INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

ONLINE BANKING -

A CASE STUDY FOR DYNAMIC DATABASE-DRIVEN

CLIENT/SERVER SYSTEM

HONG WANG

A MAJOR REPORT

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

JANUARY 2000

Abstract

Online Banking - A Case Study for Dynamic Database-driven Client/Server System

Hong Wang

With the arrival of the age of the Internet, the financial service community is provided with tremendous opportunities and challenges. Today's providers must consider the Internet and on-line services besides their traditional non-online services. In this report, the analysis, design and implementation details of an on-line banking application have been described. Many state-of-the-art technologies were used to build the dynamic database-driven client/server system. These techniques include scripting languages, cascading style sheets, dynamic HTML, active server pages, active data objects, active server framework, design-time controls, ODBC and others. The implementation requires the creation of web site, web pages and links to the database. The database-driven web application was coded using VBScript, JavaScript, Oracle and Microsoft Visual InterDev 6.0. The system functions well and can perform most of day-to-day banking activities.

Acknowledgements

I would like to thank my supervisor, Dr. Lixin Tao, for his great advice and help through the process of doing this project. I would like to express my gratitude to Dr. B.C. Desai for many comments and suggestions on my major report. My family provided support that ensured the completion of this project.

Contents

List of Figures

Chapter 1

1 Introduction

Electronic Commerce (e-commerce) is the buying and selling of goods and services or the transferring money over the Internet or an Intranet. It is the integration of communications, data management, and security capabilities to allow business applications to automatically exchange information related to the sale of goods and services. This can involve stores or banking activities. Communications services support the transfer of information from the originator to the recipient. Data management services define the exchange format of the information. Security mechanisms authenticate the source of information, guarantee the integrity of the information received, prevent disclosure of the information to inappropriate users, and document that the information was received by the intended users. Standards have been established to make the process easier and more secure.

E-commerce differs from traditional commerce in the way information is exchanged and processed. Traditionally, information has been exchanged through direct, person-to-person contact or through the use of the telephone or mail systems. In e-commerce, information is conveyed via a communication network or some other electronic media. In addition, the information accompanying a typical traditional business transaction is usually acted upon by individuals involved in fulfilling the transaction; on the other hand, with e-commerce, much of the transaction processing is automated. In this way, on-line banking and e-commerce can provide the transacting parties with immediate, convenient and secure means by which to transfer financial value.

With the arrival of the age of the Internet, few would argue that e-commerce is simply the most efficient distribution channel from the standpoint of both access and cost. E-commerce will be the backbone of the new global economy in the next century, a marketplace that can be entered from anywhere, which can give access to an unimaginably diverse range of products and services. For the lucky few financial institutions who have mastered e-banking, it will enable much more refined, real-time tailoring of products and services to their customers. It will give them a better ability to develop and price products, and to track market success. The Internet is becoming a familiar, trusted mass medium; and anytime, anywhere electronic banking is a perfect fit for today's and tomorrow's time-pressured lifestyles. Any bank that is not planning to link its customer's accounts to its Web site is gambling with its own survival. Online banking will play a greater part for the consumers of the next generation. As such, online security will have to match the risk of more people using open networks for bank transactions. Internet banking is taking off. The purpose of this project is to explore state of the art technology to develop a dynamic database-driven Web application package of e-banking system.

1.1 Problem Description

This online banking system is designed for financial institutions to deal with their basic banking services. The system allows customers to open accounts, view account balances and statement details, transfer funds between accounts, pay bills and change personal information and passwords. Any customer is able to scan interest details, along with frequently requested information for checking, savings, and certificate of deposit accounts quickly and easily without interacting with bank clerks. The system is

accessible to the potential and current clients with different privilege levels. In addition, multiple users can access it at the same time, rather than personally standing in lines for ATM or banking clerks. The online banking system provides access 24 hours a day to customers' accounts information. Therefore customers can enjoys the convenience of financial controls of their accounts through Internet. The customers can also pay the bills for merchants without writing checks and buying postal stamps. In addition, the customers can move funds between bank accounts in a cost-free way.

According to standard of software industry, the system described above belongs to dynamic Web application. This application is very data-centric. Multiple databases are critical to the operation of the online banking. These databases contain all of the information about account types and details for each account type, and maintain customer records, their account information and account transactions. This enormous amount of information changes very frequently.

Customers can access this information they need via a set of Web pages. The ability to obtain current information via a Web browser is a competitive advantage for on-line banking system. Customers will appreciate the personalized and interactive interfaces that they use to make contact with an online banking. This differentiates online banking from its non-online competitors.

1.2 Tools Used

The online banking system will be developed in Windows NT/98 environment. The following tools will be used for its development:

- Microsoft Visual InterDev 6.0, VBScript and JavaScript: Used for building the web applications.
- Oracle 7.3 or higher: Used as database management to handle data storage, retrieval, management and recovery of all persistent data.
- HTML 4.0: Used for designing web pages.
- ODBC: Serves as the connectivity between the Web server and the database.
- Microsoft Internet Explorer 4.0 or higher: Serves as a web browser to support user interface that drives the display and handles all graphic layouts.
- Microsoft Image Composer 1.5 and Paint Shop Pro 3.12: Used to produce images.

This report is organized as follows: Chapter 2 describes the technology used in the system. Chapter 3 focuses on system requirements and analysis. System designs are documented in Chapter 4. Chapter 5 provides the implementation details, and conclusions are given in Chapter 6.

Chapter 2

2 Technology Overview

2.1 Why is Visual InterDev needed?

The World Wide Web was originally designed to transfer and publish static contents. A user obtained a page by first specifying its Uniform Resource Locator (URL) to a browser, then a Hypertext Transfer Protocol (HTTP) request was sent to the appropriate server. The HTTP response was received containing the Hypertext Markup Language (HTML) source code for that page. Finally the browser interpreted and displayed the information.

The development of new software technologies soon made it possible to provide more interactive experiences for users. An HTML file can refer to a Java applet or an ActiveX control. A client can download and execute the software objects dynamically on the machine. It also became possible to initiate execution of a program on a server when a Web page was requested. The Common Gateway Interface (CGI) was one of the techniques used for this purpose. A CGI program can dynamically generate an HTML response for the browser.

However, a variety of problems had arisen from CGI and other approaches. First, for each HTTP request, a new process had to be initiated, which degraded server performance. Second, it was very complicated to program Web applications by such approaches. This is partly because HTTP is a "state-less" protocol. A connection is not maintained between a client and the server and so each request is independent. Another

issue was that there was no standard way to perform the tasks that were frequently necessary for Web applications, such as dynamic generation of content from server databases.

These problems fostered the need for an integrated development environment. Microsoft Visual InterDev 6.0, a development environment, is designed to solve these problems. It belongs to a member of the Microsoft Visual Studio family of tools and is used to create and manage dynamic Web applications quickly. Multiple tools are included which enable the rapid design, development, testing, deployment, and maintenance of dynamic Web applications. The major features of Visual InterDev include site and page design, database integration, database design and management, debugging, site management, and team development. It is powerful and can be used to create database-driven Web applications, design the structure and appearance of a site, build Web pages, debug client and server scripts, maintain the integrity of a site, and operate in a team environment.

2.2 Technologies

Now we examine the technologies that are necessary to build the dynamic Web application --- online banking system.

2.2.1 Client Script

A client script is a program that is embedded in a Web page. The code is interpreted by a browser, which provides interactive experience for an end user. Client scripts can be written in scripting languages such as VBScript, JavaScript[2,5,6,10] or JScript. Each element on a Web page is represented by an object that may have properties, events, and

methods. This is known as the Dynamic HTML Object Model. Client script can process events generated by these elements.

2.2.2 Server Script

A server script is a program embedded in a Web page that is located in the server side. The code is interpreted by server software and it plays a key role in a dynamic Web application. The Web page containing server script is called an Active Server Page (ASP) file. Server script is never sent to a client machine directly; instead, it dynamically generates HTML and client script that are sent back to a browser. The Active Server Framework (ASF) technology is used to process server script (see Figure 1). Server scripts can be written in scripting languages such as VBScript, JavaScript or JScript.

2.2.3 Active Server Framework

The Active Server Framework (ASF), a key technology to build dynamic Web applications, is a software executed in conjunction with a Web server. Its primary role is to process Active Server Pages (ASP). The framework is available in a web server such as Microsoft Internet Information Server (IIS)[4] or Personal Web Server (PWS)[4]. IIS or PWS is software that is used to process HTTP requests.

Figure 1 depicts the processing of an ASP. The sequence begins when a browser requests a .asp file. That file is read and processed by the Web server. The scripting engine of the ASF processes any server-side script commands. The response to the browser is HTML and client-side script. When a server-side script is processed, it may dynamically construct HTML and/or client-side script and direct the ASF to write this to a browser.

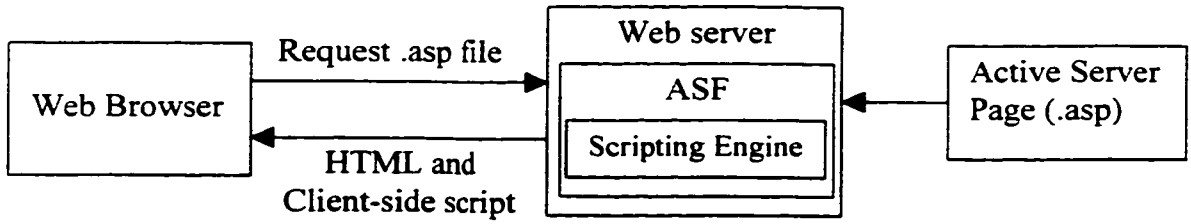


Figure 1: The processing of an ASP by ASF

In the online banking application, some other software technologies are also used, including Cascading Style Sheets (CSS)[1,12], Dynamic HTML (DHTML)[12], Open Database Connectivity (ODBC)[4], Structured Query Language (SQL)[7], Active Data Objects (ADO)[4] and Design-Time Controls (DTC)[4,9].

2.2.4 Scripting Languages

Client scripts and server scripts can be written in scripting languages such as VBScript, JavaScript or JScript. VBScript is a scripting language that was developed by Microsoft Corp. At this time, only Internet Explorer can interpret VBScript client code. Netscape browsers do not support this language. Only Internet Information Server can interpret VBScript server code. JavaScript is a scripting language that was developed by Netscape. Both Internet Explorer and Netscape Communicator support this language. Jscript is a Microsoft's version of JavaScript.

Chapter 3

3 Systems Requirements and Analysis

3.1 User Characteristics

There are two kinds of users for the online banking system: Potential Customers and Current Customers. They may be people or companies. They have different needs and privilege levels to access the system.

- **Potential Customers:** The Potential Customers do not have any accounts at the online banking. A Potential Customer can open an account, which is his/her/its first account at the online banking. After opening the first account, the potential customer became a current customer.
- **Current Customers:** The Current Customers have at least an account at the online banking. They can:
  - ✓ Open additional accounts.
  - ✓ Check balances.
  - ✓ Query transaction history.
  - ✓ Pay bills.
  - ✓ Transfer funds between accounts.
  - ✓ Check or update personal information.
  - ✓ Change password.

3.2 System Access Privilege Control

The potential customers have the lowest access privilege level. They can only look at some services information and open an account. The current customers have a relatively higher access privilege level. A user ID and password are needed whenever a current customer enters the existing accounts and opens new accounts. Once logged in, a current customer can change some information, such as password, address and phone number.

3.3 Functional Requirement Definition

3.3.1 Database Requirements

In order for on-line banking to work, the following files/databases are needed:

Customer File

The customer file contains all relevant information about customers: SIN, name, date of birth, gender, address, phone number, user ID and password. For an individual, SIN, name, date of birth and gender can not be changed once submitted by the customer when opening the first account. For a company, SIN means its registration numbers; date of birth means its creation date; the gender is N/A. This file is used to check or update personal information and change password. It is appended after a potential customer opened the first account and updated after a current customer modified the personal information and changed password.

Account Type File

The account type file contains a list of all account types that the online banking provides, including account type, account description, and interest rate. The online banking provides the following account types: checking, savings, investment, deposit, GIC and students. No customer can modify this file.

Customer Account Information File

The customer account information file has a complete list of accounts already opened by current customers, including account number, open date, balance, SIN and account type for each account. This file is appended after a potential customer or a current customer opened a new account. It is used to check balances and get all account numbers and types held by a customer.

Payment File

The payment file contains all information about customers' transactions, including paying account number, receiving account number, bill number, pay date, due date and paid amount. If account numbers of both paying and receiving accounts belong to the same customer, it means the customer is transferring fund. Otherwise, it means the customer is paying bills. For either transferring fund or paying bills, paying account must belong to the customer. This file is used when customers transfer funds and check balance, view transaction history, transfer funds and pay bills.

3.3.2 Service Requirements

The services include most of day-to-day banking activities by ATM and banking clerks, except withdrawing cash. The services are described as follows:

Open an account: Both potential customers and current customers can open a new account. But there is some difference. Potential customers need to fill out forms for opening an account by providing personal information and account information. Current customers just fill out forms with account information. Their personal information has already been stored in the database, and there is no need to provide personal information again. Before submitting a form, a customer can modify, reset for refilling, and cancel it. Once a customer submits application for an account, the customer will be informed whether the account has been opened successfully. The current customers can update some personal information, such as password, address and phone number.

Pay bills: Customers can select one of their accounts they want to use for bill payment. This is paying account. Customers can input amount they want to pay for each bill. The current date will be displayed in the columns of pay date and due date automatically. The customer do not need to input the pay date since it is set as the current date and the customer can not modify it. For due date, the current date illustrates the input format of due date. It is also convenient for the customer to modify since it is likely that the customer just changes day rather than year and month. A customer can input amount to be paid, whom a bill should be paid for and bill number. The balance will be reduced from the paying account. Meanwhile, the balance will be increased for the receiving account.

Transfer funds: A customer can transfer money between his/her/its accounts. Before submitting transfer requests, the customer needs to select where funds are transferred from and to, and transfer amount. The transfer date is automatically displayed as the current date. The balance will be reduced from the transfer-from account. Meanwhile, the balance will be increased for the transfer-to account.

Account summary: A customer can get an account summary for his/her/its all accounts, including account number, account type and balance. The accounts' changes can be reflected immediately.

Transaction history: A customer can get transaction history for his/her/its all accounts. This reflects all transactions performed during some last period, for example, last week, half of month, one month. Transaction history includes transaction date, account number of transaction-from, account number of transaction-to and transferring amount. His/her/its accounts' changes can be reflected immediately.

Service for changing information: Customers can modify some information, such as address, phone number and password. For password change, the old password should be provided by the customer.

3.4 Non-Functional Requirements

Non-functional requirements define the system properties such as its reliability and constraints. For online banking system, the non-functional requirements can be classified into two categories: product non-functional requirements and external non-functional requirements.

3.4.1 Product non-functional requirements

3.4.1.1 Computer Hardware and Software Requirements

The online banking system should perform all its functionality efficiently on the following hardware and software environments:

- **Server side:**
  1. Minimum hard disk space: 1 GB
  2. Minimum RAM: 64 MB
  3. An IBM compatible Personal Computer (PC) with CPU of Intel Pentium 200 or higher
  4. Operating System: Microsoft Windows NT 4.0 or higher, Microsoft Windows 95/98
  5. Database Management System (DBMS): Oracle 7.3 or higher

- **Client side:**
  1. Minimum hard disk space: 500 MB
  2. Minimum RAM: 16 MB
  3. Any IBM compatible Personal Computer (PC) with CPU of Intel Pentium 200 or higher CPU can run the required software efficiently
  4. Operating System: Windows 95/98, Windows NT 4.0 or higher
  5. Minimum modem connection rate: 14400 bps (bits/per second)
  6. Internet Browser: Microsoft Internet Explorer 4.0 or higher

3.4.1.2 Performance Requirement

The online banking system should be a multi-user system, which means multiple different users can access the system concurrently and the system can still work correctly and efficiently.

3.4.1.3 Efficiency Requirement

In general, the average response time of the system should be less than thirty seconds. However, in the case of a very complicated query that demands a lot of calculations and joining tables, the maximum respond time is set to five minutes.

3.4.1.4 Usability Requirement

The interfaces of the system should be designed to be clear, simple and east to use and understand. To increase the user friendliness, on-line help should be built into the system.

3.4.1.5 Robustness Requirement

In any natural disasters such as a sudden loss of electricity, the system should not lose any information or data. The system should maintain the integrity of the database all the time.

3.4.1.6 Reliability Requirement

The system should be reliable to the users and its failure rate should be less than 0.05%.

3.4.2 External non-functional requirements

3.4.2.1 Safety Requirement

To insure the safety of the system, identification facilities must be implemented in order to have different levels of privileges to access the system. In this way, unauthorized access to the system can be prevented.

3.4.2.2 Privacy Requirement

The private information of each individual stored in the system should be secured. Only the owner of data is allowed to access the information. User ID and password are designed to achieve this goal.

Chapter 4

4 Systems Design

4.1 Design Rationale

The following key issues are considered in the design of Online Banking System:

- **Choice of Architecture**

The online banking system has to respond to asynchronous events from the customers and the database. The customers are allowed to open accounts, check balance, view transaction history, pay bills, transfer funds, change address and password thus to interact with the system. The dynamic nature of the various objects such as account, payment, transaction and balance contribute the system's dynamic characteristics. A combination of interactive and dynamic architecture best addresses these requirements.

- **Scalability**

One of the key issues for any online system is its ability to scale up. The online banking system should be capable of handling increasing number of users with minimum effect on the response time. By maintaining a set of *active objects* on the client machine, the *response time* is vastly improved as most of the user requests are served locally. At the same time the online banking system could handle more number of connections in a given time. The overheads involved in such client side processing are negligible compared to the features it provides.

- **Concurrency**

The online banking system receives asynchronous events that need to be addressed simultaneously. The system is divided into suitable independent sub-systems to handle such concurrent events.

- **Dynamic Update of Data**

Data displayed on the client machine has to be continuously updated as new events are recorded on the database. The *Event Manager* that receives the "pushed" event from the database in turn interacts with the *Account Controller* to incrementally refresh the displayed information on the *User Interface.*

- **Consistency of data**

One of the key issues in the design is the need to maintain consistency of data between the Client, Server and the Database. The *Account Controller* and *Event Manager* co-ordinate to maintain consistency of data across the system.

- **Persistence**

In an effort to provide fault tolerance to the online banking system all active transactions are stored and continuously updated in the database. In case of a server crash, the transaction objects with their latest balance could be retrieved.

4.2 Implementation of Associations

We identified five sub-systems for the online banking system: User Interface subsystem, Event Handler subsystem, Client Banking subsystem, Server Banking subsystem, Database subsystem. In this section, we will discuss their associations.

4.2.1 Association between Subsystems

- **Two-way association between the *User Interface subsystem* and the *Event handler subsystem*:** *Icon* has a pointer to the *User Input handler.* And the *Display Handler* has *pointer references* to all *views* in the *User Interface subsystem.*
- **Two-way association between the *Event handler subsystem* and the *Client Banking subsystem*:** It is implemented by providing *pointer references* in the *Account Handler, Payment Handler* and *Customer Handler* to the *Account Controller.* The *Account Controller*, in turn, has a *pointer reference* to the *Display Handler.*
- **Two-way association between the *Client Banking subsystem* and the *Server Banking subsystem*:** This is implemented by providing a *pointer reference* in the *Client Object Manager* to the *Server Object Manager.* And also the *Event Manager* has a pointer reference to the *Account Controller.*

The above associations support the design rationale of supporting concurrency. For examples, when the *Client Object Manager* is requesting new objects from the *Server Object Manager*, the *Event Manager* could be pushing new events to the *Account Controller.*

- **Two-way association between the *Server Banking subsystem* and the *Database subsystem*:** this is implemented by providing pointer references in the *Account Manager, Payment Manager, Customer Manager and Server Object Manager* to the *database.* The Manager Classes are provided with a pointer reference to the database to store and to restore information to/from the database. The *Database* employs

"push" type of event notification to enable instant update of information and maintain perfect data consistency between the client, server and the database at all times. The *Database* has a pointer reference to the *Event Manager* to support this.

The above association enables support to the design rationale of dynamic update of data, object persistence and partly to system scalability.

4.2.2 Association within Subsystems

4.2.2.1 Event Handler Subsystem

All associations in this subsystem are one-way associations. They are:

- **Between *User Input Handler* and the *Account Handler*:** implemented providing a *pointer reference* in the *User Input Handler* to the *Account Handler.*
- **Between *User Input Handler* and the *Payment Handler*:** implemented providing a *pointer reference* in the *User Input Handler* to the *Payment Handler.*
- **Between *User Input Handler* and the *Customer Handler*:** implemented providing a *pointer reference* in the *User Input Handler* to the *Customer Handler.*
- **Between *User Input Handler* and the *Display Handler*:** implemented providing a *pointer reference* in the *User Input Handler* to the *Display Handler.*

4.2.2.2 Client Banking Subsystem

All associations in this subsystem are one-way associations. They are:

- **Between the *Account Controller* and the *Client Object Manager*:** by providing a *pointer reference* in the *Account Controller* to the *Client Object Manager.*

- **Between the *Account Controller* and the *Active objects*:** by providing *pointer reference* in the *Game Controller* to the *Active objects.*

4.2.2.3 Server Banking Subsystem

All associations in this subsystem are one-way relationships. They are implemented:

- **Between Server Object Manager and Account/Payment/Customer Manager:** by providing *pointer reference* in the *Server Object Manager* to the *Account Manager, Payment Manager* and *Customer Manger.*
- **Between the Event Manager and the Customer Manager:** by providing *pointer reference* in the *Event Manager* to the *Customer Manager.*
- **Between the Account Manager and the Account:** by providing a *pointer reference* in the Acc*ount Manager* to the *List of Accounts.*
- **Between the Payment Manager and Payment:** by providing a *pointer reference* in the *Payment Manager* to the *List of Payments.*
- **Between the Customer Manager:** by providing a *pointer reference* in the *Customer Manager* to a *List of Customers.*

4.3 Architecture

Figure 2 illustrates the major components of the online banking system architecture. The key tasks of the online banking system is to handle concurrent events from external users and maintain *consistency* between data resident on the client, server, database and the data displayed to the end user. Note that Open Account View 1 is for the current customers and Open Account View 2 is for the potential customers.

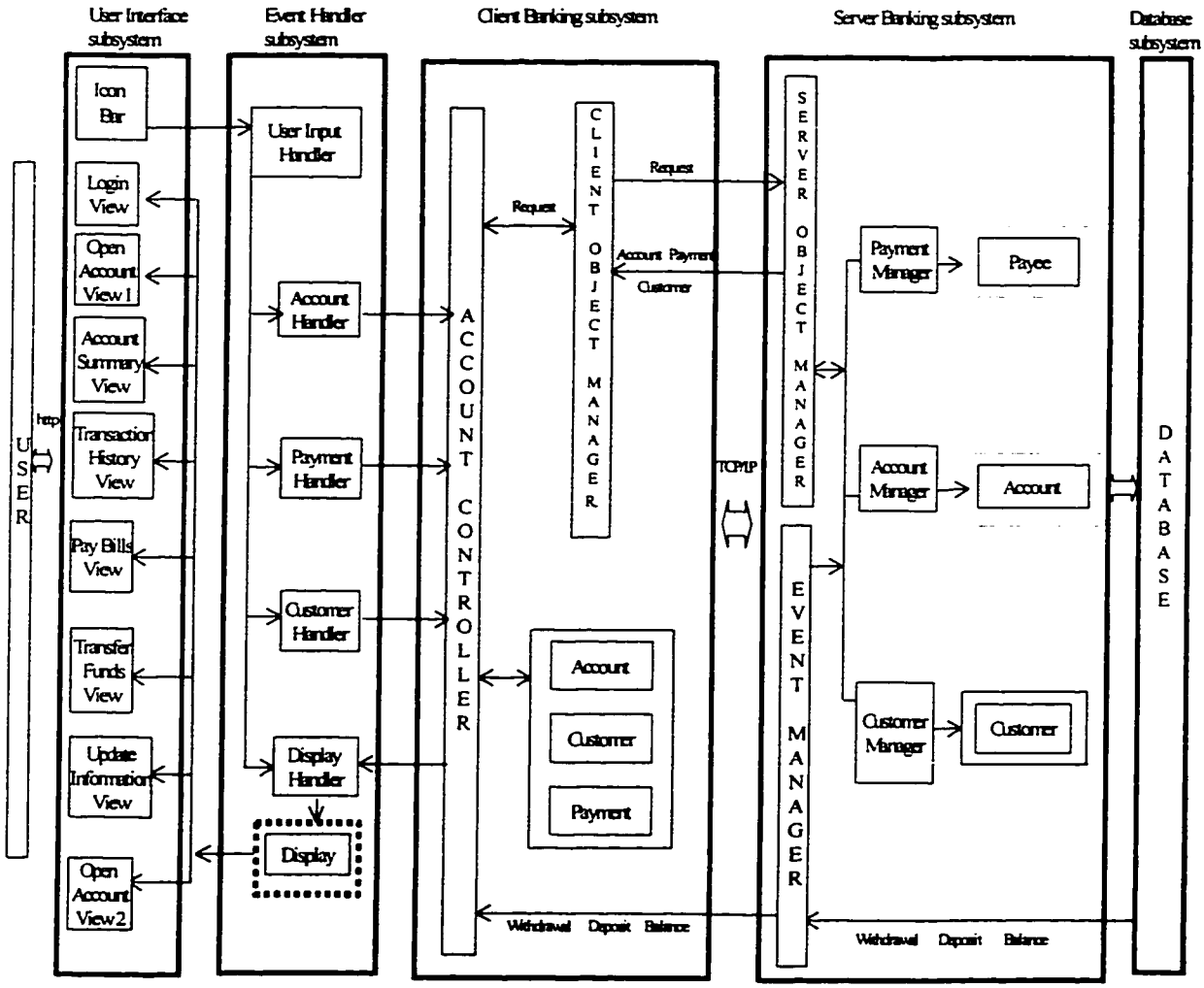


Figure 2: Subsystems with Event and Information Flow

The Online Banking System is a web-based client/server software application system, which will be applied on a distributed computing environment. The technique of layered architecture can provide one basis for geographic distribution. From function and performance viewpoints, specification of functions within distinct layers allows us to keep similar functions closely aligned. Layering design can improve simplicity, understandability and maintainability of the system. We identified the following various sub-systems of the online banking system:

- User Interface subsystem
- Event Handler subsystem
- Client Banking subsystem
- Server Banking subsystem
- Database subsystem

User Interface subsystem provides a user friendly graphical user interface for the customer to provide and retrieve information.

Event Handler subsystem isolates the *User Interface subsystem* from the *Client Banking subsystem*. It separates the physical events from the logical events. This enables much easier debugging during implementation and also during upgrading the system when new features are added.

Client Banking subsystem interacts with the *Server Banking subsystem* to fetch various objects based on user requests. It is also responsible to continuously update the objects resident on the client machine based on the events received from the *Event Manager* which is located in the *Server Banking subsystem.*

Server Banking subsystem serves various clients' requests by fetching objects from the database. It receives *"pushed"* events from the *Database* and updates the objects resident on the *Client Banking subsystem* and *Server Banking subsystem.* For the client side update, the *Event manager* communicates with the *Client Banking subsystem.* And, the *Event Manager* is responsible for the server side update. The *Event Manager* plays a key role in maintaining consistency of data between the Client, the Server and the Database.

Database subsystem serves objects requested by the server. It also *"pushes"* events to the *Event Manager* as new events are recorded on the database.

4.4 System Topology

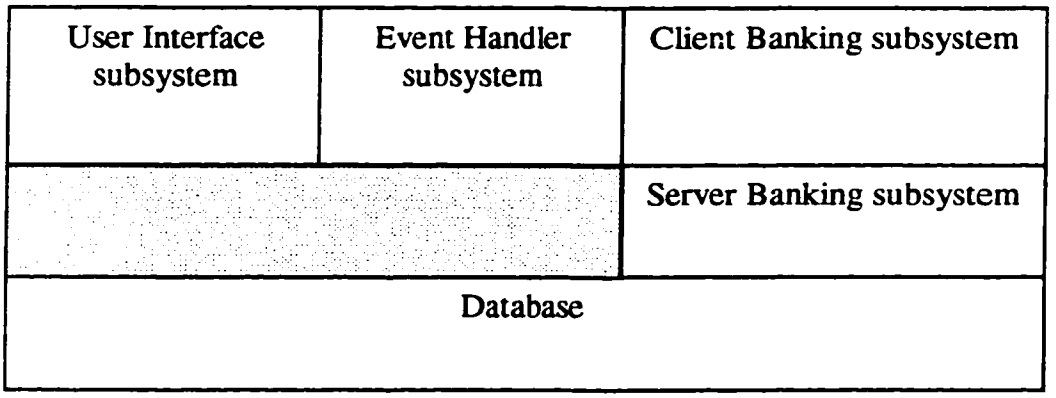


Figure 3: System Topology

Figure 3 displays the layers and partitions of the architecture. The Client / Supplier is denoted by the Client being above the Supplier, while a peer-to-peer relationship is denoted by the subsystems being side-by-side.

The *Event Handler subsystem* serves the *User Interface subsystem* by notifying the *Account Controller* to fetch requested objects. The *Event Handler subsystem* uses the *User Interface subsystem* to display information to the customers. The two sub-systems serve each other to form a peer-to-peer relationship.

The *Event Handler subsystem* and *Client Banking subsystem* have a peer-to-peer relationship. The *User Interface subsystem* is isolated from the *Client Banking subsystem* by the *Event Handler subsystem.*

The *Database subsystem* serves the requests made by the *Server Banking subsystem,* and pushes events as they are recorded on the database. They have Client / Supplier relationship.

4.5 Module Description

This section contains the description of each module in subsystems. It explains the overall function and purpose of that module.

4.5.1 Module in User Interface Subsystem

There is only one module in this subsystem: user interface module.

4.5.3.2 User Interface Module

The User Interface module is composed of all levels of user interfaces on the client site, which are the first and the only means for the user and the Online Banking system to communicate with each other. Inside this module, not only can all interfaces be developed using the same set of tools and be supported by the same interface applications, but also are all interfaces tightly and dynamically inter-related with each other. Any particular sequential combination of a set of user interfaces actually

represents the sequence of user inputs and system responses. Outside this module, from a bird's view of the system, this user interface module as a whole performs the sole function of accepting user input on one side and passing the user message to every module in Event Handler subsystem on the other side or vice versa.

4.5.2 Modules in Application Subsystem

In the online banking system, we can call Event Handler subsystem, Client Banking subsystem and Server Banking subsystem as Application subsystem. Through user interface, user triggers an event by clicking mouse or typing input on keyboard and different modules in Event Handler subsystem, Client Banking subsystem and Server Banking subsystem can handle and response the different fired events.

The Application subsystem contains two major modules as depicted in Figure 4:

1) **The Security Module**: This module performs the security check for user access to the system information and services. It verifies the user input, and passes the checked result to the User Interface subsystem by displaying the different pages according to the different checking result. This module is the bridge between user interface module and other modules.

2) **The Customer Service Module:** This module provides services including opening accounts, checking balances, paying bills, transferring funds, view transaction history, change addresses and passwords. This module provides services for current customers of Online Banking system with user IDs and passwords.

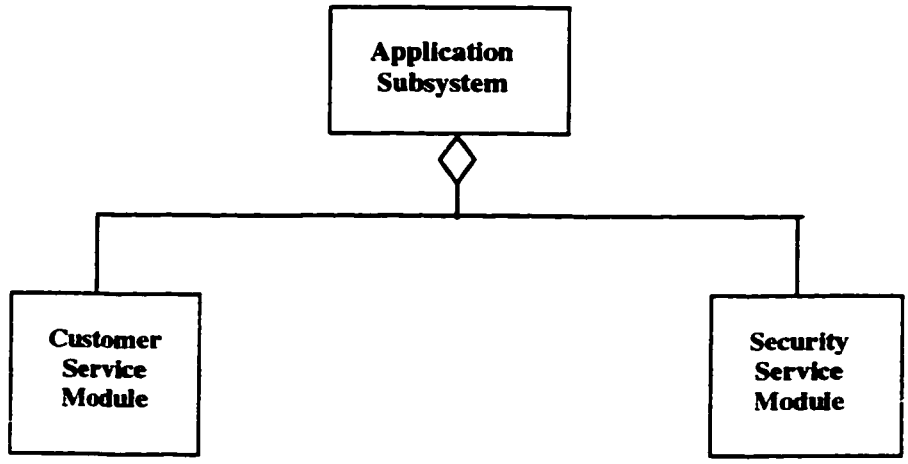


Figure 4: Modular Structure of Event Handler Subsystem

The following sections introduce the internal component organization of these modules in the Application subsystem.

4.5.2.1 Security Module

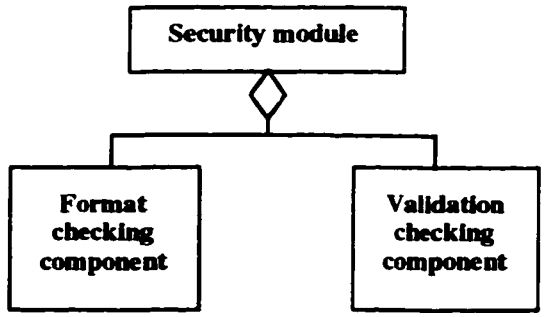


Figure 5: Modular Structure of security module

The Security Module is shown in Figure 5. It is to check user whether has the right to get the services requested. If the services a user asks for need security check, the user interface will show a security check interface, ask user to input user identified ID and password. The user interface passes the message to Security module. There are two phases. In phase I, the format of ID and password are checked without involving database. ID should consist of seven digits and password should consisted of string less than eight characters. If input format is correct, further check involved data stored database is implemented in phase II. If the user passes through the security check correctly, the security module passes the control to other module corresponding to the service user requested. Otherwise, the security module will pass a message to user interface, and inform that the user has no right to get the services.

4.5.3.4 Customer Service Module

The internal architecture of the customer service module is shown in Figure 6. This module is composed of two components:

1. The customer information component. This component processes user requests of changing telephone number, address and password.
2. The customer banking component. This component is further composed of four sub-components:
    - Open accounts
    - Pay bills
    - Check balances
    - Transaction history

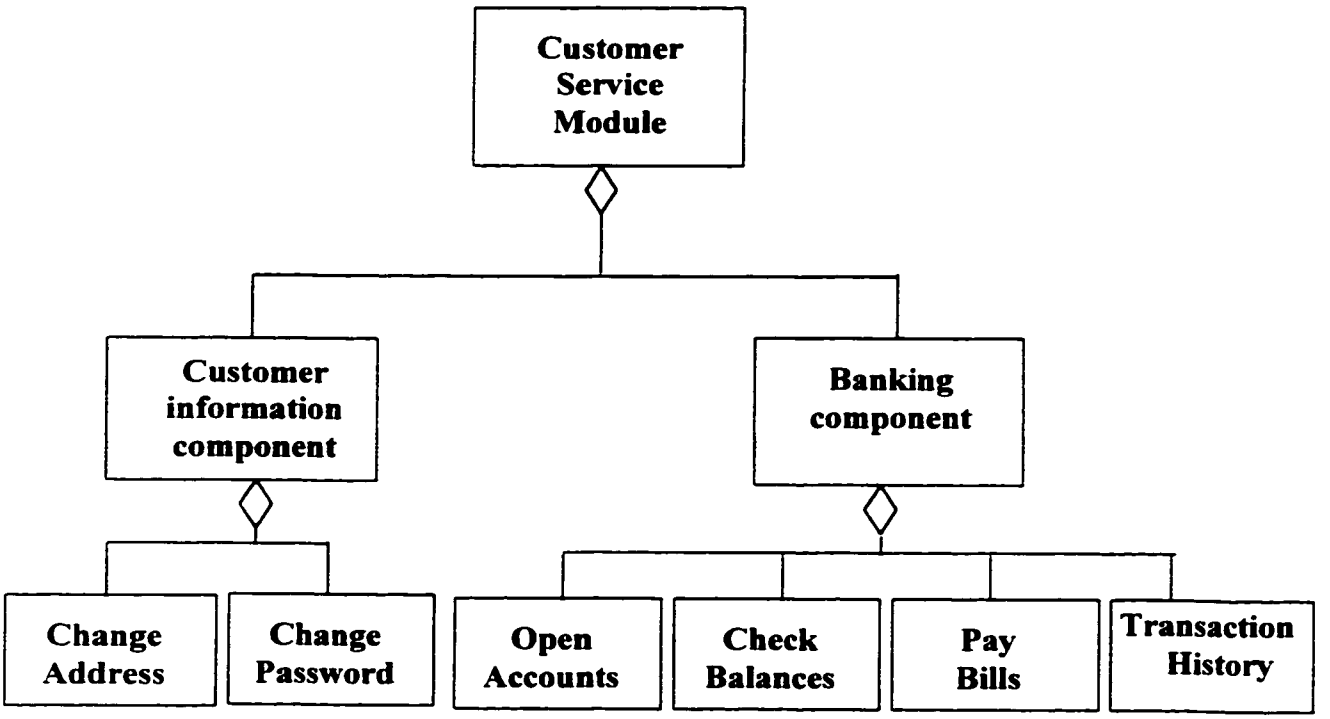


Figure 6: Customer Service Module

4.5.3 Database Subsystem

The database subsystem stores all the data that the system can provide to its users and the inter-relationship between these data as shown in ER diagram in Figure 7. Upon user requests, the Event Handler subsystem may accordingly search database for information, update data, or save new data into the database through Client Banking subsystem and Server Banking subsystem. The Database subsystem is connected to Server Banking subsystem by ODBC as middle ware. Through the query it can get data from or save data to database. The query is attached with each component if that component need to access database.

4.5.3.1 Entity-Relationship Model

The E-R diagram for the database is showed in Figure 7. We can have a relatively clear view of the entire entity sets and the relationships among them. There are four tables all together. Three basic tables store the general information of the bank and one relationship table connects the three basic tables.

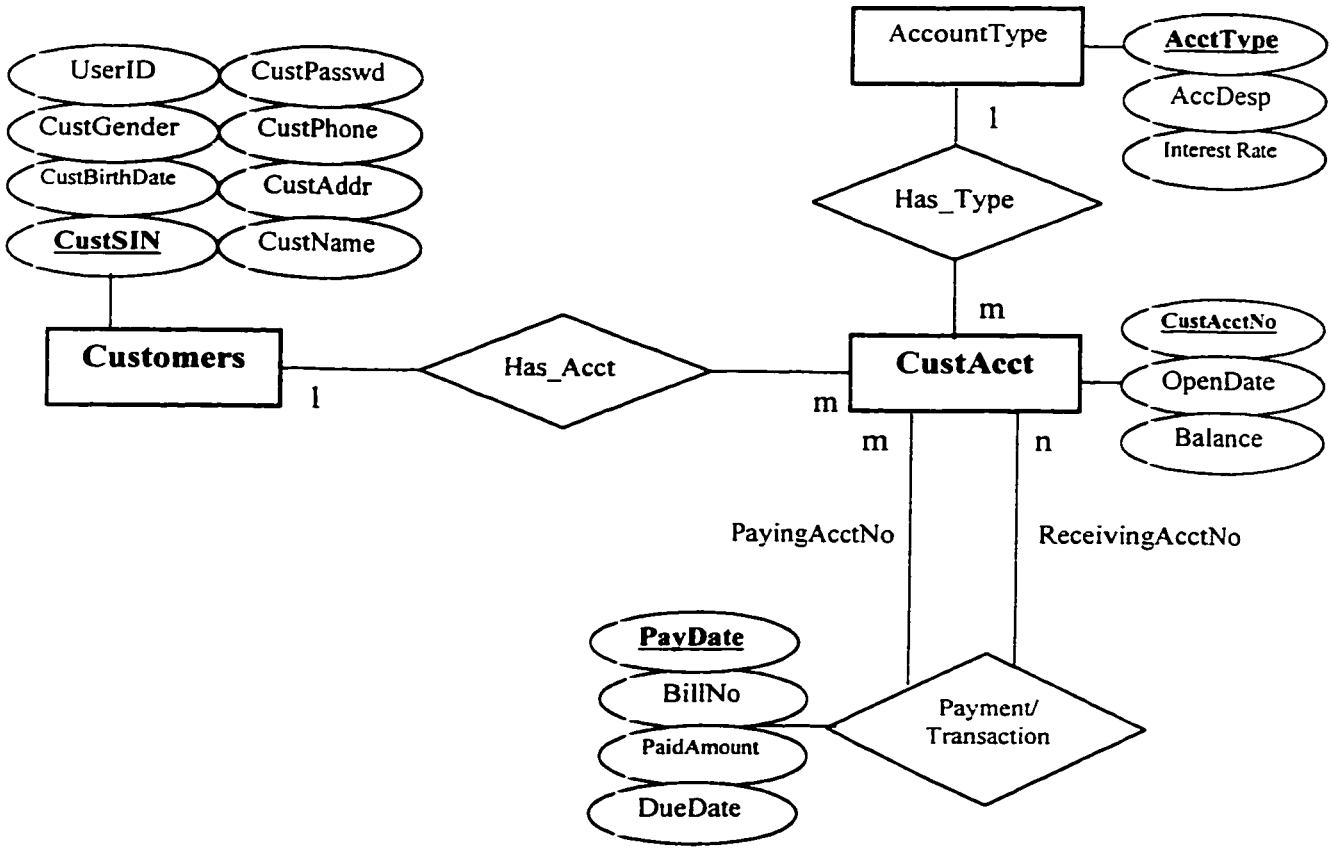


Figure 7: E-R Model

4.5.3.2 Assumptions

The database schema is designed under the following assumptions:

- A customer is identified by a SIN. Every customer must supply his/her/its name, date of birth, gender, address, and phone number. Once a customer supplied his/her/its SIN, name and date of birth and gender, the customer can not change this information himself/herself/itself.
- Each customer can have several different kinds of bank accounts. For each kind of account types, a customer can not have more than one account number.
- Each account is held by one individual. There is no joint account.
- No monthly charge for any service.

4.5.3.3 Database Schema

The notations for the database schema are:

- ❖ Primary Keys: in bold and underline.
- ❖ Foreign Keys: in bold, italic and underline.

Entity Tables:

1 **Customers** (**CustSIN**, CustName, CustBirthDate, CustGender, CustAddr, CustPhone, UserID, CustPasswd)

A customer can not change CustSIN, CustName, CustBirthDate and CustGender.

2 **CustAcct** (**CustAcctNo**, OpenDate, Balance, ***CustSIN***, ***AcctType***)

Where Primary key (**CustAcctNo**),

Foreign key (CustSIN) references Customers (CustSIN),

Foreign key (AcctType) references AccountType (AcctType).

3 **AccountType** (**AcctType**, AccDesp, InterestRate)

Relationship Table:

4 **Payment** (***PayingAcctNo, ReceivingAcctNo***, BillNo, PayDate, DueDate, PaidAmount)

Where Primary key (**PayingAcctNo, ReceivingAcctNo, PayDate**)

Foreign key (PayingAcctNo) references CustAcct(CustAcctNo),

Foreign key (ReceivingAcctNo) references CustAcct(CustAcctNo),

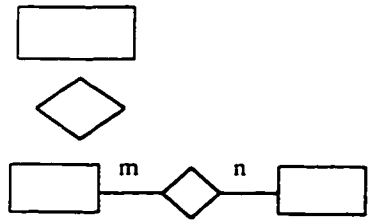
Check (PayingAcctNo <> ReceivingAcctNo).

4.5.3.4 Integrity Constraints

The referential integrity constrains are taken into account in the following cases:

- When an attribute is added, modified or deleted from an entity, all the associated relations should be updated if necessary. For example, if a new customer is approved to open an account, once his/her/its personal information is recorded in the **Customers** table, this information and all account information must be added in the **CustAcct** table.
- If a relation R refers to a relation S via a set of attributes which forms the primary key of S and the foreign key of R. Then, for a record in R, the value of the foreign key must be equal to the primary key of a record in S. For example, relation **CustAcct** refers to **Customers** via the attribute ***CustSIN*** that is the primary key of **Customers** and a foreign key of **CustAcct**. Then the value of *CustSIN* in **CustAcct** must be equal to the value of *CustSIN* in **Customers**.
- In **Payment** table, *PayingAcctNo* can not be equal to *ReceivingAcctNo*. That means that transferring funds at the same account is not permitted.

Note:



Represents an entity

Represents a relationship

Represents a m-n relationship

The attributes in italic are the primary key.

The word in bold represents an entity.

Chapter 5

5 Implementation

5.1 Project Setup

First, create a Web project for this online banking application. The project name is **OraScottPro**. Then, add a data connection **ConnectOraScott** for the data source named **OraScottDSN** to the project. The data connection is located in the DataEnvironment of global.asa of the project OraScottPro.

5.2 Site Diagram

A site diagram provides valuable information about the structure of a Web application. It defines the navigation relationships among a set of pages and specifies the pages that appear on the global navigation bar. Figure 8 illustrates the site diagram containing the pages for the online banking application.

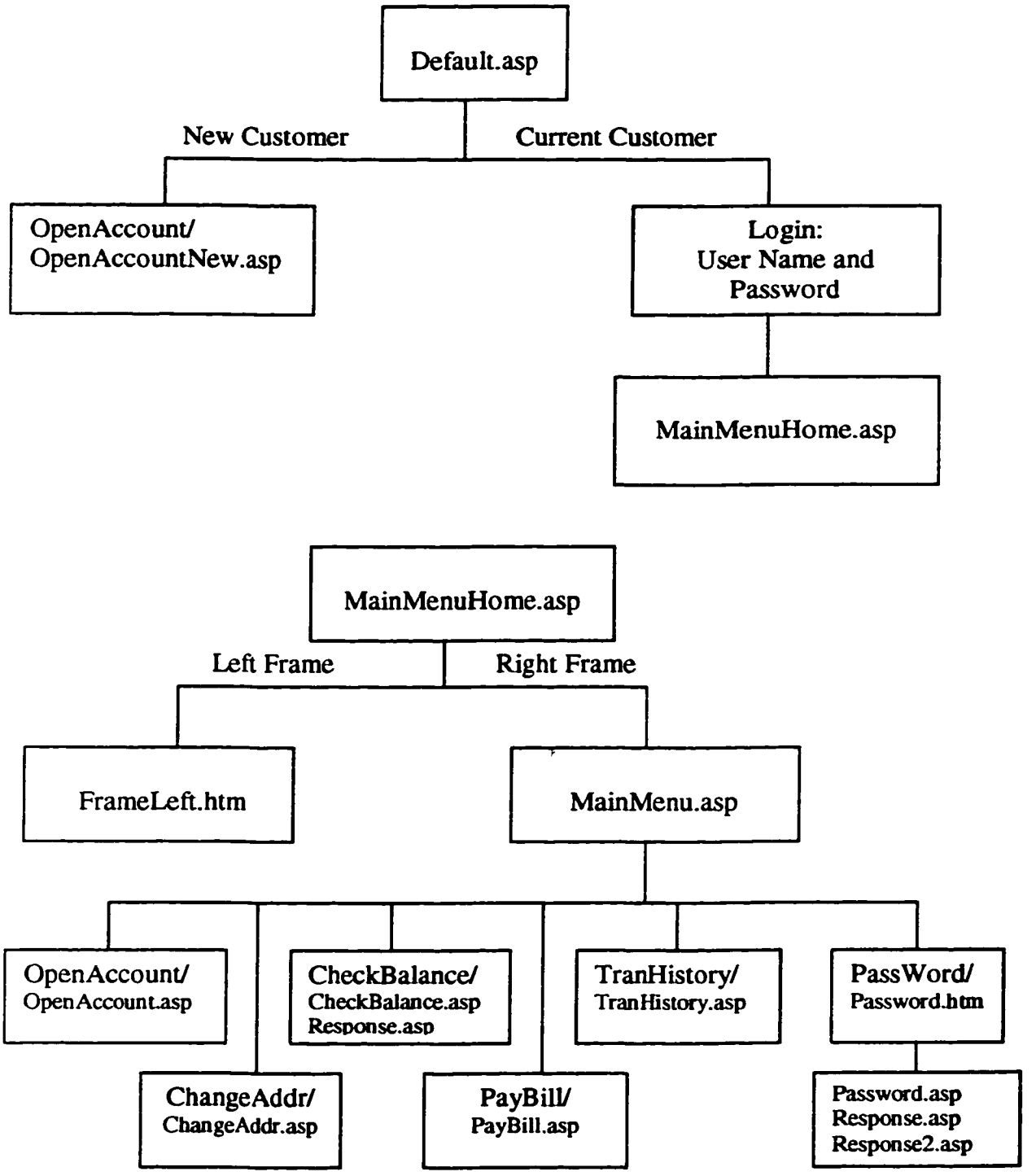


Figure 8: Site Diagram

5.3 Pages Specifications

5.3.1 Home Page

File Name: Default.asp

Function:

Welcomes users and lets users browse the home page of the online banking application.

Description:

Once a user connects to the online banking through the Internet, the home page will appear on the screen as shown in Figure 9. There are two options, "Open an account" and "Login" for a new customer and a current customer, respectively. A potential customer can use the mouse to click on "Open an account". A current customer can use the mouse to click on "Login" after inputting User ID and Password. Then, the system checks if information given by the user is correct. Before login, the user can "Clear" to input again or exit. After a correct login, the system will permit the user to use authorized options. Otherwise, a re-login is required.

Input:

The user uses the mouse to click on one of the possible options on the user's screen. For current customers, they need to provide their user IDs and passwords.

Output:

The system will pop up another window on the screen to show the user one corresponding Web page. For a new customer, it links the user to the hyperlink Figure 17. For a current customer, it links the user to the hyperlink Figure 10. Or error messages are given for incorrect login.

Constraints:

Among two options, the potential customers can access "Open an account" option. The current customers, who have user IDs and passwords, can login to access more options.

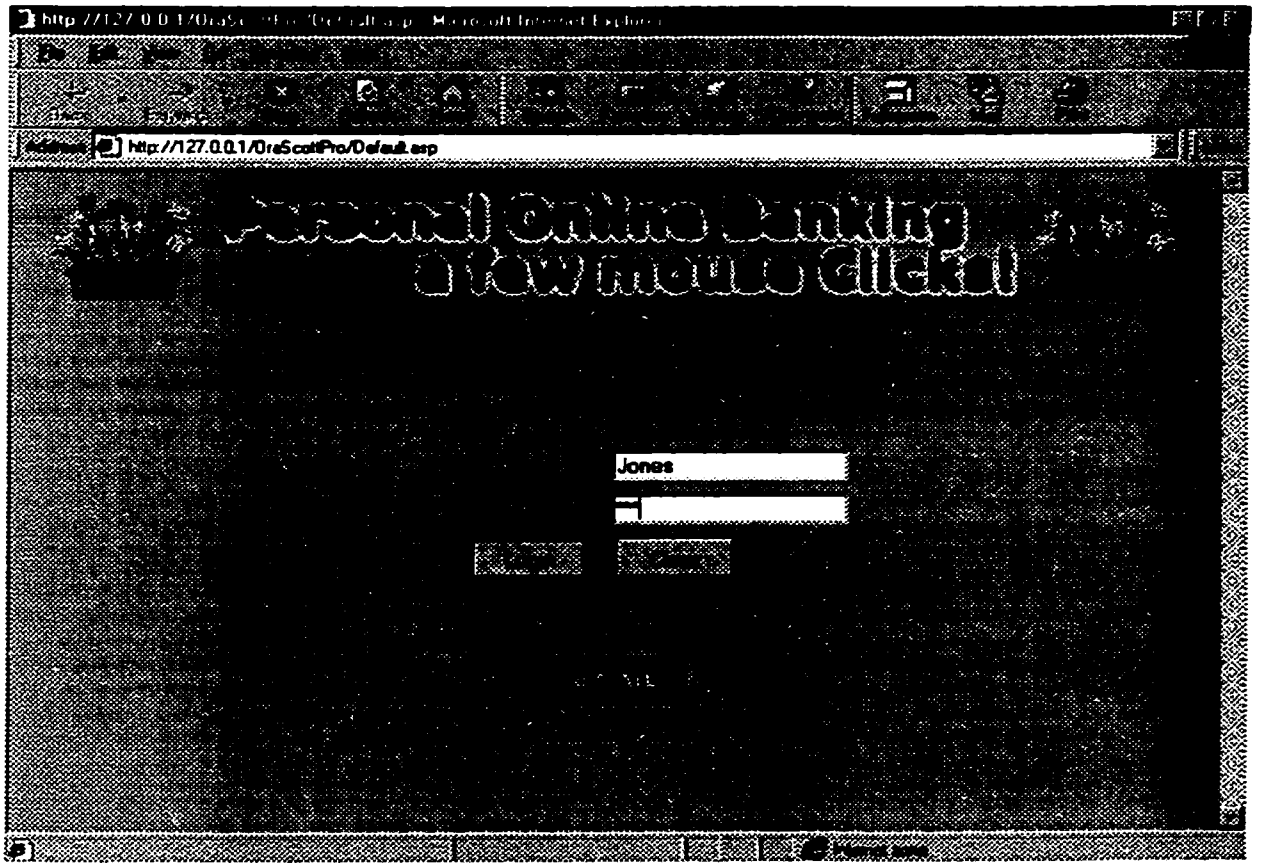


Figure 9: Home Page for Online Banking

5.3.2 Main Menu for Current Customers

File Names: MainMenuHome.asp, FrameLeft.htm, MainMenu.asp.

Function:

Lets current customers browse the main menu and selects options for them.

Description:

After a current customer logs in from the home page, the main menu will appear on the screen as shown in Figure 10. There are eight options: **Home, Open Account, Pay Bills, Check Balance, Transaction History, Change Address, Change Password, Log Out.** A current customer can use the mouse to click on one of options. There are dynamic advertising banners for commercial purposes.

Input:

The user uses the mouse to click on one of the possible options on the user's screen at the right and left sides of the screen.

Output:

The system will pop up another window on the screen to show the corresponding Web page. The following are some examples of these cases:

Home option: Links the user to this main menu.

Open Account option: Links the user to the hyperlink Figure 11. A form appears for opening an account.

Pay Bill option: Links the user to the hyperlink Figure 12. The user can pay bills and transfer funds. If both paying account and receiving account belong to the same customer, the transfer means transferring money from one account to another account for this customer. Otherwise, the transfer means paying bills and transferring money to other customers' accounts.

Check Balance option: Links the user to the hyperlink Figure 13. A list of available balances for all accounts owned by the user appears, including three columns: account type, account number, current balance.

Transaction History option: Links the user to the hyperlink Figure 14. A list of transactions appears, including date, description, from, to, transferring amount for the selected account in some last statement period.

Change Address option: Links the user to the hyperlink Figure 15. This part permits the user to update address and phone number.

Change Password option: Links the user to the hyperlink Figure 16. This part permits the user to change password.

Log Out option: Links the user to the hyperlink Figure 9. This option lets the user to logout from the main menu and go back to home page.

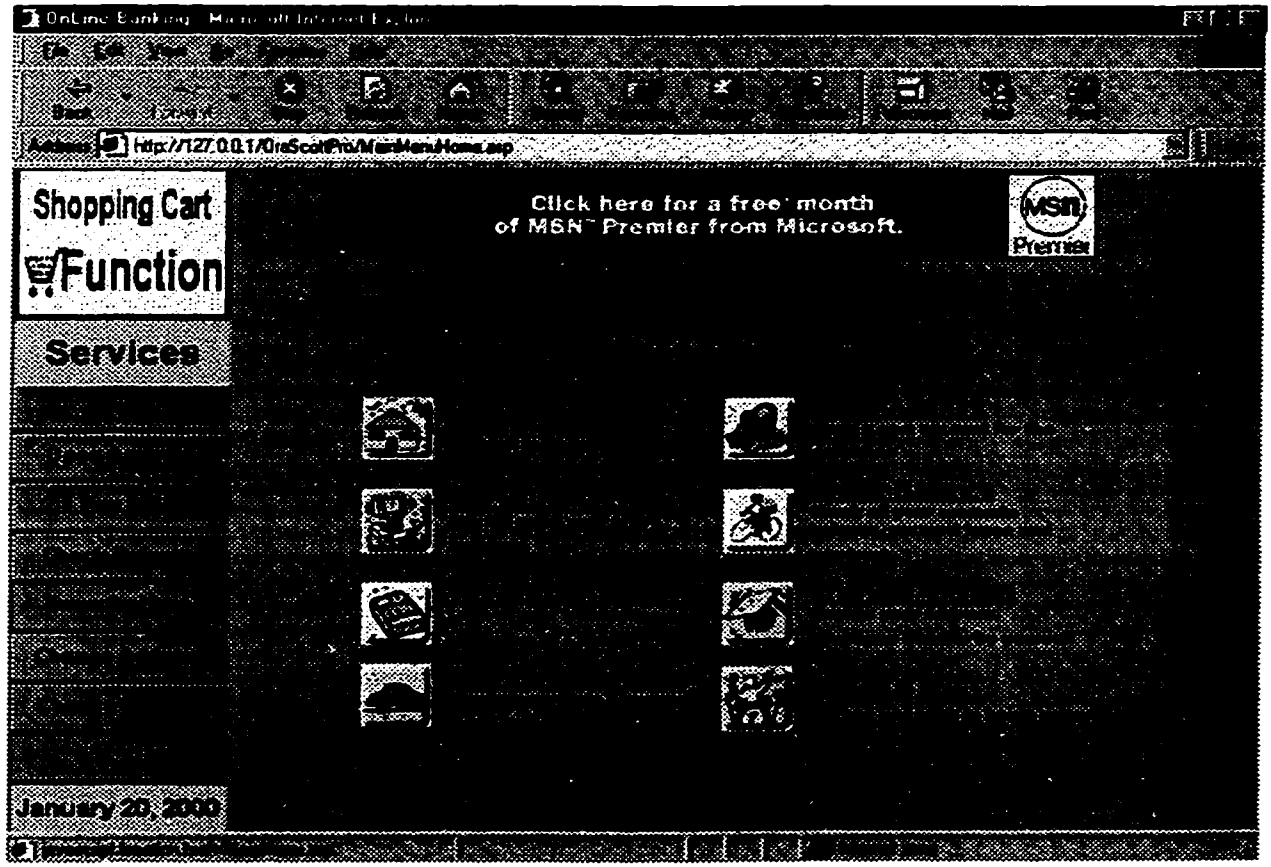


Figure 10: Main Menu for Current Customers

5.3.3 Open an Account for Current Customers

File Name: OpenAccount/OpenAccount.asp.

Function:

Open an account for current customers.

Description:

This Web page gives a form to be filled out by the user as shown in Figure 11. The user can submit the form after filling. During filling the form, they can reset the form, or cancel the application before submission. Before the application can be submitted, the online banking must check if the user fills all necessary information. After submitting the application, confirmation information will be shown.

Input:

The user fills the form and submits it by clicking on "Submit" button. For some information, the user can use the mouse to select one from the list boxes, such as account type. Current customers do not need to input personal information again when they open a new account, since all personal information has already been recorded in the database.

Output:

If all information has been properly filled out, the system will display acknowledge information to the user. Otherwise, error messages will appear on the screen and indicates that more information is needed before submission.

Constraints:

- When an existing customer opens an account, the system checks if the SIN input by the customer is the same as his/her/its SIN stored in the table Customers. First, check if the SIN input by the customer exists; if yes, then check if the user ID and

password input by the customer while logging in are corresponding to that with the same SIN in the table Customers.

- A current customer can not open more than one account for the same type. If the user selects the same account type as that he/she/it has already had, the system will prompt the user to re-select account type.
- The minimum deposit required to open account is $50.00.

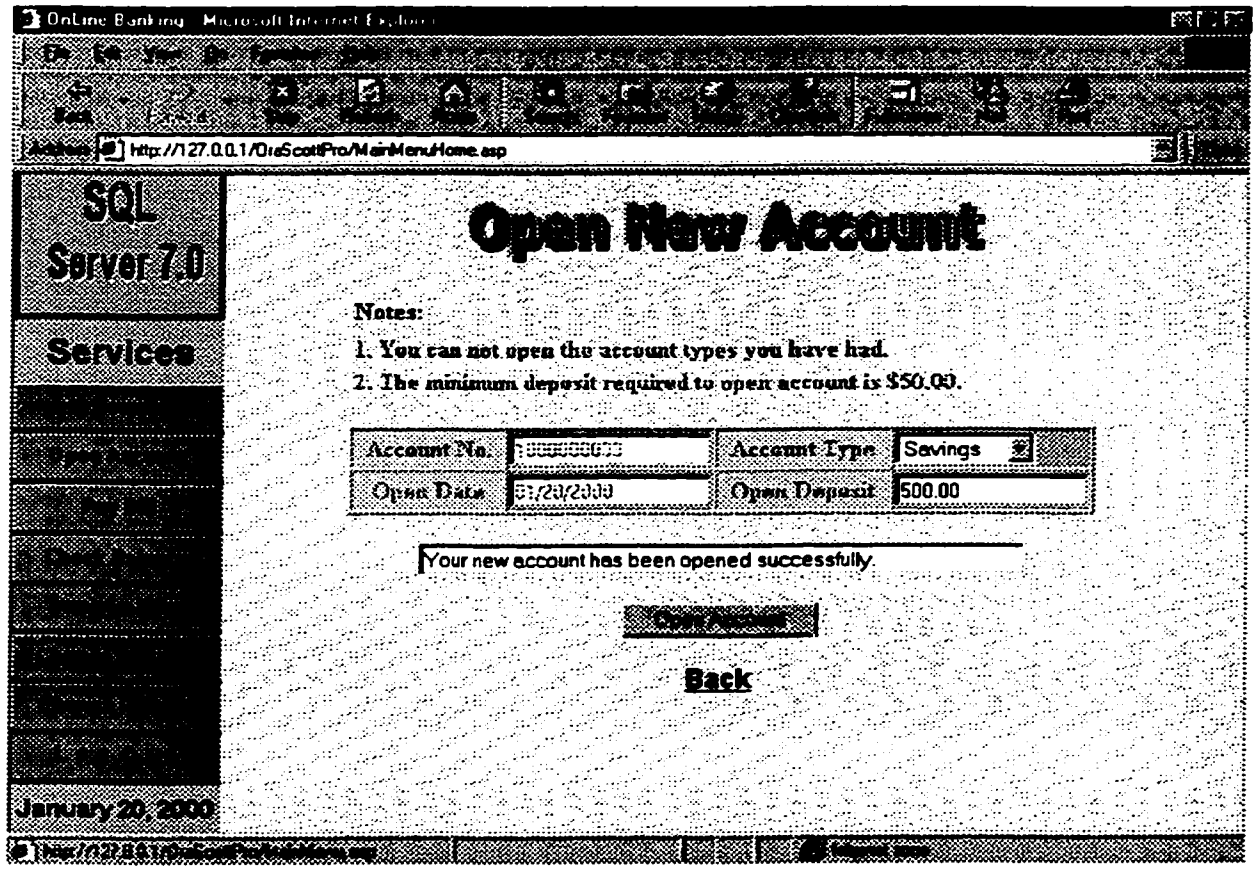


Figure 11: Open an Account for Current Customers

5.3.4 Pay Bill

File Name: PayBill/PayBill.asp.

Function:

Current customers use it to pay bills or transfer money from an account to another account.

Description:

This Web page can give two options "Pay" and "Reset". There is a list box for the user to select a paying account which is owned by the user. Below the list box, there is a form to be filled by the user as shown in Figure 12. For "Pay" option, the user can use it to pay pending bills after filling out the form, including receiving account number, bill number, pay date, due date and amount. He/She/It can reset paying bills by clicking "Reset" button. He/She/It can cancel paying bills by clicking "Back". This Web page also provides the user to transfer money between two accounts. The user needs to select transfer funds from which account and transfer funds to which account and inputs the amount he/she/it wants to transfer. For transferring funds, both the paying account and the receiving account are owned by the same user. The user can submit the transfer after selecting and inputting all information. He/She/It also can cancel the transfer before submission. Before the application can be submitted, the system checks if the user fills all needed information. After submitting the transfer, confirmation information is given.

Input:

The user needs to select a paying account and input receiving account number, due date, bill number and amount. The pay date is displayed as the current date automatically. Then, press "Pay" button to pay, "Reset" button to reset or "Back" to cancel.

Output:

If all information has been properly selected and filled out, the system will display acknowledge information to the user. Otherwise, error messages will appear on the screen and indicates that more information is needed before submission.

Constraints:

- When a customer pays bill, the system checks if current balance in the chosen paying account is larger than the paid amount for the bill.
- When a customer pays bills, the customer must pay them one by one.

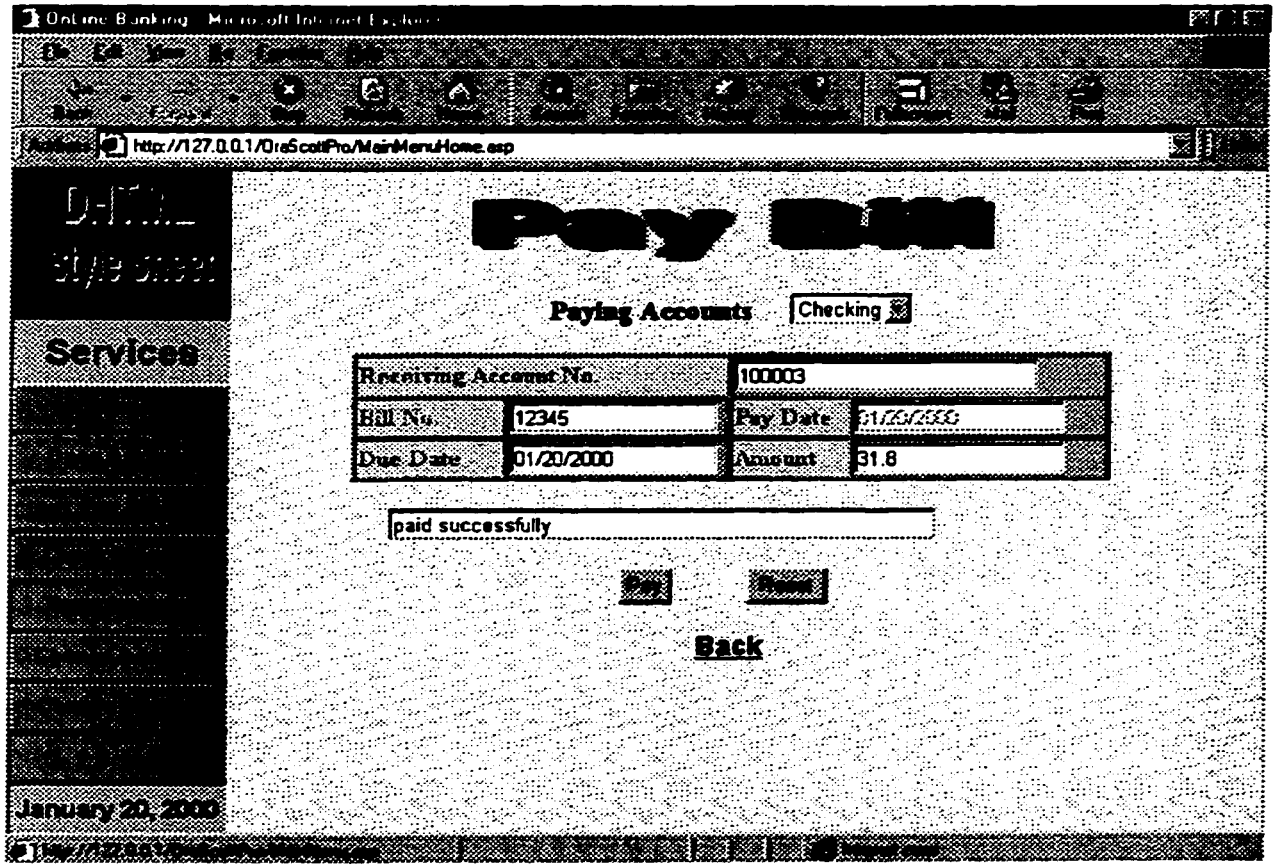


Figure 12: Pay Bills

5.3.5 Check Balance

File Name: CheckBalance/CheckBalance.asp

Function:

Current customers use it to check the current balances of all accounts and generate an account summary.

Description:

This Web page can give a list of available balances for all accounts owned by the user as shown in Figure 13.

Input:

No input is needed.

Output:

The system displays Figure 13.

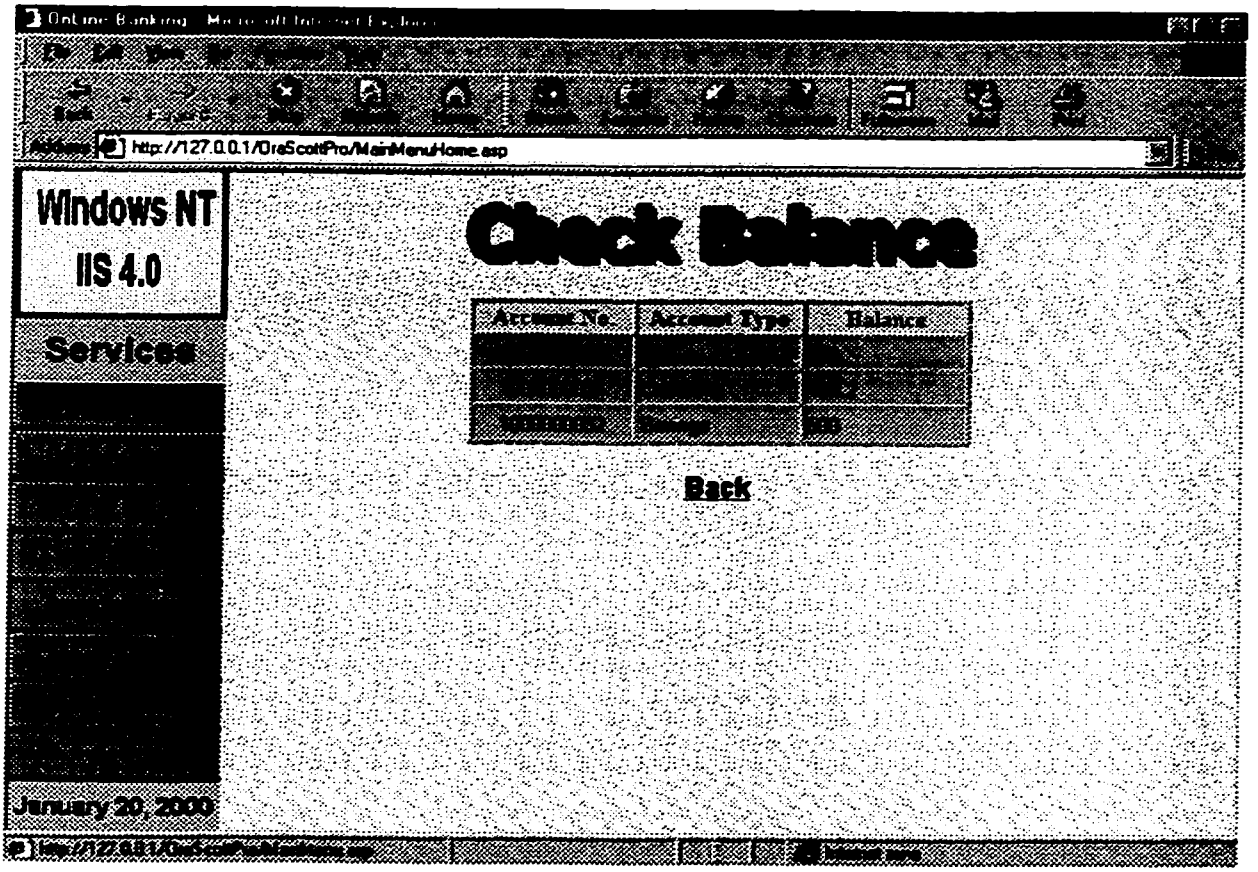


Figure 13: Check Balance

5.3.6 Transaction History

File Name: TranHistory/TranHistory.asp.

Function:

Current customers use it to view transactions for all their accounts for a last period.

Description:

This Web page can give a list of transactions after the user selects an account from a list box of accounts and a statement period, if there is any transaction in the period for the selected account. Otherwise, a message prompts the user that there is no transaction for the period for this account.

Input:

The user press "Submit" button after selecting an account from a list box and a statement period. Before submission, the user can cancel his request by pressing "Cancel" button.

Output:

A list of transactions appears as shown in the Figure 14, including date, description, from, to, amount, if there is any transaction in the period for the selected account. Otherwise, there is a prompt to indicate that there is no transaction for the given account in the selected period.

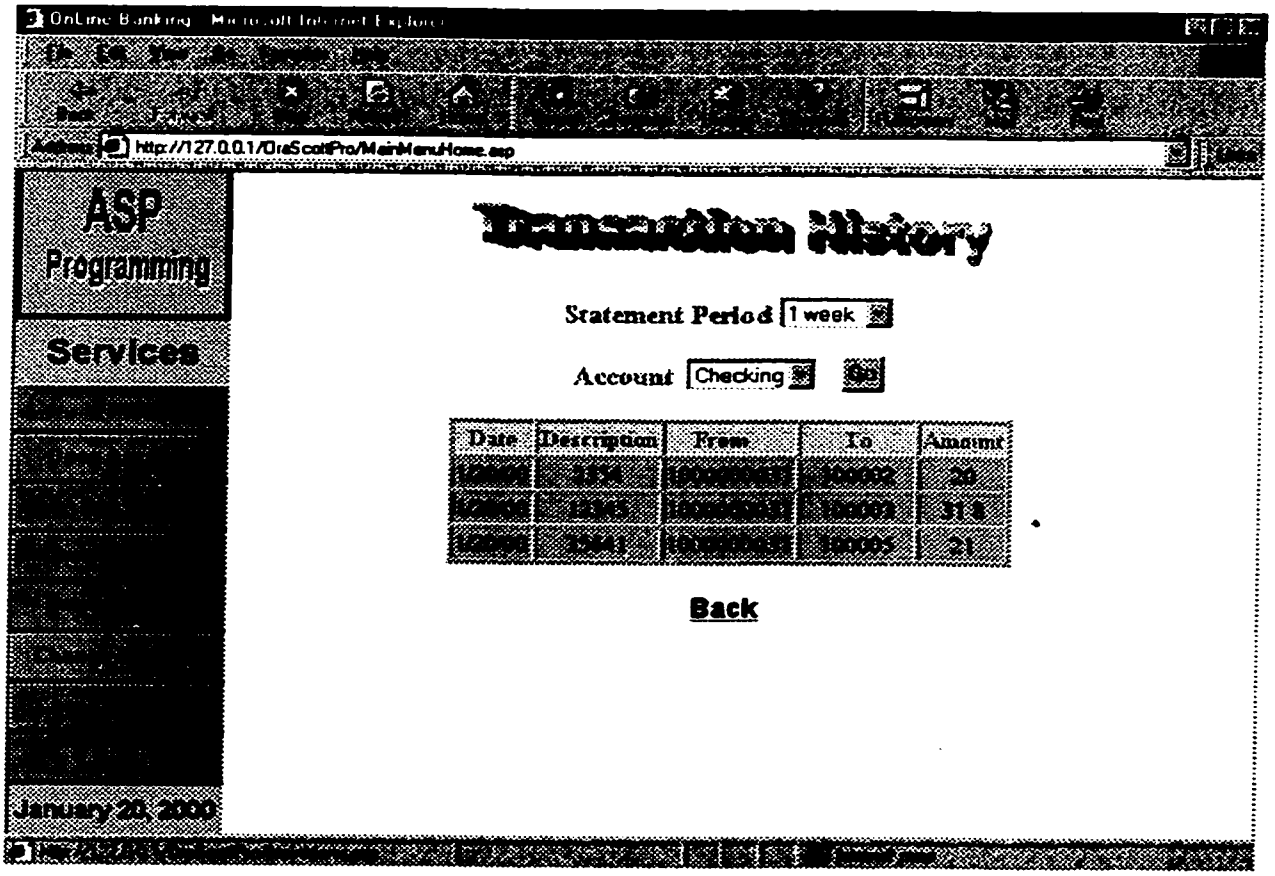


Figure 14: Transaction History

5.3.7 Change Address

File Name: ChangeAddr/ChangeAddr.asp.

Function:

Current customers change their addresses and phone numbers.

Description:

This Web page gives the user to modify his/her/its address and phone number. When the current customers press "Change Address" at the main menu, his/her/its personal information has been displayed in this page as shown in the Figure 15. Based on the login information (User ID and password), the system automatically searches the personal information from the table Customers the given customer. After the user modified the address and phone number, he/she/it presses "Modify" button to change the old address and phone number. Other personal information, including SIN, name, gender and date of birth, is also displayed on this page, but can not be changed. He/She/It also can cancel his/her/its modification before submission by click on "Back". After submitting the modification, confirmation information is provided.

Input:

The user needs to input information to be modified, either address or phone number or both. After input all the information, the user submits by clicking on "Modify" button.

Output:

Confirmation information will be given. Otherwise, an error message will appear on the screen and indicates that more information is needed before modification.

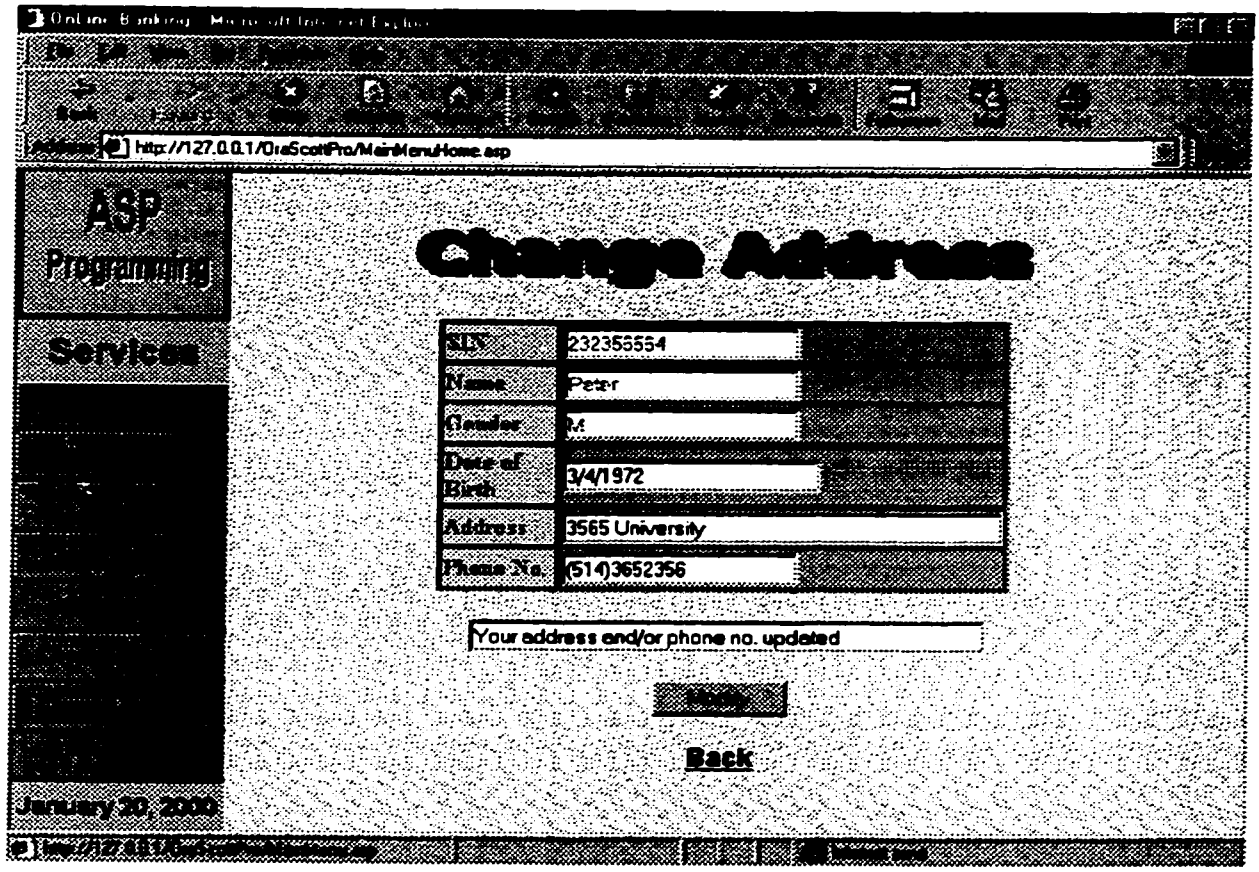


Figure 15: Change Address

5.3.8 Change Password

File Names: PassWord/Password.htm, Password.asp, Response.asp and Response2.asp.

Function:

Current customers change their passwords.

Description:

This Web page provides the user to change his/her/its password. The user inputs the current password and a new password. The new password needs to be re-entered for confirmation. After the user modified the password, he/she/it presses "Change" button to finish the change. He/She/It also can reset the modification by clicking on "Clear"

button and cancel the modification before submission by click on "Back". After submitting the modification, confirmation information is provided.

Input:

The user needs to input information to be modified, including current password and new password. After input all the information, the user submits by clicking on "Change" button.

Output:

Confirmation information will be given. Otherwise, an error message will appear on the screen and indicates that more information is needed before modification.

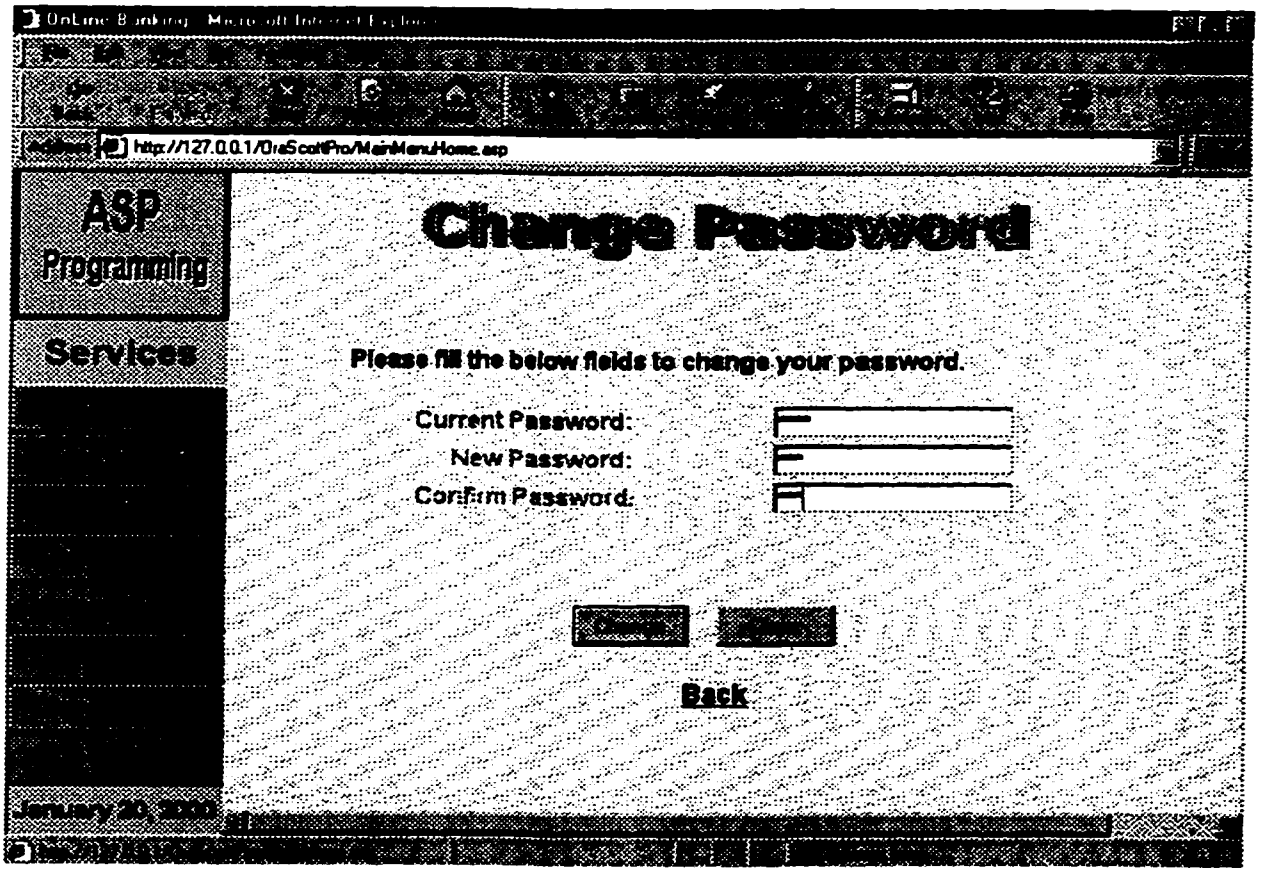


Figure 16: Change Password

5.3.9 Log out

File Name: OraScottPro/Default.asp.

Function:

The current customer uses it to log out and go back to home page of the online banking system.

Description:

When the current customers finished all options they wanted to use, they use "Log Out" to close their files. After they press "Log Out", the system will close the user's account information and go back to home page described in Figure 9.

Input:

No input is needed.

Output:

Home page Figure 9 appears.

5.3.10 Open Account for New Customers

File Name: OpenAccount/OpenAccountNew.asp.

Function:

A potential customer uses it to open an account.

Description:

This Web page gives two forms to be filled out by the user as shown in Figure 17, including personal information and account information. The user can submit the forms

after filling. During filling the forms, the user can reset the forms. He/She/It also can cancel the application before submission. Before the application can be submitted, the online banking must check if the user has filled all necessary information. After submitting the application, confirmation information will be shown to the user.

Input:

The user fills two forms and submits it by clicking on "Submit" button. For some information, the user can use the mouse to select one from the list boxes, (such as including day and month of date of birth (DOB), gender and account type. Before submitting the application, a customer can modify, reset for refilling, and cancel it.

Output:

If all information has been properly filled out, the system will display acknowledge information to the user. Otherwise, error messages will appear on the screen and indicate that more information is needed before submission.

Constraints:

- When a new customer opens an account, the system checks if the re-entered password is the same as the password. If yes, the re-entered password is stored; otherwise, the system requires the customer to input password and re-entered password again.
- When a new customer opens an account, he/she/it is required to choose a User ID for later login. After the customer chooses the User ID, the system checks if there exists the same User ID in the table Customers. Thus, each User ID uniquely corresponds to one customer to increase security.

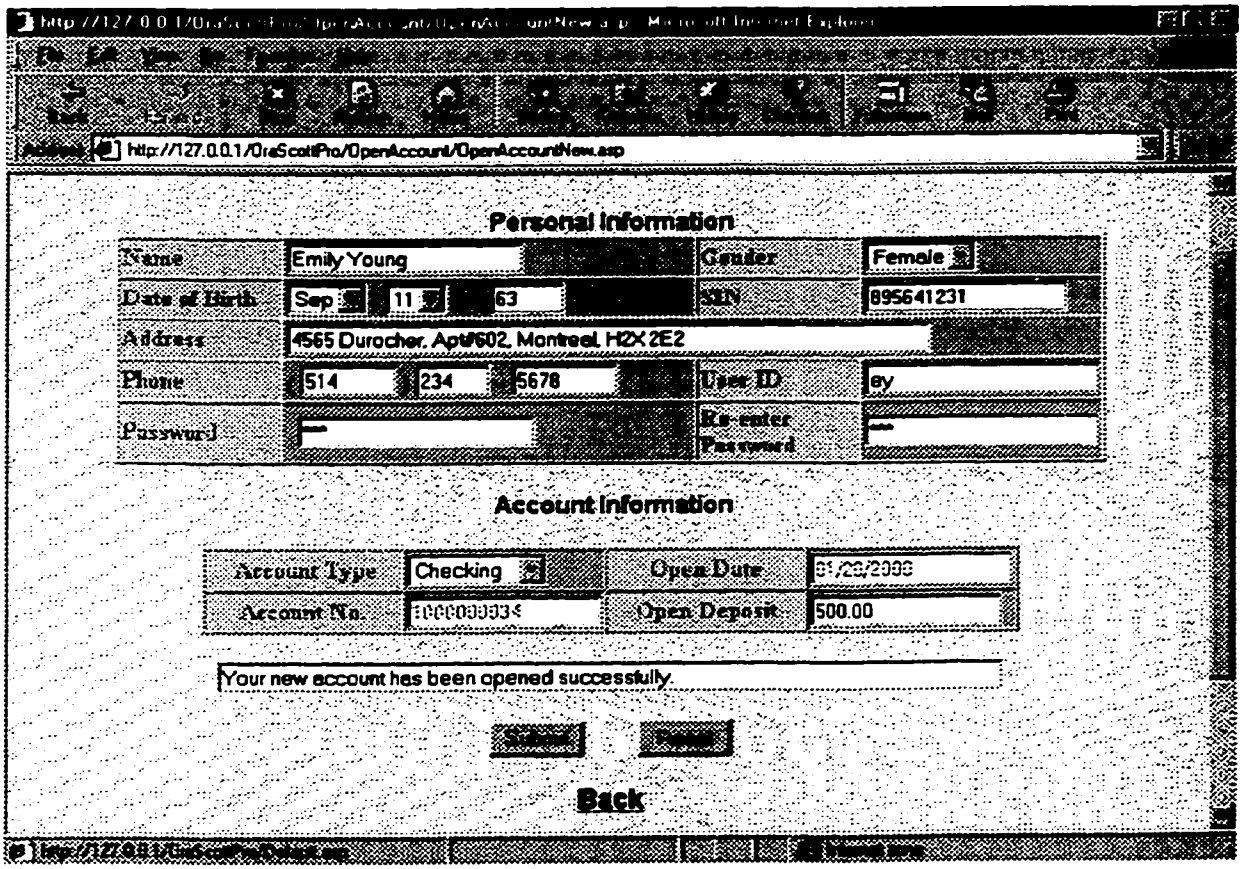


Figure 17: Open an Account for New Customers

5.4 Install procedure for financial institutions

5.4.1 Create ODBC System Data Source

ODBC Data Source Name: OraScottDSN.

Creation Procedure:

1. On the desktop, double click **My Computer.**
2. Double click **Control Panel.**
3. Double click **ODBC.**

4. In ODBC Data Source Administrator window, select **System DSN** and then click **Add** to create a System Date Source.
5. In Create New Data Source window, select the driver **Microsoft ODBC for Oracle** to set up a data source, and then click **Finish.**
6. After 5, in Microsoft ODBC for Oracle Setup window, input **Data Source Name** as **OraScottDSN** and then click **OK.**
7. Exit ODBC Data Source Administrator window to go back to desktop.

5.4.2 Run start file bank.sql under Oracle SQL*PLUS environment

Reference to Appendix C for the file bank.sql. Run the start file bank.sql to set up database by creating tables and views. The user name and password to enter Oracle SQL*PLUS is scott and tiger, respectively. Execute the command under Oracle SQL*PLUS environment:

SQL>start bank.sql

5.4.3 Copy all files for the online banking system under the home directory of the Web server

In order to copy all files for the online banking system under the home directory of the Web server, we need two steps:

1. Create a sub-directory for the online banking system under the home directory of the web server. Under Windows 95/98 environment, the web server is Personal

Web Server (PWS). Under Windows NT 4.0, the web server is Internet Information Server (IIS). For example, under Windows 95/98, the home directory is C:\Inetpub\wwwroot. Create the sub-directory **OraScottPro** under the directory wwwroot.

2. Copy OraScottPro.zip to the directory **OraScottPro**. This .zip file includes all the files for the online banking system. Then, unzip OraScottPro.zip to the directory **OraScottPro**.

Chapter 6

6 Conclusion

We have designed and implemented an on-line banking web application, which is database-driven web-based client/server system. This report documented in details the technology and software used in the system, system requirements and analysis, system design and implementations. User manual and source codes were also provided. The system has been designed to exploit advanced software and modern technology, including the use of standard browsers, Visual InterDev, JavaScript, VBScript, web technology and secure database techniques, which enabled the robustness and security. The whole system functioned well and performed all the designed tasks of banking activities during the demonstration. Further extension and upgrade are possible, which may make the system offer the full range of home banking activities through Internet, including reordering checks and stopping payments, loan service and credit card service, etc.

The Internet explosion and the possibility of direct digital interaction with large numbers of home consumers presents tremendous opportunity and challenges to most financial institutions to serve the bank's convenience-driven markets for banking services and to reduce the business costs. With the speed with which the Internet is so widely accepted, on-line banking will be a driving force for bank's business. It gives the user the opportunity to bank at home, at the office, or anywhere from a computer. As we approached the 21st century, on-line banking is getting more popular everyday.

There are two major developments regarding the Internet that will have a major impact on the future of online banking services. The first is high speed, broadband access, and the second is the subsequent convergence of the personal computer and television set. Finally, wireless cable and digital satellite delivery complement these new media which will revolutionize online services. The bottom line of high speed, broad bandwidth delivery is that the banks now have full multimedia capabilities providing for video, telephony, and sound on the web sites, and the convergence of computers and television presents another major opportunity in the online delivery of financial services. Home banking is truly the ultimate in convenience for the consumers.

Bibliography

[1] Thomas A. Powell, *HTML: The Complete Reference*, McGraw-Hill, 1999.

[2] Steven Holzner, *JavaScript Complete*, McGraw-Hill, 1998.

[3] George Koch and Kevin Loney, *Oracle 8: The Complete Reference*, McGraw-Hill, 1997.

[4] Alex Homer, Dave Sussman and Brian Francis, *Professional Active Server Pages 3.0*, Wrox press, 1999.

[5] Vivian Neou, *HTML 3.2 CD with JavaScript for Windows95*, Prentice Hall PTR, 1997.

[6] Murray, William H., *JavaScript and HTML 4.0 User's Resource*, Upper Saddle River, 1999.

[7] Jan L. Harrington, *SQL Clearly Explained*, AP Professional, 1998.

[8] Don Benage, *Using Microsoft Visual Studio*, Que Corporation, 1998.

[9] Gary Cornell, *Learn Microsoft Visual Basic scripting edition now*, Microsoft Press, 1998.

[10] Eric Ladd, *Using HTML 4.0, Java 1.1, and JavaScript 1.2*, Que Corporation, 1998.

[11] William G. Page, *Using Oracle 8*, Que Corporation, 1998.

[12] J. Cranford Teague, *DHTML for the World Wide Web*, Peachpit Press, 1998.

Appendix A User Manual

To start the online banking, a user can access the following web site by Microsoft Internet Explorer browser version 4.0 or higher:

http://Web Server IP Address/OraScottPro/Default.asp, here Web Server IP Address is dynamic depending on a specific financial institution's Web Server IP Address.

After visiting the above web site, customers can use the online banking services as follows:

1. For the current customers, they need to input their User IDs and Passwords to login.
2. For the new customers, they do not have User IDs and Passwords before they open their first accounts on the online banking. They can click Open an Account on the home page to open an account.
3. For the current customers, after login, they can access their accounts. They can:
   - ❖ open accounts by clicking on "Open Account";
   - ❖ pay bills or transfer funds by clicking on "Pay Bill";
   - ❖ check balances and get account summary by clicking on "Check Balance";
   - ❖ view transactions happened in the past statement period by clicking on "Transaction History";
   - ❖ change addresses and phone numbers by clicking on "Change Address";
   - ❖ change passwords by clicking on "Change Password";
   - ❖ go back to home page by clicking on "Log Out".

Appendix B SQL Start File

This appendix lists the SQL start file for the online banking system. The start file bank.sql is used for quick and simple reporting on all of the tables and views in the database. Its purpose is to illustrate all column definitions for each table and view in the database, including attribute name, type, size, etc. Before creating tables and views, we drop them first and then create new ones each time when the system is setup. The start file also includes some initial data in each table for illustration purpose.

Start file name: bank.sql;

Contents are listed as follows:

```
rem file name: bank.sql
drop table Payment;
drop table CustAcct;
drop table AccountType;
drop table Customers;
/*
rem ******** table Customers ***************
*/
create table Customers (
CustSIN        number(9),
CustName       varchar2(24),
CustGender     char(4),
CustBirthDate  date,
CustAddr       varchar2(60),
```

```
CustPhone      varchar2(12),
UserID         varchar2(12),
CustPasswd     varchar2(12),
primary key (CustSIN));

insert into Customers values
(100000001, 'Jone Smith', 'M', To_date('12-May-1946', 'DD-MON-YYYY'),
'123 High Steet West', '514-3443981','Jones','er2');
insert into Customers values
(100000002, 'Bart Sarjeant','M', To_date('22-May-1966', 'DD-MON-YYYY'),
'345 Guy Steet West', '514-3556982', 'bsar','pe94');
insert into Customers values
(100000003, 'Elbert Talbot', 'M', To_date('12-Aug-1966', 'DD-MON-YYYY'),
'2432 Young Steet East', '416-5674211', 'djke3','dfe6');
insert into Customers values
(100000004, 'Print Canada', 'AN', To_date('12-Aug-1966', 'DD-MON-YYYY'),
'321 Young Steet West', '416-5674211', 'sprint','dkr');
insert into Customers values
(100000005, 'Bill Canada', 'AN', To_date('12-Aug-1966', 'DD-MON-YYYY'),
'321 Gay West', '514-3344211', 'ball','de34');
insert into Customers values
(100000006, 'Hydro Quebec', 'AN', To_date('12-Aug-1966', 'DD-MON-YYYY'),
'321 St. Catherin', '514-5674211', 'Hydro','kd4');
```

```
/*
rem ********** table AccountType ********************
*/
create table AccountType(
AcctType      varchar2(16),
AccDesp       varchar2(50),
InterestRate  number(4,2),
primary key   (AcctType));

insert into AccountType values
('Checking', 'Checking is a transacton account.', 0.12);
insert into AccountType values
('Savings',  'Savings is for accumulating savings.', 2.15);
insert into AccountType values
('GIC', 'GIC is Guaranteed Investment Certificates.',  4.11);
insert into AccountType values
('Students', 'Students account is for students.', 0.15);
insert into AccountType values
('Deposit', 'Deposit makes money grow.', 2.10);
insert into AccountType values
('Investment', 'Investment is for the maximum return.', 4.19);
```

```
/*
rem ********* table CustAcct *****************
*/
create table CustAcct(
CustAcctNo    number(10),
CustSIN       number(9),
AcctType      varchar2(16),
OpenDate      date,
Balance       number(10,2),
primary key (CustAcctNo),
foreign key (CustSIN) references Customers,
foreign key (AcctType) references AccountType);

insert into CustAcct values
(100001, 100000001, 'Checking', To_date('02-May-1950', 'DD-MON-YYYY'),
3200.32);
insert into CustAcct values
(100002, 100000002, 'Checking', To_date('24-Sep-1990', 'DD-MON-
YYYY'),4567.83);
insert into CustAcct values
(100003, 100000004, 'Savings', To_date('16-Aug-1949', 'DD-MON-
YYYY'),100.00);
insert into CustAcct values
```

```
(100004, 100000005, 'Savings', To_date('09-Dec-1951', 'DD-MON-
YYYY'),200.00);
insert into CustAcct values
(100005, 100000006, 'Savings', To_date('17-Jun-1961', 'DD-MON-
YYYY'),300.00);

/*
rem ********** table Payment ********************
*/
create table Payment(
BillNo              varchar2(10),
PayingAcctNo        number(10),
ReceivingAcctNo number(10),
PayDate             date,
DueDate             date,
PaidAmount          number(10,2),
primary key (PayingAcctNo, ReceivingAcctNo, PayDate),
foreign key (PayingAcctNo) references CustAcct(CustAcctNo),
foreign key (ReceivingAcctNo) references CustAcct(CustAcctNo),
check (PayingAcctNo <> ReceivingAcctNo));


insert into Payment values
```

```
(10234, 100001, 100003, To_date('12-May-1999', 'DD-MON-YYYY'),
To_date('12-May-1999', 'DD-MON-YYYY'),150.65);
insert into Payment values
(13434, 100001, 100004, To_date('16-Dec-1999', 'DD-MON-YYYY'),
To_date('18-Dec-1999', 'DD-MON-YYYY'),250.85);
insert into Payment values
(13224, 100002, 100005, To_date('16-Dec-1999', 'DD-MON-YYYY'),
To_date('18-Dec-1999', 'DD-MON-YYYY'),350.85);

/*
rem ==========View from here======
*/
/*
rem ********** SortedAcctType**********
*/
create or replace view SortedAcctType as
select  AcctType  from  AccountType group by AcctType;
select * from SortedAcctType;
```

Appendix C Source Code

Some selected source code for the application is presented in this part.

File Name: Default.asp

```
<html>
<head>
<script language="JavaScript">
function TodayDate() {
now = new Date();
monthString = new Object();
monthString[0]="January";
monthString[1]="February";
monthString[2]="March";
monthString[3]="April";
monthString[4]="May";
monthString[5]="June";
monthString[6]="July";
monthString[7]="August";
monthString[8]="September";
monthString[9]="October";
monthString[10]="November";
monthString[11]="December";
document.writeln(monthString[now.getMonth()] + " " + now.getDate() + ", " +
now.getYear())
 }
</script>
<link rel="stylesheet" type="text/javascript" href="OnLineBank.js">
<link rel="stylesheet" type="text/css" href="OnLineBank.css">
</head>
<body>
<center><img SRC="images/FLOWER3.gif" WIDTH="96" HEIGHT="74">
```

```
        <img src="images/logon.gif" WIDTH="524" HEIGHT="75">
        <img SRC="images/FLOWER3.gif" WIDTH="96" HEIGHT="74">
</center>
<p align="center">
<form action="MainMenuHome.asp" method="post" id="form1" name="form1">
<div align="center">
        <table border="0" cellPadding="1" cellSpacing="5" width="55%">
        <tr><td align="right">
                <strong><font FACE="Verdana, Arial, Helvetica" SIZE="3"
color="crimson">New Customer? </strong></font>
            </td>
            <td> <a href="../OraScottPro/OpenAccount/OpenAccountNew.asp">
                <strong><font FACE="Verdana, Arial, Helvetica" SIZE="3"
color="mediumblue"><U>Open an Account
                        </U></font></strong></a>
            </td>
        </tr>
        <tr><td>  </td></tr>
        <tr>
            <td align="right">
              <font FACE="Verdana, Arial, Helvetica" SIZE="3" color="darkred"><b>
                        Customer Login:</font> </B>
            </td>
        </tr>
        <tr>
            <td align="right">
             <font FACE="Verdana, Arial, Helvetica" SIZE="2" color="mediumblue">
             <b>User ID:</font></B></td>
            <td>
             <input id="text1" name="Username"></td></tr>
        <tr>
```

```
            <td align="right"><font FACE="Verdana, Arial, Helvetica" SIZE="2"
color="mediumblue"> <b> Password:</font></STRONG> </B></td>
            <td> <input id="password1" name="password" type="password"></td></tr>
        </table>
 </div>
<div align="center">
   <table border="0" cellPadding="1" cellSpacing="1" height="36" style="HEIGHT:
36px; WIDTH: 190px" width="29.5%">
          <tr>
                  <td><input id="submit1" name="submit1" style="HEIGHT: 28px;
WIDTH: 75px" type="submit" value="Login"></td>
                  <td><input id="reset1" name="reset1" style="HEIGHT: 28px;
WIDTH: 75px" type="reset" value="Clear"></td>
              </tr>
        </table>
 </div>
</form>
<center><font face="Arial" size="2" color="darkblue">
<script language="JavaScript">
   TodayDate();
</script>
<a href="mailto:hwang@cs.concordia.ca">
<img border="0" height="32" src="images/e-mail.gif" width="92"></a></font>
</center>
<center><font face="Arial" size="2" color="darkblue"> 
<img SRC="images/Copyright.gif" WIDTH="250" HEIGHT="30"> </center></font>
</body>
</html>
```

File Name: MainMenuHome.asp

```
<%
'validate the input data
UserID = Trim(Request.Form("Username"))
Passwd = Trim(Request.Form("password"))

If UserID="" or Passwd="" then
   str = server.URLEncode("Please input your User ID and Password.")
        Response.Redirect("Response.asp?message=" & str)
End if

Set connection = server.CreateObject("ADODB.connection")
connection.Open "OraScottDSN","scott","tiger"

'sql
sql="SELECT * " & _
        "FROM Customers " & _
   "WHERE UserID = '"&UserID&"' and " & _
   "CustPasswd ='"&Passwd&"' "

Set recordset = connection.Execute(sql)

'check if the input sid exists
if recordset.EOF then
        str = server.URLEncode("Your User ID or/and Password are incorrect! Try it
again.")
        Response.Redirect("Response.asp?message=" & str)
end if

Session("UserID") = recordset("UserID")
```

```
Session("Passwd") = recordset("CustPasswd")
%>

<HTML>
<HEAD>
<TITLE> OnLine Banking </TITLE>
</HEAD>
<FRAMESET COLS="136,*" BORDER="0" FRAMESPACING="0"
FRAMEBORDER="0">
    <FRAME SRC="FrameLeft.htm"  NAME="LeftFrame" NORESIZE
SCROLLING="no" MARGINWIDTH="0" MARGINHEIGHT="0">
    <FRAME SRC="MainMenu.asp" NAME="MainFrame" NORESIZE
SCROLLING="auto" MARGINWIDTH="0">
</FRAMESET>
</HTML>
```

File Name: FrameLeft.htm

```
<html>
<head>
<title> OnLine Banking </title>
<meta NAME="generator" CONTENT="OnLine Banking">

<script language="JavaScript">
function TodayDate() {
now = new Date();
monthString = new Object();
monthString[0]="January";
monthString[1]="February";
```

```
 monthString[2]="March";
 monthString[3]="April";
 monthString[4]="May";
 monthString[5]="June";
 monthString[6]="July";
 monthString[7]="August";
 monthString[8]="September";
 monthString[9]="October";
 monthString[10]="November";
 monthString[11]="December";
document.writeln(monthString[now.getMonth()] + " " + now.getDate() + ", " +
now.getYear())
 }

var sid="mundania";

function dynamic_image_on(imgName) {
        if (version == "3") {
        ActiveImage = eval(imgName + "_active.src");
        document [imgName].src = ActiveImage;
        }
   }

function dynamic_image_off(imgName) {
        if (version == "3") {
        DefaultImage = eval(imgName + "_default.src");
        document [imgName].src = DefaultImage;
        }
  }
    while(sid!="mundania"){};
```

```
whichbrowser = navigator.appName; //identify browser
        Ver = parseInt(navigator.appVersion);
        if (whichbrowser == "Netscape" && Ver >= 3 || whichbrowser == "Microsoft
Internet Explorer" && Ver > 3) version = "3";
        else version = "2";
        if (version == "3") { // if 3.0, load images

       //image to show when not highlighted

        img_1_default = new Image();
        img_1_default.src = "Images/HOME.gif";
        img_2_default = new Image();
        img_2_default.src = "images/OPEN.gif";
        img_3_default = new Image();
        img_3_default.src = "images/Pay.gif";
                        img_4_default = new Image();
        img_4_default.src = "images/Check.gif";
                        img_5_default = new Image();
        img_5_default.src = "images/Tran1.gif";
                        img_6_default = new Image();
        img_6_default.src = "images/Addr.gif";
        img_7_default = new Image();
        img_7_default.src = "images/Pass.gif";
        img_8_default = new Image();
        img_8_default.src = "images/ChanUser.gif";

       //image to show when highlighted

        img_1_active = new Image();
        img_1_active.src = "images/HomeON.gif";
        img_2_active = new Image();
```

```
            img_2_active.src = "images/OpenOn.gif";
            img_3_active = new Image();
            img_3_active.src = "images/PayOn.gif";
            img_4_active = new Image();
            img_4_active.src = "images/CheckOn.gif";
            img_5_active = new Image();
            img_5_active.src = "images/TranOn.gif";
            img_6_active = new Image();
            img_6_active.src = "images/AddrOn.gif";
            img_7_active = new Image();
            img_7_active.src = "images/PassOn.gif";
            img_8_active = new Image();
            img_8_active.src = "images/ChanUserOn.gif";

}
// done hiding-->

function mouseHover(state)
{
    var row = window.event.srcElement;
    var colorChange = ((state == "over") ? "#ffff00" : "");
    row.style.backgroundColor = "red";
}

</script>

<script language="javascript" RUNAT="Server">

/*****************************************************
* mouseOver()
*****************************************************/
```

```
/* mouseOver is called when the mouse move over the catalog
        table. The row that the mouse is over gets highlighted.
*/

function mouseOver() {

        var element = window.event.srcElement;

        while (element.tagName != 'TR')
                element = element.parentElement;

        element.style.backgroundColor = "yellow";
}

/*****************************************************
* mouseOut()
*****************************************************/
/* mouseOver is called when the mouse move off a row in the catalog
        table. The hilighted row retured to normal.
*/
function mouseOut() {
        var element = window.event.srcElement;

        while (element.tagName != 'TR')
                element = element.parentElement;

        element.style.backgroundColor = "white";}
</script>
<link rel="stylesheet" type="text/css" href="FrameLeft.css">
</head>
<body>
```

```
<center>
<img SRC="images/CART_Anim.gif" WIDTH="130" HEIGHT="100">
</center>

<table ALIGN="left" BORDER="0" WIDTH="50%" cellspacing="0" cellpadding="0">

<tr><td height=45 align="center" bgcolor="lightgreen"><font size=5
color="darkred"><b> Services
      </td>
</tr>

<tr><td><a href="../OraScottPro/MainMenu.asp"
onmouseout="dynamic_image_off('img_1')"
onmouseover="dynamic_image_on('img_1')" target="MainFrame">
        <img alt="Home" border="0" name="img_1" src="images/HOME.gif"
WIDTH="137" HEIGHT="32"></a>
      </td></tr>
<tr><td><a href="../OraScottPro/OpenAccount/OpenAccount.asp"
onmouseout="dynamic_image_off('img_2')"
onmouseover="dynamic_image_on('img_2')" target="MainFrame">
        <img alt="Open Account" border="0" name="img_2" src="images/OPEN.gif"
WIDTH="137" HEIGHT="32"></a>
        </td></tr>
<tr><td><a href="../OraScottPro/PayBill/PayBill.asp"
onmouseout="dynamic_image_off('img_3')"
onmouseover="dynamic_image_on('img_3')" target="MainFrame">
        <img alt="Pay Bill" border="0" name="img_3" src="images/Pay.gif"
WIDTH="137" HEIGHT="32"></a>
        </td></tr>
```

```
<tr><td><a href="../OraScottPro/CheckBalance/CheckBalance.asp"
onmouseout="dynamic_image_off('img_4')"
onmouseover="dynamic_image_on('img_4')" target="MainFrame">
        <img alt="Check balance" border="0" name="img_4" src="images/Check.gif"
WIDTH="137" HEIGHT="32"></a>
        </td></tr>
<tr><td><a href="../OraScottPro/TranHistory/TranHistory.asp"
onmouseout="dynamic_image_off('img_5')"
onmouseover="dynamic_image_on('img_5')" target="MainFrame">
        <img alt="Transaction History" border="0" name="img_5"
src="images/Tran1.gif" WIDTH="137" HEIGHT="32"></a>
        </td></tr>
<tr><td><a href="../OraScottPro/ChangeAddr/ChangeAddr.asp"
onmouseout="dynamic_image_off('img_6')"
onmouseover="dynamic_image_on('img_6')" target="MainFrame">
        <img alt="Change Address" border="0" name="img_6" src="images/Addr.gif"
WIDTH="137" HEIGHT="32"></a>
        </td></tr>
<tr><td><a href="../OraScottPro/PassWord/Password.htm"
onmouseout="dynamic_image_off('img_7')"
onmouseover="dynamic_image_on('img_7')" target="MainFrame">
        <img alt="Change Password" border="0" name="img_7" src="images/Pass.gif"
WIDTH="137" HEIGHT="32"></a>
        </td></tr>
<tr><td><a href="../OraScottPro/Default.asp"
onmouseout="dynamic_image_off('img_8')"
onmouseover="dynamic_image_on('img_8')" target="_top">
        <img alt="Log Out" border="0" name="img_8" src="images/ChanUser.gif"
WIDTH="137" HEIGHT="32"></a>
        </td></tr>
```

```
<tr><td height=35 align="center" bgcolor="lightgreen"><font size=3
color="darkred"><b>
        <script language="JavaScript">
   TodayDate();
        </script></font>
        </td></tr>
</table>
</body>
</html>
```

File Name: MainMenu.asp

```
<html>
<head>
<title>OnLine banking</title>
<link rel="stylesheet" type="text/css" href="OnLineBank.css">
</head>
<body>
<center>
<%Set ad = Server.CreateObject("MSWC.AdRotator")%>
<%= ad.GetAdvertisement("RandomAd.txt") %>
<br>
<font FACE="Verdana, Arial, Helvetica" SIZE="1">
<a HREF="javascript:location.href='MainMenu.asp'">Refresh AD</a>
</font></center>
<center>
<font FACE="Verdana, Arial, Helvetica" SIZE="1"> 
</font></center>
```

```
<p align="center"><strong><font FACE="Verdana, Arial, Helvetica" SIZE="4"
color="mediumblue">
Select One Service:
</font></strong></p>

<p align="center">
<table align="center" WIDTH="75%" BORDER="0" CELLSPACING="1"
CELLPADDING="5">
        <tr>
                <td><a href="../OraScottPro/MainMenu.asp" target="MainFrame">
                        <img SRC="images/H_Home.gif" border="0" WIDTH="45"
HEIGHT="44"></a></td>
                <td><a href="../OraScottPro/MainMenu.asp"
target="MainFrame">Home</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Go back main
        menu</b></font>
                </td>
                <td><a href="../OraScottPro/OpenAccount/OpenAccount.asp"
target="MainFrame">
                        <img SRC="images/H_furniture.gif" border="0" WIDTH="45"
HEIGHT="44"></a></td>
                <td><a href="../OraScottPro/OpenAccount/OpenAccount.asp"
target="MainFrame">Open Account</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Open a new account for the customer</b></font>
                </td>
        </tr>
        <tr>
                <td><a href="../OraScottPro/PayBill/PayBill.asp" target="MainFrame">
```

```
                    <img SRC="images/H_hardware.gif" border="0" WIDTH="45"
 HEIGHT="44"></a></td>
                <td><a href="../OraScottPro/PayBill/PayBill.asp" target="MainFrame">
                Pay Bill</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Pay bill using your
        own accounts</b></font></td>
                <td><a href="../OraScottPro/CheckBalance/CheckBalance.asp"
 target="MainFrame">
                    <img SRC="images/H_motion.gif" border="0" WIDTH="45"
 HEIGHT="44"></a></td>
                <td><a href="../OraScottPro/CheckBalance/CheckBalance.asp"
 target="MainFrame">Check Balance</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Check all your accounts balance, account types and
 numbers</b></font></td>
        </tr>
        <tr>
                <td><a href="../OraScottPro/TranHistory/TranHistory.asp"
 target="MainFrame">
                    <img SRC="images/H_textbooks.gif" border="0" WIDTH="45"
 HEIGHT="44"></a></td>
                <td><a href="../OraScottPro/TranHistory/TranHistory.asp"
 target="MainFrame">Transaction History</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Look at your transaction history up to 1 month</b></font>
                </td>
                <td><a href="../OraScottPro/ChangeAddr/ChangeAddr.asp"
 target="MainFrame">
                    <img SRC="images/H_tutors.gif" border="0" WIDTH="45"
 HEIGHT="44"></a></td>
```

```
                <td><a href="../OraScottPro/ChangeAddr/ChangeAddr.asp"
 target="MainFrame">Change Address</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Change your address, telephone number by yourself</b></font>
                </td></tr>
        <tr>
                <td><a href="../OraScottPro/PassWord/Password.htm"
 target="MainFrame">
                        <img SRC="images/H_vehicles.gif" border="0" WIDTH="45"
 HEIGHT="44"></a> </td>
                <td><a href="../OraScottPro/PassWord/Password.htm"
 target="MainFrame">Change PassWord</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Change your login password for security</b></font>
                </td>
                <td><a href="../OraScottPro/Home.htm" target="_top">
                        <img SRC="images/H_parties.gif" border="0" WIDTH="45"
 HEIGHT="44"></a></td>
                <td><a href="../OraScottPro/Default.asp" target="_top">
                Log Out</a>
                <font FACE="Verdana, Arial, Helvetica" SIZE="1"><b>
                <br>Log out and back to home page</b></font>
                </td>
        </tr>

</table>
</p>
</body>
</html>
```

File Name: OpenAccount.asp

```
<%@ Language=VBScript %>
<% 'VI 6.0 Scripting Object Model Enabled %>
<!--#include file="../_ScriptLibrary/pm.asp"-->
<% if StartPageProcessing() Then Response.End() %>
<FORM name=thisForm METHOD=post>
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<SCRIPT ID=serverEventHandlersVBS LANGUAGE=vbscript RUNAT=Server>

Sub bntSave_onclick()

        UserID =Session("UserID")
        Passwd =Session("Passwd")
        Accttype = Trim(lstAcctType.getValue(lstAcctType.selectedIndex))
        OpenDate = Trim(txtOpenDate.value)
        Balance = Trim(txtBalance.value)
        Balan = CDbl(Balance)
        Custsin= FindCustsin(UserID, Passwd)
        accno=txtAcctNo.value

        If  Balance = "" then
                txtMsg.value="The open balance is required."
                exit sub
       Elseif Balan<50 then
          txtMsg.value="Your minimum deposit is less than $50."
    exit sub
```

```
        Elseif CheckAcctType(Custsin, Accttype)<>"" then
          txtMsg.value="You have had "&AcctType&" account, please select another
type."
      exit sub
    End if

    'after checking the balance is not empty and then change it to be Double Precision.
    Set connection=Server.CreateObject("ADODB.Connection")
      connection.Open "OraScottDSN","scott","tiger"

   sqlNewAccount = "INSERT INTO CustAcct (CustAcctNo, CustSIN, AcctType,
OpenDate, Balance) " & _
        "VALUES ("&accno&", "&Custsin&", "&Accttype&"', " & _
        "To_date('"&OpenDate&"', 'MM/DD/YYYY'), "&Balan&")"

        Set rstOpenAccount=connection.Execute(sqlNewAccount)
        connection.close
        txtMsg.value="Your new account has been opened successfully."

End Sub

</SCRIPT>
</head>

<script language="VBScript" RUNAT=Server>

'get CustSIN of the existing customer from the table Customers.
Function FindCustsin(ByVal userid, passwd)

  Set connectionC=Server.CreateObject("ADODB.Connection")
    connectionC.Open "OraScottDSN", "Scott", "Tiger"
```

```
        sql_CSIN="SELECT CustSIN FROM Customers " & _
              " WHERE UserID ="&userid&"' and " &_
              " CustPasswd ="&passwd&""
     Set recordsetC=connectionC.Execute(sql_CSIN)

     FindCustsin=CLng(Trim(recordsetC("CustSIN")))
     connectionC.close
  End function

  'check if this customer has had this kind of account types
  function CheckAcctType(Custsin, Accttype)
     Set connection=Server.CreateObject("ADODB.Connection")
       connection.Open "OraScottDSN","scott","tiger"

     accttypesql="select AcctType from CustAcct " & _
     "where (CustSIN="&Custsin&" and AcctType="&Accttype&")"
     Set recaccttype=connection.Execute(accttypesql)

     if recaccttype.EOF then
       CheckAcctType=""
     else
           CheckAcctType=recaccttype("AcctType")
     end if
     connection.close
  end function

  Function CreateAccNo()
          'create a CustAccNo
          Set connection=Server.CreateObject("ADODB.Connection")
       connection.Open "OraScottDSN","scott","tiger"
```

```
        accnosql="select CustAcctNo from CustAcct where CustAcctNo>=all("&_
          "select A.CustAcctNo from CustAcct A)"
        Set maxaccno = connection.Execute(accnosql)
        CreateAccNo=clng(trim(maxaccno("CustAcctNo")))+1

        connection.close
End Function

Sub ChangeDate_onclick()
  orgdate=Date()
  dd=CStr(DatePart("d",orgdate))
  mm=CStr(DatePart("m",orgdate))
  yy=CStr(DatePart("yyyy", orgdate))
  dd1=CInt(dd)
  mm1=CInt(mm)

  if dd1 > 0 And dd1 <= 9 then
    dd="0"+dd
  elseif mm1 > 0 And mm1 <= 9 then
    mm="0"+mm
  end if

  chdate=mm+"/"+dd+"/"+yy
  txtOpenDate.value=chdate

End Sub


Sub thisPage_onshow()

      Call ChangeDate_onclick()
```

```
        accno=CreateAccNo()
    txtAcctNo.value=accno
    txtBalance.value="50.00"

End Sub


</script>
<body bgColor=linen>
<center><br>
<img SRC="../images/OpenAcct.gif" WIDTH="334" HEIGHT="43">
</center>
<p>

<table Align="center" width="75%">
 <tr><td><STRONG><FONT color=brown face="">Notes:
        </FONT></STRONG>
      </td></tr>
 <tr><td><FONT color=royalblue face=""><STRONG>1. You can not open the account
        types you have had.</FONT> </STRONG></FONT>
      </td></tr>
<tr><td><FONT color=royalblue face=""><STRONG>2. The minimum deposit
required to open account is $50.00. </STRONG></FONT>
     </td></tr>
</table>

<p align="center">
<table border="1" cellPadding="1" cellSpacing="1" width="75%" bgColor=#eea0ab
borderColor=white borderColorDark=slategray borderColorLight=white>
<TR>
  <TD bgColor=wheat>
```

```
      <P align=center><FONT color=mediumslateblue
          face=""><STRONG>Account No.</STRONG></FONT></P></TD>
    <TD style="WIDTH: 15px" width=15></TD>
    <TD bgColor=wheat><P align=center><STRONG><FONT
          color=mediumslateblue face="">Account ype</FONT></STRONG></P></TD>
    <TD style="WIDTH: 15px" width=15></TD></TR>

    <tr><td bgColor=wheat>
     <P align=center><FONT color=mediumslateblue
          face=""><STRONG>Open</FONT> <FONT color=mediumslateblue
          face="">Date</FONT> </STRONG> </P>
        <td style="WIDTH: 15px" width=15> </td>
      <td bgColor=wheat>
        <div align="center"><FONT color=mediumslateblue
        face=""><STRONG>Open Deposit</STRONG></FONT></div>
      <td style="WIDTH: 15px" width=15>
      </td></tr>
</table></p>

 <p align=center><strong>
   <font face="Arial" size="4">
   <A href="../MainMenu.asp">Back</A>
   </font></strong></p>

<script language="javascript" RUNAT = "Server">


/*****************************************************
* mouseOver()
*****************************************************/
/* mouseOver is called when the mouse move over the catalog
```

```
        table.  The row that the mouse is over gets highlighted.
 */

 function mouseOver() {

        var element = window.event.srcElement;

        while (element.tagName != 'TR')
                element = element.parentElement;

        element.style.backgroundColor = "yellow";
 }


/*****************************************************
* mouseOut()
*****************************************************/
/* mouseOver is called when the mouse move off a row in the catalog
        table.  The hilighted row retured to normal.
*/
function mouseOut() {
        var element = window.event.srcElement;

        while (element.tagName != 'TR')
                element = element.parentElement;

        element.style.backgroundColor = "white";
}

</script>
```

```
</body>
<% ' VI 6.0 Scripting Object Model Enabled %>
<% EndPageProcessing() %>
</FORM>
</html>
```

File Name: OpenAccountNew.asp

```
<%@ Language=VBScript %>
<% ' VI 6.0 Scripting Object Model Enabled %>
<!--#include file="../_ScriptLibrary/pm.asp"-->
<% if StartPageProcessing() Then Response.End() %>
<FORM name=thisForm METHOD=post>
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">

<SCRIPT ID=serverEventHandlersVBS LANGUAGE=vbscript RUNAT=Server>

Sub bntSave_onclick()

        Accttype = Trim(lstAcctType.getValue(lstAcctType.selectedIndex))
        OpenDate = Trim(txtOpenDate.value)
        Bal = Trim(txtBalance.value)
        name=Trim(txtName.value)
        gender=Trim(lstGender.getValue(lstGender.selectedIndex))
        bday=Trim(lstbday.getValue(lstbday.selectedIndex))
        bmonth=Trim(lstbmonth.getValue(selectedIndex))
        byear=Trim(txtbyear.value)
```

```
dob=bday&"-"&bmonth&"19"&byear
custsin = Trim(txtCustSIN.value)
address=Trim(txtAddr.value)
phone1=Trim(txtCode.value)
phone2=Trim(txtTelCode1.value)
phone3=Trim(txtTelCode2.value)
phone="("&phone1&")"&phone2&phone3
useID=Trim(txtUserID.value)
passwd=Trim(txtPasswd.value)
repasswd=Trim(txtRePasswd.value)
accno=txtAcctNo.value
Balan = CDbl(Bal)


If  name = "" then
        txtMsg.value="Your name is required, please input it."
        exit sub
Elseif len(byear)<>2 then
        txtMsg.value="Please input your last two-digit year of birth."
        exit sub
Elseif  len(custsin) <>9 then
        txtMsg.value="Please input your 9-digit SIN."
        exit sub
Elseif  address = "" then
        txtMsg.value="Your address is required, please input it."
        exit sub
Elseif  len(phone1) <>3 then
        txtMsg.value="Please input 3-digit area code."
        exit sub
Elseif  len(phone2) <>3 then
        txtMsg.value="Please input first 3 digits of your phone no."
```

```
            exit sub
        Elseif  len(phone3) <>4 then
            txtMsg.value="Please input last 4 digits of your phone no."
            exit sub
        Elseif useID = "" then
            txtMsg.value="Please select your user ID."
            exit sub
        Elseif FindUserID(useID) then
            txtMsg.value="This UserID has been chosen by somebody else, please
select another one."
            exit sub
        Elseif  passwd = "" then
            txtMsg.value="Your password is required."
            exit sub
        Elseif  repasswd = "" then
            txtMsg.value="Please re-enter your password."
            exit sub
        Elseif passwd<>repasswd then
            txtMsg.value="Two passwords are not same. Please re-enter your
password."
          txtPasswd.value=""
          txtRePasswd.value=""
            exit sub
        Elseif Bal = "" then
            txtMsg.value="The open balance is required."
            exit sub
        Elseif Balan<50 then
          txtMsg.value="The minimum deposit to open account is $50."
    exit sub
      Elseif CheckAcctType(custsin, Accttype)<>"" then
```

```
            txtMsg.value="You have had "&AcctType&" account, please select another
type."
      exit sub
   End if

   'after checking the balance is not empty and then change it to be Double Precision.
   custsin=CLng(custsin)


   Set connection=Server.CreateObject("ADODB.Connection")
     connection.Open "OraScottDSN","scott","tiger"

        'append a record in the Customers table
        custsql= "INSERT INTO Customers(CustSIN, CustName,CustGender, " & _
        "CustBirthDate, CustAddr,CustPhone,UserID, CustPasswd) VALUES(" &
custsin & ",'" & name & "'," & _
        "'"& gender &"', To_date('"& dob &"', 'DD-MON-YYYY'), '" & address & " ','"
& phone & "', "& _
        " '"&useID&"','"& repasswd &"')"

        Set newcustrecord = connection.Execute(custsql)

        'Add a new record to CustAcct
        sqlNewAccount = "INSERT INTO CustAcct (CustAcctNo, CustSIN, AcctType,
OpenDate, Balance) " & _
        "VALUES ("&accno&", "&custsin&", '"&Accttype&"', " & _
        "To_date('"&OpenDate&"', 'MM/DD/YYYY'), "&Balan&")"

        Set rstOpenAccount=connection.Execute(sqlNewAccount)
        connection.close
        txtMsg.value="Your new account has been opened successfully."
```

```
End Sub

</SCRIPT>
</head>

<script language="VBScript" RUNAT=Server>

'Check if userid chosen by the new customer has already been selected by searching
'the table customers.
Function FindUserID(ByVal userid)

  Set connectionC=Server.CreateObject("ADODB.Connection")
     connectionC.Open "OraScottDSN", "Scott", "Tiger"
     sql_UserID="SELECT UserID FROM Customers " & _
          " WHERE UserID = '"&userid&"'"
  Set recordsetC=connectionC.Execute(sql_UserID)

  if not recordsetC.EOF then
          FindUserID=TRUE
  else
          FindUserID=FALSE
  end if

  connectionC.close

End function

'check if this customer has had this kind of account types
function CheckAcctType(Custsin, Accttype)
```

```
    Set connection=Server.CreateObject("ADODB.Connection")
      connection.Open "OraScottDSN","scott","tiger"

    accttypesql="select AcctType from CustAcct " & _
    "where (CustSIN="&Custsin&" and AcctType="&Accttype&")"
    Set recaccttype=connection.Execute(accttypesql)

    if recaccttype.EOF then
      CheckAcctType=""
    else
          CheckAcctType=recaccttype("AcctType")
    end if

    connection.close

end function


Function CreateAccNo()

        'create a CustAccNo
        Set connection=Server.CreateObject("ADODB.Connection")
     connection.Open "OraScottDSN","scott","tiger"

        accnosql="select CustAcctNo from CustAcct where CustAcctNo>=all("&_
          "select A.CustAcctNo from CustAcct A)"
        Set maxaccno = connection.Execute(accnosql)
        CreateAccNo=clng(trim(maxaccno("CustAcctNo")))+1

        connection.close
```

```
End Function


Sub ChangeDate()

  orgdate=Date()
  dd=CStr(DatePart("d",orgdate))
  mm=CStr(DatePart("m",orgdate))
  yy=CStr(DatePart("yyyy", orgdate))
  dd1=CInt(dd)
  mm1=CInt(mm)

  if dd1 > 0 And dd1 <= 9 then
    dd="0"+dd
  elseif mm1 > 0 And mm1 <= 9 then
    mm="0"+mm
  end if

  chdate=mm+"/"+dd+"/"+yy
  txtOpenDate.value=chdate

End Sub


Sub thisPage_onshow()

      Call ChangeDate()
      txtAcctNo.value=CreateAccNo()
      txtBalance.value="50.00"

End Sub
```

```
</script>
<body bgColor=linen>

<center>
<img SRC="../images/OpenAcct.gif" WIDTH="334" HEIGHT="43">
</center>

<TABLE align=center background="" bgColor=blanchedalmond border=1
cellPadding=1 cellSpacing=1 width=85%>
  <P align=center><FONT color=maroon face=Arial size=3><STRONG>Personal
Information  </STRONG></FONT></P>
  <TBODY>
  <TR>
    <TD width=20% bgColor=wheat style="BACKGROUND-COLOR: peachpuff;
COLOR: mediumslateblue"><STRONG>Name</STRONG></TD>
    <TD width=45% style="BACKGROUND-COLOR: #eea0ad"></TD>
    <TD style="BACKGROUND-COLOR: peachpuff"><STRONG><FONT
color=mediumslateblue  face="">SIN</FONT></STRONG>
    <TD style="BACKGROUND-COLOR: #eea0ad">
      <STRONG  style="BACKGROUND-COLOR: #eea0ad"></STRONG>
  <TR>
    <TD style="BACKGROUND-COLOR: peachpuff"><FONT
      color=mediumslateblue face=""  style="BACKGROUND-COLOR:
#f5deb3"><STRONG style="BACKGROUND-COLOR: peachpuff"
      >Address</STRONG></FONT></TD>
    <TD colSpan=3 height=6 style="BACKGROUND-COLOR: #eea0ad; HEIGHT:
6px"></TD></TR>
  <TR>
    <TD style="BACKGROUND-COLOR: peachpuff"><FONT
```

```
        color=mediumslateblue face=""
        style="BACKGROUND-COLOR: #f5deb3"><STRONG
style="BACKGROUND-COLOR: peachpuff"
        >Phone</STRONG></FONT>
      <TD height=6 style="BACKGROUND-COLOR: #eea0ad; HEIGHT: 6px">
      <TD style="BACKGROUND-COLOR: peachpuff"><STRONG><FONT
color=mediumslateblue face="">User ID</FONT></STRONG>
      <TD style="BACKGROUND-COLOR: #eea0ad">
   <TR>
      <TD style="BACKGROUND-COLOR: peachpuff"><STRONG><FONT
color=mediumslateblue  face=""  style="BACKGROUND-COLOR:
peachpuff">Password</FONT></STRONG></TD>
      <TD height=6 style="BACKGROUND-COLOR: #eea0ad; HEIGHT: 6px"> 
      </TD>
      <TD style="BACKGROUND-COLOR: peachpuff"><FONT size=2
style="BACKGROUND-COLOR: peachpuff"><STRONG><FONT
color=mediumslateblue><FONT>Re-enter
        Password </FONT></FONT></FONT><FONT
        color=mediumslateblue> </FONT></STRONG>
      <TD style="BACKGROUND-COLOR: #eea0ad">
      </TD>
</TR></TBODY></TABLE>

<P align=center style="BACKGROUND-COLOR: linen"> <FONT color=maroon
face=Arial size=3
style="BACKGROUND-COLOR: #ffdab9"><STRONG style="BACKGROUND-
COLOR: linen">Account Information</STRONG></FONT>
<P align=center style="BACKGROUND-COLOR: linen">
<table align="center" border="1" cellPadding="1" cellSpacing="1" width="70%"
bgColor=#eea0ab borderColor=white borderColorDark=slategray
borderColorLight=white>
```

```
  <TR>
   <TD bgColor=wheat width=30%
     style="BACKGROUND-COLOR: peachpuff">
    <P align=center><FONT color=mediumslateblue
       face=""><STRONG> Account Type</STRONG></FONT></P></TD>
   <td width=30%> </td>
   <td bgColor=wheat style="BACKGROUND-COLOR: peachpuff">
       <div align="center"><FONT color=mediumslateblue
       face=""><STRONG>Open
       Deposit</STRONG></FONT></div>
   <td style="WIDTH: 15px" width=15></td>
</tr></table></P>

<INPUT name=reset style="HEIGHT: 25px; WIDTH: 60px" type=reset value=Reset>
</center>

<p align=center><strong><font face="Arial" size="4"><A
href="../Default.asp">Back</A>
</font></strong></p>

<script language="javascript" RUNAT = "Server">

/*****************************************************
* mouseOver()
*****************************************************/
/* mouseOver is called when the mouse move over the catalog
       table. The row that the mouse is over gets highlighted.
*/

function mouseOver() {
```

```
        var element = window.event.srcElement;

        while (element.tagName != 'TR')
                element = element.parentElement;

        element.style.backgroundColor = "yellow";
}


/*************************************************
* mouseOut()
*************************************************/
/* mouseOver is called when the mouse move off a row in the catalog
        table. The hilighted row retured to normal.
*/
function mouseOut() {
        var element = window.event.srcElement;

        while (element.tagName != 'TR')
                element = element.parentElement;

        element.style.backgroundColor = "white";
}

</script>

<% ' VI 6.0 Scripting Object Model Enabled %>
<% EndPageProcessing() %>
</FORM>
</html>
```

File Name:PayBill.asp

```
<%@ Language=VBScript%>
<% ' VI 6.0 Scripting Object Model Enabled %>
<!--#include file="../_ScriptLibrary/pm.asp"-->
<% if StartPageProcessing() Then Response.End() %>
<FORM name=thisForm METHOD=post>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<META content="text/html; charset=unicode" http-equiv=Content-Type>
<%

Sub bntPay_onclick()

        Accts = Trim(lstAccts.getValue(lstAccts.selectedIndex))
        ReAcct = Trim(txtReAcctNo.value)
        BillNo=Trim(txtBillNo.value)
        PayDate = Trim(txtPayDate.value)
        DueDate = Trim(txtDueDate.value)
        Amount=Trim(txtAmount.value)

        'validate the input data
        If  ReAcct = "" then
        txtMsg.value="The receiving account number is required."
                exit sub
        Elseif  Amount = "" then
                txtMsg.value="Amount is required, please input it."
                exit sub
```

```
    End if

'after checking the balance is not empty and then change it to be Double Precision.
Amt = CDbl(Amount)
ReAcctNo=CLng(ReAcct)


Set conne=Server.CreateObject("ADODB.Connection")
  conne.Open "OraScottDSN","scott","tiger"

    'look up custacctno corresponding to accts
    sqlcustacctno="select custacctno from custacct A, customers C " & _
    " where (UserID ="&Session("UserID")&" and " &_
    " CustPasswd ="&Session("Passwd")&"'"&_
" and C.CustSIN=A.CustSIN and AcctType="&Accts&")"

    Set rstacctno= conne.Execute(sqlcustacctno)

    acctno=CLng(trim(rstacctno("custacctno")))

    'check if balance is enough for PaidAmount in table CustAcct
    checkbalancesql="select balance from custacct " & _
    "where custacctno="&acctno&""

    Set rstcheckbal=conne.Execute(checkbalancesql)
    bal=CDbl(Trim(rstcheckbal("balance")))

    If bal<Amt Then

      txtMsg.value="Your balance is "&bal&", not enough for "&Amt&"."
      Exit Sub
```

```
        Else

                'append a record in the Payment table
                paysql= "INSERT INTO Payment(BillNo,PayingAcctNo,
ReceivingAcctNo, " & _
                "PayDate, DueDate, PaidAmount) VALUES("'&BillNo&"', "&acctno&","
& _
                ""'& ReAcctNo &", To_date("'& PayDate &"', 'MM/DD/YYYY')," &_
                'To_date("'&DueDate&"','MM/DD/YYYY'), "&Amt&")"

                Set paybillrecord = conne.Execute(paysql)

                'update paying account balance
                updpayingbal= "UPDATE CustAcct Set Balance=Balance-"&Amt&" "
&_
                "Where CustAcctNo=" & acctno & ""

                Set rstupdpaybal=conne.Execute(updpayingbal)

                'update receiving account balance
                updreceivingbal= "UPDATE CustAcct Set Balance=Balance+"&Amt&""
& _
                "Where CustAcctNo="& ReAcctNo &""

                Set rstupdrebal=conne.Execute(updreceivingbal)

          txtMsg.value="paid successfully"
        End if

        conne.close
```

```
End Sub

Sub ChangeDate()

  orgdate=Date()
  dd=CStr(DatePart("d",orgdate))
  mm=CStr(DatePart("m",orgdate))
  yy=CStr(DatePart("yyyy", orgdate))
  dd1=CInt(dd)
  mm1=CInt(mm)

  if dd1 > 0 And dd1 <= 9 then
    dd="0"+dd
  elseif mm1 > 0 And mm1 <= 9 then
    mm="0"+mm
  end if

  chdate=mm+"/"+dd+"/"+yy
  txtPayDate.value=chdate
  txtDueDate.value=chdate

End Sub

Sub thisPage_onshow()

        Call ChangeDate()

End Sub

%>
</head>
```

```
<BODY bgColor=linen>
<center><br>
<img SRC="../images/PayBill.gif" WIDTH="334" HEIGHT="43">
</center>

<p align=center>
<FONT color=mediumslateblue face=""><STRONG><FONT color=navy face=""
size=4>Paying Accounts    </FONT></STRONG>
<FONT color=chocolate></FONT>
</p>

<p >
<TABLE align=center border=1 cellPadding=1 cellSpacing=1 width=75%
bgColor=#a0d1f8 borderColor=#800000 borderColorDark=#800000
borderColorLight=#808080 style="BACKGROUND-COLOR: #a0d1f8">
  <TR>
    <TD align=left colSpan=2 style="BACKGROUND-COLOR: peachpuff; TEXT-
ALIGN: left;  width=50%><P><FONT face="" size=2><FONT face="" size=3><FONT
      color=mediumslateblue face=""><STRONG>Receiving</STRONG></FONT>
      <STRONG><FONT color=mediumslateblue face="">Account
No.</FONT></STRONG>
    </TD>
    <TD colSpan=2 width=50% style="BACKGROUND-COLOR: #eea0ad">
    </TD>
  <TR>
    <TD style="BACKGROUND-COLOR: peachpuff"><STRONG><FONT
color=mediumslateblue face="">Due Date</FONT></STRONG></TD>
    <TD style="BACKGROUND-COLOR: #eea0ad">
    </TD>
</TR></TABLE>
```

```
<INPUT id=reset1 name=reset1 type=reset value=Reset> <b> 
</p>
<p align=center><strong>
   <font face="Arial" size="4">
   <A href="../MainMenu.asp">Back</A>
</font></strong></p></B></FONT>
</BODY>
<% 'VI 6.0 Scripting Object Model Enabled %>
<% EndPageProcessing() %>
</FORM>
</HTML>
```

File Name: CheckBalance.asp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<META content="text/html; charset=unicode" http-equiv=Content-Type>
<%

'validate the input data
UserID=Session("UserID")
Passwd=Session("Passwd")

Set connection = server.CreateObject("ADODB.connection")
connection.Open "OraScottDSN","scott","tiger"
```

```
'sql
sqlCheckIn="SELECT * " & _
        "FROM Customers " & _
   "WHERE UserID = '"&UserID&"' and " & _
   "CustPasswd ='"&Passwd&"' "

Set rstCheckIn = connection.Execute(sqlCheckIn)

'check if the input sid exists
If rstCheckIn.EOF Then
        str = server.URLEncode("UserID or/and PassWord are incorrect! Try it again.")
   Response.Redirect("../CheckBalance/Response.asp?message=" & str)
End if

Set connection = server.CreateObject("ADODB.connection")
connection.Open "OraScottDSN","scott","tiger"

'sql checking if the user exists
sqlBalance="select A.CustAcctNo, A.AcctType, A.Balance " & _
   "FROM CustAcct A  , Customers C  " & _
   "WHERE UserID = '"&UserID&"'  AND " & _
   "CustPasswd = '"&Passwd&"'  AND " & _
   "C.CustSIN=A.CustSIN "

Set rstBalance = connection.Execute(sqlBalance)

If rstBalance.EOF then

        str = server.URLEncode("You have not had an account yet.")
        Response.Redirect("Response.asp?message=" & str)
else
```

```
%>
<BODY bgColor=linen>
<center><br>
<img SRC="../images/Check1.gif" WIDTH="334" HEIGHT="43">
</center>
<p><center><table border="1" bgColor=#efa0ac align=center>
   <TBODY style="BACKGROUND-COLOR: #eea0ad">
 <tr Align="middle">
 <td width="30%" style="BACKGROUND-COLOR: peachpuff; WIDTH:
30%"><P><font Size="3" face=""><FONT
        color=mediumslateblue><FONT
        face="Times New Roman"><FONT><STRONG>Account No.</FONT>
        </STRONG></P></FONT></FONT></FONT></td>
 <td width="30%" style="BACKGROUND-COLOR: peachpuff; WIDTH: 30%"><font
Size="3" face=""><FONT
        color=mediumslateblue face="Times New Roman"><STRONG>Account
Type</FONT></STRONG></font></td>
 <td width="30%" style="BACKGROUND-COLOR: peachpuff; WIDTH:
30%"><FONT color=mediumslateblue face="Times New Roman"
        size=3><STRONG>Balance</STRONG></FONT></td>
</tr>

<%
Do Until rstBalance.EOF
%>
<tr>
<td Align="middle"><font Size="2" face="Arial">
        <%= rstBalance("CustAcctNo") %></font>
<td Align="left"><font Size="2" face="Arial">
```

```
        <%= rstBalance("AcctType") %></font>
<td Align="left"><font Size="2" face="Arial">
        <%= rstBalance("Balance") %></font>
        <%
        rstBalance.MoveNext

Loop
        connection.close
End If
%>
</FONT></td>
</tr></TBODY>
</table></center>

<p align=center><strong>
  <font face="Arial" size="4">
  <A href="../MainMenu.asp">Back</A>
</font></strong></p></B>
</BODY>
</HTML>
```

File Name: TranHistory.asp

```
<%@ Language=VBScript%>
<% 'VI 6.0 Scripting Object Model Enabled %>
<!--#include file="../_ScriptLibrary/pm.asp"-->
<% if StartPageProcessing() Then Response.End() %>
<FORM name=thisForm METHOD=post>
```

```
<HTML>
<HEAD bgColor=linen>
<META content="text/html; charset=unicode" http-equiv=Content-Type>
</head>
<BODY bgColor=linen>
<center><br>
<img SRC="../images/trans.gif" WIDTH="334" HEIGHT="43">
 </center>

<p align=center><STRONG><FONT color=mediumslateblue face="" size=4>Statement
Period</FONT></STRONG>
</p>
<p align=center><FONT color=mediumslateblue face=""
size=4><STRONG>Account </STRONG></FONT>
<p>

<%
Sub bntGo_onclick()

  Accts = Trim(lstAccts.getValue(lstAccts.selectedIndex))
  Period= CLng(Trim(lstPeriod.getValue(lstPeriod.selectedIndex)))

  Select Case Period
   Case 6
    NewDate=DateAdd("d", -6, Date())
   Case 14
    NewDate=DateAdd("d", -14, Date())
   Case 29
    NewDate=DateAdd("d", -29, Date())
  End Select
```

```
        Set conne=Server.CreateObject("ADODB.Connection")
      conne.Open "OraScottDSN","scott","tiger"

        'look up custacctno corresponding to accts
        sqlcustacctno="select custacctno from custacct A, customers C " & _
        " where (UserID ="&Trim(Session("UserID"))&"' and " &_
        " CustPasswd ="&Trim(Session("Passwd"))&"'"&_
   " and C.CustSIN=A.CustSIN and AcctType="&Accts&")"

        Set rstacctno= conne.Execute(sqlcustacctno)

        acctno=CLng(trim(rstacctno("custacctno")))

        'get corresponding records from the Payment table
   transql="SELECT PayingAcctNo,ReceivingAcctNo,PayDate, BillNo," & _
   "PaidAmount from Payment " & _
   "Where (PayingAcctNo="& acctno &" or " & _
   "ReceivingAcctNo="& acctno &") and
PayDate>To_date('"&NewDate&"','MM/DD/YY')"

   Set tranrecord = conne.Execute(transql)

   If tranrecord.EOF Then
%>

   <p>
   <table border="1" bgColor=#efa0ac align=center borderColorDark=lightslategray
borderColorLight=white>
   <TBODY style="BACKGROUND-COLOR: #eea0ad">
         <tr>
```

```
        <td align="middle" style="BACKGROUND-COLOR:
 peachpuff"><STRONG><FONT color=mediumslateblue face=""
          style="BACKGROUND-COLOR: peachpuff">No
          transaction during the period.</FONT></STRONG>
        </td>
       <%
        conne.close
        Exit Sub
      ELSE
      %>
          <p>
          <table border="1" bgColor=#efa0ac align=center borderColorDark=lightslategray
 borderColorLight=white>
          <TBODY style="BACKGROUND-COLOR: #eea0ad">
          <tr Align="middle">
            <td width="50" style="BACKGROUND-COLOR:
 peachpuff"><STRONG><FONT color=mediumslateblue face="">Date
          </FONT></STRONG></td>
            <td width="70" style="BACKGROUND-COLOR:
 peachpuff"><STRONG><FONT  color=mediumslateblue face=""
          style="BACKGROUND-COLOR:
 peachpuff">Description</FONT></STRONG></td>
            <td width="70" style="BACKGROUND-COLOR:
 peachpuff"><STRONG><FONT color=mediumslateblue face=""
          style="BACKGROUND-COLOR: peachpuff">From</FONT></STRONG>
 </td>
            <td width="70" style="BACKGROUND-COLOR:
 peachpuff"><STRONG><FONT color=mediumslateblue face=""
          style="BACKGROUND-COLOR: peachpuff">To</FONT></STRONG> </td>
```

```
        <td width="50" bgColor=#eea0ad style="BACKGROUND-COLOR:
peachpuff"><STRONG><FONT color=#7b68ee  style="BACKGROUND-COLOR:
peachpuff">Amount</FONT></STRONG></td>
        </tr>
  <%
        Do Until tranrecord.EOF
              fpayingno=tranrecord.Fields("PayingAcctNo")
              freceivingno=tranrecord.Fields("ReceivingAcctNo")
              fdate=tranrecord.Fields("PayDate")
              fdesc=tranrecord.Fields("BillNo")
              famount=tranrecord.Fields("PaidAmount")

  %>
<tr><td>
        <font Size="3">
        <%= tranrecord("PayDate") %>
        </font></td>
      <td Align="middle">
        <font Size="3">
        <%= tranrecord("BillNo") %>
        </font></td>
      <td Align="middle" bgColor=#eea0ad>
        <font Size="3">
        <% =tranrecord("PayingAcctNo")%>
        </font></td>
 <td Align="middle">
        <font Size="3">
        <%=tranrecord("ReceivingAcctNo")%>
        </font></td>
<td Align="middle">
        <font Size="3">
```

```
            <%=tranrecord("PaidAmount")%>
            </font></td>
        <%
            tranrecord.MoveNext
    Loop
End If
    conne.close
End Sub
%>

</tr></TBODY></table>

<p align=center><strong><font face="Arial" size="4">
    <A href="../MainMenu.asp">Back</A>
</font></strong></p>

</BODY>
<% 'VI 6.0 Scripting Object Model Enabled %>
<% EndPageProcessing() %>
</FORM>
</HTML>
```

File Name: ChangeAddr.asp

```
<%@ Language=VBScript %>
<% 'VI 6.0 Scripting Object Model Enabled %>
<!--#include file="../_ScriptLibrary/pm.asp"-->
<% if StartPageProcessing() Then Response.End() %>
```

```
<FORM name=thisForm METHOD=post>

<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<script ID="serverEventHandlersVBS" LANGUAGE="vbscript" RUNAT="Server">

Sub thisPage_onshow()

        Call Btn_Query_onclick()

End Sub

Sub Btn_Query_onclick()

        TheID=Session("UserID")
        Pass =Session("Passwd")

        sqlCustomer = "Select * From Customers  " & _
                "Where UserID = '"&TheID&"' AND " & _
                "CustPasswd = '"&Pass&"' "

        rstCustomer.setSQLText(sqlCustomer)
        rstCustomer.open

End Sub


Sub Btn_Modify_onclick()

        addr= Trim(txtAddress.value)
```

```
        phon= Trim(txtPhone.value)

        TheID=Session("UserID")
        Pass =Session("Passwd")

        IF addr = "" then
                txtMessage.value="The address is required"
                exit sub
        ElseIF phon = "" then
                txtMessage.value="The phone no. is required"
                exit sub
        end if
        txtMessage.value=""

        Set connection=Server.CreateObject("ADODB.Connection")
        connection.Open "OraScottDSN","scott", "tiger"

        sql_Modify="UPDATE Customers SET " & _
        " CustAddr='" &addr& "', " & _
        " CustPhone='" &phon& "' " & _
        " WHERE UserID= '"&TheID&"' AND " & _
        " CustPasswd = '"&Pass&"' "

        Set rstChangeAddr=connection.Execute(sql_Modify)

   txtMessage.value="Your address and/or phone no. updated"

End Sub

</script>
</head>
```

```
<body bgColor="linen">

<p align="center"><strong>
<font color="maroon" face="Arial" size="5"></font></strong>

<IMG height=40 src="../images/Addres.gif" width=401>

<p>
<table border="1" cellPadding="1" cellSpacing="1" height="123" style="HEIGHT:
147px; WIDTH: 372px" width="65.74%" align="center" bgColor=#a0d1f8
borderColor=#800000 borderColorDark=#808080
borderColorLight=#800000><TBODY style="BACKGROUND-COLOR: peachpuff">
  <tr>
    <td align="right"
    style="BACKGROUND-COLOR: peachpuff; COLOR: mediumslateblue">
      <DIV align=left><STRONG>SIN</STRONG></DIV></td>
    <td style="BACKGROUND-COLOR: #eea0ad">
    </td></tr>
  <tr>
    <td align="right" style="BACKGROUND-COLOR: peachpuff"><DIV
align=left><STRONG><FONT color=mediumslateblue
      face="Times New Roman" size=3
      style="BACKGROUND-COLOR:
peachpuff">Name</FONT></STRONG></DIV></td>
    <td style="BACKGROUND-COLOR: #eea0ad">
</td></tr>
 <TR>
  <TD align=right style="BACKGROUND-COLOR: peachpuff"><DIV
align=left><STRONG><FONT color=mediumslateblue face=""
      style="BACKGROUND-COLOR:
peachpuff">Gender</FONT></STRONG></DIV></TD>
```

```
    <TD style="BACKGROUND-COLOR: #eea0ad"></TD></TR>
   <TR>
    <TD align=right style="BACKGROUND-COLOR: peachpuff"><DIV
 align=left><STRONG><FONT color=mediumslateblue
          face="">Date of Birth</FONT></STRONG></DIV></TD>
    <TD style="BACKGROUND-COLOR: #eea0ad"></TD></TR>
    <tr>
      <td align="right" style="BACKGROUND-COLOR: peachpuff"><DIV
 align=left><STRONG><FONT color=mediumslateblue
          face="">Address</FONT></STRONG></DIV></td>
      <td style="BACKGROUND-COLOR: #eea0ad">
 </td></tr>
   <tr>
      <td align="right" style="BACKGROUND-COLOR: peachpuff"><DIV
 align=left><STRONG><FONT color=mediumslateblue
          face="">Phone</FONT></STRONG> <STRONG><FONT
 color=mediumslateblue face="">No</FONT></STRONG>.</DIV>
      <td style="BACKGROUND-COLOR: #eea0ad"></td>
 </tr></TBODY></table></p>

 <p align=center><strong>
   <font face="Arial" size="4">
   <A href="../MainMenu.asp">Back</A>
 </font></strong></p>

 </body>
 <% 'VI 6.0 Scripting Object Model Enabled %>
 <% EndPageProcessing() %>
 </FORM>
 </html>
```

File Name: Password.htm

```
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<title></title>
</head>

<body bgColor=linen>
<p>
<center><img src="../images/Passwd0.gif" WIDTH="393" HEIGHT="40">
</center>

<form action="Password.asp" method="post" id="form1" name="form1">
<div>
<blockquote>
   <p align="left"><strong><FONT
   face=Arial><FONT color=darkblue>
   </FONT></FONT></strong> </p>
   <p align="left"><strong><FONT
   face=Arial
color=darkblue>          Please
fill the below fields to change your password.
  </FONT></strong></p>
   <blockquote>
      <blockquote>
         <table border="0" cellPadding="1" cellSpacing="1" width="75%">
            <tr>
               <td><strong><FONT color=seagreen face=Arial>Current
Password:</FONT></strong></td>
```

```
                <td>
                  <input id="password1" name="OldPass" type="password"></td></tr>
              <tr>
                <td><font color="navy"
 face="Arial"><strong>      <FONT
                  color=chocolate face="">New
                  Password:</FONT></strong></font></td>
                <td>
                  <input id="password2" name="NewPass1" type="password"></td></tr>
              <tr>
                <td><font face="Arial"><strong><FONT
                  color=chocolate><FONT>Confirm
                  Password</FONT></FONT></font><FONT><FONT
                  color=chocolate>:</FONT></FONT></STRONG></td>
                <td>
                  <input id="password3" name="NewPass2"
 type="password"></td></tr></table></blockquote>
      <blockquote> </blockquote>
      <blockquote>
        <table border="0" cellPadding="1" cellSpacing="1" height="36"
style="HEIGHT: 36px; WIDTH: 190px" width="29.5%" align=center>
          <tr>
            <td><input id="submit1" name="submit1" style="HEIGHT: 28px; WIDTH:
75px" type="submit" value="Change"></td>
            <td>
<input id="reset1" name="reset1" style="HEIGHT: 28px; WIDTH: 75px" type="reset"
value=" Clear "></td></tr></table></blockquote>
      <p align=center><strong>
   <font face="Arial" size="4">
   <A href="../MainMenu.asp">Back</A>
</font></strong></p>
```

```
</form></BLOCKQUOTE></BLOCKQUOTE></DIV>
</body>
</html>
```

File Name: Password.asp

```
<%
OldPass = Trim(Request.Form("OldPass"))
NewPass1= Trim(Request.Form("NewPass1"))
NewPass2= Trim(Request.Form("NewPass2"))

TheID=Session("UserID")
Pass =Session("Passwd")

If (OldPass="" or NewPass1="" or NewPass2="" ) then

        str = server.URLEncode("Either current or new password is NULL!")
        Response.Redirect("../PassWord/Response.asp?message=" & str)

ELSEIF OldPass <> Pass  then

        str = server.URLEncode("Incorrect old password, try again.")
        Response.Redirect("../Password/Response.asp?message=" & str)

ELSEIf NewPass1<>NewPass2 Then

        str = server.URLEncode("You dont't confirm your new password!")
        Response.Redirect("../Password/Response.asp?message=" & str)
```

```
end if

Response.Write("Student Id:"&TheID&"<br>")
Response.Write("OldPass:"&OldPass&"<br>")
Response.Write("NewPass1:"& NewPass1 & "<br>")
Response.Write("NewPass2:"& NewPass2 & "<br>")

Set connection = server.CreateObject("ADODB.connection")
connection.Open "OraScottDSN", "scott", "tiger"

sql_UpdatePasswd="UPDATE Customers SET " & _
        " CustPasswd= '"&NewPass1&"' " & _
        " WHERE UserID= '"&TheID&"' AND " & _
        " CustPasswd = '"&Pass&"' "

Set rstPasswd=connection.Execute(sql_UpdatePasswd)

        Session("Passwd") = NewPass1

        str = server.URLEncode("Your password have been changed!")
        Response.Redirect("../PassWord/Response2.asp?message=" & str)

%>
```

File Name: Response.asp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
```

```
<HEAD>
<META content="text/html; charset=unicode" http-equiv=Content-Type>
<%= Request.QueryString("message") %><BODY
bgColor=navajowhite><p><center><b>
</b></center>
<P><STRONG><U><FONT face=Arial><A
href="Password.htm">Back</A></FONT></U></STRONG></P>
<P> </P>
</BODY>
</HTML>
```