# Engineering Research Center User Account Request System

By
Tanya Leigh George

A Project
Submitted to the Faculty
of Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science

Mississippi State, Mississippi

December 1998

Name: Tanya Leigh George

Date of Degree: December 18, 1998

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Julia Hodges

Title of Study: ENGINEERING RESEARCH CENTER USER ACCOUNT REQUEST SYSTEM

Pages in Study: 22

Candidate for Degree of Master of Science

Keeping track of information can be a tedious and time consuming job. Replacing manual systems with a computer database system is done to make the job easier and more effecient.

This paper details the development of the User Account Request System. The development process followed the software waterfall model. The old system involved filling out a paper version of an account request form; this form was then filed into binders. The new system replaced the form and binders with a web form and an Oracle database including an Oracle GUI form.

# ACKNOWLEDGMENTS

I would like to thank the staff of the ERC, especially the system administration, for all of their help. Specifically, I would like to thank Trey Breckenridge, my customer, for hiring me in the position of Database Developer.

I would also like to thank my committee members. Thanks to Dr. Carter for helping to bring me here to Mississippi State. Thanks to Dr. Donna Reese for being my interim Major Professor. Thanks to Dr. Hodges for being my final Major Professor and my Project Director.

My final thanks goes to Lance Burton, Jasper, and Gizmo. Without their help and continous support, I would have left a long time ago.

# TABLE OF CONTENTS


Page

ACKNOWLEDGMENTS . . . . . ii

LIST OF FIGURES . . . . . iv

CHAPTER

I. INTRODUCTION . . . . . 1

II. SYSTEM SPECIFICATION AND DESIGN . . . . . 3

Requirements Analysis and Definition . . . . . 3
System and Software Design . . . . . 4

III. IMPLEMENTATION AND UNIT TESTING . . . . . 5

Implementation . . . . . 5
GUI Form Implementation . . . . . 5
Web Form Implementation . . . . . 9
Unit Testing . . . . . 12

IV. INTEGRATION AND SYSTEM TESTING . . . . . 13

V. CONCLUSIONS AND FUTURE WORK . . . . . 15

REFERENCES . . . . . 16

APPENDICES

A. PROJECT CONTRACT . . . . . 17

B. LIST OF DELIVERABLES . . . . . 19

C. USER ACCOUNT FORM . . . . . 21

# LIST OF FIGURES

Page

FIGURE

1 User Account Request System Architecture . . . . . 4

# CHAPTER I

# Introduction

The National Science Foundation (NSF) Engineering Research Center (ERC) at Mississippi State University (MSU) is a highly computer–oriented facility. The employees work on a multi–user network requiring each employee to have a unique identifier on the computer system. All employees, from the front office staff to the researchers to the student workers, are assigned computer accounts. Additionally, professors use the ERC's computer resources in teaching their classes, so class accounts are needed in addition to the employee accounts.

Under the old system, all of these accounts required a hard copy version of the Account Request Form to be filled out and signed by the employee's supervisor or the class professor. Once the account was created, the form was filed in an active account binder. This binder could contain a few hundred forms at one time, making it very difficult to locate important information when needed. When the accounts were deleted, the matching form was moved to the deleted account binder(s). This system of binders was extremely inefficient for finding information.

To improve the efficiency of locating user account information and to speed up user account processing, a new system using a database was required. Oracle was chosen for the database system because it was the only database development system that was available, but it was also the ideal choice. The Oracle database tools have the benefit of working across most operating system (OS) platforms and have

an interface to the web. The ERC uses mainly Sun Microsystems and Silicon Graphics (SGI) workstations, making cross–platform–use a necessity.

There are two subsystems to the Oracle database system: a graphical user interface (GUI) form and a web form. The main purpose of the GUI form is to give the system administration a simplified way of entering, updating, and deleting user data. The GUI form also will act as an interface to running the OS scripts used to create, lock, and delete the user accounts. The purpose of the web form is to replace the hard copy version of the Account Request Form, including the advisor's signature, and to create a simple way of requesting class accounts.

# CHAPTER II

# System Specification and Design

This system was developed by following a waterfall model for software development, organized into five stages. They are requirements analysis and definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance. This chapter discusses the first two stages.

## Requirement Analysis and Definition

The requirements were determined by an interview with the customer, a system administrator at the ERC. The requirements were to replace their current system of tracking user account data with a more efficient one. The hard copy of the User Account Form, including the signature, was to be replaced with an online web form that would submit user requests in a queue. The filing binders were to be replaced with a GUI Oracle database form that would allow the customer to add, update, and delete user accounts. This form would also provide easy ways of mailing account expirations to employee supervisors, locking user accounts, and deleting user accounts. This GUI form would also provide a way of retrieving the user requests from the request queue to enter them into the database as well as sending the command to create the accounts on the operating system.

# System and Software Design

The system was divided into two subsystems: the GUI form and the web form. The GUI form was divided into two modules. The first module relates to the system administrators' scripts and the second one involves the database design. The first module has four units: creating a user account, creating class accounts, expiring user accounts, and deleting user accounts. The other module has three units: table design, the form in which the user can insert, update, delete and query database records, and the expiration script. The second subsystem has three modules: user identification, user account requests, and class account requests. The breakdown of the system is shown in Figure 1.

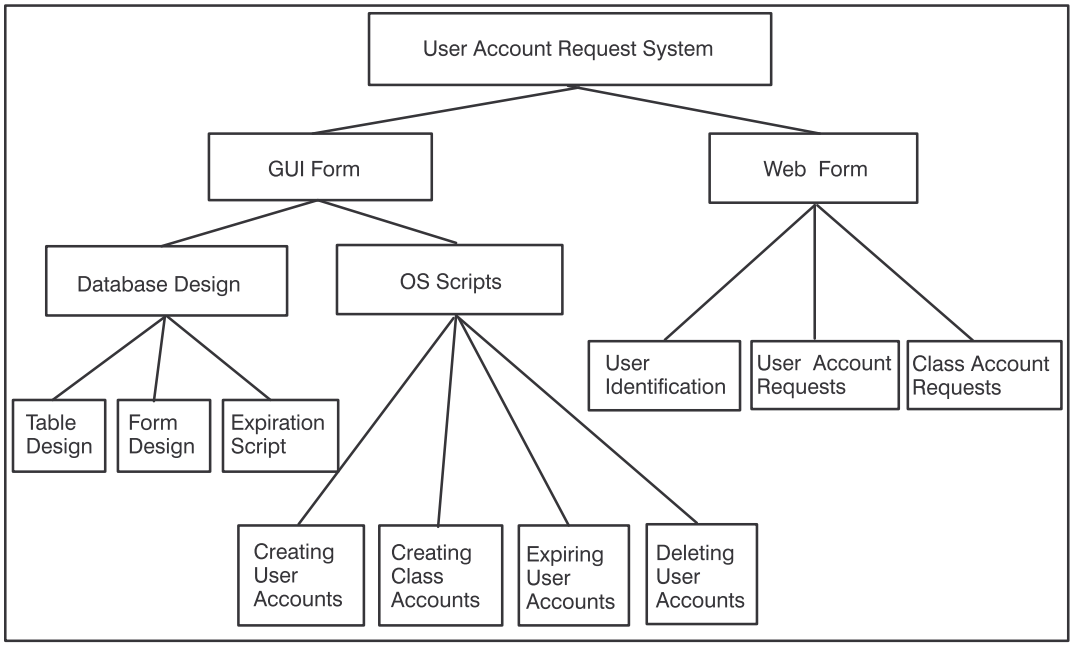


Figure 1 User Account Request System Architecture

# CHAPTER III

# Implementation and Unit Testing

The third stage of the waterfall model is implementation and unit testing. The goal of this stage is transform the design into viable units. Typically the design is transformed into code, but in this system it is a combination of code and form elements. Each unit is verified by either a static or dynamic method. Static methods include code reviews, code readings, and visual inspections. Dynamic testing methods involve code execution.

## Implementation

The order of implementation was driven by the nature of the development systems. The table was implemented before the GUI form, because the elements of the form correspond to the elements in the table. The expiration script executes from the form and uses information from the table so it was the last unit to be implemented in its module. The rest of the entire system is dependent on this module. The order of implementation for the rest of the system was arbitrary.

## GUI Form Implementation

Implementing the system began with the GUI Oracle form subsystem. A database table was created, entitled ADM_USERS, by incorporating the fields from the hard copy of the User Account Form plus a few additional fields

requested by the customer. Some of these fields have been removed from the system due to major networking changes in the ERC. In addition to the Oracle table, an Oracle sequence, ADM_USERS_SEQ, was created to populate the primary key of the table. A sequence is a database object that creates sequential numbers (Portfolio 1992). The naming convention used for the table and sequence follow what is used for all ERC database objects. The first part of the name is the department to which the object belongs. The second part describes what the object is. The sequences always have the same name as the table for which they were created with the addition of the SEQ ending. Once the table and sequence were created, the form was developed using Forms 4.0.

The style of the Oracle form was determined by customer preference and functionality. The color scheme was approved by the customer and is the color scheme of all ERC Oracle forms. The fields were grouped by their similarities and ordered by importance. The customer requested that when queried records display the active accounts before the deleted accounts. This was done by ordering by the deletion date in descending order.

There are many different item types that could be used for displaying the data fields. In this form, the most common type is text items. Text items are used for data sets that are virtually unlimited. Text items can be displayed as single or multi–line fields (Chu and Lim 1994). Lists of values (LOV's) can be used to protect data integrity.

The text fields in this form were divided into required fields, which are represented with a black text on a white background, and non–required fields, which are represented by black text on a light grey background. This was done in order to aid the user in determining which fields were necessary in order to be able to enter a new record. Two non–required fields, del_date and de-

leted_by, were given red text to help highlight the fact that the account had been deleted.

Another item type used in this form is the radio group. A radio group displays a list of values all at once to the user. The user is able to select only one of the possible choices. This type of data item is useful for representing extremely limited data sets (Chu and Lim 1994). The default value for the radio group in this form was selected by determining the most common value.

Check boxes are another item type used in the form. A check box is used for a binary value (e.g., true or false, yes or no, on or off, etc.). This type is generally used for data that has two values. A check box can be used to represent multiple value data sets, but the result is always mapped to one of two states (Chu and Lim 1994). Additional functionality was added to both the check boxes and the radio group used in this form. The selected items were additionally highlighted by changing the text color. This was done to enhance which values were selected. The enhancement was necessary in earlier versions of Oracle forms because the selected values were difficult to distinguish from the non–selected values.

The last item type used in this form to represent a database value is called a list item. A list item can be displayed as a poplist, a text list, or a combo box (Chu and Lim 1994). A poplist display was chosen to conserve space and limit the user to a predetermined set of possible values. A radio group could have been chosen instead, but space constraints made the list item the better choice.

This form frequently uses an item type called a button. Buttons perform functions and do not map directly to specific data fields. All buttons have an associated WHEN–BUTTON–PRESSED trigger that executes when the user

clicks on the button in the form. There are three different versions of the button item type in this form.

One version of a button is the small background colored buttons that are located next to some of the text fields. These buttons are called LOV buttons and are used specifically to spawn LOV's that are related to the text item by which they are located. An LOV is a popup window that displays a dynamic list of single or multi–column data from which the user can select value(s) that will be copied into the appropriate corresponding field(s). Where the values are copied is determined by the designer (Chu and Lim 1994). LOV's are used to help with data integrity since they can help prevent misspellings and typographic errors. LOV buttons are used when needed in all ERC forms.

Another version of a button in this form is the maroon colored button. These buttons were initially created from the default button palette provided by Oracle. They allow the user to do basic database functions (e.g., query, commit, clear, exit). These buttons are standard on all ERC forms, in order to minimize confusion for users who work in multiple forms.

The third version of buttons is the light blue button. These buttons are unique to each form. The functions they perform are designed specifically for the user account system. Some of these buttons are used as interfaces to the OS scripts that deal with user accounts.

Triggers are used to add functionality to the form. They encompass PL/SQL code that is written by the designer (Chu and Lim 1994). Oracle has many pre–defined triggers. The triggers used in this form are: PRE–INSERT, PRE–FORM, WHEN–RADIO–CHANGED, WHEN–CHECKBOX–CHECKED and WHEN–BUTTON–PRESSED. The PRE–INSERT trigger is used to populate the primary key field of the user data; whenever a new record is to be inserted, this trigger executes and the next unique value from the sequence is

entered into the primary key field. The PRE–FORM trigger executes when the form is first run and determines who the current user of the form is. This value is then used to populate the appropriate fields. The WHEN–CHECKBOX–CHANGED and WHEN–RADIO–CHANGED triggers are used to create the extra color highlights used with the check boxes and radio groups. The WHEN–BUTTON–PRESSED triggers are used for every button on the form and execute the appropriate action to take place once the button is pressed.

## Web Form Implementation

Using the web is ideal for the second subsystem of this project. It allows the web users (i.e., supervisors and professors) to request accounts from almost anywhere they have access to a web browser. Another added benefit is that the requests can be made any time of the day or night. This system was developed using Oracle Webserver 2.0. The purpose of this system is to replace the hard copy of the Account Request Form and its signature. In order to do this, the subsystem had to provide the following functions: web user identification, employee account requests, and class account requests. A second database table, ADM_USERS_QUEUE, was created to store the web user account requests. A corresponding sequence, ADM_USERS_QUEUE_SEQ, was also created.

Identifying and verifying the web user was the most important piece of the web system. If the web user could not be verified, the account requests would be invalid as would a hard copy without a supervisor's signature. To achieve this verification, the original thought was to store the web users' passwords in an Oracle table. It was determined that this was not a secure enough method and that the web users would consider this undesirable. Another possible idea

was to create Oracle user identifiers and passwords for every potential user of the web system. One problem with this method was that the database administrator did not relish the idea of creating all of these accounts for just this one purpose. Another problem was that the Oracle password is completely separate from the operating system password and because of this would cause too much user confusion.

The solution to this problem was to use the web user's OS login and password. OS calls made from a Pro*C program were used to verify the web user's identity. Pro*C code was used because it allowed the use of Oracle PL/SQL and Unix OS commands. Once the web user identity is determined, then their employee status is checked in the employee database. If their status is non–student, then they are allowed access. If access is granted, then their login name, the time of the identification, and a unique identifier (ID) are stored in an Oracle table, ADM_REQACCT_ACCESS. The unique ID is determined by the sequence ADM_REQACCT_ACCESS_SEQ. This information is stored in the database so that the identity of the user can be checked on every web page of the system. The time is needed because the web user is timed out after twenty minutes and would need to log back into the system to continue. The time–out is an extra security precaution.

The rest of the web system was developed in Oracle PL/SQL. An Oracle package, called User_Account, was created that contains all of the procedures and functions used in the web system. All of the package routines, with the exception of the routines that create help screens, call the security function that verifies the user's identity and reference time. The Oracle Webserver works by connecting to the Oracle RDBMS and executing stored procedures. These procedures have been written in a combination of standard PL/SQL and new

web routines. The new web routines are a series of built–in Oracle functions in PL/SQL that return HTML code.

The part of the web system that takes care of the user account requests is made up of three routines. The first routine displays a form that is based on the hard copy of the User Account Form. A few changes were needed after the ERC upgraded their network. A second routine displays a help system that defines the fields in the form and the specific restrictions for some of the fields. The final routine performs error checking on the data requested in the first routine. If there are errors, this routine displays them in a list to the web user; otherwise, the web user is notified that the request has been submitted into the account request queue.

The part of the web system that handles the class user accounts is more complicated. There are two separate routines that display forms to the user requesting information, and two matching routines that check the data for errors, display error messages if needed, and submit the data if allowed. The first set of routines ask for general data that is the same for the entire class. The second set of routines asks for the list of students. This list consists of the student name and their user login ID separated by a semi–colon. Each student record is also divided by a line return. Using the list saves the web user time; they would have otherwise had to submit after typing each student's information. The list is separated into the appropriate fields and records and is checked for errors. An error list is provided to the web user if needed. The web user is also given a list of all the approved records. Two more routines allow the web user to either update or delete students or add new students to the class. There are also a variety of routines that create help pages for this part of the system.

Two other features are part of the web system. One is a list of all accounts for which the supervisor is in charge. This is to aid the supervisor in keeping track of which accounts for which he/she is responsible. The other feature is the request queue. The queue gives the web user the ability to see how many requests are in the queue. The queue also gives the web user the opportunity to edit or delete any of his/her current requests.

## Unit Testing

In this phase of development, the units are tested separately. Each unit is tested once its design has been implemented. Generally, as in this case, the person coding the design tests the code. The purpose of testing at this level is to find coding mistakes, but design errors can still be found.

For the coded units, dynamic or functional testing was used to find defects. Functional testing is commonly referred to as black box testing. The inner workings are not specifically tested. Only the input and the result to that input are important (Beizer 1984). The type of functional testing used was equivalence partitioning. Equivalence partitioning is done by dividing input values into acceptable and unacceptable partitions. Values are then tested from each partition. The values chosen represent all the values from their equivalence partition.

For the form components, visual inspection was used for verification. Inspection was performed by both the progammer and the customer. Visual inspection involves evaluating the following aspects: arrangement efficiency, visual appeal, and appropriateness of the components.

# CHAPTER IV

## Integration and System Testing

System testing is an important part of the software development life–cycle. System testing serves two purposes: verifying the system requirements and detecting errors in the system (Sommerville 1992). Bottom–up is the strategy used in testing this system. Bottom–up testing involves testing the lower units first and then going to the next level of the hierarchy. In this system, the units were tested, followed by the modules, followed by the subsystems.

Once the individual units have passed testing, they can be integrated into the system. After the units have been connected, they are tested again. In this project, the new set of tests is a compilation of all the tests done on the units that have been integrated thus far. The units are integrated and tested until both subsystems are complete. The subsystems are not tested as a whole. Under the current design, the interaction between the two subsystems is minor and changes made in one subsystem do not affect the other. The only interaction between the two subsystems is through their database tables. They rely on each other to maintain data integrity in these tables.

Before the system was delivered to the customer, all of the requirements were verified. Functional testing was used in verifying the requirements. Upon delivery the customer validated that the system performed all the functions requested.

Due to post–delivery changes made to the system, final testing was performed multiple times. Regression testing is simply redoing the test cases after any changes have been made to the system (Jalote 1991). If regression testing is not performed, then there is a high likelihood that errors will get to the user. Even small changes in the design or implementation of a system can generate new problems. Regression testing can catch these problems, but test cases need to be well documented in order for retesting to be feasible. This type of testing is very important in the last stage of system development, operation and maintenance.

# CHAPTER IV

## Conclusions and Future Work

The User Account Request System is being used by the ERC System Administration as well as the ERC employees. This system has drastically reduced the amount of time required to create and maintain user accounts. It has also lessened the amount of time, effort, and the number of mistakes made in requesting user and class accounts.

There is one specific part of this project that I did not foresee. I wish I had paid more attention to the year 2000 issue. The problem could have been easily handled if it had been a consideration during the design process. By neglecting the issue, the maintainence needed to fix the problem took more work and time than it should have. Being aware of the year 2000 problem could have also improved the overall Oracle development at the ERC.

This system can be further improved as newer versions of Oracle RDBMS and Oracle products become available. One area that can definitely be improved is the web user identification system. In future versions of Webserver, the Pro*C code can be removed and replaced by a Java interface. This would improve the effeciency of any changes that might be needed in this system in the future as well as being more user friendly.

# REFERENCES


Beizer, Boris. 1984. *Software system testing and quality assurance.* New York, NY: Van Nostrend Reinhold Company Inc.

Chu, Ken, and Gina Lim. 1994. *Forms 4.5 developer's guide manual.* Redwood City, CA: Oracle Corporation.

Jalote, Pankaj. 1991. *An integrated approach to software engineering.* New York, NY: Springer–Verlag.

Portfolio, Tom. 1992. *PL/SQL User's guide and reference version 2.0.* Redwood City, CA: Oracle Corporation.

Sommerville, Ian. 1992. *Software engineering.* Reading, PA: Addison–Wesley.

# APPENDIX A

# Project Contract

# Engineering Research Center
# User Account Management System
## Project Contract

At the Engineering Research Center (ERC), most of the research is done using a Unix–based multi–user computer system. Individual computer accounts are required for computer access. In addition to ERC employees, a number of Mississippi State University classes use the ERC's computing resources. The Systems Administration at the ERC is responsible for creating, maintaining, and deleting all the ERC user accounts. These responsibilities take a lot of the administrators' time.

Under the current system, the new user or her supervisor must obtain a User Account Form from the Systems Administrator's office. Once the form is completed and submitted, account creation can take anywhere from an hour to a few weeks. The process of creating the accounts is very repetitive and tedious, since for each account the exact same steps are required. This is true for both employee accounts and class accounts.

My task is to create a system that requires minimal effort and time to use for both the requester and administrator. A database system is appropriate for the type of recordkeeping and data processing required. Therefore, the first step is to create a database system to keep track of all user account information. The second step is to replace the hardcopy version of the User Account Form with a web–based request system. The last step is to create ways of simplifying the creation, locking, and deletion of user accounts from within the database form.

Due to availability constraints, the system will be developed using Oracle tools. The system must run on Sparc Stations using a Solaris operating system and on Silicon Graphic machines using an IRIX operating system.

The project deliverables are as follows:

- Software Requirements Specification & Design Document
- Oracle Form & PL/SQL code for Web interface
- User Manual
- Test Plan & Test Results


Dr. Julia E. Hodges, Professor
Department of Computer Science, MSU
Project Director and Major Professor

Dr. Donna S. Reese, Associate Professor
Department of Computer Science, MSU

Dr. Bradley D. Carter, Professor
Department of Computer Science, MSU

Tanya George
Masters Student (CS), MSU

# APPENDIX B

# List of Deliverables

- Software Requirements Specification & Design Document
- Oracle Form & PL/SQL code for Web interface
- User Manual
- Test Plan & Test Results

# APPENDIX C

# User Account Form