

**JSTL**
JSTL is a component technology within the Java 2 Enterprise Edition (J2EE) specification and is controlled by Sun Microsystems. JSTL is a set of simple and standard tag libraries that encapsulates the core functionality commonly needed when writing dynamic JSP pages.

**PVC**
Provost & Vice Chancellor for Academic Affairs
SS chair
Search and Screen committee chair

SS committee
Search & Screen committee

Struts
It is a centralized framework that is based on the MVC architecture which uses xml files to configure the different modules.

SQL
Structured Query Language (SQL) is a popular computer language used to create, modify and query databases.

UML
Unified Modeling Language™ (UML) is an industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems standardized by the Object Management Group. UML simplifies the complex process of software design by using “blueprints” for software construction.

WEAPSS
Web-based Employment Application & Processing Support System
1. BACKGROUND INFORMATION

The mission of the Human Resources Department at the University of Wisconsin-La Crosse (UW-L) is to recruit, develop, and retain the high caliber diverse workforce, necessary for the university to achieve its mission and goal. The recruiting process by itself is a complicated process for any university. Most complications arise due to a number of factors such as the need for advertising for vacancies in different publications, the need for approval in each step of hiring by higher level authorities, interviewing candidates and making decisions, approval of the decisions made by an interviewing committee by higher level authorities and confirming funds with budget department before hiring candidates. These things not only slow down the process of hiring but also consume a lot of time. In the case of an urgent hiring, it is often not feasible.

In each step of the hiring process for the part-time instructional academic staff positions, UW-L requires approval by a number of persons at different levels such as the Budget Department, the Affirmative Action Office, the Human Resources office, and the Dean of the concerning college/school, etc. Further college and/or departments may create applicant pools that can be used to fill part-time vacancies to teach a 50% Full Time Equivalent (FTE) or less load or up to two classes. Pools are valid for two years, from the date the recruitment packet is approved by the Assistant to the Chancellor for Affirmative Action and Diversity. If additional candidates are needed during the two year period, the pool is re-advertised [9].

The steps involved in hiring for part time needs for instructional academic staff are given below

- Approval to Recruit and Advertise
- Receiving and Processing Applications
• Screening Applicants and Interviewing Candidates
• Hiring Process
• Closing of a Search

Approval to Recruit and Advertise

Initially, a Search and Screen committee (SS committee) has to get approval to initiate a pool search by submitting a list of required items to the Dean, the office of Administration and Finance, the Provost/Vice Chancellor, Affirmative Action & Diversity and Human Resources (in that sequence). On approval by the concerned offices, the chair of the SS committee receives approval to recruit and advertise (via Email notification) from Human Resources. The Advancement office then contacts the committee regarding vacancy advertisements and their placement in the publications listed on the Position vacancy and Recruitment Effort Plan forms.

Receiving and Processing Applications

Upon receipt of each application, the SS committee sends each candidate an acknowledgment letter indicating the required application materials that were received, along with those that are still missing and a deadline date for their submission including the Request for Confidentiality form (L-1825) and Equal Employment Opportunity Form (EEO form), as well as the Informational Programs & Disclosure Information. Within two days after the deadline date of the search, the SS committee sends the complete listing of applicants who did not request confidentiality to Human Resources using the Alphabetical Listing of Nominees and Applicants form L-1737.
Screening Applicants and Interviewing Candidates

The SS committee reviews the applications and determines whether the candidates are qualified and assigns the reason code and tier to their applications. The SS committee then submits the Pool Search Request to Interview and Applicant List form L-1349, with applicants placed in either Tier 1, 2 or 3, to the Dean and the Assistant to the Chancellor for Affirmative Action and Diversity (in that sequence) for their approval.

During the remaining two-year period the SS committee may receives other applications from candidates and the committee may wish to add to the pool. In this case the SS committee sends an updated L-1349 form to the Dean and the AAD for their approval.

Hiring Process

After the interviewing process has been completed, the list of qualified candidates needs to be approved for hiring by the Dean, the Assistant to the Chancellor for Affirmative Action and Diversity, and Administration and Finance. Finally on approval, the candidate(s) are contacted with an offer letter.

Closing of a Search

When the candidate accepts the offer, committees complete the Personnel Action form L-1347 and forward it to the Dean, the Provost, and HR (in that sequence). HR will then issue a contract letter for the candidate [9].
Disadvantage of the current system

- **Paper intensive**
  From the initial stage of hiring (Approval to hire) to the closing of a search, the process is fully based on paper forms.

- **Time Consuming**
  It takes significant amount of time to route application materials from one level to another level for approval in each stage of the hiring process.

- **Prone to missing application**
  Applications are prone to be lost not only during the processing but also during the storage. There is an unacceptably high chance of misplacing the job applications while it is in the processing phase.

- **Physical Storage requirement for applications**
  For each job vacancy, there are always a significant number of applications and to store them safely and in correct order is a big issue especially when applications have to be kept for a pool search for the period of two years.

- **Time for pool search**
  Whenever there is vacancy, staff in the HR department try to search the applications from application pool which matches the given requirements for the position. This requires a lot of time since each application needs to be checked to see if it meets the given requirements.
Need for Web Application

The current paper based system may put UW-L at a disadvantage as it searches for employees from the point of view of getting qualified applicants in less time since according to Pew Internet Project survey, fifty two million Americans uses internet for jobs search, and more than 4 million do so on a typical day[12].

Considering the above mentioned issues, the sponsor wants the entire hiring process to be completely web based in order to automate the entire hiring workflow. The goal is to eliminate a series of paper approval documents needed to conduct and finalize a search to replace these documents with electronic forms and approvals in order to streamline and to improve the recruitment and hiring process.
2. A BRIEF INTRODUCTION TO SOFTWARE LIFE CYCLE MODELS

A software lifecycle is a model that describes all of the activities to engineer a software product. Many process models exist in software engineering literature such as the waterfall model, evolutionary rapid prototyping, rapid application development, incremental prototyping, and the spiral model. Each of them has its own strength and weaknesses. So, choosing the right model depends on number of factors like specification, the nature of the project, developer skills, project size, deadlines and resources available for developing the software [1]. One of oldest models is the waterfall model which is used when the requirements are easy to establish and are stable, the development is customer specific, and changes are not foreseen in the near future. In this model, development proceeds in a stepwise manner from requirements, through design, implementation, testing and finally operation.

![Waterfall Model](image)

Figure 1. Waterfall Model
But when the requirements are difficult to establish in clear terms, or when customers are not sure about their requirements or when it is too risky to develop the whole system at once, the “the water fall model” becomes very ineffective. In such cases the prototyping models (PRM) are mostly used. In a prototyping model, a prototype is built to ascertain the requirements, solution and, in some cases, the most appropriate technology. The advantage of using the prototype model is that with in less time and effort, the customers’ requirements are confirmed. The risk of losing time and a large investment in hardware and development of software is reduced [11]. One of the widely used prototyping life cycle models is the Rapid Application Development model (RAD). In this process, a developer creates the prototype as soon as he/she gets the requirements for a particular module, tests it and evaluates it with the client and makes appropriate changes. This process is repeated until a working system emerges that encompasses the true set of customer and system requirements.

The benefits of using rapid prototyping are

- The prototype helps customers and developers to understand the requirements for the system.
- It encourages active participation among different types of users and developer.

Another, widely used prototyping model is the evolutionary prototyping model. Evolutionary prototyping uses multiple iterations of requirements gathering and analysis, design and prototype development. After the each iteration, the result is analyzed by the customer. Their response creates the next level of requirements and defines the next iteration. Use of evolutionary prototyping is especially beneficial when requirements are changing rapidly or there is a lack of understanding of what is required.

An alternative process model which combines the advantages of evolutionary prototyping with the control required for large scale development is the incremental model. Incremental software development entails the planning, development and release
of software products in a sequence of increments or stages, where each additional increment adds operational functionality, or capability, not available in previous releases. Although incremental development is iterative in nature, it has long been recognized as an effective way to get the user interested and actively involved in the development of the system in order to ensure a closer fit to real needs and greater level of user satisfaction. The gradual introduction provides time for the client to adjust to the system while also allowing for adjustment of expectations and responsiveness to change. It is also seen as one of the basic ways to enhance risk management and reduce the potentials of loss risk. Regarding the technical benefits from the adoption of incremental development is earlier resolution of lot of implementation problems and design errors [2].

From the customer’s perspective, benefits are early (and on going), delivery of functionalities enhance confidence in the developer’s ability to deliver the right system and greater involvement leading to enhanced familiarity, reduced uncertainty, better informed decision making on the requirements.

Some of the major benefits of using the incremental model are

- Less cost and time is required to make the first delivery
- Easier to test and debug during a smaller iteration
- Easier to manage risk because risky pieces are identified and handled during its iteration
- It keeps a constant reality check, from customer feedback.[1]
In this project the sponsors were not sure of all the requirements at the beginning. So, in order to give them an example of the web interfaces and collect requirements effectively in less time, the developer initially used a rapid prototyping model to collect requirements by developing a throw-away prototype. Following the initial prototype development the developer primarily used an Incremental prototyping model. During the development phase of the last increment, some new requirements were requested to be added to the system by the sponsor such as: specifying the actual salary, ability for administrative users to view up to date résumés, as well as applications of applicants. Those requirements were important ones and have significant importance for this project but adding them conflicted with the definition of Incremental prototyping. So, the developer started following an evolutionary prototyping model instead, allowing new requirements to be added. In this way, the developer was able to include the new requirements and complete the project.
3 DEVELOPMENTS OF THE WEAPSS

3.1 COLLECTING SOFTWARE REQUIREMENTS

During the initial requirement gathering phase, the developer attended a series of meetings with sponsors on weekly basis and couple of meetings with the Information Technology Services (ITS) department web team in order to understand and get a picture of the problem domain of the project as well as the technology and tools to be used for development. The developer also studied the existing process in detail which is paper based except for a provision to advertise the vacancies using the static HTML webpage, on the university website. This process was time consuming and inconvenient for both university personals and job applicants.

Within a few weeks of conducting meetings and going through detailed study of the existing process, the developer started to develop a throw-away prototype using JSP. The purpose of this prototype was to gather requirements as soon as possible and start the incremental development phase.

Some of the requirements collected for WEAPSS are listed below.

Jobseeker

- should be able to view all the active jobs.
- should be able to create and post online resume.
- should be able to add/edit personal info in his/her account.
- should be able to search jobs and apply online.
- should be able to view the applied jobs.
Human Resource Department

- should able to post the vacancies.
- should be able to activate and deactivate the vacancies posted.
- should be able to view the resumes in the resume bank of the system.
- should be able to search the candidates based on different fields like education, experience or/and skill, in the data store.
- should get notification from the system whenever there is an application for a job posted, until the position is filled.
- should get notification once the vacant position posted for approval is approved by top levels.
- should be able to view the reports of ongoing/past hiring process.

Some general requirements are:

- All the data must be validated in order to prevent inconsistency.
- the user with administrative rights must be able to grant privileges to a set of users depending upon their role in the recruitment process.
- each type of user should have different interfaces depending upon their roles.
- system should assist the AAO, the Finance and Administration, the SS committees, the Deans and the Human Resource Department’s staffs in recruiting process.
- system should be easily maintainable, changeable for change in requirements in future.
- system should track recruitment number, all applicants, and paper work.
3.2 USER CLASSIFICATION AND CHARACTERISTICS

WEAPSS is a multi user system. There are mainly three types of users: administrative users, system administrators and applicants.

- Administrative users are responsible for managing jobs and processing applications. This type of users is further subdivided to Committee chair, Committee member, Affirmative Action, the Dean, Finance & Administration and Human Resource.
- System administrator users are responsible for managing administrative users as well as other system administrator users.
- Applicants are able to post their resumes, apply for jobs, view job details, search jobs and update their profiles.

Login authentication for all types of users is done by their email address and password (created at the time of registering to the system).

Administrator Users:

<table>
<thead>
<tr>
<th>User Type</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource</td>
<td>• Add/Browse/Edit/ Delete job.</td>
</tr>
<tr>
<td></td>
<td>• Reactivate job for pool search /Re-advertise job, if job’s initial posted date is less than two years from the current date.</td>
</tr>
<tr>
<td></td>
<td>• Checking applicant’s criminal background before hiring.</td>
</tr>
<tr>
<td></td>
<td>• View all job hiring details (including administrator users’ name and date of approval).</td>
</tr>
<tr>
<td></td>
<td>• View current job(s) processing status and details</td>
</tr>
</tbody>
</table>
Including applications.

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committee Chair</td>
<td>• Assign tier and reason code to applications.</td>
</tr>
<tr>
<td></td>
<td>• Recommend candidates for interviewing and hiring.</td>
</tr>
<tr>
<td>Committee Member</td>
<td>• View all applications as well as updated resume of the applicants.</td>
</tr>
<tr>
<td>Affirmative Action</td>
<td>• Approve/Disapprove the lists of applicants on the basis of their ethnic background, for interviewing and/or hiring.</td>
</tr>
<tr>
<td>Dean</td>
<td>• Approve/Disapprove the list of applicants for meeting as well as for hiring.</td>
</tr>
<tr>
<td></td>
<td>• Specify actual salary amount for job of whose candidate has accepted the job offer already.</td>
</tr>
<tr>
<td>Finance and Administration</td>
<td>• Approve/Disapprove the salary and candidates for the job.</td>
</tr>
</tbody>
</table>
**System Administrator Users:**

<table>
<thead>
<tr>
<th>User Type</th>
<th>Responsibilities</th>
</tr>
</thead>
</table>
| Administrator | • Add/edit/delete college, its dean and department.  
              | • Add/edit/delete education major.                                                |
|             | • Add/edit/delete all types of users except the users of type Applicant, Committee member and Committee Chair. |

**Applicant:**

<table>
<thead>
<tr>
<th>User Type</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicant</td>
<td>• Register to the system.</td>
</tr>
<tr>
<td></td>
<td>• Create multiple online resumes and store them in the system.</td>
</tr>
<tr>
<td></td>
<td>• Apply for a job.</td>
</tr>
<tr>
<td></td>
<td>• Search job(s) by different criteria.</td>
</tr>
<tr>
<td></td>
<td>• View job details.</td>
</tr>
<tr>
<td></td>
<td>• View applied jobs</td>
</tr>
</tbody>
</table>
Use Case Diagrams below shows the capabilities of each type of users.

Figure 3. Use Case Diagram for System Administrator User.
Figure 4. Use Case Diagram for Applicants.
Figure 5. Use Case Diagram for administrative users
3.3 USER INTERFACE DESIGN

The WEAPSS application contains a web-based graphical user interface, designed using Adobe Photoshop for template design and Macromedia Dreamweaver for integrating design with programming code. Each interface is written using JSTL tags with html, JavaScript and CSS. JSTL provides standard implementation for typical presentation layer tasks such as data formatting and iterative or conditional content. One of the advantages of using JSTL for presentation layer is that there is no longer a need for writing actual Java code instead JSTL tags are used. JSTL tags are also much cleaner and simpler than JSP scriptlets. This kind of separation allows better encapsulation and easier maintenance of WEAPSS. The code behind the JSP pages is written in Java using the Struts framework. Since applicants for jobs may not be computer literate, each interface was properly tested for user friendliness and finally approved by customer. User interface design of current system is based on templates; so whenever there is a change in template, the pages using that template are automatically updated allowing for site wide modifications quickly.

The use of templates for WEAPSS allows the entire site to be flexible, easy to update, and provide consistency and continuity between JSP pages. Templates allow the designer to make changes to the overall design of the site quickly by revising the template file directly.

WEAPSS consists of one base template provided by the university in order to make the layout of the WEAPSS consistent with other websites of the university. For each type of user there are separate struts templates using the base template for background. This strategy was used in order to make WEAPSS easier for update in term of web design. The entire application consists of around 30 struts forms.
4 DESIGN

The sponsor of the project wanted the web team of ITS department to be responsible for the maintenance of the current project in future. After a few meetings with ITS web team, the developer got overall information of the technology and the tools to be used for the project. The web team of the ITS department was currently using Struts as framework, JDeveloper as a tool and Oracle as the backend database for creating their web projects.

Struts Framework is a robust, mature, well tested and documented framework which uses the front controller pattern and is an implementation of the model 2 architecture. In model 2 architecture, request from a client browser are sent to a controller servlet, which determines the nature of the request and passes it off to the appropriate handler for that request type. Each handler is associated with a particular model, which encapsulates business logic to perform a specific and discrete set of functions. Once the operation is completed, the results are sent back to the controller which determines the appropriate view and displays it.

Figure 6. Model 2 Architecture
Advantage of using Struts

- **Centralized File-Based Configuration**
  Struts values/mapping is represented in XML or property files. This loose coupling means that many changes can be made without modifying or recompiling the underlying Java code, and any major change can be made by editing a single file. This approach also lets developers focus on their specific tasks (implementing business logic, presenting certain values to clients, etc.) without needing to know about the overall system layout.

- **HTML Tags**
  Struts Framework provides a set of custom JSP tags to create HTML forms that are associated with JavaBeans components. This bean/form association serves two useful purposes:
  - Access to initial form-field values from Java objects.
  - Re-display forms with some or all previously entered values intact.

- **Validation**
  The Struts framework offers the flexibility of client side and server side validation. The main advantage of using the Struts validation is that it enables decoupling of validation from the application with a configurable XML file for both client and server side validations. Client side validations for the forms are enabled by JSP tags that generate the validation script. This saves the request being sent to the server but requires that java script be enabled by browser. Server-side validations can be performed by the model form beans extending the ValidatorForm. Validations for the system are handled in the validate method of the Action Forms. The error messages associated with the validation are specified in the resource bundle. [13]
4.1 HIGH-LEVEL ARCHITECTURAL DESIGN

The high-level architecture design of the WEAPSS is depicted in Figure 7. Users interact with WEAPSS through JSP pages with the browser from the client machine for all functionalities of the system. WEAPSS resides on a web server running Oracle application server.

WEAPSS is a three-tier web application (i.e. a web server, a database and web browsers using the WEAPSS site). One of the main advantages of using this type of architecture is that it is flexible and separates the “logic” of the web application from the “content” or data of the web application. In a three-tier application, the web browser sends HTTP requests to the web server which in turn requests a service and passes parameters to the application tier. The application tier services the request by making queries and updates against the database. Results are passed back to the presentation tier which creates the user interface in HTML and finally this result is returned to client browser.

Figure 7. High Level Architecture of WEAPSS
4.2 DATABASE DESIGN OF WEPASS

Figure 9 shows the database design for the project using an entity relationship diagram. In this design, each entity represents a table in the database along with the primary key (PK) and foreign key (FK), for each table.

During the database design of WEAPSS, there were many user requirement issues to be solved with the efficient database design.

Figure 8. Entity Relationship diagram of WEAPSS
Each job can be re-activated/advertised for any number of times during the two years from the initial approval to recruit. For each activation of the job, the requirement may or may not be identical to that of previous requirement. To solve this problem the developer created the “JOB_ACTIVATION” table which allows storing the activation of a job many times and the “JOB_REQUIREMENT” table which stores unique job requirements for each activation of the job. Advertisement for a job is done on the website only if there are no candidates available in current applicant pool which meet given job requirements. Most of time activations will not be advertised on the website. To address this issue the “JOB_ADVERTISE” table was created which has an entry only if the job is advertised.

WEAPSS allows the administrative users to view the applicants’ information in two ways, one by their application (not updatable by applicant once they have applied) and another by the resume (which is allowed to be updated by applicants at anytime).

One of the sponsor’s requirements was to keep track of each administrative user approving applicants for any job. So to address this problem a table named ”APPROVAL” was created which keeps track of each approval of a job by storing the administrative user’s id and approval date.

To keep the track of the processing of each application for an activation of the job, table “APPLICATION_PROCESS” was created so that it will store each application’s processing status.

One of the issues to be addressed was lock management to synchronize concurrent access to objects from multiple transactions. To solve this issue, each table has a field called “tablename_access” which is set to an integer value. Whenever a user retrieves a set of data from table, the value of “tablename_access” is also retrieved and on the time of update, first the system checks to see the value “tablename_access” matches with the
current value of “tablename_access” in respective table. If it matches change occurs and the value of “tablename_access” is incremented by 1; otherwise, no change will occur and message will be displayed.

4.3 ARCHITECTURE OF WEAPSS

WEAPSS is a web based system with users interacting with the system from the client machines. It follows MVC 2 Architecture and is divided into Model, View and the Controller.

Model:

In WEAPSS, model layer is responsible for providing the data from the database and saving the data into data store. This layer consists of all business classes and data access classes of WEAPSS. There are 6 data access classes and 23 business classes in this application. Figure 9. shows these classes and their relationships. Among these classes Job, Job_Activation, Job_Application, Resume, Applicant and School are the aggregate classes.

- Job Class:
  Job class stores the information about the recruitment number and its activations.

- Job_Activation class:
  Job_Activation class represents each activation of a job. This class stores the information about of job’s qualification & experience requirements, affirmative hiring goal, fund(s) and other general requirements such as job description, responsibilities etc.
- **Job_Application class:**
  This class represents real world’s job application which consists of information about applicant’s education backgrounds, work experiences and personal references.

- **Resume Class:**
  Resume class stores the information about applicant’s educational background, work experience and personal references as the Job_Application class does. The only difference between the Job_Application class and this class is that resumes are allowed to be updated over time but application as not.

- **User Class:**
  User class stores the information about users including their username, password and user type. User Class represents the administrative users and the System administrator for whom detail information are not required to be stored in the data store as described by the problem domain.

- **Applicant Class:**
  Applicant Class is the specialization of the User class and as the name suggests, it represents applicants. It consists of the detail information including applicant’s contact address, phone number and their criminal background information.

- **School Class:**
  This class stores name of the school and its department information.

The entire application revolves around these classes. For each of these classes there are respective data access classes. For example, there is a DB_Application class responsible for retrieving and storing all the information of the Job_Application class.
Figure 9. Class Diagram of WEAPSS.

View:

View represents the user view of the WEAPSS and is responsible for taking the input from the user, dispatching the request to the controller and then receiving responses from the controller and displaying the result to the user. There are 60 JSP pages each coded
using JSTL, HTML, Struts HTML, Struts Templates and Java Script in this layer. Each of these JSP pages is associated with respective Action classes via action mapping. Figure 10 shows the icons of some of JSP pages such as, DisplayJobs.jsp, Profile.jsp etc.

Due to size limitation of the size of the thesis only some major jsp pages are described in this manuscript.

<table>
<thead>
<tr>
<th>JSP Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayJobs</td>
<td>This page displays all the current active jobs in the system.</td>
</tr>
<tr>
<td>newRegistration</td>
<td>This page consists of the registration form for applicants.</td>
</tr>
<tr>
<td>CommitteeViewApplicants</td>
<td>This JSP page allows committee member to browse a list of applicants for a job.</td>
</tr>
</tbody>
</table>

Figure 10. Portion of ControlFlow Diagram of struts-config.xml.
<table>
<thead>
<tr>
<th>Page Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommitteeViewApplications</td>
<td>This page allows committee member/chair to view application details. The detail of the resume is displayed on this page for Committee member/chair to view.</td>
</tr>
<tr>
<td>CommitteeViewResume</td>
<td>This page provides a search form with field like keyword, education and experience etc for searching the resumes residing in the system.</td>
</tr>
<tr>
<td>PoolSearch</td>
<td>This page displays the report of each hiring. It consists of all detail information including the information about fund, committee member job descriptions, applicants’ information etc.</td>
</tr>
<tr>
<td>ViewReport</td>
<td>This webpage represents online resume for applicants and consists of number of inputs fields for education background, work experience, personal experience and criminal background information.</td>
</tr>
</tbody>
</table>
Controller:

Controller is the intermediary between Model and View. Controller is responsible for receiving the request from a client. Once a request is received from a client it executes the appropriate business logic from the Model and then produces the output to the user using the View component. In WEAPSS, Action class, DynaActionForm and struts-config.xml are the part of Controller. There are thirty Action classes such as HRAction, JobAction, searchResumeAction, displayJobAction, Registration etc. Each one of them process the specified HTTP request and create the corresponding HTTP response (or forward to another web component that will create it), with provisions for handling exceptions thrown by the business logic. Whenever there is a similar functionalities between the action classes, LookupDispatchAction classes is used instead of simple action class in order to reduce number of action classes, since LookupDispatchAction classes allows action classes to have more than one method excluding the method for validation.

Due to the size limitation of this manuscript only major Action classes and its functionalities are described below.

<table>
<thead>
<tr>
<th>Action Class</th>
<th>Functionalities</th>
</tr>
</thead>
</table>
| JobAction    | • Create a new job by pool search/advertising in website.  
• Reactivate job for pool search /Re-advertise job, if job’s initial posted date is less than two years from the current date.  
• Update/delete jobs.  
Search jobs by keyword, department or |
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRAction</td>
<td>• Generate hiring reports.</td>
</tr>
<tr>
<td></td>
<td>• Retrieve job details including each and every details of its activations/advertising in websites.</td>
</tr>
<tr>
<td></td>
<td>• Display all the hiring completed jobs and its detail report.</td>
</tr>
<tr>
<td></td>
<td>• Display all jobs on display and its detail report.</td>
</tr>
<tr>
<td></td>
<td>• Display all reactivated jobs and its detail report.</td>
</tr>
<tr>
<td></td>
<td>• Display all the jobs requesting for criminal background check.</td>
</tr>
<tr>
<td>CommitteeAction</td>
<td>• Display all jobs and its detail for committee.</td>
</tr>
<tr>
<td></td>
<td>• Update application processing status.</td>
</tr>
<tr>
<td>ResumeAction</td>
<td>• Create/browse/update/delete resumes.</td>
</tr>
<tr>
<td></td>
<td>• Display all jobs pending for approval by the Dean.</td>
</tr>
</tbody>
</table>
4.4 TESTING

Testing was started right away once the requirement document was completed, after getting feedback on the throwaway prototype. Review and inspection of the requirement document was done in a number of iterations and by the end of the requirement phase there were around 50 defects found. Once the defects were fixed the requirement document was then frozen and the incremental software development life cycle was followed to engineer the project. On completion of each module, unit testing was
conducted followed by integration and functional testing. Functional testing was carried out using boundary value testing.

Once the entire software was developed, thorough system testing was conducted by the sponsor and developer for a period of one month to ensure the reliability of the product. Some of the test cases are shown below.

**Scenario:** Advertise Job

**Test Cases for posted, deadline and starting dates:**

<table>
<thead>
<tr>
<th>Posted Date</th>
<th>Deadline Date</th>
<th>Starting Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current date</td>
<td>Greater than posted date</td>
<td>Greater than deadline date</td>
</tr>
<tr>
<td>Previous day date</td>
<td>On posted date</td>
<td>One day less than deadline date</td>
</tr>
<tr>
<td>Invalid date</td>
<td>Invalid date</td>
<td>Invalid date</td>
</tr>
<tr>
<td>Date greater than two years from previous posted date (if re advertising)</td>
<td>Date greater than two years from previous posted date (if re advertising)</td>
<td>Date greater than two years from previous posted date (if the reposting)</td>
</tr>
</tbody>
</table>

**Scenario:** Dean specifying actual salary for a job

**Test Cases for specifying salary amount:**

<table>
<thead>
<tr>
<th>Actual Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater or Equals to minimum salary</td>
</tr>
<tr>
<td>Less than minimum salary</td>
</tr>
</tbody>
</table>
Scenario: Adding a new job via pool search

Test Cases for number of applicant selected from pool:

<table>
<thead>
<tr>
<th>Number of Applicant selected from pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
</tr>
<tr>
<td>Greater than zero</td>
</tr>
</tbody>
</table>

4.5 DEPLOYING WEAPSS

WEAPSS is deployed by installing the application on a web server running Oracle Application Server (OAS). While deploying WEAPSS, it should be packaged into Web Archive (WAR) file and loaded onto a OAS. Assembling a WAR file for a Struts application can be done by creating a WAR deployment profile using JDeveloper. This will create a new deployment profile and add it to the project. A Struts runtime jar file is automatically packaged into the WAR file and so any further configuration of deployment server is not needed. Once the deployment profile is created, it can be deployed directly to Oracle application server.
4.6 SECURITY

WEAPSS is a multi-user web based application which stores and processes crucial information about each applicant and job. Some of the information that are processed in WEAPSS are very sensitive like salary information, applicant’s criminal background information. Therefore, security was of high concern throughout the development process of the project.

To ensure strong authentication and encryption of data being transmitted over Internet, 128-bit encrypted Secure Socket Layer (SSL) connection is being used for deploying this project on the Oracle Application Server. The Secure Socket Layer protocol was created by Netscape to ensure secure transactions between web server and browsers. The protocol uses a third party, a Certificate Authority (CA), to identify one end or both end of the transactions. By convention, URLs that require an SSL connection starts with https://. Although the throughput of data for a web server transmitting via HTTPS is often as little as one-tenth that of data transmission via HTTP, since security is major issue for WEAPSS, we have to compromise speed with security.

During designing database also, special care was taken to store encrypted passwords for user and regarding the input data validation, each form uses JavaScript functions to validate the value(s) before being sent for processing to controller class.
5 LIMITATIONS

WEAPSS is an initial effort to automate the entire hiring workflow, eliminating a series of paper approval documents needed to conduct and finalize a search. While WEAPSS has many capabilities, it still has some limitations.

One of the limitations of WEAPSS is that there is no provision for uploading a resume by an applicant. So each time an applicant wants to post his/her resume, he/she has to fill up the online resume form. Further, there is a provision for storing each approval or disapproval at each stage of hiring process made by administrator user, and so electronic signature for approving and disapproving at the each stage of application processing was not implemented.

As per the requirement specification, applicants can specify at most three items under the experience, education and personal reference section of a resume. The database is designed to accommodate unlimited items in these areas and therefore these limitations could be easily removed.
6 CONTINUING WORK

This section lists the requirements and enhancement to be implemented in future versions of this product:

- Email notification to applicants about interviews and job offers.
- Generate hiring report in printable version
- Generate Resume and Applications of applicants in printable version
- More friendly user interface design
- Tracking module in order to keep track of contract of hired person throughout the employment history in the university and generate a report whenever required
- Page navigation system for the entire system
- Stored procedures could be used to make queries to database more secure and the faster.
7 CONCLUSION

WEAPSS is a powerful online recruitment and application processing support system which is capable of storing and maintaining different types of user accounts, resumes, applications, jobs, and keeping track of the steps in the hiring process. It allows applicants to search for jobs based on different criteria and to post application. In addition, WEAPSS allows applicant to create online resumes which can be posted for multiple jobs. In the two years, following the first approval for recruitment for a position, WEAPSS allows HR staff to re-initiate the pool search and re-advertise the position for a number of times in case of a vacancy.

The main focus of WEAPSS is to streamline advertisement, hiring processes, save administrative time, eliminate redundant processes, and accelerate communication with candidates. Throughout each phase of the recruitment process, WEAPSS facilitates much more streamlined, standardized approach than the existing, paper-based recruitment process. Tasks such as sorting, coding, filing, and routing application materials which were previously performed manually, can now be performed automatically.

In conclusion, WEAPSS promises to be a fast, easy, cost-efficient, and effective tool for addressing a fast approaching human resource crisis for meeting ongoing part-time needs for instructional academic staff.


8 BIBLIOGRAPHY

APPENDIX A: SAMPLE WEAPSS SCREEN SNAPSHOTS

Index page Screen
### Browse Jobs Screen

<table>
<thead>
<tr>
<th>RecruitmentNo</th>
<th>Position</th>
<th>Department</th>
<th>PostedDate</th>
<th>Deadline</th>
<th>Opening</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1234</td>
<td>Associate</td>
<td>Mathematics</td>
<td>2008-03-03</td>
<td>2008-03-19</td>
<td>1</td>
</tr>
<tr>
<td>0908505</td>
<td>associate Lecturer</td>
<td>Mathematics</td>
<td>2008-03-04</td>
<td>2008-03-19</td>
<td>1</td>
</tr>
<tr>
<td>0908503</td>
<td>Assistant Professor</td>
<td>Educational Studies</td>
<td>2008-03-05</td>
<td>2008-03-19</td>
<td>1</td>
</tr>
<tr>
<td>0908505</td>
<td>ASSISTANT/Associate</td>
<td>Mathematics</td>
<td>2008-03-12</td>
<td>2008-03-20</td>
<td>1</td>
</tr>
<tr>
<td>0908501</td>
<td>Assistant professor</td>
<td>Mathematics</td>
<td>2008-03-12</td>
<td>2008-03-25</td>
<td>1</td>
</tr>
</tbody>
</table>
Human Resources: Division of Administrative & Finance

Human Resources: Division of Administrative & Finance

Job Detail Screen

Recruitment Number: 095S801
University Registrar
Records and Registration Department

POSITION DESCRIPTION

Description/Primary Responsibilities:
Reporting to the Vice Chancellor for Academic Affairs, the University Registrar provides vision, leadership, and management oversight for all major activities and the supervision of the staff within the Office of the Registrar. Manages registration, course scheduling, degree certification, transfer credit evaluation and articulation, commencement, and production of the University Catalog. Works collaboratively as a team with Career Services, Academic Advising Center, Financial Aid and admissions to deliver enrollment services. List of major titles with percent of time: University Curriculum and Academic Policy Activities (40%) - Administer academic policies and standards, including the interpretation of such rules - Serve as elected secretary to the graduate and undergraduate curriculum committees and provide curriculum service to department chairs in program, course and form preparation - Consultant for the university's action related to the enforcement of the Family Educational Rights and Privacy Act - Consultant to the university's General Education Committee, Graduate Council and to the Committee on Academic Policies and Standards Records Activities (35%) - Manage all aspects of the student registration process, including development of class schedule, class room scheduling, examination scheduling and assisting with the development of the academic calendar - Manage all operational activities related to the collection, dissemination and maintenance of student academic records - Maintain a secure environment for the storage and distribution of all academic records and related data - Coordinate the publication of the undergraduate and graduate catalog - Maintain official course master files for graduate and undergraduate courses, programs, majors, minors and concentrations - Enter and verify the certification and verification of athletic eligibility, veteran eligibility, enrollment and degree completion - Develop and disseminate students' academic transcripts and degree audits - Oversees administration of the TES (Transfer Information System) project, and re-entry application process - Respond to external requests for information - Manage development of Commencement programs and commencement ceremony scripts Office Administration (20%) - Responsible for staff training, supervision and evaluation - Responsible for budget development and oversight Other (5%) - Effectively work and communicate with professional associations on matters related to national policies and records related issues (AACRAO and WACRAO) - Work collaboratively as a team with Career Services, Academic Advising Center, Financial Aid and Admissions to deliver enrollment services

POSTED ON: 3/6/2008
APPLY BEFORE: 3/27/2008

QUALIFICATIONS:
Bachelor Degree in Operations Management or Operations Management - Experience with management of an office and evaluation of employees - Knowledge of budget preparation and management - Ability to manage complex administrative databases - Demonstrated ability to establish effective relationships with students, faculty and staff and the ability to competently interact with a culturally and ethnically diverse population of students, faculty, and staff

APPOINTMENT

Appointment Percentage: 100.0%

STARTING DATE: 7/29/2008

LOCAL INFORMATION

Information about UW-La Crosse: The University of Wisconsin-La Crosse, founded in 1909, is one of 13 four-year campuses in the University of Wisconsin System. It offers a wide array of undergraduate programs in business, education, the sciences, the arts, health science, recreation, physical education, and liberal studies as well as a variety of graduate programs. About 9,000 students attend classes on the campus, 110-acre campus.

La Crosse (pop. 51,000) is set among scenic bluffs along the winding Mississippi River at the Minnesota/Wisconsin border. With Minneapolis 2 1/2 hours to the northwest and Madison, Wisconsin, 2 hours to the southeast, La Crosse enjoys small-town charm with larger city benefits. Densely wooded coulees (gorges between bluffs), three rivers, and lush marsh provide prime locales for running, skiing, hunting, fishing, skating, boating, camping, biking and other outdoor pursuits.

For more information about UW-La Crosse: http://www.uwlax.edu

As an Affirmative Action, Equal Opportunity Employer, the University of Wisconsin-La Crosse is engaged in an effort to be a leader in Wisconsin's movement toward increased diversity and inclusiveness. Women, persons of color, and individuals with a disability are encouraged to apply. If you have a special need/accommodation to aid your participation in our hiring process, please contact the individual above to make appropriate arrangements.

Employment will require a criminal background check. A pending criminal charge or conviction will not necessarily disqualify an applicant. In compliance with the Wisconsin Fair Employment Act, UW-La Crosse does not discriminate on the basis of arrest or conviction record.

Under Wisconsin Statutes, we are required to provide, upon request, a list of nominees and applicants. A written request can exclude one from this list. Persons agreeing to be final candidates (to be interviewed) will have their identities revealed as final candidates.

Apply for this Job

Job Detail Screen

41
Applicant Registration Screen
1st part of Online Resume Screen
2nd part of Online Resume Screen
Human Resources: Division of Administrative & Finance

New Resume

Criminal background information:

University of Wisconsin-La Crosse

JOB APPLICATION SUPPLEMENT FOR FINALISTS:
CONVICTIONS AND/OR PENDING CRIMINAL CHARGES CONFIDENTIAL NOTE TO FINALISTS:
The University of Wisconsin-La Crosse (UW-La Crosse) conducts criminal background checks for all new hires. A criminal conviction or pending criminal charge may be a factor in the hiring decision.

The information requested below is required to conduct a criminal history background check. Discrimination on the basis of age, gender, race or any other protected class status under federal or state law is prohibited by UW-La Crosse policy.

A record of conviction and/or pending criminal charges is not an absolute bar to employment. Such information will be considered only if there is a substantial relationship between the circumstances of the conviction and/or pending charge and the position being applied for. The completion of this form is part of your application process. You must complete the form accurately and completely, disclosing all convictions and/or pending criminal charges for any felony or misdemeanor. Applicants who fail to complete the form or who provide false or incomplete information will not be further considered for employment.

Social Security Number: __________
Date Of Birth: __/__/____

If other names is used:
First Name: __________ Middle Initial: __________ Last Name: __________

If current address has been residence less than 7 years:
Address(Line 1): __________ Address(Line 2): __________
Cty: __________ State/Province: __________ Zip Code: __________

Have you ever been convicted of a felony or misdemeanor?
Note: Failure to disclose a conviction for any crime (meaning a felony or misdemeanor) will be considered an intentional omission.

Do you have any criminal charges pending against you?

Skills:

Microsoft Word: __________ Microsoft Access: __________ Microsoft PowerPoint: __________
Microsoft Excel: __________ Microsoft Front Page: __________ Internet Explorer: __________
Microsoft Outlook: __________

Submit
Applied Jobs Screen

Administrator/Administrative User’s Login Screen
Add Job Screen
Browse Currently Displaying jobs Screen

Job’s History Screen
### Human Resources: Division of Administrative & Finance

**Assistant/Associate (08M-056)’s Report:**

**Position:** Assistant/Associate

**Recruitment No.:** 08M-056

**Description:** The University of Wisconsin-La Crosse Physical Therapy Program seeks a licensed physical therapist faculty member. Candidates must have an advanced degree (e.g., professional or clinical doctorate). ABPTS certified clinical specialist strongly desired. Candidates must have the ability to teach physical therapy students in the area of orthopedics and engage in scholarship.

**Qualification:**

- P.K Experience: Yes
- I.Ed.: Teaching experience: Yes
- Operating: Yes
- Appointment No.: 1.0 FTE
- Posted Date: 2008-3-5
- Deadline Date: 2008-3-10
- Starting Date: 2008-3-31

**Affirmative Action Hiring Goals of the Department/Unit:**

Asian/Pacific Islander
American Indian/Native American

**Funds Information:**

<table>
<thead>
<tr>
<th>Source</th>
<th>Fonds</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Salary Information:**

- **Minimum Salary:** $40,000
- **Contract Date:** 08/01/2008 - 07/31/2011
- **Actual Salary:** $40,000

**Committee Information:**

- **Name:** Rajiv Singh, rajiv@uwlac.edu
- **Position:** Chair
- **Number of Applicants:** 2

---

### Current Status: HR approval to hire

**All Candidates**

<table>
<thead>
<tr>
<th>Name</th>
<th>Position &amp; Reason</th>
<th>Approved Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rajiv Singh</td>
<td>Committee (Resident)</td>
<td>2008-3-10</td>
</tr>
<tr>
<td></td>
<td>Committee (Resident)</td>
<td>2008-3-10</td>
</tr>
<tr>
<td></td>
<td>Committee (Resident)</td>
<td>2008-3-10</td>
</tr>
</tbody>
</table>

**Candidates Information:**

<table>
<thead>
<tr>
<th>#ID</th>
<th>Name</th>
<th>Email</th>
<th>Education</th>
<th>Graduate Degree</th>
<th>Experience</th>
<th>Application</th>
<th>Resume</th>
<th>Reason code</th>
<th>Tier Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>427</td>
<td>Rajiv Singh</td>
<td><a href="mailto:rajiv@uwlac.edu">rajiv@uwlac.edu</a></td>
<td>Bachelor</td>
<td>Physical Therapy</td>
<td>1 year</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>428</td>
<td>Robert Johnson</td>
<td><a href="mailto:robert@uwlac.edu">robert@uwlac.edu</a></td>
<td>Master</td>
<td>Physical Therapy</td>
<td>1 year</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Meeting Invitation**

**Name:** Rajiv Singh

**Position & Reason:** Committee (Resident) Meeting attended, Continued

**Approved Date:** 2008-3-11

**Candidates Information:**

<table>
<thead>
<tr>
<th>#ID</th>
<th>Name</th>
<th>Email</th>
<th>Education</th>
<th>Graduate Degree</th>
<th>Experience</th>
<th>Application</th>
<th>Resume</th>
<th>Reason code</th>
<th>Tier Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>427</td>
<td>Rajiv Singh</td>
<td><a href="mailto:rajiv@uwlac.edu">rajiv@uwlac.edu</a></td>
<td>Bachelor</td>
<td>Physical Therapy</td>
<td>1 year</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>428</td>
<td>Robert Johnson</td>
<td><a href="mailto:robert@uwlac.edu">robert@uwlac.edu</a></td>
<td>Master</td>
<td>Physical Therapy</td>
<td>1 year</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Part 1 of Hiring report**
Part 2 of Hiring report

Part 3 of Hiring report
Pool Search job’s screen for Committee

Assign Reason code & Tier screen
### Personal Information:

- **Name:**
- **Address (Line 1):**
- **City, State, Zip Code:**
- **Non U.S. Address:**

### Education Background:

<table>
<thead>
<tr>
<th>Institution Name and Address</th>
<th>Degree</th>
<th>Major</th>
<th>Graduation Date</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>university of Wisconsin</td>
<td>Master Degree</td>
<td>Computer Science</td>
<td>9/2004</td>
<td>Test</td>
</tr>
</tbody>
</table>

- **Education Experience:** 1 years
- **P-E Experience:** No
- **Research Experience:** 2 years

### Skill:

<table>
<thead>
<tr>
<th>Microsoft Word</th>
<th>Never Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Access</td>
<td>Never Used</td>
</tr>
<tr>
<td>Microsoft Power Point</td>
<td>Never Used</td>
</tr>
<tr>
<td>Microsoft Excel</td>
<td>Never Used</td>
</tr>
<tr>
<td>Microsoft Front Page</td>
<td>Never Used</td>
</tr>
<tr>
<td>Microsoft Outlook</td>
<td>Never Used</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>Never Used</td>
</tr>
</tbody>
</table>

View Applicant’s Resume Screen
### Browse Pending jobs for approval screen

<table>
<thead>
<tr>
<th>Candidate ID</th>
<th>Name</th>
<th>Email</th>
<th>Education Degree</th>
<th>Graduate Date</th>
<th>Experience</th>
<th>Ph experience</th>
<th>View</th>
<th>Reason Code</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>JohnDoe</td>
<td><a href="mailto:john.doe@gmail.com">john.doe@gmail.com</a></td>
<td>Bachelor Degree</td>
<td>5/1980</td>
<td>No</td>
<td>1 year</td>
<td>Pending</td>
<td>Interview</td>
<td></td>
</tr>
<tr>
<td>491</td>
<td>JaneSmith</td>
<td><a href="mailto:jane.smith@gmail.com">jane.smith@gmail.com</a></td>
<td>Master Degree</td>
<td>5/1980</td>
<td>No</td>
<td>1 year</td>
<td>Pending</td>
<td>Interview</td>
<td></td>
</tr>
</tbody>
</table>

### Meeting Invitation Screen

<table>
<thead>
<tr>
<th>Meeting Invitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate ID</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>E17</td>
</tr>
<tr>
<td>E26</td>
</tr>
</tbody>
</table>

**Meeting Invitation**

- [Meeting Invitation]
- [Disapprove Job]
<table>
<thead>
<tr>
<th>Candidate ID</th>
<th>Name</th>
<th>Email</th>
<th>Education</th>
<th>Graduate Year</th>
<th>Experience</th>
<th>EM experience</th>
<th>Reason Code</th>
<th>Tier</th>
<th>Select Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>422</td>
<td>John Smith</td>
<td><a href="mailto:john.smith@email.com">john.smith@email.com</a></td>
<td>Bachelor</td>
<td>2004</td>
<td>2 years</td>
<td>Yes</td>
<td>A1</td>
<td>1</td>
<td>Select Priority</td>
</tr>
<tr>
<td>423</td>
<td>Jane Doe</td>
<td><a href="mailto:jane.doe@email.com">jane.doe@email.com</a></td>
<td>Bachelor</td>
<td>2005</td>
<td>1 year</td>
<td>No</td>
<td>B1</td>
<td>2</td>
<td>Select Priority</td>
</tr>
<tr>
<td>424</td>
<td>William Jones</td>
<td><a href="mailto:william.jones@email.com">william.jones@email.com</a></td>
<td>Bachelor</td>
<td>2006</td>
<td>2 years</td>
<td>No</td>
<td>C1</td>
<td>3</td>
<td>Select Priority</td>
</tr>
</tbody>
</table>

Select Qualified Applicants from interview Screen

Browse Jobs (Dean Screen)
View Job Detail & Approve/Disapprove for Interview (Dean Screen)

View Job Detail & Approve/Disapprove for Hiring (Dean Screen)
View Job Detail & Approve/Disapprove for Hiring (AAD Screen)

View Job Detail & Approve/Disapprove for Hiring (FAD Screen)
Add Administrative/ Administrator User Screen

Pool Search Screen
### Applicant Search Result

**Name** | **Email** | **Phone** | **Degree Major** | **Graduated** | **Years** | **Years** | **H.Edu.** | **Teaching** | **Ethnic Background** | **Job History** |
---|---|---|---|---|---|---|---|---|---|---|
Sean Smith | sjsj@hotmail.com | | Computer Science | 1/1/00 | 1 | 1 | | | Asian Pacific Islander | | |
Nina Tong | nina@hotmail.com | | | | | | | | | | |
Robert Johnson | robert@university.com | | Mechanical Engineering | 1/1/00 | 2 | 1 | | | Asian Pacific Islander | | |
Jeff Chen | jchen@university.com | | Computer Science | 1/1/00 | 3 | 2 | | | African American | | |
William Wang | wtw@hotmail.com | | Electrical Engineering | 1/1/00 | 2 | 2 | | | African American | | |
Brian Singh | bks@gmail.com | | Electrical Engineering | 1/1/00 | 1 | 1 | | | African American | | |
Jeff Brown | jbrown@university.com | | Mechanical Engineering | 1/1/00 | 1 | 1 | | | African American | | |
Alex Green | agreen@university.com | | Electrical Engineering | 1/1/00 | 1 | 1 | | | African American | | |
Alice Smith | asmith@university.com | | Computer Science | 1/1/00 | 1 | 1 | | | African American | | |
Tara Evans | taevans@university.com | | Computer Science | 1/1/00 | 1 | 1 | | | African American | | |