Alumni Finder and Résumé Database Service

TEXAS A&M UNIVERSITY - CORPUS CHRISTI
Graduate Project Report

Michael Jeffries
4 Jun, 1998

Committee Members

Dr. Mario Guimaraes
Chairman

Dr. Patrick R. Michaud
Member

Dr. Carl Steidley
Member
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover Letter</td>
<td>i</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>Background and Rationale</td>
<td>2</td>
</tr>
<tr>
<td>Narrative</td>
<td>5</td>
</tr>
<tr>
<td>Environment</td>
<td>11</td>
</tr>
<tr>
<td>Procedure</td>
<td>12</td>
</tr>
<tr>
<td>References</td>
<td>39</td>
</tr>
<tr>
<td>Appendix A: Database</td>
<td>A-1</td>
</tr>
<tr>
<td>Appendix B: Forms</td>
<td>B-1</td>
</tr>
<tr>
<td>Appendix C: List Generation Table Data</td>
<td>C-1</td>
</tr>
<tr>
<td>Appendix D: Procedures Printout</td>
<td>D-1</td>
</tr>
<tr>
<td>Appendix E: User Documentation</td>
<td>E-1</td>
</tr>
</tbody>
</table>
Background and Rationale

The Alumni Finder and Résumé Database Service (AFRDS) is a service which combines the discipline of computer science with business in a unique fashion. Computer Science provides the technology which drives the information that will be required by business.

Recruiting is one of the key functions a business must perform. In a successful business, the success is often built on high quality. This means not only high quality in product and service but also in personnel, since the first two factors (product and service) are directly affected by the personnel responsible for each. In order for a corporation to employ high-quality personnel it can either wait for the people to come to it, or it can actively seek out these capable, high-quality personnel. Most successful corporations also recognize that waiting for good people is a recipe for disaster. Consequently, corporations actively pursue high-quality new recruits by maintaining recruiters in their personnel department. These recruiters have many tools in their arsenal including the résumés they review. In fact, these résumés are the starting point in the recruiting process as they allow the recruiters to review the credentials of many prospective employees quickly to determine who should be given interviews.

Interviewing is the only chance that students or prospective recruits have to sell themselves to a corporation. To get this opportunity to showcase the knowledge and abilities a prospect has, the prospect must first get the attention of a corporation or
employer. The means by which this typically occurs is via the résumé, unless a prospect has contacts and has been able to get an introduction via networking. Most students do not have networking available to them, and thus must follow the route of using a résumé. Numerous ways (via services, via schools, or via the Internet) exist for prospects to prepare résumés, but many prospects are still unaware of the power and capability of the Internet as a recruiting tool and may neglect to use this method.

By placing a résumé on the Internet, it becomes available 24 hours a day, 7 days a week to potential employers world wide. Employers are constantly searching for the most talented, high-quality individuals to hire. A great way to initially show an employer that an individual is this type of high-quality person is to present them with a résumé on the Internet. In today’s world of high technology solutions, the Internet is merely the latest technology to gain wide-spread use. By showing familiarity and comfort with the Internet, a prospect ensures corporations that they are looking at a prospect which is ready to transition to the next century.

By providing students and alumni with an easy-to-use, easy-to-access means to post résumés, the AFRDS fills a vital need for students and alumni (users) to make their talents and desires known to the corporate recruiters responsible for hiring them. This type of service is currently available to people on the Internet, however, these services are typically not free services, and most have a minimal price of $39.95. While many users would be willing to pay for this service (many already do), most users would prefer to be able to save the expense. The Texas A&M University-Corpus Christi Placement Service currently provides a method for students to post résumé information but this is a proprietary system that requires a fee and use of a disk to fill out forms. This information
must then be taken from the disk and placed in the résumé system. The AFRDS is a service that is available at any time from any location which has access to the Internet, so users will be able to reach a wider audience than just regional recruiters. The system proposed provides users with an easy-to-access (via the Internet), easy-to-use (form driven) method to post résumé information.

By providing corporate recruiters with a recruiter-friendly means to search for prospects based on various search factors (degree expected, field of study, experience, grade point average), the AFRDS fills the vital need for recruiters to obtain key information about college graduates from the university without requiring access to a proprietary system.

Finally, the functionality inherent within the résumé service module of the AFRDS can easily be extended to serve as an alumni locator service. This would allow former students and faculty members to make their locations known to others as well as find other friends with whom they might have lost touch.
Narrative

The Alumni Finder and Résumé Database Service (AFRDS) is a relational database system with various tables, as shown in Appendix A, and the World Wide Web (WWW) user interface used to access it. It contains demographic information about each user as well as information on the educational background of, experience levels of, and references for each user posting résumés.

This project consists of the design and implementation of both the underlying relational database and the interface system required to access the database via the WWW. The advantage of this type of system is the ease-of-access, power, and flexibility it provides to the end-users. Expected end-users include the prospects posting résumés, the recruiters reviewing it, and the alumni members posting/obtaining locator information. The system is designed and implemented with user-friendly features. This user-friendliness is obtained by using web forms which incorporate the use of a mouse to select hot spots (command buttons) which initiate actions to retrieve or post information to the database. Additionally, the system uses graphics and selectable images to communicate with the user and help navigate through the forms and data provided by the database.

Major Components

The final product is based on a relational database containing various tables which
allow for data input and data retrieval. It also generates a log-in form, and a main-switchboard form (navigation form) which determines how to proceed. The primary available options are to View the Database, Add to the Database, or Modify the Database. When the user chooses to add or modify the database, the data insertion features of the AFRDS uses dynamically generated forms to collect the user’s information and store it in the database tables. The data-retrieval features of the database are likewise form driven, and are also accomplished using dynamically-generated forms based on the user’s search criteria. Specifically, the AFRDS generates greeting/navigation forms/displays, four input forms, and two primary output displays. Finally, it generates the necessary navigation forms and buttons to guide the user through the AFRDS. An example of how the data might appear for a user can be seen by accessing some of the dummy data inserted for testing and demonstration purposes.

A. Database

A relational database which is organized as depicted in the entity-relationship (ER) diagram shown in Appendix A stores the user-provided information. The AFRDS then dynamically generates displays from the database as requested by the users. The format for data collection is handled by the forms component of the AFRDS. The main function of the database is to serve as a container for information and provide the necessary security features associated with data entry and data access. The Database Management System converts the user-provided search criteria obtained from the forms into selection criteria for data from the tables using the underlying database architecture.
B. Forms

The forms are the user’s interface to the database discussed above.

The user’s first encounter with the AFRDS involves the greeting and navigation forms.

These forms are described below:

1. *Welcome Page* - allows for full or view only access.
2. *Consent & Warning Page* - warns users about data usage
3. *Log-In Page* - requires username and a password, supports initial sign-up
4. *Main Navigation Page* - includes the buttons to view, add, modify and delete data as well as change the user’s password
5. *Error Page* - is accessed upon failure to achieve successful log-in

The *input forms* are:

<table>
<thead>
<tr>
<th>Demographic Information</th>
<th>Education*</th>
<th>Experience*</th>
<th>Other*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Schools Attended</td>
<td>Job History</td>
<td>References</td>
</tr>
<tr>
<td>Address</td>
<td>Dates Attended</td>
<td>Dates Worked</td>
<td>Objectives</td>
</tr>
<tr>
<td>Phone</td>
<td>Courses Taken</td>
<td>Duties Assigned</td>
<td>Additional Information</td>
</tr>
<tr>
<td>Gender*</td>
<td>Course Descriptions</td>
<td>Skills</td>
<td></td>
</tr>
<tr>
<td>Ethnicity*</td>
<td>Overall GPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classification*</td>
<td>Major GPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Professional Credentials*</td>
<td>Major</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal Qualities*</td>
<td>Degree Obtained</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* *Used only if posting a résumé*

Each of the above forms have buttons at the base of the form for navigation between forms and to submit data. With the exception of the personal data form, they
also have buttons to add/update additional data:

A. *Submit* used to post information to the database - **button**

B. *Next* used to skip to the next form in the entry sequence - **button**

C. *Done* used to post information to the database and return to the main navigation page - **button**

D. *Reset* used to reset the form to a blank form - **button**

E. *Add Additional* used to add additional data for the type of addition or modification being processed - **button**

F. *Update Additional* used to add additional data for the type of addition or modification being processed - **button**

The **output forms** are:

1. *Alumni Multi-listing Display* includes: name, address, work phone, e-mail, weblink, classification

2. *Résumé Multi-listing Display* includes: name, work phone, degree, major, overall GPA, major GPA, e-mail, weblink

3. *Traditional Résumé Display* includes: demographics (background & header), education, experience, narrative discussion, references
C. Operational Flow

The AFRDS has a general flow of operations that supports intuitive use of the system for both data-retrieval and data-entry. The first screen is the Welcome screen which has two buttons, one for view only access and one for full access including edit authorization. From there, the user proceeds to a warning screen which provides him with notice of privacy rights and other pertinent information of which he should be aware. After acknowledging this warning, the user accesses a login screen which allows the user to either login using an existing userid and password or allows the user to create a new userid and password for the current session and future use. After this screen has been successfully completed, the user arrives at a main navigation screen that allows them to:

- View alumni data (search)
- View résumé data (search)
- Modify/Delete from the database - also sends user to a simplistic search criteria screen
- Add to the database - sends user to a data-entry screen (starting with personal data)
- Change the password
- Exit

If the user opts to view alumni or résumé data, then the appropriate portion of the database, either alumni locator data or résumé information, is displayed in order based on the user’s ordering criteria. The database records meeting the specified search criteria,
which includes choice of whether to view locator information or résumé information, are
returned in a tabular multi-listing format. From this display the user can e-mail the
record owner or obtain more information about an individual by selecting either the web
link for access to the individual’s web site or clicking the Show Résumé button for
review of the individual’s résumé. If the user opts to modify the database, they are sent to
a search form, identical to that used to select the Alumni for viewing, which searches
based on the alphabetic restriction chosen (i.e. A-E, F-J etc.) and returns the data with a
radio button selector to identify which data to modify. The user is not allowed to modify
or delete data that is not his/her own. This security feature is implemented by requiring
the user to re-enter their password prior to modification or deletion. Finally, if the user
opts to add to the database, the user is sent to a data-collection form to obtain the
necessary information to add locator information or create a résumé. Prior to allowing
access the database is checked for previous entries as only one personal data entry per
user is allowed. After entering the information to be stored in the database, the user can
then activate the View Alumni or View Résumé button to view his/her locator
information or résumé. If the user chooses to start over, the information provided is
deleted and the user re-enters the data. After completing the data-entry, the user is
returned to the main navigation switchboard screen. Help during a session for the various
data entry and search selection forms is provided by using JavaScript to post a help
message in the browser's status bar which changes as the user moves from field to field
within a given form.
Environment

The interface system displays the Résumé forms and results to any web browser such as Netscape. The ORACLE 7 relational database system running on a DEC Alpha 3000 is used to store and provide the résumé information. The Hypertext Markup Language (HTML) is used to display the résumé results in both the tabular multi-listing format and the traditional résumé report. The Oracle Web Server (OWS) is used to access the Oracle database via the Internet. It is also used to dynamically generate the above mentioned HTML reports as previously discussed. The procedures used were developed using the Oracle Programming Language/Structured Query Language (PL/SQL) and Javascript. SQL*Plus is used to query the database.
Procedure

The following steps were taken during the design process of the Alumni Finders and Résumé Database System (AFRDS).

1. Interviewed, Mrs. Stein, the director of student counseling services, to determine what type of system is currently in place and what kind of system is desired.

2. Researched the Internet sites currently available to perform similar services for users.

3. Researched several Oracle-Web interface software products such as the Database Interface (DBI) working with a Common Gateway Interface (CGI) and the Oracle Web Server. Determined that the best approach would be to use the inherent capabilities of the Oracle Web Server which was specifically designed to work with the Oracle Database.

4. Evaluated cost effectiveness of alternative software and hardware.

5. Designed a database to house the alumni and résumé data (Appendix A).

6. Developed a Javascript HELP system to aid users during form data entry.

The following steps were taken to complete the development of the (AFRDS):

1. Implemented the system using HTML, Oracle 7.0, and Oracle Web Server.

2. Installed and tested the system with test/sample data.
3. Developed user documentation - Appendix E to this report.

The figure below gives a graphic view of how the AFRDS connects and functions:

**Alumni Finder and Resume Database System Layout**

![Diagram of alumni finder and resume database system layout](image)

**Figure 1**
A. Database

1. Storage Tables

Tables are included which are used to store the data provided by the input forms discussed in the input forms section below. These tables and their relationships are shown in greater detail in Appendix A. However, a brief review is provided to familiarize the reader with the basics of how the database is organized. The tables (shown below in italics) used to store the data are describe in greater detail in Appendix A and shown below along with some representative fields:

**Personal Data** - (PID*, SSN, LNAME, FNAME, MI, ADDRESS, CITY,
STATE, ZIP, COUNTRY, HOME PHONE, WORK
PHONE, FAX, EMAIL, WEBLINK, PERSONAL
QUALITY (3 MAX), PROFESSIONAL
CREDENTIALS (3 MAX), GENDER)

**School Data** - (SID*, SCHOOL, START DATE, END DATE, OVGPA,
MAJGPA, DEGREE, MAJOR, COURSE
DESCRIPTION, COURSE KEYWORD (5 MAX))

**Employment Data** - (EMPID*, COMPANY, CITY, STATE, HIRE DATE,
END DATE, JOB_TITLE)

**Other Data** - (OID*, RFNAME, RLNAME, RCITY, RSTATE,
RWPHONE, RHPONE, TIME KNOWN,
RELATE, RCOMPANY, RJOB_TITLE)

Security - (SSN*,USERNAME*,PASSWORD*,SESSION_ID,TIME_IN )
*
- Mandatory fields for full access, all others are optional for the user

2. List Generation Tables

In order for the user to have a wide range of choices, a large amount of list-formatted data must be organized and stored in tables for use by the list boxes on the form. These tables and their relationships, like the storage tables, are shown in greater detail in Appendix A. The tables required are shown below:

States - (SABBREV, SNAME)
 Colleges - (COLID, COLLEGE)
 Majors - (MAJID, COLID, MAJOR)
 Degrees - (DID, DTYPE)
 Personal Qualities - (PQID, QUALITY)
 Career Interest - (CARID, FIELD, POSITION)
 Location Preference - (LOCID, LOCATION)

3. Security

Security for the database is accomplished through the use of the Oracle Database security capabilities like granting privileges to select, insert, update or delete on a table,
and execute on a procedure. It also uses the Oracle Web Server and the developed PL/SQL login procedures which control the login and authentication for return users, and username/password validation for first-time users. The fashion in which this works is that the user reaches a login page where they enter their username/password combination. If they have not already entered a username/password combination then they must repeat their password and supply an ssn to be added to the security table in the database. This table has fields for the ssn, username, password, session_id (assigned by Oracle at each login) and a time_in field to keep track of login time. The user is limited to 2 hours usage prior to being required to login again. This feature was added to aid with innocent mistakes that users sometimes make when leaving a terminal. This table is then checked with each login and the time_in field is checked against the system time to determine if the 2 hour limit has been exceeded requiring a re-login. The checkTime procedure used to verify the 2 hour limit is shown in Appendix D along with the rest of the procedures which comprise the AFRDS.
B. Forms

Access to the database is accomplished through the use of forms. The greetings and input forms which are used are discussed below:

The greetings and interaction forms are:

1. Welcome Page,
2. Consent & Warning - warns about use of information,
3. Log-In Page - includes username and a password,
4. Error Page - accessed upon failure to achieve successful log-in, and
5. Main Navigation Page - includes the buttons to view, add, modify/delete, or exit the database as well as a button to update the user's password

The input forms are:

1. Demographic Information - includes: Name, Address, Gender, Ethnicity, Classification
2. Education - includes: Schools attended, Courses taken, Course Descriptions, Degrees Obtained, Major Areas of Study, Grade Point Averages
3. Experience - includes: Job history and duties, Location of Job,
4. Other - includes: Objectives, References, Additional Information,
5. *Search* - includes: Form fields to narrow the search.

These forms can be found in Appendix B and use, the post method of data transfer.

The output forms are:

1. *Alumni Display* (includes: Name, Address, Work Phone, E-Mail, WebLink, Classification)

2. *Résumé Tabular Display* (includes: Name, Work Phone, GPAs, Major, Degree, E-Mail, WebLink)

3. *Résumé Traditional Display* (includes: Background, Education, Experience, References, Entire Resume)

These forms are generated using dynamic SQL for the tabular listings and static SQL for the traditional listings.

1. **Greetings/Interaction Forms**

On entering the web site, the user is greeted with a welcome page. This page is simply a description page which explains the purpose of the AFRDS system. At the bottom of this page is an option to have full access or enter to view data only. Activation of either of these buttons causes the browser to go to the next page which contains the consent to allow use and display of the provided information, as well as a general
warning to the users. At the bottom of this page is be three buttons. The first button is a "Do Not Accept" button, which shows a brief message informing the user that acceptance is required to continue on in the system. The second button is an acceptance button "Accept", which moves the user to the next page in the system. The final button is the "Exit" button which takes the user to the TAMUCC Alumni Web Page. After acknowledging and accepting the warning, the user is transferred to the log-in page where they enter a userid and a password. After entering the information the user selects a button at the bottom of the page to login for access to the database. If this is the first time a user has logged in, then the user is directed to enter a new username/password combination along with the user's social security number. This information is then sent to the database where it is stored and then the user is required to login again. After the first login the user logs in and is transferred to the next screen in the system which is the main navigation screen. If the login attempt fails the user is given an error message and asked to login again. The main navigation screen has activation buttons that allow the user to either view alumni data, view résumé data, add data, modify data, delete data, change the password, or exit.

2. Input Forms

The first input form is used to obtain demographic information from the user. This form is a pre-designed input form that is dynamically generated at time of access by the OWS combining text boxes, list boxes, and radio buttons. A listing of the type of input to be used for each field is shown on the next page.
Identifier: SSN (obtained from security table)

Text Box: Last Name, First Name, Middle Initial, Address, City, ZIP Code, Birth Date, Home Phone, Work Phone, FAX, E-mail Address, Web Link, Professional Credentials (3), Country

List Box: Classification (Senior, Graduate Student, Alumni ... etc.)

Ethnic Background (African-American, Asian, Caucasian, Hispanic, Native-American, Other, Not Supplied)

State (AL, AZ, ... WY)

Personal Quality (Industrious, Meticulous, Insightful ... etc.) (3)

Career Interest (Art, Architecture, Electrical Eng., ... etc.)

Location Preference (Belgium, San Diego, Asia ... etc.)

Radio Button: Gender (Male, Female) (can be left blank if desired by user)

The second input form is used to obtain education-related information from the user. Like the demographic input form, the education input form is a pre-designed form which is dynamically generated at time of access. It also combines text boxes and list boxes. A listing of which type input is used for each field is shown on the next page:
The third input form is used to obtain experience-related information from the user. The employment input form is a pre-designed form that is dynamically generated by the OWS combining text boxes, list boxes, and radio buttons. A listing of which type input is used for each field is shown below:

Text Box: \textit{Company, Address, City, State, Zip Code, Hire Date, End Date, Position}

Text Area: \textit{Position Description}

Radio Button: \textit{Experience} (None, 0-12 months, 12-24 months, 2-5 Years, > 5 Years)

The fourth input form is used to obtain reference information and any additional narrative information from the user. The reference input form is a pre-designed form using text boxes and text areas (multi-line text). A listing of which type input is used for each field
is shown below:

**Text Box:**  
*Reference (Last Name, First Name, Address, City, State),*  
*Time Known, Relationship, Phone (Home, Work)*

**Text Area:**  
*Narrative Discussion*

The final input forms are used to pass queries for specific data to the database and are comprised of some of the key fields used in the first three input forms. These search criteria input forms are pre-designed forms using a text box and list boxes to select which resume information to display, or radio buttons to select which alumni information to display, which user records to update, or which user record to delete.

**Text Box:**  
*GPA*

**List Box:**  
*College, Major, Degree, Order By*

**Radio Buttons:**  
*Alphabetic Choice (for all but résumés)*

3. **Output Forms**

Prior to accessing the output forms, the user must first go through a search criteria form (discussed earlier) which narrows the focus of the data returned. The search forms
are shown in Appendix B.

The Alumni display takes some pre-chosen fields from the database and displays them in a tabular (multi-user) format. The key fields are chosen to provide location information to interested people. The output contains two links (a web link for access to an individual's web site, and an e-mail link to send e-mail to an individual’s e-mail account). The fields to be displayed are shown below:

NAME, ADDRESS, CITY, STATE, WORK PHONE, E-MAIL, WEB LINK

The Résumé display also takes some pre-chosen fields from the database and again displays them in a tabular (multi-user format). The key fields are chosen this time to provide a snapshot of an individual’s background to interested people (usually recruiters). The output also contains two links - a web link for access to an individual’s web site, and an e-mail link to send e-mail to an individual's e-mail account. The fields to be displayed are shown below:

SELECTION BOX*, NAME, WORK PHONE, MAJOR, DEGREE, GPA, E-MAIL, WEblink

* - a radio button to indicate if you wish to view a particular résumé

In both of the multi-listing outputs forms, tabular data is placed into a pre-formatted HTML table that expands as required to support additional data by the amount of records returned from the search criteria. An example of this form can be found in

23
Appendix B.

The final output form takes the stored database fields and display them in a traditional résumé format.
C. PL/SQL Procedures

The connectivity between the database and the Internet is achieved using the Oracle Web Server. The procedures required and their functions with respect to the forms and displays are shown below in Figure 2:

Figure 2

The proposed procedures and their functions and relationships are discussed in the remaining pages.
The AFRDS system is built around two core PL/SQL packages and one auxiliary help package which the Oracle Web Server (OWS) uses.

<table>
<thead>
<tr>
<th>Control Package</th>
<th>Java Package</th>
<th>Forms Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_afrds</td>
<td>pers_msg</td>
<td>pers_data_form</td>
</tr>
<tr>
<td>consent</td>
<td>school_msg</td>
<td>school_data_form</td>
</tr>
<tr>
<td>entry</td>
<td>employ_msg</td>
<td>employ_data_form</td>
</tr>
<tr>
<td>login</td>
<td>other_msg</td>
<td>other_data_form</td>
</tr>
<tr>
<td>process_login</td>
<td>login_msg</td>
<td>in_pers</td>
</tr>
<tr>
<td>new_user</td>
<td>up_pass_msg</td>
<td>in_school</td>
</tr>
<tr>
<td>user_check</td>
<td>search_msg</td>
<td>in_employ</td>
</tr>
<tr>
<td>main</td>
<td>browser_chk</td>
<td>in_other</td>
</tr>
<tr>
<td>main_hdr</td>
<td></td>
<td>up_pers</td>
</tr>
<tr>
<td>main_body</td>
<td></td>
<td>up_school</td>
</tr>
<tr>
<td>main_select</td>
<td></td>
<td>up Employ</td>
</tr>
<tr>
<td>alt_main</td>
<td></td>
<td>up_other</td>
</tr>
<tr>
<td>alt_main_hdr</td>
<td></td>
<td>delete_select</td>
</tr>
<tr>
<td>help</td>
<td></td>
<td>delete_process</td>
</tr>
<tr>
<td>search</td>
<td></td>
<td>delete_all</td>
</tr>
<tr>
<td>where_build</td>
<td></td>
<td>delete_pers</td>
</tr>
<tr>
<td>auth_form</td>
<td></td>
<td>delete_school</td>
</tr>
<tr>
<td>auth_chk</td>
<td></td>
<td>pick_school</td>
</tr>
<tr>
<td>update_password_form</td>
<td></td>
<td>delete_employ</td>
</tr>
<tr>
<td>process_uppass</td>
<td></td>
<td>pick_employ</td>
</tr>
<tr>
<td>show_resumes</td>
<td></td>
<td>delete_other</td>
</tr>
<tr>
<td>res_hdr</td>
<td></td>
<td>pick_other</td>
</tr>
<tr>
<td>res_menu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>menu_process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>show_back</td>
<td></td>
<td></td>
</tr>
<tr>
<td>show_educ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>show_employ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>show.refs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>show_all</td>
<td></td>
<td></td>
</tr>
<tr>
<td>checkTime</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As can be seen, there are numerous procedures to prevent any one procedure from becoming too cumbersome or difficult to work with. The modularization of the procedures makes for easier debugging and correction of errors as well as improves the speed and ease with which the AFRDS can be modified or improved.
Control Package:

The first procedure the user encounters is the `start_afrds` procedure which is used to create the entryway to the rest of the AFRDS system. This procedure allows for two methods of entry. The first method requires no login and gives the user access to the View Alumni and View Résumé procedures via the alternate main procedure. The second method requires a login by the user to gain access to the entire system which allows for the insertion, modification and deletion of data as well as for the viewing of database data. This procedure then forwards the user’s desire to the consent procedure.

The `consent` procedure displays a warning to the user about data entry, any restrictions and expected decorum while using the system. It provides three buttons to either accept or not accept the terms and conditions of entry, and an exit button. If the restrictions are not accepted, then the user continues to view the consent page. If the accept button is activated then the user is allowed to continue and access the system as requested at the welcome page. The final button on the entryway is powered by a javascript which returns the user to the Alumni Association's homepage.

The procedure used to process the user's entry selection is accordingly called the `entry` procedure and it either directs the user to the `alt_main` procedure for view only access or the `login` procedure for full access of all the capabilities of the AFRDS system.

The `alt_main` and `alt_hdr` procedures function the same way as their counterparts, the `main` and `main_hdr` procedures but limits access to viewing only by not providing a session_id for edit authorization.

The `login` procedure is used to create a login form that the user fills out. The data is sent to the next procedure, the `process_login` procedure which then examines the login
data submitted along with the user's notification of login as either a new user or return user. It takes this information and pass it along to either the `new_user` procedure or the `user_check` procedure.

The `new_user` procedure takes all of the data and inserts it into the security table. As it does this it also causes a session id to be created along with a time stamp indicating when the user logged in. This is used to limit full access to data to 2 hours before the user is required to login again to obtain a new session_id and reset the login time. After logging in the first time the user is required to login again using the new data. This next login like any future logins then accesses the `user_check` procedure.

The `user_check` procedure checks the security table in the database to verify the username/password combination provided is authorized for access. Upon determining that access is authorized this procedure makes a call to one of Oracle's native procedures, the DBMS_SESSION.UNIQUE_SESSION_ID procedure to obtain a session_id for the user and then transfers control to the main procedure. From this point on, all procedures use the session_id procedure as an input to verify authorized access.

The `main` procedure is simply a procedure to create two frames called the "hdr" and the "body". It does this by making a call to a lower level procedure which creates all of the HTML coding to create generic frames. This `framerows` procedure simply takes inputs provided by a calling procedure and creates the necessary frames. In the case of the main procedure, the hdr and body frames are created with the source for the frames being two other procedures.

The first of the two procedures is the `main_body` procedure which simply displays another welcome message beneath a graphic of the University. It also has links to the
Alumni Association's web page and to the Career Planning & Placement Center's job links web page.

The workhorse of the two procedures is the `main_hdr` procedure which simply creates a form with buttons on it to navigate throughout the AFRDS. It does this by calling the `buttons` procedure and assigning action to the `main_select` procedure. An important note is that the assigned action has as its target location, the "body" frame so that further information is displayed in the body frame while the hdr frame maintains the navigation buttons. One additional button is created separately which uses Javascript and the onClick event to change the parent window location of the browser to the desired location, in this case the Alumni Association web page.

The `buttons` procedure, like the framesrow procedure is a low level procedure which takes input from a calling procedure and creates generic buttons which are all assigned to the action source in the open form call.

The `main_select` procedure in turn takes the action label passed from the user selection of a desired action and determines what procedure in the AFRDS should be called. Prior to calling any other procedure, `main_select` calls the `checkTime` function to verify that the 2 hour session limit has not been exceeded. This procedure then calls either the `pers_data_form` procedure to add data, the `search` procedure to view alumni, view résumés, modify data, or delete data, the `update_password_form` procedure to change the user's password, or the `exit` Javascript to exit AFRDS and go to a specified location.

The `pers_data_form` procedure is covered in a discussion of the Forms Package.
The **search** procedure is the basic procedure which begins the data lookup, modification or deletion actions. It first creates either 1) a simple form with a list box selection to allow the user to narrow the scope of the data searched to a particular college such as that of Engineering, or Science and Technology, etc., if searching for résumés; or 2) a simple form with radio buttons to alphabetically narrow the search if searching for alumni or updating/deleting records. After making a choice, if searching for résumés, the procedure calls itself but this time transfers to a different input form consisting of list boxes and text boxes to create more specific searches. This refine search criteria form lets the user choose by degree type, major, minimum overall GPA, or any combination of these. It also allows the user to determine what order the data is displayed in up to two sorts deep. After submitting the user’s choices for a refined search, the search procedure passes the data to the **where_build** procedure. If searching for alumni data or updating/deleting records, the user is transferred directly to the **where_build** procedure.

The **where_build** procedure then takes the incoming data as variables and creates a query using dynamic SQL for the résumé search (SQL which changes based on the input provided). The dynamic SQL uses the following string variables: **select_list**, **table_list**, **where_clause**, and **order_by_clause** to pass data into the built-in Oracle procedure DBMS_SQL.PARSE. For the alumni search and when updating/deleting records, the DBMS_SQL.PARSE procedure uses a prepared input with the only change to the input coming from the where clause which is modified to search for alphabetical matches if desired by the user. A few other DBMS_SQL procedures, the DEFINE_COLUMNS, COLUMN_VALUES, OPEN_CURSOR, FETCH_ROWS, and CLOSE_CURSOR are also called to make the dynamic SQL work properly. These
however are straight-forward procedures to use and are more forgiving than the PARSE procedure which is the heart of the dynamic SQL. The specific use of these built-in procedures is shown in the code printouts provided in Appendix D. In all cases except for the view alumni request, the procedure creates radio buttons for selecting a particular record to work with. This radio button has the record ssn associated with it to allow follow-on procedures to identify which record is to be deleted, modified, or shown in résumé format. If not viewing alumni, it also creates a submit button which says either “Update record”, “Delete record”, or “Show Resumes” to continue with the user’s desired action.

If a user indicates the desire to update or delete a record then a further requirement to re-enter the user’s password calls the **auth_form** procedure. This is done so that if a user were to walk away from a terminal or somehow leave a terminal operating with a valid session in progress, then another user is not able to gain access and destroy or modify data without the proper password.

The **auth_form** procedure simply creates a single password box form which obtains the user’s password for verification against the security table. The data entered in the password box is the passed to the **auth_chk** procedure which performs the actual verification and creates appropriate error messages on an unauthorized access attempts. If successful authorization occurs then control is transferred to the Form’s Package and the **pers_data_form** for updates or the **delete_pers** procedure for deletions.

If the user indicates they wish to view résumés, then the **where_build** procedure transfers control to the **show_résumés** procedure which operates much the same way as the main procedure does in that it simply creates frames. This results in the main
navigation buttons of the AFRDS being supplant by a new set of résumé navigation buttons. The “hdr” frame then has as its source the res_menu procedure while the “body” frame has the res_hdr and any subsequent procedures as its source.

The res_hdr procedure simply creates a PL/SQL query that is sent to the database to retrieve the first name, middle initial, last name, address, city, state, zip code, work phone, and e-mail from the personal data table. This procedure uses a static SQL query instead of the dynamic SQL query previously discussed.

The res_menu procedure works like the main_hdr procedure to create a form with a set of navigation buttons to access the appropriate portion of the desired résumé. Again the buttons are created with a call to the buttons procedure. The action assignment for the from created is the menu_process procedure which simply takes the button pressed and accesses the desired display procedure as well as again calling the res_hdr procedure. This is done especially for when viewing the entire résumé so that if the user prints out the frame they will get a full résumé printout. In this case it simply acts as a flow director. Like the main navigation procedure, before continuing to display any data, the menu_process procedure calls the checkTime function to verify the 2 hour session limit has not been exceeded. The choices available on the résumé menu buttons are: Background ; Education ; Employment ; References ; Entire Résumé ; and Return to Main.

The corresponding procedures which get called by the menu_process procedure are:

    show_back ; show_educ ; show_employ ; show.refs ; show_all ; Javascript

These procedures are discussed in a group since each procedure essentially does the same thing. Each procedure creates an HTML table and makes a call to the database with the
appropriate query. The display of multiple data sets for each query is accomplished by
looping until the fetch procedure initiated by Oracle no longer returns any data. At that
time the procedure finishes the table design by closing the last row and the table itself.
The show_all procedure works the same way except that it keeps the table open as it
shifts from one query submission to the next until the last query is completed at which
time it closes the last table row and the table itself. The Javascript procedure simply
redirects the parent.hdr.location and the parent.body.location to new sources.

The final procedures in the Control Package is the update_password_form
procedure and the process_uppass procedure. Update_password_form simply creates a
form with three password fields, one for the old password, one for the new password and
one for a verification password. This data is then passed to the process_uppass
procedure by clicking on the “Update password” button on the form. If any field is
incorrect such as an improper old password or unmatching new passwords, then an error
is generated and the user is told to try again. Upon successful update of the password the
user is notified and then returned to the main_body procedure for the user’s next entry
choice. Finally, AFRDS can be accessed at the followin URL:

http://robin.tamu.edu:8890/apps/owa/jeffries.control_proj.start_afnds
Forms Package:

The first procedure to be called when adding or modifying data is the *pers_data_form* procedure. Like each of the form generation procedures to follow, *pers_data_form* has code in it to determine if the user is attempting to add new data or to modify existing data. Unlike the other form generation procedures, this procedure does not allow creation or addition of a second set of personal data for the user’s SSN. Instead it creates the form shown in Appendix B. If the user has indicated they are adding data then this procedure assigns the *in_pers* procedure as the form action. If updating, data the *up_pers* procedure is called by the form action assignment. This form is a straightforward form to insert or update data and has no mandatory field in it since the user’s SSN is automatically inserted if entering data the first time or is used to select the record from the personal table to update. To ensure that the next available PID is used, the procedure queries the database personal table to obtain the maximum PID currently in the database. This value is then increased by 1 by the code in *pers_data_form* and this value is used for the PID of the new data being inserted.

The *in_pers* procedure simply takes the data from the *pers_data_form* and places it into an SQL insert statement for insertion into the personal table. It has the form:

```sql
INSERT into jeffries.personal

(PID,LNAME,FNAME......)

VALUES (pid,lname,fname ......) - where the upper case labels correspond to table columns and the lower case labels correspond to the variables passed in from the

*pers_data_form*.
Based on the submit value the procedure determines whether to call the next form from the `school_data_form` procedure or to recall the `main_body` procedure.

The `up_pers` procedure is similar to the `in_pers` procedure except that data already exist in the table for the user and it is simply being modified or updated using a format such as follows:

```
UPDATE jeffries.personal
SET LNAME = lname
WHERE SSN = user_ssn;  
```

- where the upper case labels correspond to table columns and the lower case to the variables passed in from the form.

Again, based on the submit, the procedure determines whether to call the next form from the `school_data_form` procedure or to recall the `main_body` procedure.

The next procedure, the `school_data_form`, is similar to the `pers_data_form` procedure with two main differences. The first difference is that a button has been added to allow for the addition of more than one school record for a given user. The second difference is that because there can be multiple entries for a given user, when updating data, it selects all the possible school data records associated with the user’s SSN and if there is more than one record associated with the user’s SSN it displays a selection box, school, start date, end date and overall gpa in a table. The selection box (radio button) is assigned the sid from each record for which data is shown.

When adding data, the data is then sent to the `in_school` procedure which functions identically to the `in_pers` procedure except that it adds data to the school table and allows the user to select to add additional school data resulting in the `school_data_form` procedure being recalled. If the data is submitted with no additions
then the procedure inserts the data into the school table and calls the `employ_data_form` procedure to load the next input form. If the user decides to stop at this point the data is inserted and the `main_body` procedure is again called.

When updating data, the data is sent to the `up_school` procedure which functions identically to the `up_pers` procedure except that it uses the SID instead of the SSN for selecting which record is to be updated in the school table. This is because in the school table the SSN does not have to be unique.

The remaining data entry and update procedures function identically to the `school_data_form`, `in_school`, and `up_school` procedures except that they work with data in different tables and display different data to the user when conducting updates.

The `employ_data_form` procedure displays a selection box corresponding to the empid, company, hire date, end date, and job title for each record corresponding to the user’s SSN. While the `other_data_form` procedure displays a selection box corresponding to the oid, name, time known, and relationship for each record corresponding to the user’s SSN.

These procedures in turn call either the `in_employ` or `up_employ` procedures or the `in_other` or `up_other` procedures respectively to either insert new data or update existing data.

The remaining procedures in the forms package perform the delete functions and simply delete data from each table for the user’s SSN. The procedures are the `delete_select`, `delete_process`, `delete_all`, `delete_pers`, `delete_school`, `pick_school`, `delete_employ`, `pick_employ`, `delete_other` and `pick_other` procedures. They delete data from the appropriate tables by submitting an SQL delete statement of the form:
DELETE from tablename

WHERE SSN = user_ssn;

The delete_select procedure simply allows the user to choose whether to delete all
records or specific records for the user. The choice is sent to the delete_process
procedure which then transfers to either the delete_all procedure to delete all records or to
the delete_school procedure to begin selecting which specific records to delete. The
procedure does not initiate personal data record deletion when specific records are to be
deleted as this would result in having unaccessible records (via AFRDS) in the database.

The delete_all procedure simply processes five delete statements to delete data
from the various tables using SQL code similar to that previously discussed.

The delete_school, delete_employ, and delete_other procedures each display
which records are available for deletion along with a selection button for each of the
records. The specific record to be deleted is selected and the choice is submitted to each
of the pick_school, pick_employ, and pick_other procedures, respectively, for processing.
These pick_xxx procedures then perform the deletion of the specific record using the
SID, EMPID, OID (vs. SSN) to select the proper record to delete.

These two packages form the core of the AFRDS and perform all required tasks
from form generation to data processing and data submission.
Java Package:

The final package is a set of procedures written using JavaScript encased in Oracle Web Server statements. The JavaScript procedures are used to convey the various help messages which appear in the web browser status bar as the user navigates from one field to the next within a given form. The JavaScript is activated by either the `onClick` or the `onFocus` events which are associated with form elements.

The following table show the correlation of JavaScript procedures to the Control and Forms Package procedures:

<table>
<thead>
<tr>
<th>Java Package Procedure</th>
<th>Package</th>
<th>Correlating Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>pers_msg</td>
<td>Forms</td>
<td><code>pers_data_form</code></td>
</tr>
<tr>
<td>school_msg</td>
<td>Forms</td>
<td><code>school_data_form</code></td>
</tr>
<tr>
<td>employ_msg</td>
<td>Forms</td>
<td><code>employ_data_form</code></td>
</tr>
<tr>
<td>other_msg</td>
<td>Forms</td>
<td><code>other_data_form</code></td>
</tr>
<tr>
<td>login_msg</td>
<td>Control</td>
<td><code>login</code></td>
</tr>
<tr>
<td>up_pass_msg</td>
<td>Control</td>
<td><code>update_password_form</code></td>
</tr>
<tr>
<td>search_msg</td>
<td>Control</td>
<td><code>search</code></td>
</tr>
</tbody>
</table>

One final procedure, the `browser_chk` procedure, exists in the Java package to determine which browser type and version is being used. This information is used to aid in proper formatting of the display and output of the traditional résumé. This final package completes the AFRDS making for a flexible and powerful locator and résumé service.
References


