DESIGN AND DEVELOPMENT OF AN ONLINE DATABASE SYSTEM FOR BANKS USING MYSQL AND PHP

BY

ANJO RUTH

PGD/MCS/2007/1221

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHS / COMPUTER SCIENCE, FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE POST-GRADUATE DIPLOMA IN COMPUTER SCIENCE

APRIL 2009

CERTIFICATION

This is to certify that this research work "**DESIGN AND DEVELOPMENT OF AN ONLINE DATABASE SYSTEM FOR BANKS USING MYSQL AND PHP"** was carried out by **ANJO RUTH** with **REGNO PGD/MCS/20007/1221** of the Department Of Maths Computer, School Of Science And Science Education, Federal University of Technology Minna.

Project Supervisor Dr Victor O. Waziri	12/5/2009 Date
Head of Department Dr N. I. Akinwande	Date
Dean Postgraduate School Professor S. L. Lamai	Date
External Examiner	Date

DEDICATION

This project is dedicated to God Almighty the author and the finisher of my faith.

ACKNOWLEDGEMENT

I wish to express my greatest appreciation to the Almighty God who has been so good to me through out my life.

I express my thanks to my project supervisor Dr Victor Waziri for his patience, support and understanding during this research work. To Dr. Yahaya Abdullahi and his assistant Mr. A. Ndanusa my departmental coordinator.

And also, my sincere gratitude, to the HOD Maths/ Computer Science Dr N. I Akinwande.

Not forgetting my bosses Dr A. Ochai Librarian, University of Jos, Dr S. Akintunde Deputy University Librarian, University of Jos and my colleagues at the University of Jos Library.

I must also not fail to express my gratitude to my family, my brother, sister, aunty, mother and most especially my father Mr Joshua Oreoluwaketun Anjo.

Thank you all.

ABSTRACT

This study shows how databases can be managed, and allow users, to store and retrieve data in a structured way and also allow them to be integrated with the web. PHP and MySQL are quickly becoming the de facto standard for rapid development of dynamic, database-driven web sites. This project is developed to bring databases and the web together. PHP has many excellent libraries that provide fast, customized access to DBMSs and is an ideal tool for developing application logic. However, the important component for web database application development is the applications interface that is used to access the database server. Its major advantage is that it permits multiple users to access a database at the same time in a methodical way. An additional benefit of this project is it ability to manage large amounts of related information, and its good search structures. It is recommended that banks should make more use of this project because of its large database storage.

TABLE OF CONTENTS

Title Page		i
Certification		ii
Dedication		iii
Acknowledgement		iv
Abstract		v
Table of Contents		vi
List of Figure		ix
Chapter One		1
1.0	Introduction	1
1.1	Background of the Study	1
1.2	Statement of the Problem	2
1.3	Objectives of the Study	2
1.4	Significance of the Study	3
1.5	Methodology	3
1.6	Scope of the Study	4
1.7	Definition of Terms	4
Chapter Two		6
2.0	Introduction	6
2.1	History of MySQL, PHP and Online Banking	7
	2.1.1 History of MySQL	7
	2.1.2 History of PHP	9
	2.1.3 History of Online Banking	11
2.2	Citations and Concept Definition	13

2.3 Advantages of MySQL and PHP 15

2.3.1 Features of PHP 17

2.3.2 Features of Online Banking 17

2.4 Relationship between MySQL and PHP 19

2.5 Examples of Banking Systems or Software already in Existence. 20

Chapter Three 22

3.0 Introduction 22

3.1 Problem Identification 23

3.2 MySQL And PHP 23

3.2.1 MySQL Compatibility 24

3.2.2 PHP Compatibility 25

3.3 Characteristics of MySQL and PHP based System 26

3.4 Advantages of MySQL and PHP 28

3.4.1 Cost 28

3.4.2 Ease of Use 28

3.4.3 Cross-platform compatibility 30

3.4.4 Not tag-based 30

3.4.5 PHP: Stability 31

3.4.6 Speed 31

3.4.7 Open source licensing 32

3.4.8 Communication 33

3.4.9 Fast Feature Development 34

3.4.10 Popularity 35

3.5 MySQL Versions 36

3.6 PHP versions 37

3.7 Defining the Banking System 40

Chapter Four (APPLICATION OF
PHP AND MYSQL TO BANKING) 42
4.0 Introduction 42
4.1 Applications of MYSQL and PHP in banking. 42
4.2 Phases of Creating a
MySQL and PHP Application 43
4.2.1 Web Design (Basic Architecture) 43
4.2.2 System Requirements 46
4.3 Online Banking System Overview 47
4.3.1 Application Module 47
4.4 Functional Overview 48
Chapter Five 56
5.0 Summary 56
5.1 Recommendation 56
5.2 Conclusion 57
Reference 58
Appendix A 59
Appendix B 60
Appendix C 67

LIST OF FIGURE

Figure 3.1	24
Figure 3.2	41
Figure 3.3	41
Figure 4.1	43
Figure 4.2	45
Figure 4.3	48
Figure 4.4	50
Figure 4.5	51
Figure 4.6	51
Figure 4.7	52
Figure 4.8	52
Figure 4.9	53
Figure 4.10	53
Figure 4.11	54
Figure 4.12	54
Figure 4.13	55
Figure 4.14	55

CHAPTER ONE

1.0 Introduction

MySQL and PHP has become one of the most practical and widely used applications. They allow programmers to make sites dynamic, that is to make them customizable and for them to contain real-time information. Thus, this project provides an online database resource for banks.

The aim of this project is to allow users to create, edit, transfer and check their accounts online without going to the banks.

Tasks which would have taken hours because of queues in some banks and ATM will be done with speed and accuracy with this database.

1.1 Background of the Study

MySQL is a very fast, robust, *relational database management system (RDBMS)*.A database enables you to efficiently store, search, sort, and retrieve data. The MySQL server controls access to your data to ensure that multiple users can work with it concurrently, to provide fast access to it, and ensure that only authorized users can obtain access. Hence, MySQL is a multi-user, multi-threaded server. It uses *SQL (Structured Query Language)*, the standard database query language worldwide. MySQL has been publicly available since 1996, but has a development history going back to 1979. It has now won the *Linux Journal* Readers' Choice Award on a number of occasions.

PHP is a server-side based scripting language designed specifically for the Web. PHP was conceived in 1994 and was originally the work of one man,

Rasmus Lerdorf. It was adopted by other talented people and has gone through three major introduction rewrites to bring us the broad, matured product we see today. As of October 2002, it was in use on more than nine million domains worldwide, and this number is growing rapidly by the day.

The term online became popular in the late '80s and referred to the use of a terminal, keyboard and TV (or monitor) to access the banking system using a phone line. 'Home banking' can also refer to the use of a numeric keypad to send tones down a phone line with instructions to the bank. Online services started in New York in 1981 when four of the city's major banks (Citibank, Chase Manhattan, Chemical and Manufacturers Hanover) offered home banking services using the videotext system. Stanford Federal Credit Union was the first financial institution to offer online internet banking services to all of its members in Oct, 1994.

1.2 Statement of the Problem

It has been noted that the application of online database system in banks has been slow because of some security problems. And also, if you want to open an account you have to go to the bank and as a result slow down work and also make transfer and enquiry difficult. As a result, this online database system was designed to unite all banks and it makes general banking transaction easier.

1.3 Objectives of the Study

The objectives of the study are:

i. To design an online database for banks that will provide complementary assistance to bank customers without first going to a bank for transactions.

ii. To design a software package that will insure timely processing and retrieving of information.

iii. To provide a system that is very effective, efficient, secured and reliable.

iv. To provide a system that will allow access to creation, editing, transfer or online transaction from any bank.

1.4 Significance of the Study

It is pertinent to note that this project work will throw more light on the use of MySQL and PHP in developing a real-time banking database and it would serve as bedrock for information analysis on online banking.

There the significance of the study includes the following:

i. It would help to determine how the use of MySQL and PHP can be use to ease the banking routine.

ii. It would enable users to log in and open new accounts.

iii. It shall help to preserve and authenticate the banking databases.

1.5 Methodology

Research methodology and investigation states the various methods employed during data collection for the study and the steps taken to design the online banking system. They include the following:

a. Web design

b. Design of files and Databases

c. Design of system user requirements.

d. System Specification.

e. Determine precisely what output will be required for the new system.

1.6 Scope of the Study

The scope of this project has been limited to all commercial Banks. Some of the areas that will be handled by this package include:

a. Name of all the commercial banks

b. Logging in and authenticating users

c. Managing passwords

d. Creation of account (Current or Savings) depending on the banks

e. Editing of customers data

f. Funds transfer.

1.7 Definition of Terms

ASP: stands for active server page.

Computer: is an electronic data processor. An electronic device that accepts, processes, stores, and outputs data at high speeds according to programmed instructions.

Database: is a collection of data on computer. It is also a systematically arranged collection of computer data, structured so that it can be automatically retrieved or manipulated.

DBMS: is an acronym for Database Management System**.** It is a computer program devised to create, store, and manipulate databases.

HTML: it is the markup language used for creating documents on the World Wide Web. It is an acronym for HyperText Markup Language.

HTTP: is the client/server protocol that defines how messages are formatted and transmitted on the World Wide Web. *Full form* HyperText Transfer Protocol.

OOP: is an acronym for object-oriented programming (OOP). OOP promotes clean modular design, simplifies debugging and maintenance, and assists with code reuse.

MySQL: is a database server that manages databases. It is a relational database system that is used to store information. It can store many types of data from something as tiny as a single character to as large as complete files or graphics.

PHP: it is a recursive acronym that stands for Hypertext Preprocessor. PHP is a scripting language that is usually embedded or combines with the HTML of a web page.

RDBMS: stands for Relational Database Management System.

SQL: stands for Structured Query Language. It is a computer language or a standardized language that approximates the structure of natural English for obtaining information from databases.

Conclusion

In conclusion, the main objective of the study is to provide assistance to bank customers or users without going to the bank for transaction. This chapter explains the background of the study, the statement of the problem, the objectives, significance and the scope of the study. It also explains the method used in the design and development.

CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

Design and development of an Online Database System for Banks is a project developed with PHP and MySQL to aid online banking. It is an application of artificial intelligence of a high security standard. It will also perform most of the banking operations in real time.

Hugh E. Williams and David Lane (2002) in their book "Building Effective Database – Driven Websites, Web Database Application with PHP and MySQL" Second Edition, Describes PHP as a scripting language that is usually embedded or combined with HTML of a webpage. PHP has many excellent libraries that provide fast customized access to Database Management System (DBMS) and is an ideal tool for developing application.

In addition, Hugh and David (2002) described MySQL as a database server that manages database and supports a database language to create and delete databases and to manage search data, and it permits multiple user to manage complex relational data and it also permit multiple user to access a database at the same time in a methodical way. It is data – oriented, user oriented security etc.

PHP and MySQL are easy low cost way of bringing together the web and databases to build applications. PHP is particularly suited to web database applications because of its integration tools for the web and database environment.

The applications that will be used to design the online banking database systems are PHP, MySQL and Apache. Also, Jay Greenspan and Brad Bulger (2002) in their book "MySQL/PHP Database Applications" describes PHP and MySQL as fast, easy, cross- platform (that is works as well on a wide variety of systems), it accesses everything because it is constantly being improved.

According to the Microsoft Encarta 2008, a bank is any financial institution that receives, collects, transfers, pays, exchanges, lends, invests, or safeguards money for its customers. The basic services a bank provides are checking accounts, which can be used like money to make payments and purchase goods and services; savings accounts and time deposits that can be used to save money for future use; loans that consumers and businesses can use to purchase goods and services; and basic cash management services such as check cashing and foreign currency exchange. Four types of banks specialize in offering these basic banking services: commercial banks, savings and loan associations, savings banks, and credit unions.

2.1 History of MySQL, PHP and Online Banking

2.1.1 History of MySQL

MySQL, pronounced either "My Ess Que El" or "My Sequel," is an open source relational database management system. It is based on the structure query language (SQL), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD,

DROP, INSERT, and UPDATE can be used with MySQL.

PHP was originally created by Rasmus Lerdorf, and he oversaw production of the first release and PHP/FI 2.0. PHP 3 was rewritten from the ground up by Zeev Suraski and Andi Gutmans, and the three of them are the "language architects" behind PHP.

In the PHP group, the developers who primarily work on development of the language core, there are: Thies C. Arntzen, Stig Bakken, Shane Caraveo, Andi Gutmans, Rasmus Lerdorf, Sam Ruby, Sascha Schumann, Zeev Suraski, Jim Winstead, and Andrei Zmievski. Each of them has worked exceptionally hard, along with the help of many others, to put PHP where it is today.

MySQL is named after co-founder Monty Widenius's daughter, My. The name of the MySQL Dolphin (our logo) is “Sakila,” which was chosen by the founders of MySQL AB from a huge list of names suggested by users in our “Name the Dolphin” contest. The winning name was submitted by Ambrose Twebaze, an Open Source software developer from Swaziland, Africa. According to Ambrose, the feminine name Sakila has its roots in SiSwati, the local language of Swaziland. Sakila is also the name of a town in Arusha, Tanzania, near Ambrose's country of origin, Uganda.

MySQL was developed because of the need of a database system that was extremely fast and flexible. Unfortunately (or fortunately, depending on your point of view), they could not find anything on the market that could do what they wanted. So, they created MySQL, which is loosely based on another database management system called mSQL. The product they

created is fast, reliable, and extremely flexible. It is used in many places throughout the world. Universities, Internet service providers and nonprofit organizations are the main users of MySQL, mainly because of its price (it is mostly free). Lately, however, it has begun to permeate the business world as a reliable and fast database system.

MySQL was developed in 1996. They created it because they needed a relational database that could handle large amounts of data on relatively cheap hardware. Nothing out there could provide what they needed, so they created it themselves.

MySQL is the fastest relational database on the market. It outperforms all the leading databases in almost every category. It has almost all the functionality the leading databases have, but it does not carry the hefty price tag.

MySQL can be used for a variety of applications, but is most commonly found on Web servers. A website that uses MySQL may include Web pages that access information from a database. These pages are often referred to as "dynamic," meaning the content of each page is generated from a database as the page loads. Websites that use dynamic Web pages are often referred to as database-driven websites.

2.1.2 History of PHP

PHP originally stood for Personal Home Page. PHP, which now stands for " Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be

embedded into HTML. It began in 1994 as a set of Common Gateway Interface binaries written in the C programming language by the Danish/Greenlandic programmer Rasmus Lerdorf. Lerdorf initially created these Personal Home Page Tools to replace a small set of Perl scripts he had been using to maintain his personal homepage. The tools were used to perform tasks such as displaying his résumé and recording how much traffic his page was receiving. He combined these binaries with his Form Interpreter to create PHP, which had more functionality. PHP included a larger implementation for the C programming language and could communicate with databases, enabling the building of simple, dynamic web applications. Lerdorf released PHP publicly on June 8, 1995 to accelerate bug location and improve the code. This release was named PHP version 2 and already had the basic functionality that PHP has today. This included Perl-like variables, form handling, and the ability to embed HTML. The syntax was similar to Perl but was more limited, simpler, and less consistent.

Zeev Suraski and Andi Gutmans, two Israeli developers at the Technion IIT, rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive initialism *PHP: Hypertext Preprocessor*.The development team officially released PHP 2 in November 1997 after months of beta testing. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. Suraski and Gutmans then started a new rewrite of PHP's core, producing the Zend Engine in 1999. They also founded Zend Technologies in Ramat Gan, Israel.

On May 22, 2000, PHP 4, powered by the Zend Engine 1.0, was released. On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II. PHP 5 included new features such as improved support for object-oriented

programming, the PHP Data Objects extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements.[9] The most recent update released by The PHP Group is for the older PHP version 4 code branch. As of August, 2008 this branch is up to version 4.4.9. PHP 4 is no longer under development nor will any security updates be released.

In 2008, PHP 5 became the only stable version under development. Late static binding has been missing from PHP and will be added in version 5.3.[12][13] PHP 6 is under development alongside PHP 5. Major changes include the removal of `register_globals`, magic quotes, and safe mode.

PHP does not have complete native support for Unicode or multibyte strings; Unicode support will be included in PHP 6. Many high profile open source projects ceased to support PHP 4 in new code as of February 5, 2008, due to the GoPHP5 initiative, provided by a consortium of PHP developers promoting the transition from PHP 4 to PHP 5.

It runs in both 32-bit and 64-bit environments, but on Windows the only official distribution is 32-bit, requiring Windows 32-bit compatibility mode to be enabled while using IIS in a 64-bit Windows environment. There is a third-party distribution available for 64-bit Windows.

2.1.3 History of Online Banking

According to Bainbridge Ross, "History of Online Banking." 16 Aug. 2006. *EzineArticles.com*. 22 Jan 2009 <http://ezinearticles.com/?History-of-Online-Banking&id=270075>, Online banking (or Internet banking) allows customers to conduct financial transactions on a secure website operated by

their retail or virtual bank, credit union or building society. The concept of online banking as we know it today dates back to the early 1980s, when it was first envisioned and experimented with. However, it was only in 1995 (on October 6, to be exact) that Presidential Savings Bank first announced the facility for regular client use. The idea was quickly snapped up by other banks like Wells Fargo, Chase Manhattan and Security First Network Bank. Today, quite a few banks operate solely via the Internet and have no 'four-walls' entity at all.

In the beginning, its inventors had predicted that it would be only a matter of time before online banking completely replaced the conventional kind. Over the past few years the nature of banking has undergone major changes with the widespread use of new media in lieu of traditional high street banking. This trend began in 1989 with the introduction of First Direct, the first telephone-only bank account. The bank subsequently launched an internet 'branch' in 1999 and WAP is on the way. Bainbridge, R. (2006, August 16). *History of Online Banking*. Retrieved January 22, 2009, from http://ezinearticles.com/?History-of-Online-Banking&id=270075

Most of the internet facilities allow customers to pay bills, set up standing orders and direct debits as well as check account statements. The transferal of money to other accounts by using BACS, CHAPS and SWIFT is also available depending on the bank. These websites may also have useful information on other finance issues such as loans, insurance and mortgages. Some also provide tailored products such as the Natwest Bankline cash

manager, which is a software package used to monitor cash flow or Barclay's is one of the few banks that offers online euro services.

2.2 Citations and Concept Definition

MySQL is a database server that manages databases. It supports a database language to create and delete databases and to manage and search data. The MySQL command interpreter is commonly used to create databases and tables in web database applications and to test queries.

Many database-driven websites that use MySQL also use a Web scripting language like PHP to access information from the database. MySQL commands can be incorporated into the PHP code, allowing part or all of a Web page to be generated from database information. Because both MySQL and PHP are both open source (meaning they are free to download and use), the PHP/MySQL combination has become a popular choice for database-driven websites.

MySQL is a relational database system that is used to store information. MySQL can store many types of data from something as tiny as a single character to as large as complete files or graphics. Although it can be accessed by most programming languages, it is often coupled with PHP because they work together with ease.

Information stored in a MySQL database hosted on a web server can be accessed from anywhere in the world with a computer. This makes it a good

way to store information that needs the ability to change over time, but also needs to be accessed over the net. Some examples that can utilize MySQL are a web message board or a customer's shipping status.

PHP is a scripting language originally designed for producing dynamic web pages. It has evolved to include a command line interface capability and can be used in standalone graphical applications.

While PHP was originally created by Rasmus Lerdorf in 1995, the main implementation of PHP is now produced by The PHP Group and serves as the *de facto* standard for PHP as there is no formal specification.[3] PHP is free software released under the PHP License, however it is incompatible with the GPL due to restrictions on the usage of the term *PHP*.

PHP is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge. PHP is installed on more than 20 million websites and 1 million web servers.

PHP is an open source project and is particularly suited to web database applications such as MySQL because of its integration tools for the web and database environments.

Therefore, PHP is a recursive acronym that stands for Hypertext Preprocessor. PHP is a scripting language that is usually embedded or combines with the HTML of a web page. PHP has many excellent libraries

that provide fast, customized access to access to DBMSs and is an ideal tool for developing application logic.

This project will help in tracking allowing customers to track their bank balances. And they add value to risks and threats the existing services offered by the bank

2.3 Advantages of MySQL and PHP

The advantages of MySQL and PHP are:

i. **Performance**

MySQL is undeniably fast. You can see the developers' benchmark page at the mysql.com Web site. Many of these benchmarks show MySQL to be orders of magnitude faster than the competition.

PHP is very efficient. Using a single inexpensive server, you can serve millions of hits per day. Benchmarks published by Zend Technologies (http://www.zend.com) show PHP out performing its competition.

ii. **Database Integration**

PHP has native connections available to many database systems. In addition to MySQL, you can directly connect to PostgreSQL, mSQL, Oracle, dbm, filePro, Hyperwave, Informix, InterBase, and Sybase databases, among others. Using the *Open Database Connectivity Standard (ODBC)*, you can connect to any database that provides an ODBC driver. This includes Microsoft products, and many others.

iii. **Built-in Libraries**

Because PHP was designed for use on the Web, it has many built-in functions for performing many useful Web-related tasks. You can generate GIF images on-the-fly, connect to other network services, send email, work with cookies, and generate PDF documents, all with just a few lines of code.

iv. **Cost**

MySQL is available at no cost, under an Open Source license, or at low cost under a commercial license if required for your application.

PHP is free. You can download the latest version at any time from http://www.php.net for no charge.

v. **Learning PHP**

The syntax of PHP is based on other programming languages, primarily C and Perl. If you already know C or Perl, or a C-like language such as C++ or Java, you will be productive using PHP almost immediately.

vi. **Portability**

MySQL can be used on many different Unix systems as well as under Microsoft Windows.

PHP is available for many different operating systems. You can write PHP code on the free Unix-like operating systems such as Linux and FreeBSD, commercial Unix versions such as Solaris and IRIX, or on different versions of Microsoft Windows. Your code will usually work without modification on a different system running PHP.

vii. **Source Code**

You can obtain and modify the source code for MySQL.

You have access to the source code of PHP. Unlike commercial, closed-source products, if there is something you want modified or added to the language, you are free to do this. You do not need to wait for the manufacturer to release patches. You don't need to worry about the manufacturer going out of business or deciding to stop supporting a product.

viii. **Ease of Use**

Most modern databases use SQL. If you have used another RDBMS, you should have no trouble adapting to this one. MySQL is also easier to set up than many similar products.

2.3.1 Features Of PHP

The features of PHP are:

i. It is that it is available for Microsoft Windows, for many versions of UNIX, and with any fully functional Web server. MySQL is similarly versatile.
ii. It encourages encapsulation.
iii. PHP supports Object-oriented programming (OOP) which opens the door to cleaner designs, easier maintenance, and greater code reuse.

2.3.2 Features of Online Banking

Features commonly unique to online banking include:

i. Support of multiple users having varying levels of authority

ii. Transaction approval process.

iii. Wire transfer

Other features fall broadly into several categories:

a. **Transactional** (e.g., performing a financial transaction such as an account to account transfer, paying a bill, and apply for a loan, new account, etc.)

- Electronic bill presentment and payment - EBPP
- Funds transfer between a customer's own checking and savings accounts, or to another customer's account
- Investment purchase or sale
- Loan applications and transactions, such as repayments

b. **Non-transactional** (e.g., online statements, check links, co browsing, chat)

- Bank statements

c. **Financial Institution Administration** - features allowing the financial institution to manage the online experience of their end users

d. **ASP/Hosting Administration** - features allowing the hosting company to administer the solution across financial institutions.

2.4 Relationship between MySQL and PHP

PHP and MySQL compliment each other to do what neither can do alone. PHP can collect data, and MySQL can in turn store the information. PHP can create dynamic calculations, and MySQL can provide it with the variables it uses. PHP can create a shopping cart for your web store, but MySQL can then keep the data in a format PHP can use to create receipts on demand, show current order status, or even suggest other related products. The code used to connect PHP and MySQL is

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
# HTTP="true"
$hostname_details = "127.0.0.1";
$database_details = "bankdatabasesystem";
$username_details = "root";
$password_details = "12345678";
$details = mysql_pconnect($hostname_details, $username_details,
$password_details) or trigger_error(mysql_error(),E_USER_ERROR);
?>
```

Although PHP and MySQL can each be used independently, when you put them together it opens up countless possibilities for your site. As the internet progresses, it becomes more and more necessary to deliver dynamic content to keep up with the demands of web surfers and their desire to have information instantly delivered to them online. By learning to use PHP and MySQL you can deliver this information to them on demand.

2.5 Examples of Banking Systems or Software already in Existence.

Examples of banking software in existence include:

i. **FinnOne™:** is a product of Nucleus Software. It is a comprehensive suite for Retail Banking applications comprising of modules like Customer Acquisition System, Loan Management, Delinquency and Recovery Management, Deposits and Finance Against Securities.

ii. **Finacle:** by Infosys Technologies addresses the core banking & web based cash management requirements of retail, corporate and universal banks worldwide. It provides an insight on how banks can align their technological investments with their business objective to meet the numerous challenges that the industry now faces. This is a software used by most banks in Nigeria such as First bank, Spring bank etc.

iii. **SMART C.U.T:** is completely integrated banking system for managing financial products, services and related information.

iv. **CyberBank:** by Technisys is an enterprise-class family of solutions that enables financial institutions to rapidly deploy multi-channel financial services applications. CyberBank provides end-users with customized interactions, enabling institutions to forge a unique, enduring and profitable relationship.

v. **Corniche:** by **Megasol Technologies** Designed for the management of private banks, offshore banks, trust companies and funds, Corniche provides true multiple currency accounting with a uniform back-office

interface, merchant and card services, payment interfaces and powerful Corniche on-line banking facilities. Manage multiple banks,

vi. **eDominate Suite:** by iNet and provides online lending software; loan origination systems for mortgage, commercial, and consumer lending; a customer-specific cross-selling engine for automating product promotions; and an information management system monitoring all of these systems.

However, for the project so far, one can always abstract the advantages of designing a banking system with MySQL and PHP.

Conclusion

The design and development of an online database system for banking using PHP and MySQL is not the only system that can use PHP and MySQL there are others, for example:

- Shopping Cart Application
- Winestore System etc

In conclusion, the Online Database System is developed because of the need for banks to manage large amount of data anytime and anywhere.

CHAPTER THREE

3.0 Introduction

PHP and MySQL, individually or together, is a panacea for most Web development problem, and they present a lot of advantages. PHP is built by Web developers for Web development and supported by a large and enthusiastic community. MySQL is a powerful standard compliant RDBMS that comes in at an extremely competitive price point, even more so if you qualify for free use. Both technologies are clear-cut cases of the community banding together to address its own needs.

No matter what kind of system is being studied, there will always be one problem or the other. Development or an improvement on the existing system is always based on the problems identified.

Problems can be identified, both now and in the future, as evidence that objectives are not being achieved. However, objectives are often rather abstract, and it may be easier for members of the public to understand a strategy based on clearly identified problems. This problem-oriented approach to strategy formulation is an alternative to starting with objectives, but does still need to be checked against the full list of objectives.

In this chapter, the problems of the existing system will be highlighted. Note the system to be created will not replace the existing system but it will help it work more efficiently.

Also, the interaction between MySQL and PHP will be explained.

3.1 Problem Identification

Problems are identified based on the strengths, weaknesses, opportunities and threats encountered in the process of conducting day-to-day operation of the banks.

The problems identified during the course of the investigation on the Present system include: -

a. Time consuming. That is, the rate and time of queuing in bank.
b. Costly to maintain due to high consumption of stationeries, purchase of filing cabinets and more personnel are required.
c. It is very slow because of the manual labour that delays transaction process.
d. Individuals can not access their account information online.
e. There is great possibility for missing, misplacement and omission of documents.

3.2 MySQL And PHP

PHP is a server-side scripting language, which can be embedded in HTML or used as a standalone binary. Proprietary products in this niche are Microsoft's Active Server Pages, Macromedia's ColdFusion, and Sun's Java Server Pages. Some tech journalists used to call PHP "the open source ASP" because its functionality is similar to that of the Microsoft product—although this formulation was misleading, as PHP was developed before ASP. Over the past few years, however, PHP and server-side Java have

gained momentum, while ASP has lost mindshare, so this comparison no longer seems appropriate.

3.2.1 MySQL Compatibility

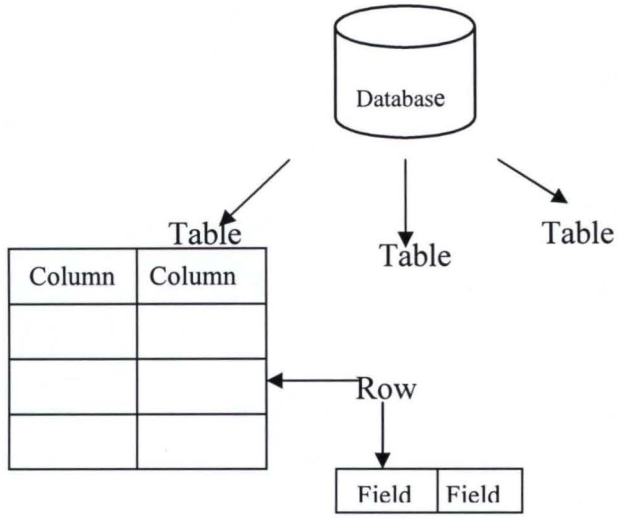

Figure 3.1 *The anatomy of a database*

Database contains a number of tables. Each table is made up of a series of columns. Data is stored in rows, and the place where each row intersects a column is known as a field. Figure 3.1 depicts this breakdown.

MySQL is more than just a database. It is a system that manages databases. It controls who can use them and how they are manipulated. It logs actions and runs continuously in the background. This is different from what you may be used to. Most people think about Microsoft Access or Lotus Approach when they think about databases. These are databases, but they are not management systems. A DBMS can contain many databases. Users connect to the database server and issue requests. The database server queries its databases and returns the requests to the issuers. Databases, such as Approach and Access, are a step down from this type of system. They

share their files with multiple users, but there is no interface controlling the connections or answering requests.

There are many uses for a DBMS such as MySQL. Uses can range from help desk systems to Web site applications. The important thing to remember is that MySQL is large enough and quick enough to function in almost any situation.

3.2.2 PHP Compatibility

PHP is strongly influenced by other programming languages, such as Perl and C. PHP supports object-oriented programming (OOP). OOP promotes clean modular design, simplifies debugging and maintenance, and assists with code reuse.

PHP is the engine behind millions of dynamic web applications. Its broad feature set, approachable syntax, and support for different operating systems and web servers have made it an ideal language for both rapid web development and the methodical construction of complex systems.

One of the major reasons for PHP's success as a web scripting language is its origins as a tool to process HTML forms and create web pages. This makes PHP very web-friendly. Additionally, it is a polyglot. PHP can speak to a multitude of databases, and it knows numerous Internet Protocols. PHP also makes it simple to parse browser data and make HTTP requests.

This Research is a collection of solutions to common tasks of banking with PHP. Using cookie or session authentication instead of HTTP Basic

authentication makes it much easier for users to log out: you just delete their login cookie or remove the login variable from their session.

Another advantage of storing authentication information in a session is that you can link users' browsing activities while logged in to their browsing activities before they log in or after they log out.

3.3 Characteristics of MySQL and PHP based System

i. The first characteristic of a MySQL and PHP based System is that it can be used by more than one person at a time. This is a requirement at any level of banking. More than one person may need to have access to their information at a given time. This is critical for banks to function successfully. MySQL and PHP meets this requirement. It can have up to 101 simultaneous connections. This doesn't mean that only 101 people can use this application. It means it can have 101 connections going on at the same time—which is a little different. A connection is the time it takes for a user to receive the data that he or she has requested. In the case of MySQL, this is hardly any time at all. Most database systems in the same class as MySQL allow fewer simultaneous connections. Currently, the only DBMS to offer more connections is Microsoft SQL Server.

ii. The next characteristic that a MySQL and PHP based System must have is security. When dealing with mission-critical information, only people with the need to know should be allowed to view it. Security keeps malicious people at bay; without it, disasters can happen. MySQL meets this requirement. The security in MySQL is

unparalleled. Access to a MySQL database can be determined from the remote machine that can control which user can view a table. The database can be locked down even further by having the operating system play a role in security as well. Very few databases in the same class as MySQL can compare to the level of security that MySQL provides.

iii. One other characteristic of a MySQL and PHP based System is flexibility. How flexible is the application? Can it change to meet the ever-changing needs of business? How deep can you make those changes? How hard is it to change? MySQL answers these questions very well. It is extremely flexible and easy to use. MySQL can run on almost any platform. If a new CIO wants to change from Windows NT to Linux, fine—MySQL can adapt. MySQL also comes with the source code. If there are any deep-level changes that you need to make, you can edit the source and make these changes yourself. If MySQL is missing a feature that you can't live without, just add it yourself. No other database on the market can offer you that kind of flexibility. MySQL also has several application-level interfaces in a variety of languages.

iv. In addition to the previously discussed characteristics, databases at the a MySQL and PHP based System must be able to work together. Data warehousing is a technique that combines all the data in a bank. Because of the flexibility and speed that MYSQL and PHP has to offer, they can work well in any situation.

v. Yet another feature of MySQL and PHP is its portability—it has been ported to almost every platform. This means that you don't have to change your main platform to take advantage of them. MySQL also has many different application programming interfaces (APIs). They include APIs for Perl, TCL, Python, C/C++, Java (JDBC), and ODBC.

3.4 Adantages of MySQL and PHP

3.4.1 Cost

PHP costs you nothing. MySQL is open-source licensed software. The freeness of open source and Free software is guaranteed by a gaggle of licensing schemes, most famously the *GPL* (*Gnu G*eneral *P*ublic *L*icense) or *copyleft*.

Usually, open source software users can freely choose the precisely optimal cost-benefit equation for each particular situation: no cost and no warranties, or expensive but well supported. No organized attempt has been made yet to sell service and support for PHP. MySQL does sell support as part some of its licensing packages for the MySQL product. Other open source products, such as Linux, have companies such as Red Hat standing by to answer your questions, but the commercialization process is still in the early stages for PHP.

3.4.2 Ease of Use

PHP is easy to learn, compared to the other ways to achieve similar functionality. Unlike Java Server Pages or C-based CGI, PHP doesn't require you to gain a deep understanding of a major programming language

before you can make a trivial database or remote-server call. Unlike Perl, which has been semi jokingly called a "write-only language," PHP has a syntax that is quite easy to parse and human-friendly. And unlike ASP.NET, PHP is stable and ready to solve your problems anytime. In fact, it's entirely possible to use PHP just by modifying freely available scripts rather than starting from scratch—you'll still need to understand the basic principles, but you can avoid many frustrating and time-consuming minor mistakes.

Note that *Easy* means different things to different people, and for some Web developers it has come to connote a graphical, drag-and-drop, What You See Is What You Get development environment.

HTML-embeddedness

PHP is embedded within HTML. In other words, PHP pages are ordinary HTML pages that escape into PHP mode only when necessary. This means it goes through the page from top to bottom, looking for sections of PHP, which it will try to resolve.

The HTML-embeddedness of PHP has many helpful consequences:

* PHP can quickly be added to code produced by WYSIWYG editors.
* PHP lends itself to a division of labor between designers and scripters.
* Every line of HTML does not need to be rewritten in a programming language.
* PHP can reduce labor costs and increase efficiency due to its shallow learning curve and ease of use.

Perhaps the sweetest thing of all about embedded scripting languages is that they don't need to be compiled into binary code before they can be tested or

used—just write and run. PHP is interpreted (as are many newish computer languages), although the Zend Engine does some behind-the-scenes precompiling into an intermediate form for greater speed with complex scripts.

3.4.3 Cross-platform compatibility

PHP and MySQL run native on every popular flavor of Unix (including Mac OS X) and Windows. A huge percentage of the world's HTTP servers run on one of these two classes of operating systems.

PHP is compatible with the three leading Web servers: Apache HTTP Server for Unix and Windows, Microsoft Internet Information Server, and Netscape Enterprise Server (a.k.a. iPlanet Server). It also works with several lesser-known servers, including Alex Belits' fhttpd, Microsoft's Personal Web Server, AOLServer, and Omnicentrix's Omniserver application server. Specific Web-server compatibility with MySQL is not required, since PHP will handle all the dirty work for you.

3.4.4 Not tag-based

PHP is a real programming language. ColdFusion, by contrast, is a bunch of predefined tags, like HTML. In PHP, you can define functions to your heart's content just by typing a name and a definition. In ColdFusion, you have to use tags developed by other people or go through the Custom Tag Extension development process.

As a witty PHP community member once said, "ColdFusion makes easy things easy, and medium-hard things impossible." And as every programmer

will agree, once you experience the power of curly brackets and loops, you never go back to tags.

3.4.5 PHP: Stability

The word *stable* means two different things in this context:

1. The server doesn't need to be rebooted often.
2. The software doesn't change radically and incompatibly from release to release.

To our advantage, both of these connotations apply to both MySQL and PHP. Apache Server is generally considered the most stable of major Web servers, with a reputation for enviable uptime percentages. Although it is not the fastest nor the easiest to administer, once you get it set up, Apache HTTP Server seemingly never crashes. It also doesn't require server reboots every time a setting is changed (at least on the Unix side). PHP inherits this reliability; plus, its own implementation is solid yet lightweight. In a two-and-a-half-month head-to-head test conducted by the Network Computing labs in October 1999, Apache Server with PHP handily beat both IIS/Visual Studio and Netscape Enterprise Server/Java for stability of environment.
PHP and MySQL are also both stable in the sense of feature stability.

3.4.6 Speed

PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.

PHP5 is much faster for almost every use than CGI scripts. The time and resources necessary for this handoff and spawning are considerable, and there can be limits to the number of concurrent processes that can be running at any one time. Other CGI scripting languages such as Perl and Tcl can be quite slow. Most Web sites have moved away from use of CGI for performance and security reasons.

Although it takes a slight performance hit by being interpreted rather than compiled, this is far outweighed by the benefits PHP derives from its status as a Web server module. When compiled this way, PHP becomes part of the http daemon itself. Because there is no transfer to and from a separate application server (as there is with ColdFusion, for instance) requests can be filled with maximum efficiency.

3.4.7 Open source licensing

We've already dealt with the cost advantages of open source software in the "Cost" section of this chapter. The other major consequence of these licenses is that the complete source code for the software must be included in any distribution. In fact, the Unix version of PHP is released *only* as source code; so far, the development team has staunchly resisted countless pleas to distribute official binaries for any of the Unixes. At first, new users (particularly those also new to Unix) tend to feel that source code is about as useful as a third leg, and most vastly prefer a nice convenient rpm.

The most immediate pragmatic advantage is that you can compile your PHP installation with only the stuff you really need for any given situation. This

approach has performance and security advantages. For instance, you can put in hooks to the database(s) of your choice.

For all their *openness*, the licenses for MySQL and PHP are quite different. They have many similarities to be sure but also some radically different provisions, especially when it comes to when you should pay. Genuinely open source software like PHP cannot seek to limit the purposes for which it is used, the people allowed to use it, or a host of other factors. The most critical of these rights is the one allowing users to make and distribute any modifications along with the original software.

Users new to the open source model should be aware that this right is also enjoyed by the developers. At any time, Rasmus, Zend, and company can choose to defect from the community and put all their future efforts into a commercial or competing product based on PHP.

3.4.8 Communication

PHP makes it easy to communicate with other programs and protocols. The PHP development team seems committed to providing maximum flexibility to the largest number of users.

Database connectivity is especially strong, with native-driver support for about 15 of the most popular databases plus ODBC. In addition, PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time. PHP5 extends this support even further, offering a fully incorporated

GD graphics library and revamped XML support with DOM and simpleXML.

Most things that PHP does not support are ultimately attributable to closed-source shops on the other end. For instance, Microsoft has not thus far been eager to cooperate with open source projects like PHP. Potential users who complain about lack of native Mac OS 9 or .NET support on the PHP mailing list are simply misinformed about where the fault lies.

3.4.9 Fast Feature Development

Users of proprietary Web development technologies can sometimes be frustrated by the hostile speed at which new features are added to the official product standard to support emerging technologies. With PHP, this is not a problem. All it takes is one developer, a C compiler, and a dream to add important new functionality. This is not to say that the PHP team will accept every random contribution into the official distribution without community buy-in, but independent developers can and do distribute their own extensions which may be later folded into the main PHP package in more or less unitary form.

PHP development is also constant and ongoing. Although there are clearly major inflection points, such as the transition between PHP4 and PHP5, these tend to be most important deep in the guts of the parser—people were actually working on major extensions throughout the transition period without critical problems. Furthermore, the PHP group subscribes to the

open source philosophy of "release early, release often," which gives developers many opportunities to follow along with changes and report bugs.

3.4.10 Popularity

PHP is fast becoming one of the most popular choices for so-called two-tier development. Although it's not evident from this graphic, the period October 1998 through October 1999 showed 800 percent growth in the number of domains. As Web sites become even more ubiquitous, and as more of them go beyond simple static HTML pages, PHP is expected to gain ground quickly in absolute numbers of users. Although it's somewhat more difficult to get firm figures, it seems that PHP is also in a strong position relative to similar products. According to a 2002 Zend report, Microsoft Active Server Pages technology appears to be utilized on about 24 percent of Web servers, whereas ColdFusion is implemented on approximately 4 percent of surveyed domains. PHP is used on over 24 percent of all Web servers, as measured by a larger and more accurate sample, and is now said to be the most popular server-side scripting language on the Web.

PHP enjoys substantial advantages over its competitors in this development category, which has turned out to be the majority of the Internet.

3.6.11 Not Proprietary

The history of the personal computer industry to date has largely been a chronicle of proprietary standards: attempts to establish them, clashes between them, their benefits and drawbacks for the consumer, and how they are eventually replaced with new standards.

But in the past few years the Internet has demonstrated the great convenience of voluntary, standards-based, platform-independent compatibility. E-mail, for example, works so well because it enjoys a clear, firm standard to which every program on every platform must conform. New developments that break with the standard (for example, HTML-based e-mail stationery) are generally regarded as deviations, and their users find themselves having to bear the burdens of early adoption.

PHP is in a position of maximum flexibility because it is, so to speak, *antiproprietary*. It is not tied to any one server operating system, unlike Active Server Pages. It is not tied to any proprietary cross-platform standard or middleware, as Java Server Pages or ColdFusion are. It is not tied to any one browser or implementation of a programming language or database. PHP isn't even doctrinaire about working only with other open source software. This independent but cooperative pragmatism should help PHP ride out the stormy seas that seem to lie ahead.

3.5 MySQL Versions

Before we can choose, it is good to look first at what is actually available. Multiple versions of MySQL are available at the same time, which may initially be a bit confusing. Right now we have:

Upcoming Releases

* MySQL 6.0 - Alpha
* MySQL Maria Preview

* Snapshots - source code snapshots of the development trees
* Glassfish+MySQL bundle — developers edition

Older Releases

* MySQL 4.0 - Old production release
* MySQL 4.1 - GA/Production
* MySQL 5.0 - Beta

You will also notice an extra number in releases you can download, such as "4.1.10". The "10" is like the build number within the series. If you talk with someone about what version you use, it is vital to name this full version, i.e. "I am using version 4.1.10" so that people know exactly which version you are talking about.

As a hint, you can always find out which version of the MySQL server you are using by issuing the following SQL statement: SELECT VERSION(). This makes the MySQL server return a result set of one row with one column, containing a string with the exact version. Never rely on the version of your client software, as the server could be a completely different version.

3.6 PHP versions

Major Version	Minor Version	Release date	Notes
1.0	1.0.0	1995-06-08	Officially called "Personal Home Page Tools (PHP Tools)". This is the first use of the name "PHP".

2.0	2.0.0	1996-04-16	Considered by its creator as the "fastest and simplest tool" for creating dynamic web pages.
3.0	3.0.0	1998-06-06	Development moves from one person to multiple developers. Zeev Suraski and Andi Gutmans rewrite the base for this version.
4.0	4.0.0	2000-05-22	Added more advanced two-stage parse/execute tag-parsing system called the Zend engine.
	4.1.0	2001-12-10	Introduced 'superglobals' (`$_GET`, `$_POST`, `$_SESSION`, etc.)
	4.2.0	2002-04-22	Disabled register_globals by default. Data received over the network is not inserted directly into the global namespace anymore, closing possible security holes in applications.
	4.3.0	2002-12-27	Introduced the CLI, in addition to the CGI.
	4.4.0	2005-07-11	Added man pages for `phpize` and `php-config` scripts.
	4.4.8	2008-01-03	Several security enhancements and bug fixes. Was to be the end of life release for PHP 4. Security updates only until 2008-08-08, if necessary.
	4.4.9	2008-08-07	More security enhancements and bug fixes. The last release of the PHP 4.4 series.

5.0	5.0.0	2004-07-13	Zend Engine II with a new object model.
	5.1.0	2005-11-24	Performance improvements with introduction of compiler variables in re-engineered PHP Engine.
	5.2.0	2006-11-02	Enabled the filter extension by default.
	5.2.8	2008-12-08 [26]	emergent bug fix
	5.3.0	First Quarter of 2009[27]	Namespace support; Improved XML support through use of XMLReader and XMLWriter; Late static bindings [28], Jump label (limited goto), Closures, Native PHP archives, Garbage collection, Persistent Connection with mysqli
6.0	6.0.0	No date set	Unicode support; removal of ereg extension, 'register_globals', 'magic_quotes' and 'safe_mode'; Alternative PHP Cache; Removal of mime_magic and rewrite of fileinfo() for better MIME support

Note

	Meaning
Red	Old release; not supported
Yellow	Old release; still supported
Green	Current release
Blue	Future release

3.7 Defining the Banking System

The first step in designing a database is to gain a working knowledge of the current banking system. A banking system is the way a bank performs its duties to meet its goals. For example, a bank has the following banking process in its Banking system:

1. Opening an account that is, saving, current, fixed account etc.
2. Checking of Account balance that is inquiry.
3. Crediting and debiting of account.
4. Funds transfer.
5. Loan
6. Closing of account.

A database will be created in respect to the above banking process, for example at the phpmyadmin database and tables can be created. *Fig 3.2* and *Fig 3.3* shows the databases used by the online banking system.

Conclusion

In conclusion, there are many different techniques to help you gain an understanding of the banking process. The most helpful is to interview the people who work with the system everyday. These people should know the inner workings of the process. Also, there are also examples of other online system (for instance the winestore application) that can help understand the PHP and MySQL advantages and working process.

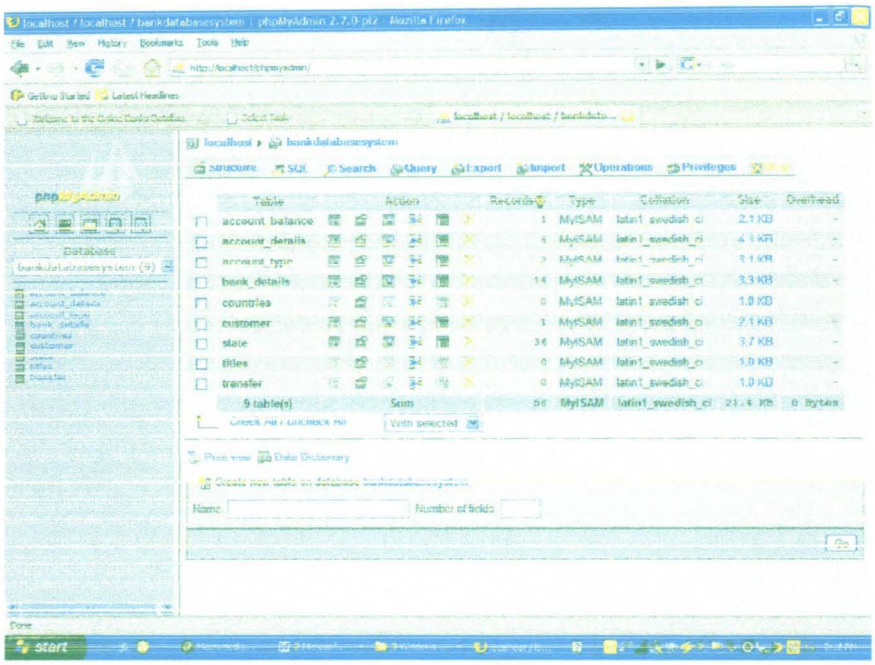

Fig 3.2 List of tables in the database (customer) used in this study

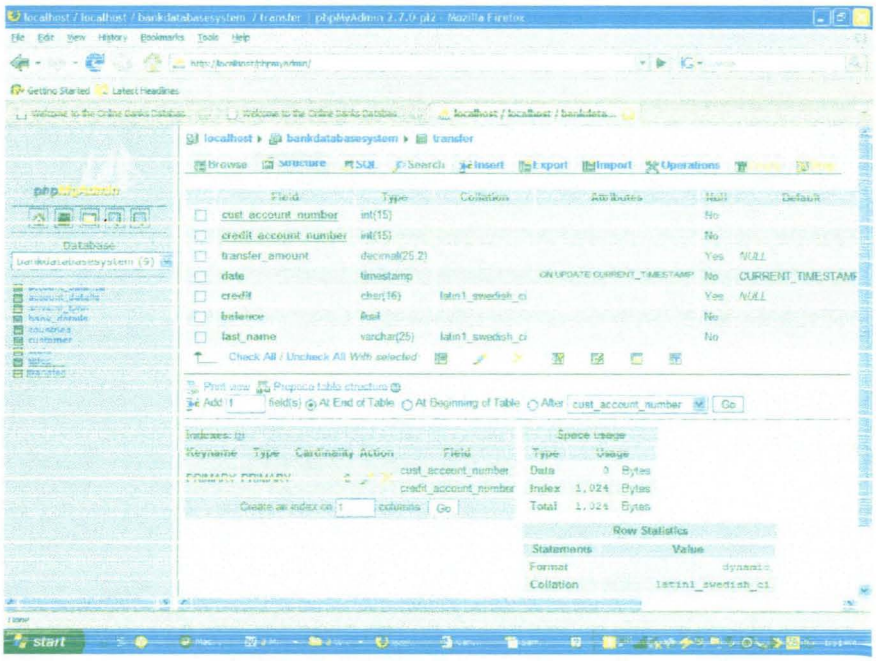

Fig 3.3 lists of record in a table (customer) in the database

CHAPTER FOUR

APPLICATION OF PHP AND MYSQL TO BANKING

4.0 Introduction

This chapter deals with the application of a database (MySQL) that is integrated with the web (using PHP). It was developed on Microsoft Windows and it works fine on Linux and Mac OS X environment. It shows the techniques for developing an online web database application that store, manage and retrieve data. The architecture describes a successful framework for the application that can run on modest hardware and process more than a million hits per day. In other words, this research work is on the design of a good portal for the twenty four (24) commercial banking institutions in Nigeria.

4.1 Applications of MYSQL and PHP in banking.

Developing applications using MySQL and PHP are straightforward. It uses a session management that is unique to interaction between a browser and a web database application. This provides a way of storing the database.

To create applications that can be read in any browser, the following will be required:

i. Programming/ Scripting Language (PHP)
ii. Database Server (MySQL)
iii. Browser (Internet Explorer)
iv. Web server (Apache)

When requests are made from the browser, web page then pulls data out of a database the request is sent to the web server which in turn calls a PHP

script. The PHP script is executed by the PHP preprocessor; it pulls data from the database. The results are then massaged by the rest of the PHP script and turned into HTML. The final HTML gets sent back to the user's browser. Fig 4.0 shows the step by step process of how MySQL, PHP and Apache work together.

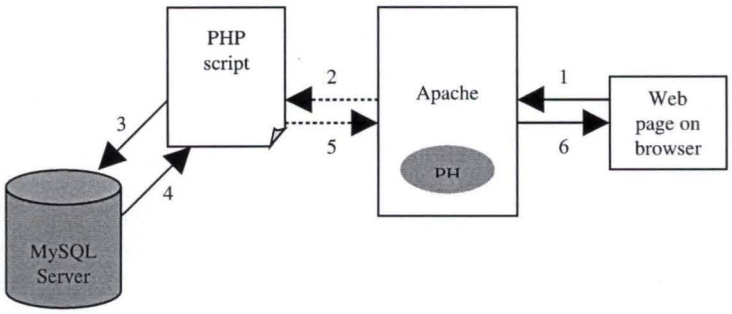

Fig 4.1 *the step by step process of how MySQL, PHP and Apache work together.*

4.2 Phases of Creating a MySQL and PHP Application

The Application phase includes: -

a. Web design

b. Design of files and Databases

c. Design of system user requirements.

d. System Specification.

e. Determine precisely what output will be required for the new system.

4.2.1 Web Design (Basic Architecture)

Web design is a process whereby Website or webpage is developed through system analysis is synthesized with related knowledge in order to achieve the desired goals.

The client

The applications you can develop with MySQL and PHP make use of a single client that is, the Web browser.

Web server

A specific application, called a Web server, will be responsible for communicating with the browser. PHP is used to broker requests between the Web server and the database server; it will also be used to perform programmatic tasks on the information that comes to and from the Web server. Figure 4.2 represents this system. But of course none of this is possible without an operating system. The Web server, programming language, and database server you use must work well with your operating system. The Apache Web server is the most popular Web server there is. It, like Linux, PHP, and MySQL, is an open-source project. Apache makes use of third-party modules.

Operating System

There are many operating systems out there. Windows XP and Linux are probably the most popular. Almost all PHP/MySQL applications are running off of some version of Unix, whether it be Linux, BSD, Irix, Solaris, HP-UX, or Windows. For that reason, the applications in this Project will work with Windows.

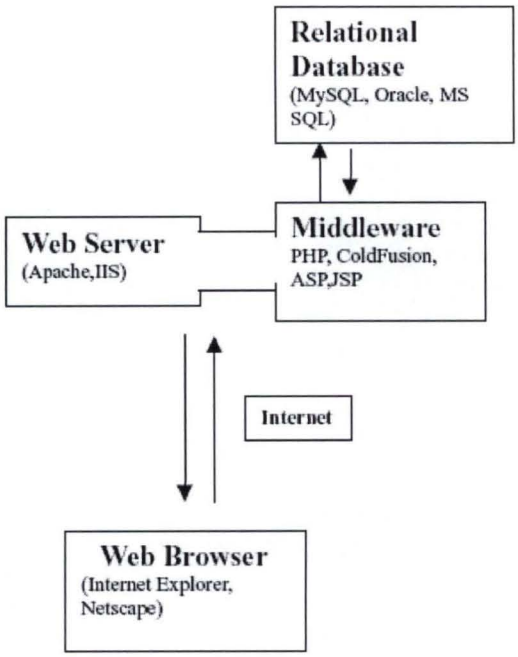

Figure 4.2: Architecture of Web applications

Middleware

PHP belongs to a class of languages known as *middleware*. These languages work closely with the Web server to interpret the requests made from the World Wide Web, process these requests, interact with other programs on the server to fulfill the requests, and then indicate to the Web server exactly what to serve to the client's browser.

The middleware is where vast majority of work is done. Here the Web server can be up and running without a whole lot of effort. And once it is up and running, you won't need to fool with it a whole lot. In addition to PHP, there are several languages that perform similar functions.

Relational Databases

Relational Database Management Systems (RDBMSs) provide a great way to store and access complex information. MySQL is the RDBMS that will be use for this project. Some of the other popular commercial RDBMSs are MySQL, Oracle, Sybase, Informix, Microsoft's SQL Server, and IBM's db2.

4.2.2 System Requirements

This is an integral requirement for the overall effectiveness of the operation of the system. This involves the software and the hardware requirement.

i. Software Requirement

* MySQL 5.0.0
* MySQL Administrator 1.1.8
* MySQL Query Browser 1.1.20
* PHP 5.1.2
* Apache 2.2 x Module
* Browser (for example Firefox, Internet explorer)
* Windows NT/XP/2002 service pack 2 or Linux
* MS – Dos

ii. Hardware Requirement

* Intel® Pentium IV and above.
* Processor speed 1.5GHz and above
* Harddisk minimum of 40GB
* RAM 128MB (512MB recommended)

4.3 Online Banking System Overview

The Online Banking System application was developed to meet the requirements outline in section 4.2.2 of this system.

The Online Banking System has many components of a typical web database application, including:

* Maintainable web pages generated with templates, and populated data from a database.
* User-driven querying and browsing, in which the user provides the parameters that limit the searching or browsing of the database. This includes one-component querying
* Data entry and validation.
* User tracking with session management techniques.
* User authentication and management.
* SQL querying that requires table locking.
* Receipt pages that avoid the reload problem.
* Robust error handling with a custom error module.
* Email and browser alerts.

4.3.1 Application Module

The application has four separate modules that are:

- Banks /Customer Management: this includes opening of account and editing of personal details.
- Crediting, Debiting of Account and transfer of funds.
- Managing Account.
- Authentication: includes logging in, logging out and changing passwords.

The application also has a set of common components, including authentication functions, a custom error handler, validation functions, a custom error handler, validation functions, and general purpose functions and constants.

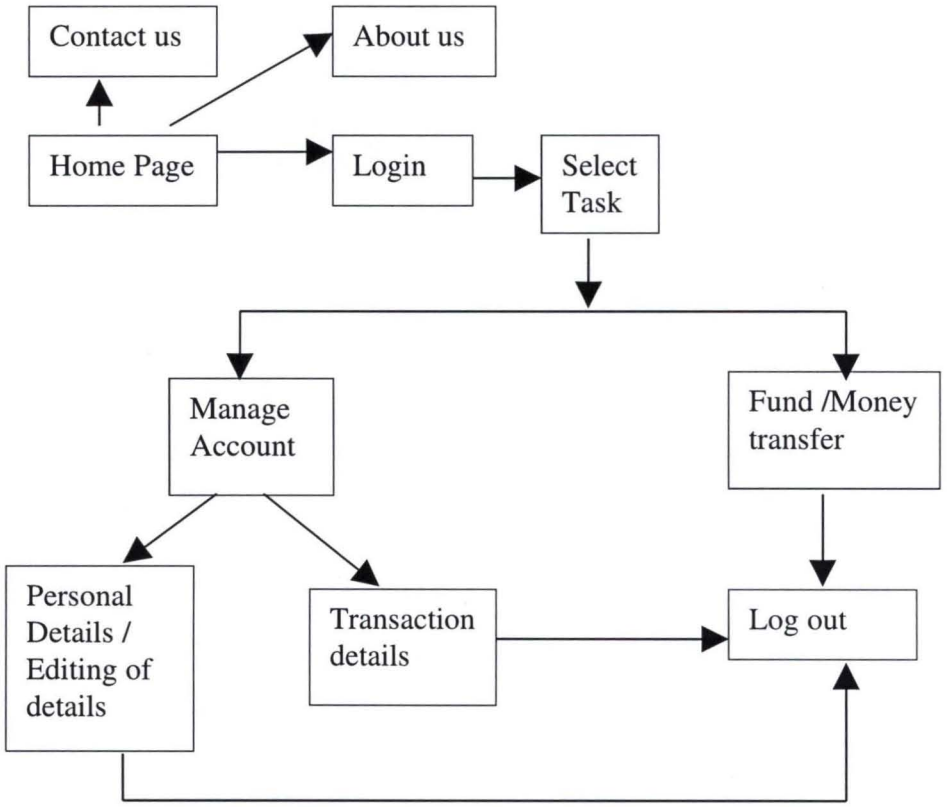

Fig 4.3 Online Banking System Architecture

4.4 Functional Overview

The Architecture of the Online Banking System is shown in Fig 4.3. It shows the flow of events.

Banks

The list of the twenty – four (24) Nigerian banks is contained the home page. The script *index.php* contains the following bank links namely:

Accessbank, Afribank, BankPHB, Citibank, Diamondbank, Ecobank, ETB, Fcmb, Fidelitybank, Finbank, Firstbank, Gtbank, Intercontinentalbank, Oceanicbank, Skyebank, Springbank, Stanbic_IBTC bank, Standard Chartered bank, Sterling_bank, UBA, Union Bank, Unity Bank, Wema bank, Zenith bank. This page allows the user to access any bank of their choice and to access the other parts of the application that is contact us and about us. This page *Fig 4.4* allows entry of all customers.

Each bank link takes you the specific bank website, where there is an authentication page that will further link you to the transaction page.

Account and PIN management

This is provided by the authpassword.php and changepassword.php scripts. To change PIN, users are required to enter their current four (4) digits Pin to reduce the risk of an unauthorized change, then enter the new PIN twice to minimize the chance of typing error.

Authentication

The authorization scripts are loginform.php and logout.php. The loginform.php script produces a form for user to enter their personal details and a four (4) digit PIN. The loginform.php *(Fig 4.5)* script validates the account number and the pin, and checks if a matching user. If so, the scripts logs the user into the login page. If the process fails, they're returned to authorization.php *(Fig 4.6)* and errors are displayed. The script logout.php *(fig 4.13)* logs the user out of the application and redirects her to the main bank page index.php; the logout script doesn't produce out.

Checks the user's credentials against a database, and registers the user as logged in by setting a session variable.

Customer details management

This is provided by the customerdetails.php, selecttask.php, manage_account.php and accountopeningform.php scripts that implement the open an account and edit details features.

The script customerdetails.php *(Fig 4.8)* presents the customer details form . The form shows entry of all customer details and a PIN number that is used for future visits to the site. The accountopeningform.php *(Fig 4.7)* is used to open account by any customer.

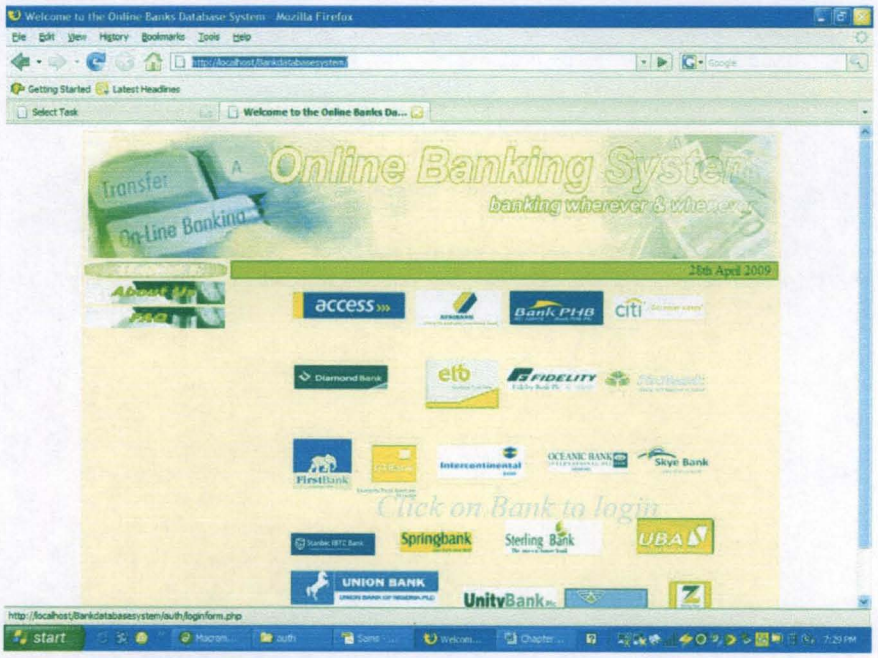

Fig 4.4 index.php

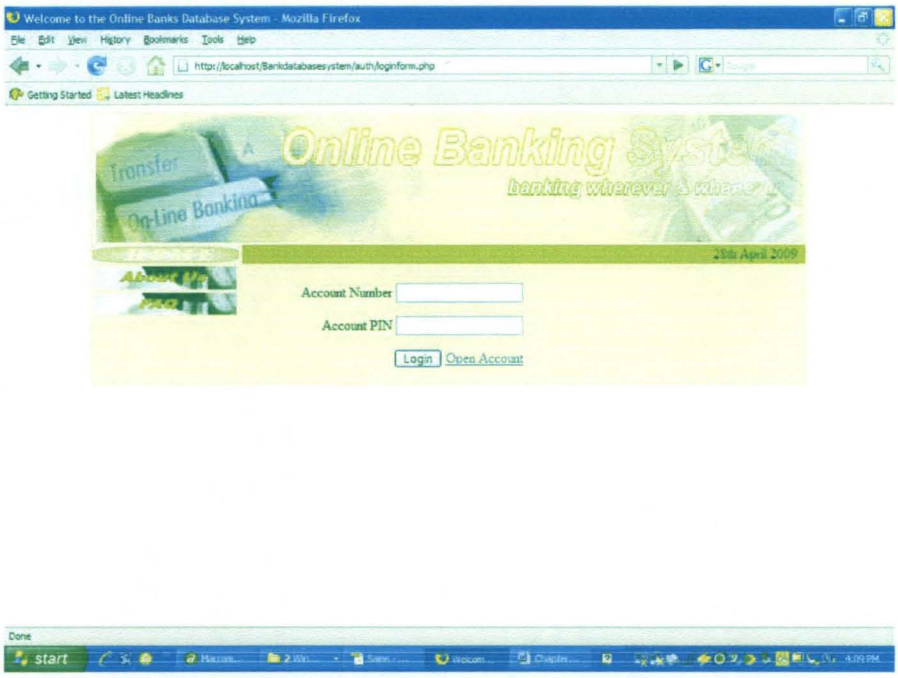

Fig 4.5 loginform.php

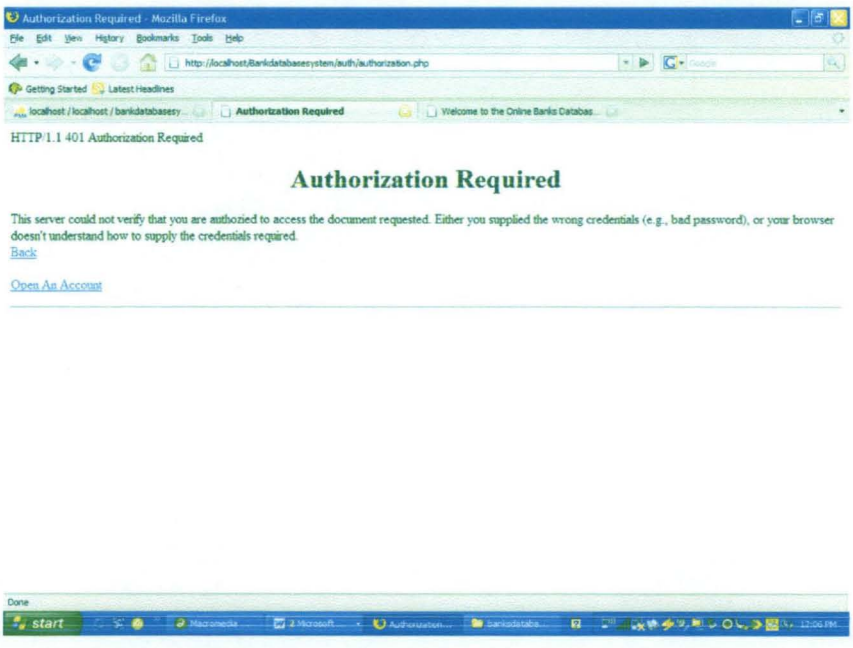

Fig 4.6 authorisation.php

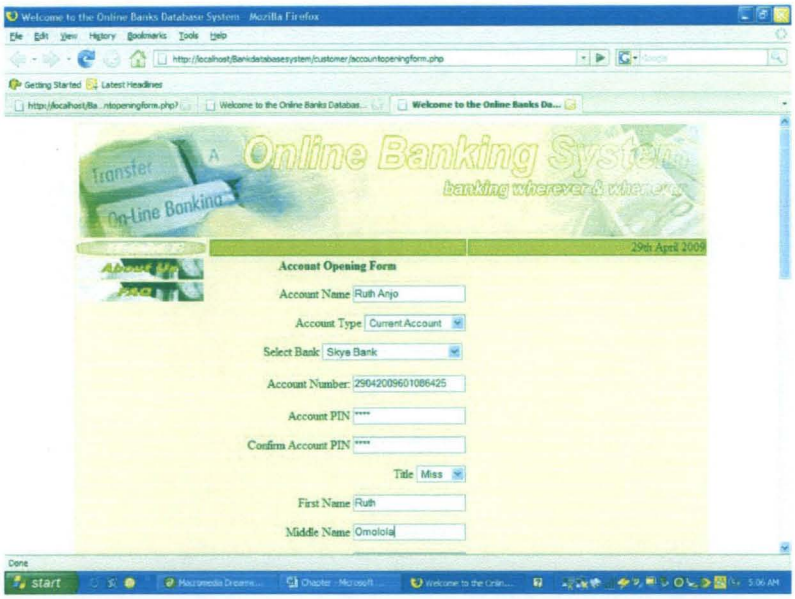

Fig 4.7 accountopeningform.php

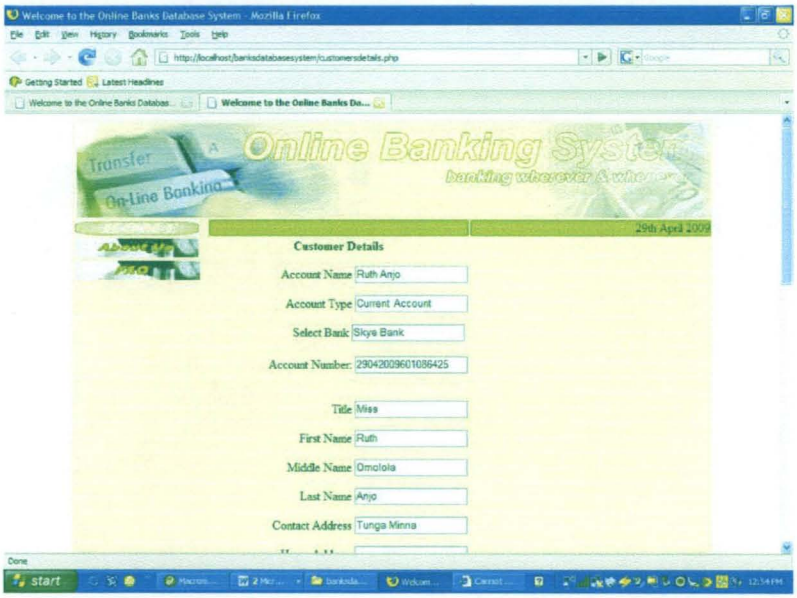

Fig 4.8 customersdetail.php

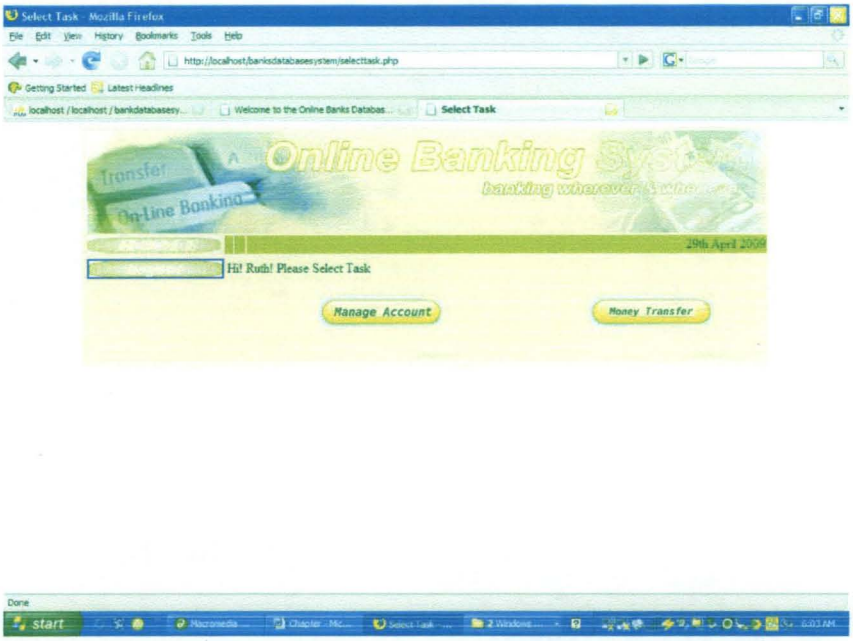

Fig 4.9 selecttask.php

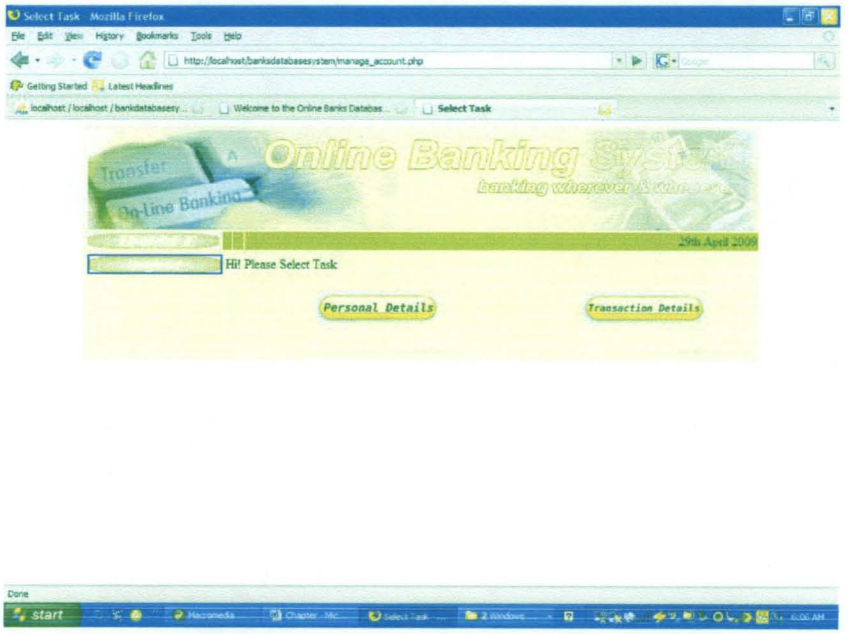

Fig 4.10 manage_account.php

Transaction

This is where most of the transactions such as money transfer (*fig 4.10moneytransfer.php*), account details (*fig 4.12 transactiondetails.php*) are done.

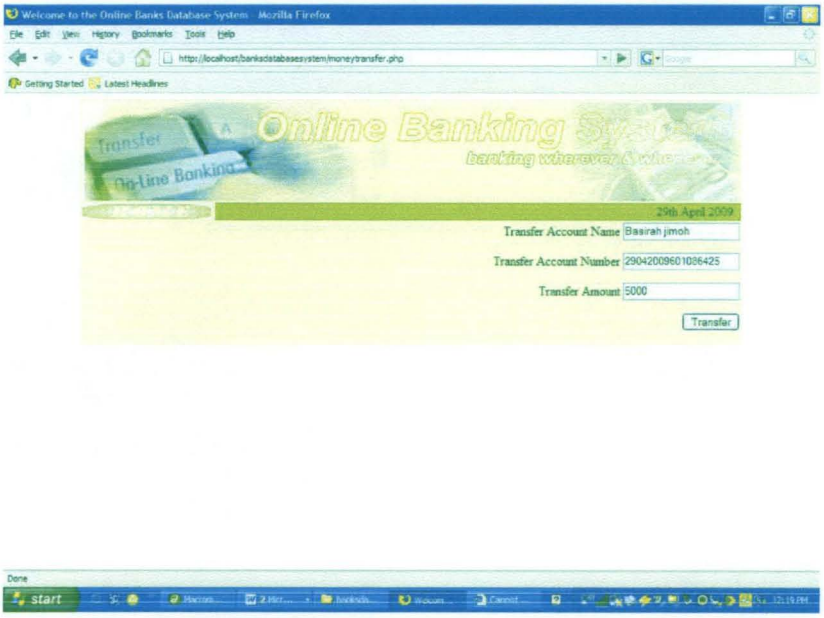

Fig 4.11 moneytransfer.php

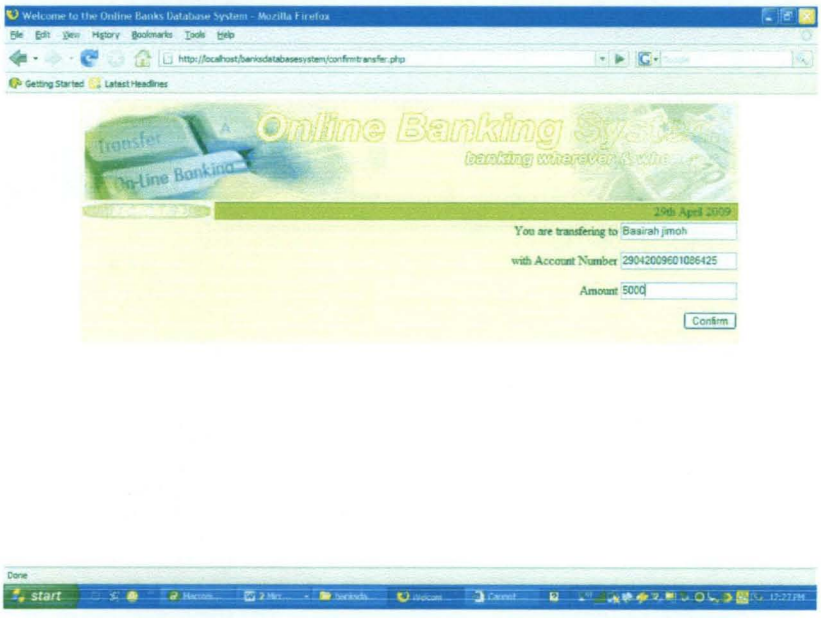

Fig 4.12 confirmtransfer.php

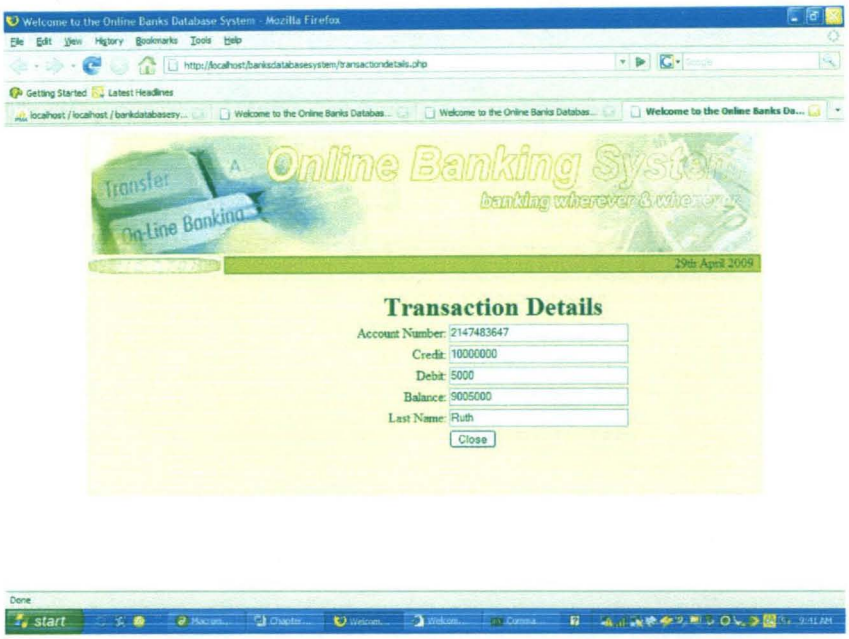

Fig 4.13 transactiondetails.php

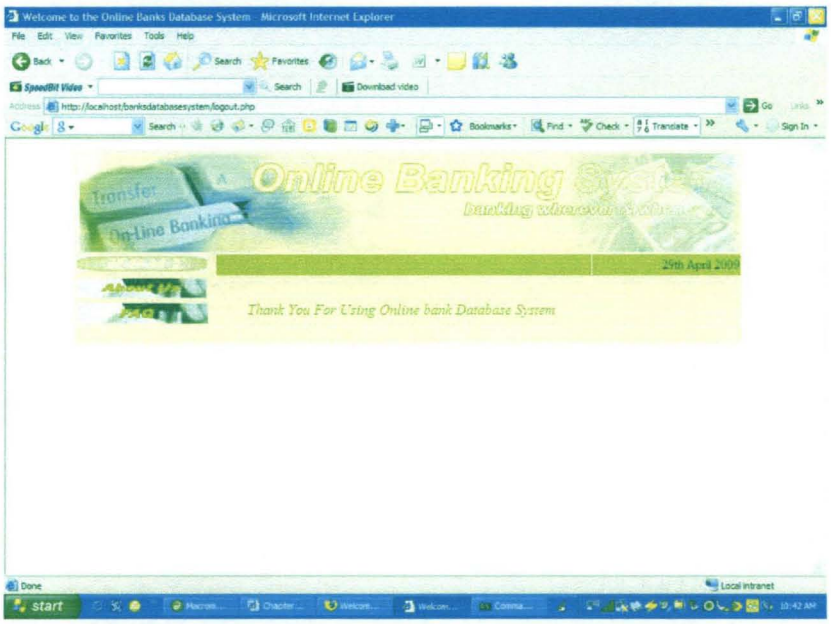

Fig 4.14 logout.php

Conclusion

In conclusion, developing this online database system involves web design, creating files and databases etc. Files uses in the development can be seen in detail in APPENDIX A.

CHAPTER FIVE

5.0 Summary

The entire work was divided into five chapters. Chapter One gives the general introduction of the study, it states the background, importance, significance, objective, scope and limitation of study. Chapter Two review past relevant and historical literature of the study. Chapter Three describes an in-depth problem of study using MySQL and PHP. Application of MYSQL and PHP to banking is the Chapter Four. Then the Summary, Recommendation and Conclusion are in the Chapter Five.

This online database system for banks using MYSQL and PHP is not replacing the already existing banking software but is going to work with the software in other for them to work faster and effectively.

The system has been programmed to accept data online, process it and give accurate feedback. The advantages of the system include ease and speed of data organization and processing.

5.1 Recommendation

This website is recommended for every commercial bank in Nigeria. It stands as a bond that fuses all banks together in-order for them to share and communicate.

To use the Online Banking System and to ensure the highest level of security, I recommend that you always make sure that you use the following browser versions.

- Microsoft Internet Explorer 6.0
- Mozilla Firefox
- Opera 3.1 and above
- Netscape Navigator 7.2 and above

If you are using an older browser version that is no longer compatible with the Online Banking System you can go to their web sites and upgrade your browser to a new version.

To ensure security, it is recommended that you take the following measures:
1. Change you pin regularly and keep it safe.

2. Always logout and close your browser after using Online Banking System.

I also recommend that more work should be done on the database by future researchers.

5.2 Conclusion

In conclusion, the project work "Design and Development of an Online Database System for Banks Using MySQL and PHP" is designed to improve the approach to offline bank transactions. The online database system for banks is a goal- driven website. Its main action is to transact money online.

Finally, the project work defined and explained MySQL and PHP. MySQL is the database that was used in the design of the system and PHP which is the programming or scripting language used in the study.

APPENDIX B

DATABASE SOURCE CODE

Bankdatabasesystem.data

```
-- phpMyAdmin SQL Dump
-- version 2.7.0-pl2
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Generation Time: Apr 29, 2009 at 03:27 PM
-- Server version: 5.0.18
-- PHP Version: 5.1.2

SET AUTOCOMMIT=0;
START TRANSACTION;

--
-- Database: `bankdatabasesystem`
--

-- --------------------------------------------------------

--
-- Table structure for table `account_balance`
--

CREATE TABLE `account_balance` (
  `cust_account_number` int(15) NOT NULL,
  `credit` decimal(25,2) default NULL,
  `debit` decimal(25,2) default NULL,
  `balance` decimal(25,2) default NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`cust_account_number`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `account_balance`
--
```

```
INSERT INTO `account_balance` VALUES (2147483647, '10000000.00',
'5000.00', '9005000.00', 'Ruth');

-- --------------------------------------------------------

--
-- Table structure for table `account_details`
--

CREATE TABLE `account_details` (
  `cust_id` int(11) NOT NULL auto_increment,
  `cust_account_number` int(15) NOT NULL,
  `account_name` varchar(50) NOT NULL,
  `account_pin` varchar(4) NOT NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`cust_id`),
  KEY `account_pin` (`account_pin`),
  KEY `cust_account_number` (`cust_account_number`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
AUTO_INCREMENT=5 ;

--
-- Dumping data for table `account_details`
--

INSERT INTO `account_details` VALUES (1, 12345678, 'Ruth Omolola',
'1234', '');
INSERT INTO `account_details` VALUES (2, 87654321, 'Basirah Jimoh',
'2345', '');
INSERT INTO `account_details` VALUES (3, 2147483647, 'Basirah
Jimoh', '1234', 'Basirah');
INSERT INTO `account_details` VALUES (4, 2147483647, 'Victor Anjo',
'2345', 'Victor');

-- --------------------------------------------------------

--
-- Table structure for table `account_type`
--
```

```
CREATE TABLE `account_type` (
  `account_type_id` int(3) NOT NULL,
  `account_type` char(25) NOT NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`account_type_id`),
  KEY `var` (`account_type`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `account_type`
--

INSERT INTO `account_type` VALUES (1, 'Current Account', '');
INSERT INTO `account_type` VALUES (2, 'Saving Account', '');

-- --------------------------------------------------------

--
-- Table structure for table `bank_details`
--

CREATE TABLE `bank_details` (
  `bank_id` int(5) NOT NULL,
  `bank_name` varchar(50) NOT NULL,
  `year` int(4) NOT NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`bank_id`),
  KEY `name` (`bank_name`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `bank_details`
--

INSERT INTO `bank_details` VALUES (301, 'Access Bank', 2009, '');
INSERT INTO `bank_details` VALUES (302, 'Afribank', 2009, '');
INSERT INTO `bank_details` VALUES (303, 'BankPHB', 2009, '');
INSERT INTO `bank_details` VALUES (304, 'Citibank', 2009, '');
INSERT INTO `bank_details` VALUES (305, 'Diamond Bank', 2009, '');
```

```
INSERT INTO `bank_details` VALUES (306, 'Ecobank', 2009, '');
INSERT INTO `bank_details` VALUES (307, 'ETB', 2009, '');
INSERT INTO `bank_details` VALUES (308, 'FCMB', 2009, '');
INSERT INTO `bank_details` VALUES (309, 'Fidelity Bank', 2009, '');
INSERT INTO `bank_details` VALUES (310, 'Finbank', 2009, '');
INSERT INTO `bank_details` VALUES (311, 'First Bank', 2009, '');
INSERT INTO `bank_details` VALUES (312, 'GTBank', 2009, '');
INSERT INTO `bank_details` VALUES (313, 'Intercontinental Bank', 2009,
'');
INSERT INTO `bank_details` VALUES (314, 'Oceanic Bank', 2009, '');

-- --------------------------------------------------------

--
-- Table structure for table `countries`
--

CREATE TABLE `countries` (
  `country_id` int(4) NOT NULL,
  `country` char(30) NOT NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`country_id`),
  KEY `country` (`country`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `countries`
--


-- --------------------------------------------------------

--
-- Table structure for table `customer`
--

CREATE TABLE `customer` (
  `cust_id` int(11) NOT NULL auto_increment,
  `cust_account_number` int(15) NOT NULL,
  `surname` varchar(50) default NULL,
```

```
  `last_name` varchar(50) default NULL,
  `middle_name` char(50) default NULL,
  `gender` enum('Male','Female') default NULL,
  `title_id` int(3) default NULL,
  `address` varchar(50) default NULL,
  `city` varchar(50) default NULL,
  `state` varchar(20) default NULL,
  `zipcode` varchar(10) default NULL,
  `country` varchar(25) default NULL,
  `phone` varchar(15) default NULL,
  `birth_date` char(10) default NULL,
  `mother_maiden_name` char(50) default NULL,
  PRIMARY KEY  (`cust_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
AUTO_INCREMENT=2 ;

--
-- Dumping data for table `customer`
--

INSERT INTO `customer` VALUES (1, 29042009601086425, 'Anjo',
'Ruth', 'Omolola', 'Female', 0, 'Tunga Minna', 'Minna', 'Niger', '234', '27',
'08037811034', '1985-03-09', 'Olowo');

-- --------------------------------------------------------


--
-- Table structure for table `state`
--

CREATE TABLE `state` (
  `state_id` int(4) NOT NULL,
  `state_name` varchar(50) NOT NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`state_id`),
  KEY `region` (`state_name`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `state`
```

```
--

INSERT INTO `state` VALUES (1, 'Abia', '');
INSERT INTO `state` VALUES (2, 'Adamawa', '');
INSERT INTO `state` VALUES (3, 'Akwa Ibom', '');
INSERT INTO `state` VALUES (4, 'Anambra', '');
INSERT INTO `state` VALUES (5, 'Bauchi', '');
INSERT INTO `state` VALUES (6, 'Bayelsa', '');
INSERT INTO `state` VALUES (7, 'Benue', '');
INSERT INTO `state` VALUES (8, 'Borno', '');
INSERT INTO `state` VALUES (9, 'Cross River', '');
INSERT INTO `state` VALUES (10, 'Delta', '');
INSERT INTO `state` VALUES (11, 'Ebonyi', '');
INSERT INTO `state` VALUES (12, 'Edo', '');
INSERT INTO `state` VALUES (13, 'Ekiti', '');
INSERT INTO `state` VALUES (14, 'Enugu', '');
INSERT INTO `state` VALUES (15, 'Gombe', '');
INSERT INTO `state` VALUES (16, 'Imo', '');
INSERT INTO `state` VALUES (17, 'Jigawa', '');
INSERT INTO `state` VALUES (18, 'Kaduna', '');
INSERT INTO `state` VALUES (19, 'Kano', '');
INSERT INTO `state` VALUES (20, 'Katsina', '');
INSERT INTO `state` VALUES (21, 'Kebbi', '');
INSERT INTO `state` VALUES (22, 'Kogi', '');
INSERT INTO `state` VALUES (23, 'Kwara', '');
INSERT INTO `state` VALUES (24, 'Lagos', '');
INSERT INTO `state` VALUES (25, 'Nassarawa', '');
INSERT INTO `state` VALUES (26, 'Niger', '');
INSERT INTO `state` VALUES (27, 'Ogun', '');
INSERT INTO `state` VALUES (28, 'Ondo', '');
INSERT INTO `state` VALUES (29, 'Osun', '');
INSERT INTO `state` VALUES (30, 'Oyo', '');
INSERT INTO `state` VALUES (31, 'Plateau', '');
INSERT INTO `state` VALUES (32, 'River', '');
INSERT INTO `state` VALUES (33, 'Sokoto', '');
INSERT INTO `state` VALUES (34, 'Taraba', '');
INSERT INTO `state` VALUES (35, 'Zamfara', '');
INSERT INTO `state` VALUES (36, 'FCT', '');
-- --------------------------------------------------
--
```

```
-- Table structure for table `titles`
--

CREATE TABLE `titles` (
  `title_id` int(2) NOT NULL,
  `title` char(10) default NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`title_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `titles`
--
-----------------------------------------------------------
-- Table structure for table `transfer`
--

CREATE TABLE `transfer` (
  `cust_account_number` int(15) NOT NULL,
  `credit_account_number` int(15) NOT NULL,
  `transfer_amount` decimal(25,2) default NULL,
  `date` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
  `credit` char(16) default NULL,
  `balance` float NOT NULL,
  `last_name` varchar(25) NOT NULL,
  PRIMARY KEY  (`cust_account_number`,`credit_account_number`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `transfer`

COMMIT;
```

APPENDIX C

PHP SOURCE CODES

Index.php

```
<html xmlns>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
>
<title>Welcome to the Online Banks Database System</title>
</head>

<body >
<table bgcolor='#FFFFCC' align="center" width="825">
 <tr>
        <td colspan="2"><div align="center"><img src="internets.gif"
alt="Welcome to the bank database system" width="817" height="148" /></
div></td>
 </tr>
 <tr>
  <td width="169"><a href="index.php"><img src="home.gif" alt="Home"
width="169" height="20" border="0" /></a>
  <div align="center"></div></td>
  <td width="644" bgcolor="#993300"><div align="right">
    <?php     echo date('jS F Y'); ?>
   </div></td>
 </tr>
 <tr>
                    <td     height="27"><a     href="aboutus.php"><img
src="mm_lodging_image.jpg" alt="About us" width="166" height="27"
border="0" /></a></td>
    <td rowspan="3"><div align="center"><a href="loginform.php"><img
src="obslogos.png" alt="Click on Bank" width="500" height="440"
border="0" /></a></div></td>
 </tr>
 <tr>
     <td height="27"><a href="faq.php"><img src="faq.jpg" alt="FAQ"
width="166" height="25" border="0" longdesc="faq.php" /></a></td>
 </tr>
```

```
 <tr>
  <td> </td>
 </tr>
</table>

</body>
</html>
```

Details.php

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
# HTTP="true"
$hostname_details = "127.0.0.1";
$database_details = "bankdatabasesystem";
$username_details = "root";
$password_details = "12345678";
$details = mysql_pconnect($hostname_details, $username_details,
$password_details) or trigger_error(mysql_error(),E_USER_ERROR);
?>
```

loginform.php

```
?php require_once('Connections/details.php'); ?>
<?php
// *** Validate request to login to this site.
if (!isset($_SESSION)) {
 session_start();
}

$loginFormAction = $_SERVER['PHP_SELF'];
if (isset($_GET['accesscheck'])) {
 $_SESSION['PrevUrl'] = $_GET['accesscheck'];
}

if (isset($_POST['textfield'])) {
  $loginUsername=$_POST['textfield'];
 $password=$_POST['textfield2'];
 $MM_fldUserAuthorization = "";
 $MM_redirectLoginSuccess = "selecttask.php";
 $MM_redirectLoginFailed = "authorization.php";
 $MM_redirecttoReferrer = false;
 mysql_select_db($database_details, $details);
```

```
 $last_name=$_GET['last_name'];
                                        $LoginRS__query=sprintf("SELECT
cust_account_number, account_pin, last_name FROM account_details
WHERE cust_account_number='%s' AND account_pin='%s'",
                    get_magic_quotes_gpc()  ?  $loginUsername  :
addslashes($loginUsername), get_magic_quotes_gpc() ? $password :
addslashes($password));

        $LoginRS  =  mysql_query($LoginRS__query,  $details)  or
die(mysql_error());
 $loginFoundUser = mysql_num_rows($LoginRS);
 if ($loginFoundUser) {
   $loginStrGroup = "";

  //declare two session variables and assign them
       $_SESSION['valid_user'] = $last_name;
  $_SESSION['MM_Username'] = $loginUsername;
  $_SESSION['MM_UserGroup'] = $loginStrGroup;

  if (isset($_SESSION['PrevUrl']) && false) {
    $MM_redirectLoginSuccess = $_SESSION['PrevUrl'];
  }
  header("Location: " . $MM_redirectLoginSuccess );
 }
 else {
 //  header("Location: ". $MM_redirectLoginFailed );

 }
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
>
<title>Welcome to the Online Banks Database System</title>
</head>
```

```
<body >
<table align="center" bgcolor='#FFFFCC' width="825">
 <tr>
        <td colspan="3"><div align="center"><img src="internets.gif"
alt="Welcome to the bank database system" width="825" height="125" /></
div></td>
 </tr>
 <tr>
   <td width="169"><a href="index.php"><img src="home.gif" alt="Home"
width="162" height="20" border="0" /></a>
   <div align="center"></div></td>
   <td colspan="2" bgcolor="#993300"><div align="right">
     <?php      echo date('jS F Y'); ?>
    </div></td>
 </tr>
<?php
// If this file is not included from the MMHTTPDB possible hacking
problem.
if (!function_exists('create_error')){
        die();
}

define('MYSQL_NOT_EXISTS', create_error("Your PHP server doesn't
have the MySQL module loaded or you can't use the mysql_(p)connect
functions."));
define('CONN_NOT_OPEN_GET_TABLES', create_error('The Connection
is not opened when trying to retrieve the tables. Please refer to
www.interaktonline.com for more information.'));
define('CONN_NOT_OPEN_GET_DB_LIST',          create_error('The
Connection is not opened when trying to retrieve the database list. Please
refer to www.interaktonline.com for more information.'));

if (!function_exists('mysql_connect') || !function_exists('mysql_pconnect')
|| !extension_loaded('mysql')){
        echo MYSQL_NOT_EXISTS;
        die();
}

// Now let's handle the crashes or any other PHP errors that we can catch
function KT_ErrorHandler($errno, $errstr, $errfile, $errline) {
```

```
	global $f, $already_sent;
	$errortype = array (
		1  => "Error",
		2  => "Warning",
		4  => "Parsing Error",
		8  => "Notice",
		16 => "Core Error",
		32 => "Core Warning",
		64 => "Compile Error",
		128 => "Compile Warning",
		256 => "User Error",
		512 => "User Warning",
		1024=> "User Notice",
		2048=> "E_ALL",
		2049=> "PHP5 E_STRICT"

	);
	$str = sprintf("[%s]\n%s:\t%s\nFile:\t\t'%s'\nLine:\t\t%s\n\n", date('d-
m-Y H:i:s'),(isset($errortype[@$errno])?$errortype[@$errno]:('Unknown '.
$errno)),@$errstr,@$errfile,@$errline);
	if (error_reporting() != 0) {
			@fwrite($f, $str);
			if (@$errno == 2 && isset($already_sent) && !
$already_sent==true){
				$error = '<ERRORS>'."\n";
				$error .= '<ERROR><DESCRIPTION>An
Warning Type error appeared. The error is logged into the log
file.</DESCRIPTION></ERROR>'."\n";
				$error .= '</ERRORS>'."\n";
				$already_sent = true;
				echo $error;
			}
	}
}
if ($debug_to_file){
		$old_error_handler = set_error_handler("KT_ErrorHandler");
}

class MySqlConnection
{
```

```
/*
 // The 'var' keyword is deprecated in PHP5 ... we will define these variables
at runtime.
  var $isOpen;
        var $hostname;
        var $database;
        var $username;
        var $password;
        var $timeout;
        var $connectionId;
        var $error;
*/
        function MySqlConnection($ConnectionString, $Timeout, $Host,
$DB, $UID, $Pwd)
        {
                $this->isOpen = false;
                $this->timeout = $Timeout;
                $this->error = '';

                if( $Host ) {
                        $this->hostname = $Host;
                }
                elseif( ereg("host=([^;]+);", $ConnectionString, $ret) )  {
                        $this->hostname = $ret[1];
                }

                if( $DB ) {
                        $this->database = $DB;
                }
                elseif( ereg("db=([^;]+);",  $ConnectionString, $ret) ) {
                        $this->database = $ret[1];
                }

                if( $UID ) {
                        $this->username = $UID;
                }
                elseif( ereg("uid=([^;]+);",  $ConnectionString, $ret) ) {
                        $this->username = $ret[1];
                }
```

```
            if( $Pwd ) {
                $this->password = $Pwd;
            }
            elseif( ereg("pwd=([^;]+);", $ConnectionString, $ret) ) {
                $this->password = $ret[1];
            }
        }

        function Open()
        {
            $this->connectionId = mysql_connect($this->hostname, $this->username, $this->password);
            if (isset($this->connectionId) && $this->connectionId && is_resource($this->connectionId))
            {
                $this->isOpen = ($this->database == "") ? true : mysql_select_db($this->database, $this->connectionId);
            }
            else
            {
                $this->isOpen = false;
            }
        }

        function TestOpen()
        {
            return ($this->isOpen) ? '<TEST status=true></TEST>' : $this->HandleException();
        }

        function Close()
        {
            if (is_resource($this->connectionId) && $this->isOpen)
            {
                if (mysql_close($this->connectionId))
                {
                    $this->isOpen = false;
                    unset($this->connectionId);
                }
            }
```

```
    }

    function GetTables($table_name = '')
    {
        $xmlOutput = "";
        if ($this->isOpen && isset($this->connectionId) &&
is_resource($this->connectionId)){
            // 1. mysql_list_tables and mysql_tablename are
deprecated in PHP5
            // 2. For backward compatibility GetTables don't have
any parameters
            if ($table_name === ''){
                $table_name = @$_POST['Database'];
            }
            $sql = ' SHOW TABLES FROM ' . $table_name;
            $results = mysql_query($sql, $this->connectionId) or
$this->HandleException();

            $xmlOutput = "<RESULTSET><FIELDS>";

            // Columns are referenced by index, so Schema and
            // Catalog must be specified even though they are not
supported

            $xmlOutput .=
'<FIELD><NAME>TABLE_CATALOG</NAME></FIELD>';
    // column 0 (zero-based)
            $xmlOutput .= '<FIELD><NAME>TABLE_SCHEMA</
NAME></FIELD>';          // column 1
            $xmlOutput .=
'<FIELD><NAME>TABLE_NAME</NAME></FIELD>';          //
column 2

            $xmlOutput .= "</FIELDS><ROWS>";

            if (is_resource($results) && mysql_num_rows($results)
> 0){
                while ($row = mysql_fetch_array($results))
{
```

```
                                        $xmlOutput .=
'<ROW><VALUE/><VALUE/><VALUE>' . $row[0].
'</VALUE></ROW>';
                                    }
                    }
                    $xmlOutput .= "</ROWS></RESULTSET>";

    }
            return $xmlOutput;
        }

        function GetViews()
        {
```

Accountopeningform.php

```
<?php require_once('Connections/details.php'); ?>
<?php
// *** Redirect if username exists
$MM_flag="MM_insert";
if (isset($_POST[$MM_flag])) {
 $MM_dupKeyRedirect="loginform.php";
 $loginUsername = $_POST['Input'];
  $LoginRS__query = "SELECT cust_account_number FROM customer
WHERE cust_account_number='" . $loginUsername . "'";
 mysql_select_db($database_details, $details);
            $LoginRS=mysql_query($LoginRS__query, $details) or
die(mysql_error());
 $loginFoundUser = mysql_num_rows($LoginRS);

 //if there is a row in the database, the username was found - can not add the
requested username
 if($loginFoundUser){
  $MM_qsChar = "?";
  //append the username to the redirect page
  if (substr_count($MM_dupKeyRedirect,"?") >=1) $MM_qsChar = "&";
                $MM_dupKeyRedirect = $MM_dupKeyRedirect .
$MM_qsChar ."requsername=".$loginUsername;
  header ("Location: $MM_dupKeyRedirect");
  exit;
 }
}
```

```
function GetSQLValueString($theValue, $theType, $theDefinedValue = "",
$theNotDefinedValue = "")
{
    $theValue = (!get_magic_quotes_gpc()) ? addslashes($theValue) :
$theValue;

  switch ($theType) {
    case "text":
      $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
      break;
    case "long":
    case "int":
      $theValue = ($theValue != "") ? intval($theValue) : "NULL";
      break;
    case "double":
        $theValue = ($theValue != "") ? "'" . doubleval($theValue) . "'" :
"NULL";
      break;
    case "date":
      $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
      break;
    case "defined":
            $theValue = ($theValue != "") ? $theDefinedValue :
$theNotDefinedValue;
      break;
  }
  return $theValue;
}

$editFormAction = $_SERVER['PHP_SELF'];
if (isset($_SERVER['QUERY_STRING'])) {
  $editFormAction .= "?" . htmlentities($_SERVER['QUERY_STRING']);
}

if ((isset($_POST["MM_insert"])) && ($_POST["MM_insert"] ==
"form1")) {
  $insertSQL = sprintf("INSERT INTO customer (cust_account_number,
surname, last_name, middle_name, gender, title_id, address, city, `state`,
```

```
zipcode, phone, birth_date, mother_maiden_name) VALUES (%s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
                GetSQLValueString($_POST['Input'], "int"),
                GetSQLValueString($_POST['textfield4'], "text"),
                GetSQLValueString($_POST['textfield6'], "text"),
                GetSQLValueString($_POST['textfield5'], "text"),
                GetSQLValueString($_POST['select3'], "text"),
                GetSQLValueString($_POST['select2'], "int"),
                GetSQLValueString($_POST['textfield7'], "text"),
                GetSQLValueString($_POST['textfield10'], "text"),
                GetSQLValueString($_POST['textfield11'], "text"),
                GetSQLValueString($_POST['textfield12'], "text"),
                GetSQLValueString($_POST['textfield14'], "text"),
                GetSQLValueString($_POST['textfield23'], "text"),
                GetSQLValueString($_POST['textfield19'], "text"));

  mysql_select_db($database_details, $details);
  $Result1 = mysql_query($insertSQL, $details) or die(mysql_error());

  $insertGoTo = "customersdetails.php";
  if (isset($_SERVER['QUERY_STRING'])) {
    $insertGoTo .= (strpos($insertGoTo, '?')) ? "&" : "?";
    $insertGoTo .= $_SERVER['QUERY_STRING'];
  }
  header(sprintf("Location: %s", $insertGoTo));
}
?>
```

logout.php

```
<?php
//initialize the session
if (!isset($_SESSION)) {
  session_start();
}

// ** Logout the current user. **
$logoutAction = $_SERVER['PHP_SELF']."?doLogout=true";
if                ((isset($_SERVER['QUERY_STRING']))                &&
($_SERVER['QUERY_STRING'] != "")){
  $logoutAction .="&". htmlentities($_SERVER['QUERY_STRING']);
}
```

```
if ((isset($_GET['doLogout'])) &&($_GET['doLogout']=="true")){
  //to fully log out a visitor we need to clear the session varialbles
  $_SESSION['valid_user'] = $last_name;
  $_SESSION['MM_Username'] = $loginUsername;
  $_SESSION['MM_UserGroup'] = $loginStrGroup;
  $_SESSION['PrevUrl'] = NULL;
  unset($_SESSION['MM_Username']);
  unset($_SESSION['MM_UserGroup']);
  unset($_SESSION['PrevUrl']);

  $logoutGoTo = "logout.php";
  if ($logoutGoTo) {
    header("Location: $logoutGoTo");
    exit;
  }
}
?>
```