Account Manager v. 1

By

Kris Wong

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

March 2004

Account Manager v. 1

by

Kris Wong

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Kris Wong	Date
Russ McMahon, Faculty Advisor	Date
James F. Sullivan, Department Head	Date

Table of Contents

Section	Page
List of Figures	ii
Abstract	iii
1. Statement of the Problem	1
2. Description of the Solution	1
2.1. User Profile	2
2.2. Design Protocols	2
2.2.1. Tiers	2
2.2.2. Client Tiers	3
2.2.2.1. User Client Tier	4
2.2.2.2. Administrator Client Tier	6
2.2.3. Middle Tier	8
2.2.4. Database Tier	10
3. Objectives of the Project	13
3.1. Windows Client	13
3.2. Windows 2000 Middle Server	14
3.3. Oracle 9i Database Server	14
4. Proof of Design	14
4.1. Windows Client	14
4.2. Windows 2000 Middle Server	16
4.3. Oracle 9i Database Server	17
5. Testing	17
6. Conclusions and Recommendations	18
6.1. Conclusions	18
6.2. Recommendations	18
Appendix A. Budget	20
Appendix B. Timeline	21

List of Figures

Figure	Page
Figure 1. Multi-Tier Architecture	3
Figure 2. Account Manager Login Screen	4
Figure 3. Account Listing	5
Figure 4. Account Transaction Listing	6
Figure 5. Admin Utility	8
Figure 6. Relationships Between Tables	13

Abstract

Account Manager is an open-source personal bank/credit card account management program. Account Manager presents the user all the tools necessary to keep track of transactions on their accounts, as well as schedule upcoming and repeating transactions. Account Manager is also a multi-tiered, client/server application. Its tiers are comprised of two Windows GUI client interfaces (one for users and one for administrators), a Windows 2000 service as a middle server, and an Oracle 9i database server. The client tier is designed to be easy to use for basic account management, while at the same time offer its users more advanced features for helping them track their finances. The middle server is designed to offer scalability, by keeping resources for each logged in user to a minimum, to allow support for hundreds of users. The Oracle 9i database is designed to offer maximum efficiency through the use of pre-compiled stored procedures. Overall, the project is designed to be scalable and flexible, while at the same time offering ease of use for the end user.

Account Manager v. 1

1. Statement of the Problem

Throughout my career at the University of Cincinnati, I have learned many new skills. As I entered my final year, it became crucial that I prepare myself to enter the job market as an IT professional. It was for this reason that I had to identify a Senior Design project that would challenge me in new ways and teach me skills that went above and beyond what I had learned to that point. The purpose of my final project at the university was not to fulfill the need of a business or an individual, as with many other students, but rather to take my skill set to a whole new level that will allow me to compete in the job market. As I have learned through my own experiences in interviews and talking with potential employers, businesses are looking for individuals that have had hands on experience with the latest technologies.

2. Description of the Solution

Account Manager is an open-source personal bank/credit card account management program. Account Manager presents the user all the tools necessary to keep track of transactions on his/her accounts, as well as schedule upcoming and repeating transactions. Account Manager makes it easy for the user to know what funds he/she has available, which makes budgeting his/her money easier. The complexity of the application is kept to a minimum, while still providing several useful features that enable the user to keep better track of his/her accounts. Some of the features that are included in Account Manager are: support for multiple users, each with their own accounts, the ability to filter transactions based on several different criteria, the ability to transfer funds

between accounts, the ability to add and edit various different transaction types, and more.

Account Manager was developed in C# for the Windows platform. There are currently no other open-source account management utilities developed for the Windows platform. Being open-source leaves the user with flexibility. The user may choose to use the application as distributed, or may to choose to extend the application in any manner he/she desires by modifying the source.

2.1 User Profile

Users of Account Manager fit into one of two categories. There are those users who will take advantage of the application as distributed, and there are those users who will wish to customize the software via modifying the source. For those users who choose to use the software as distributed, Account Manager is designed with ease of use in mind. The average computer user with knowledge of using the Windows operating system should have no problems using the application. The graphical user interface (GUI) is designed to make using the application straight forward and easy to navigate. Account Manager also caters to the more advanced user in allowing him/her to extend the source in any manner he/she pleases. This flexibility is not seen with any other applications of this kind currently available.

2.2 Design Protocols

2.2.1 Tiers

Account Manager is a multi-tiered application, composed of two Windows GUI client interfaces (one for users and one for administrators), a Windows 2000 service as a middle server, and an Oracle 9i database server. All requests from either client are

serviced by the middle server. The middle server may or may not have to communicate with the database server, depending on the request. The communication between the three tiers is illustrated in Figure 1.

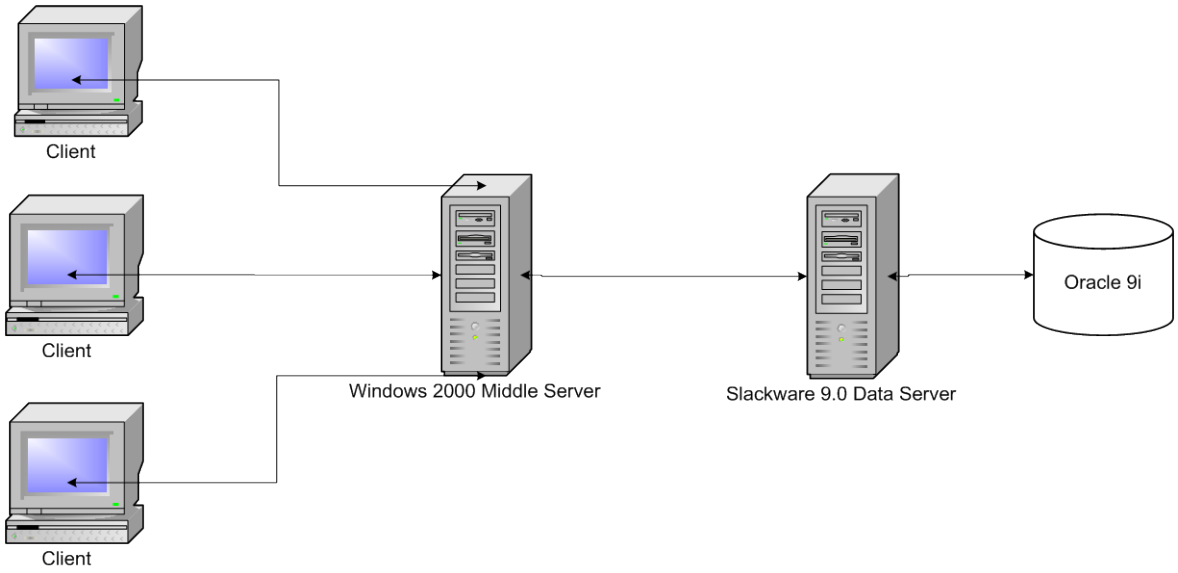

Figure 1. Multi-Tier Architecture

2.2.2 Client Tiers

2.2.2.1 User Client Tier

The user client tier is the only tier that the end user sees. The user client tier is a Windows GUI application, created using the .NET Framework Classes. When the application is first launched, the user is presented with an MDI parent form and a login form (see Figure 2. page 4). Here the user is required to identify himself/herself with a profile name and a password before he/she is able to proceed.

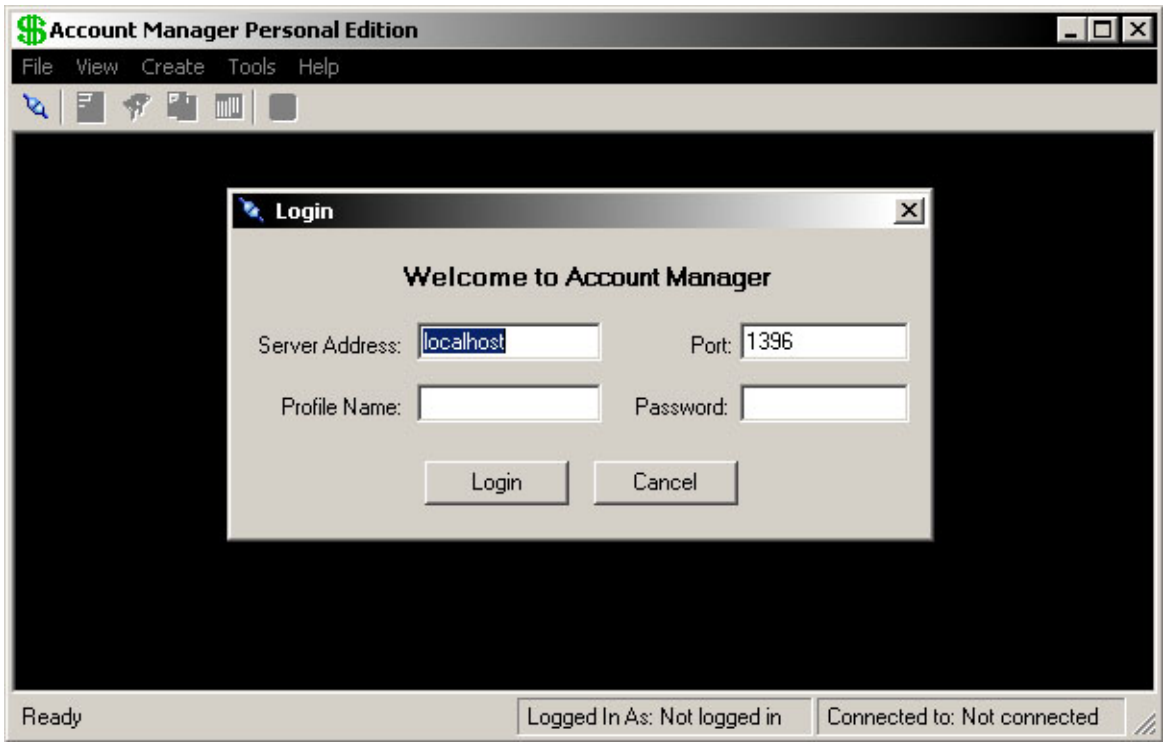

Figure 2. Account Manager Login Screen

Once the user has logged in, a form is displayed that shows a list of accounts the user has entered into Account Manager. The account name, number, type, and current balance are displayed for each account. At the bottom of the form, there is a list of pending transactions. This list contains any transactions that have been scheduled on the current day, or any previous days, that have not been processed. Also on the form are the following buttons: Details – displays a form showing a list of transactions for the currently selected account, New – displays a form to create a new account, Edit – displays a form to edit the currently selected account settings, Delete – deletes the currently selected account, and Reconcile – displays a form used to reconcile the currently selected account. The above is illustrated in Figure 3, page 5.

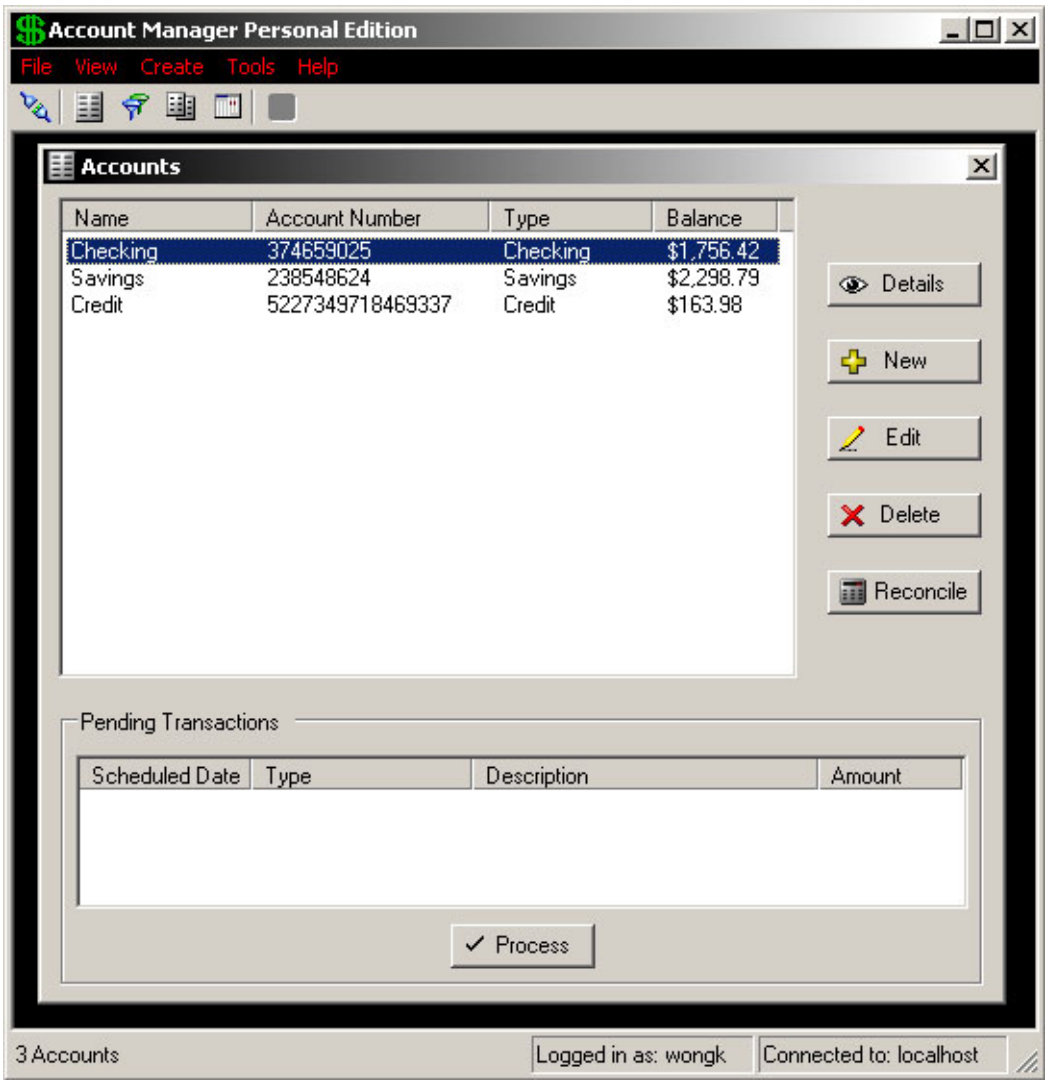

Figure 3. Account Listing

When the user clicks the Details button, a form is displayed that lists the transactions for the selected account. For each transaction the date, type, description, amount, and whether or not the transactions has been marked as cleared are displayed; the current account balance is also displayed on the form. From here the user is able to add a new transaction, edit the selected transaction, or delete the selected transaction. The user is able to filter the transactions using the filters he/she has created for the given account. The user is also able to sort the transactions by clicking on any of the column

headers; clicking on the column header sorts the transactions based on the column and also switches the sort order from ascending to descending or vice versa. The current filter, sort column, sort order, column widths, and window dimensions are all saved to the registry for each account. The next time the form is displayed it will appear exactly the same as when it was closed. The above is illustrated in Figure 4.

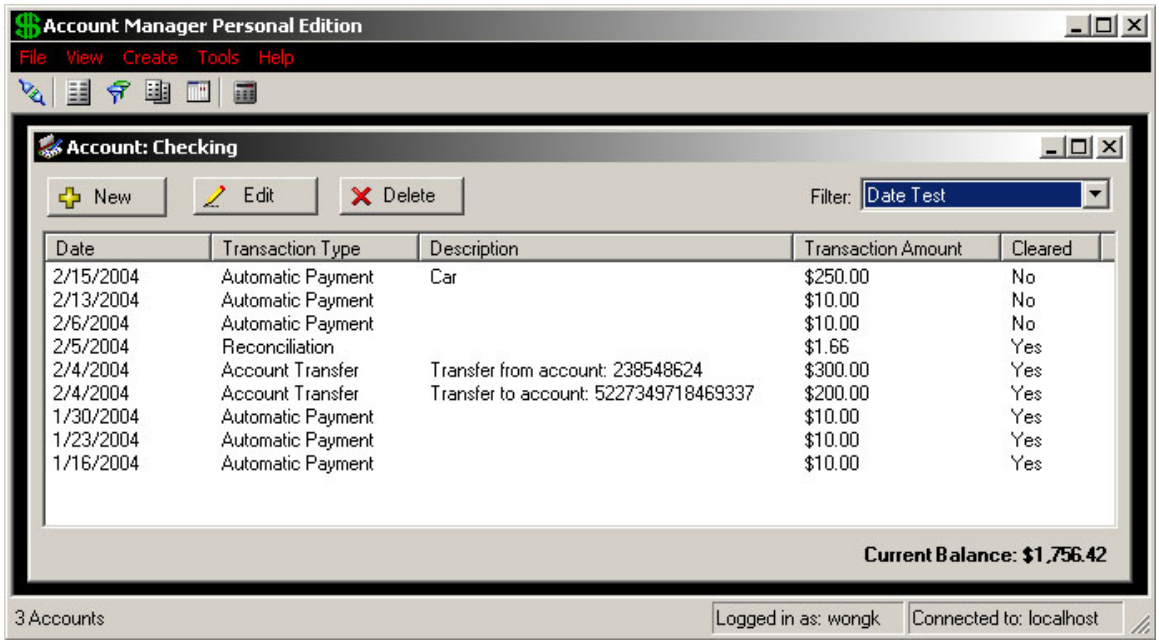

Figure 4. Account transaction listing

2.2.2.2 Administrator Client Tier

The administrator client tier is also a Windows GUI application. The admin utility is used to perform administrative tasks on the Windows 2000 middle server. When the application is started, a notify icon is placed in the system tray. The notify icon that displays is one of two different icons, depending on whether or not the Account Manager Service is running on the machine. If the service is running, an icon with a green dollar sign appears. If the service is stopped, or is not installed on the machine, an icon with a red dollar sign appears. The application checks the status of the service every

second, and if the status has changed it updates accordingly. The application can be used to start and stop the service as well.

When an administrator right clicks on the notify icon, a context menu appears. The first option on the context menu is to show the admin utility. Selecting this option brings up a login form. Logging into the admin utility is the same as logging into the client. The only difference is that an administrator profile and password are required. Administrator profiles and user profiles cannot be interchanged, meaning an administrator cannot login to the client and a user cannot login to the admin utility. If the administrator successfully logs into the admin utility, they are presented with a form that shows a list of profiles currently logged into Account Manager, as well as three buttons (See Figure 5. page 8). There is a button for logging out profiles that are logged into the server, there is a button for refreshing the list of logged in users, and there is a button for managing profiles. When the administrator clicks on the Manage Profiles button, they are presented with a form that displays all of the administrator and user profiles currently in the database. From here they are able to add new profiles, change the password of an existing profile, or delete existing profiles. The only limitation is that they cannot delete their own profile.

Figure 5. Admin Utility

2.2.3 Middle Tier

The Account Manager middle tier runs as a service under Windows 2000. This tier provides several functions for the connected clients, but its main function is to retrieve data from the database backend and communicate this data with the client and vice versa. This tier maintains a cache of logged in users, so a user may not log in more than once. Communication between the clients and the server is achieved using .NET Remoting. .NET Remoting allows the clients to use instances of objects that are actually constructed on the server. The TCP protocol, in edition to a binary message formatter, provides the means for sending messages back and forth between the client and the server. All messages that are sent between the user client and the server are encrypted using the Triple DES encryption algorithm. There are three objects that are remoted, and these three objects provide for all of the communication between the client and the server.

The first object is the CServerSession object. This object is described as a Well-Known Singleton object. This means that the object will be constructed on the server once the first client requests an instance of the object, and subsequent requests from

clients will be given a reference to the original instance. CServerSession inherits from the MarshalByRefObject class, which allows the object to be remoted, and also the IServerSession interface, which is used to describe the available methods to the client. This object has four methods available to the client: Login, Logout, GetUserSession, and GetAdminSession. Login verifies the user's profile name and password, and assuming the user has submitted valid information and the user is not already logged in, it adds the user to the cache of logged in users and lets the client know the login attempt was valid. Logout removes the user from the cache of logged in users. GetUserSession returns a CUserSession object to the client, which provides for all further communication. GetAdminSession returns a CAdminSession object to the client, which is the administrative version of the CUserSession object.

A unique CUserSession object is constructed for each client, and can only be obtained via the CServerSession.GetUserSession method. The CUserSession object has far too many methods available to the client to describe in this document, but each method serves the purpose of exchanging data between the client and the server. CUserSession inherits from its base class CSession, and the IUserSession interface. CSession inherits from MarshalByRefObject, and provides for shared functionality between the CUserSession and CAdminSession objects. This shared functionality is mainly related to object lifetime.

In the land of .NET Remoting, objects can only remain idle for so long before they are destroyed. In the case of Account Manager, objects can only remain idle for 45 seconds before they are destroyed. Both the client and the admin utility contain timers that ping the server every thirty seconds once they have been successfully logged in.

This serves to insure that old objects do not remain on the server for long periods of time if the client would happen to crash for some reason.

As with the CUserSession object, a unique CAdminSession object is created for each administrator. This object can only be obtained via the CServerSession.GetAdminSession method. CAdminSession also inherits from CSession, and from the IAdminSession interface. The CAdminSession object also has too many methods to describe in this document, but all of these methods are for the purpose of managing profiles. It is important to note two things here. When an administrator logs out a logged in profile, it may take up to thirty seconds before the client is actually logged out. This is because the server must wait for the client's next ping request to log the user out. Also, when one administrator is editing a profile, they have a lock on that profile until they are done editing it. No other administrators are able to edit this profile until the lock has been released.

2.2.4 Database Tier

The database tier is an Oracle 9i release 2 database server running on Slackware Linux 9.0. Communication between the middle server and Oracle is provided by the use of the Oracle Data Provider .NET. The tables and the relationships that compose the Account Manager database are as follows:

Profile		
Column Name	**Type**	**Null Allowed**
Name	Varchar(15)	No
Password	Varchar(50)	No
ID (PK)	Number(4,0)	No
Type	Number(1,0)	No

Account		
Column Name	**Type**	**Null Allowed**
Account_Number	Varchar(20)	No
Account_Name	Varchar(50)	No
Type	Number(1,0)	No
Profile (FK)	Varchar(15)	No
Current_Trans_Num	Number(6,0)	No
Current_Check_Num	Number(6,0)	Yes
Initial_Balance	Number(10,2)	No
ID (PK)	Number(4,0)	No

Scheduled_Transaction		
Column Name	**Type**	**Null Allowed**
ID (PK)	Number(4,0)	No
Account_Number (FK)	Varchar(20)	No
Profile (FK)	Varchar(15)	No
Due_Date	Date	No
Type	Varchar(50)	No
Description	Varchar(250)	Yes
Amount	Number(10,2)	No
Category	Number(1,0)	No
Cleared	Char(1)	No
Frequency	Number(1,0)	No
Automatic	Char(1)	No

Transaction_Type		
Column Name	**Type**	**Null Allowed**
ID (PK)	Number(4,0)	No
Name	Varchar(50)	No
Account_Type	Number(1,0)	No
Profile (FK)	Varchar(15)	No

Filter		
Column Name	**Type**	**Null Allowed**
ID (PK)	Number(4,0)	No
Name	Varchar(50)	No
Type	Number(2,0)	No
Account (FK)	Varchar(20)	No
Profile (FK)	Varchar(15)	No

Filter_Args		
Column Name	**Type**	**Null Allowed**
Filter_ID (FK)	Number(4,0)	No
Argument	Varchar(50)	No
Ndx	Number(2,0)	No

The following is the template used to create a new table for every account that is created in Account Manager. The table name follows the format: [profile name]_[account number]. Tables constructed from this template have no relationships, as there is no reason for any to exist.

[Profile Name]_[Account Number]		
Column Name	**Type**	**Null Allowed**
ID (PK)	Number(7,0)	No
Occurred_On	Date	No
Type	Varchar(50)	No
Description	Varchar(250)	Yes
Amount	Number(10,2)	No
Category	Number(1,0)	No
Cleared	Char(1)	No

Figure 6 illustrates the relationships between tables.

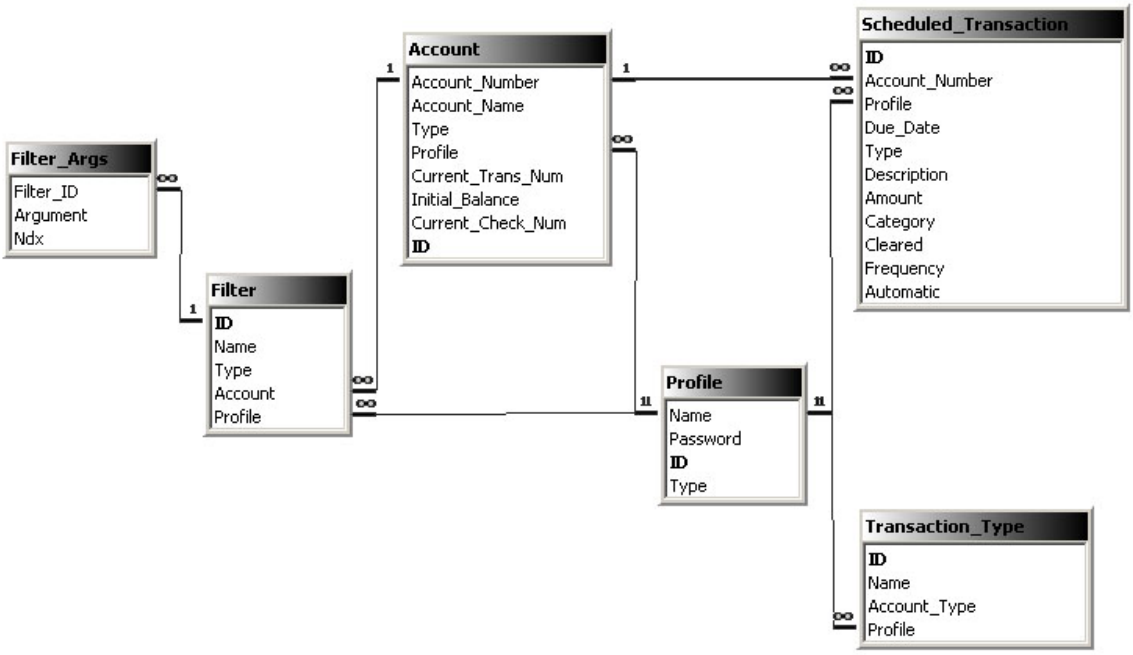

Figure 6. Relationships Between Tables

3. Objectives of the Project

3.1 Windows Client

- A Windows GUI interface developed in C# using the .NET Framework Classes
- Ease of use/navigation for the user
- Uses caching to increase efficiency and reduce network traffic
- Extendibility through being open source
- The ability to manage checking, savings, and credit card accounts, allowing the user to efficiently manage his/her money through a comprehensive feature set as follows:
  - Multiple users can use the application via the use of profiles and passwords
  - Users can manage as many accounts as necessary
  - Users can enter transactions manually for each account, or may schedule transactions that will either be applied automatically or after they are processed by the user
  - Users can transfer funds between any of their accounts
  - Users can create filters that will limit the transactions displayed for any account based on certain criteria
  - Users can edit multiple transactions at one time via the use of filters
  - Users can reconcile their accounts as necessary
  - Users can customize the transaction types that are displayed when adding or editing a transaction

3.2 Windows 2000 Middle Server

- Implemented as a Windows service
- Enables all communication from the client to the database and vice versa
- Provides a way manage profiles via a Windows GUI interface

3.3 Oracle 9i Database Server

- Tables and relationships setup as described in the "Database Tier" section above
- Stored Procedures using PL/SQL for all communication with the Middle Server

4. Proof of Design

4.1 Windows Client

The Windows Client was designed to be both visually appealing and easy to use for the average Windows user. The client has many colorful icons, and its dialogs were designed in a manner that is both appealing and ergonomic. It follows many of the nuances that are common in most Windows applications. The main menu is easy to navigate and contains many of the features of the application. There is a toolbar, which makes many of the features users are most likely to use only one click away. There is also a status bar, which displays useful information to the user.

The source of my application is open for anyone to extend or modify in any manner they desire. My code is well commented and formatted in a manner that is common to most programmers. This makes modifying the source manageable for anyone knowledgeable about the C# language.

Profiles allow the user to use the application from any computer that has the client software installed. There is no limit to the number of accounts that a given user can manage with Account Manager. Users can enter all of their transactions manually, or, if they desire, they can schedule transactions that will be applied once, daily, weekly, every

other week, or monthly. These scheduled transactions can be setup to be applied automatically when the user logs in, or after the user has processed the transaction. Users can transfer funds between accounts, so they do not have to manually enter two separate transactions. Users are able to create several different kinds of filters, depending on what data they want to filter by. These filters limit the transactions that are displayed on the transactions form. This makes it easier to sort through the transactions and find specific ones, especially as the number of transactions entered for the account increases. Using these filters, the user can edit the fields of any number of transactions all at once. For instance, a user is able to create a filter that will only show transactions that have not been cleared. At the end of the month, once the user has received an account statement from his/her bank, he/she can then reconcile the account and use a Multiple Field Edit to mark all of the non-cleared transaction as cleared. Users are able to add as many Transaction Types to their accounts as they desire, as well as edit and/or delete the default Transaction Types. These Transaction Types can be thought of as categories, allowing the user to group similar transactions together.

Along with the previously listed features, the client is also designed as a thick client. The client maintains a cache of all of the user's relevant data for as long as they are logged in. This greatly reduces the number of requests sent to the server, in that only adding new data or editing existing data will result in a request to the server. The client is able to display and use any data previously entered by retrieving it from its local cache. This greatly increases the efficiency of the application.

4.2 Windows 2000 Middle Server

The Windows 2000 Middle Server runs as a service. The service can be specified to start automatically when a user logs onto the machine. The application can only be started or stopped using the Service Control Manager in Windows. The middle server allows for all communication between the client and the database backend. Through the use of .NET Remoting, and the three objects mentioned in the Design Protocols section of this report, the client is able to communicate with the server from anywhere that a network connection can be established. Originally I had planned for the middle server to perform caching for each of its connected clients. After further consideration I decided to move the caching functionality to the client application. This increased efficiency in two ways. First, it greatly cut down on the number of requests that the client was making to the server, allowing the application to execute more quickly and cutting down on network traffic. Second, this cut down on the number of resources required for each logged in client. Using fewer resources makes the server much more scalable, and also reduces its impact on other running applications and services. I had also originally planned for the Windows GUI interface for controlling the server to be a part of the server application. Services, though, are not able to interface with the user in any way. This forced me to create a separate utility for controlling the server, which turned out to be convenient. It allowed me to create a separate Admin Utility that makes monitoring, starting, and stopping the service all easier. This is done through the use of a notify icon that resides in the Widows system tray.

4.3 Oracle 9i Database Server

There were several small changes made to the original table designs as shown in my Design Freeze, but there were no major changes made. The tables are fully normalized and designed to run with maximum efficiency. Stored procedures were also used to increase efficiency within the database itself. Since stored procedures are precompiled, this cuts down on the time required to execute a given line of SQL. This also removed almost all SQL from the server project itself, making the code both easier to follow and to maintain.

5. Testing

Unit testing has been a continual part of the development process. As I have implemented new features, and rewritten old features, I have also done testing to make sure that anything that was added was working as expected, and to make sure that anything else that was affected was still working correctly. Testing was also done by other users, to make sure the application works in other environments, and to make sure that other users feel the application is working as it should be. Other users may sometimes disagree with the developer as to how a certain feature should work. This makes it useful to consider the opinions of other users before a product is released. Testing was also done to verify that all exceptions are being handled correctly, so the application does not crash, and to confirm that all user input is being validated correctly, so that invalid data is not being sent to the database. The application was tested under Windows 98, 2000, and XP.

6. Conclusion and Recommendations

6.1 Conclusions

Account Manager has been designed to enable it users to track transactions on his/her banking and credit accounts, allowing them to more easily keep track of his/her finances. Account Manager offers flexibility in that it may be used as distributed, or the source may be modified to match the needs of the user. The three tiers of the application have been designed to work with maximum efficiency, so users can make better use of their time. All of the project deliverables have been met, and testing has been completed to make sure they have been met correctly.

6.2 Recommendations

This project, while it was successful, was also extremely complex. If I did not have an entire year to work on the project, I would not have been able to complete it. Many implications arose along the way that had to be planned for and/or taken into consideration. First, while Slackware Linux is the most "Unix-like" distribution of Linux, it is also one of the most difficult versions of Linux to administer. Installing Oracle on Slackware also presented a problem, since some of the GUI tools that are distributed with the software to make installation and setup easier would not run. This required me to setup the database manually, which was very complex. Next, I had to learn how to program in an object-oriented manner, after having programmed procedurally in all of my classes. This was a large step for me, one which required me to rewrite much of my code several times.

There are many considerations that must be taken into account when programming using a client/server model. First, I had to learn the ins and outs of .NET

Remoting, which is largely complex. I also had to learn how to incorporate encryption for communication between the client and server. This was also complex and took quite a bit of time. I had to plan how the remoted objects were designed and how they interact. I also had to find a way to keep track of what users are logged in to the application and how to log them out, should the need arise. This had a lot to do with object lifetime and posed quite a few problems. I had to find a way to synchronize all shared data and objects, so there would be no data corruption.

I learned a lot of valuable skills from this project. For any students who are thinking of doing a similar project: the learning curve is high, and it requires a lot of time to correctly plan for everything that must be taken into account.

Appendix A.

Budget

When considering all the tools that I will be taking advantage of, it also important to take into consideration the cost of these tools. The following table illustrates the cost of the required software titles, as well as some hardware considerations.

Product	Cost	Source
Visual Studio .NET Professional Edition	$1,079.00	msdn.microsoft.com
Slackware Linux 9.0	$39.95 (or download free of charge)	store.slackware.com
Oracle 9i Standard Edition	$300.00 per license (minimum of 5 licenses)	oraclestore.oracle.com
Windows 2000 Server	$1,636.00	www.dell.com
Linux Server	$3,583.00	www.dell.com
Workstation	$1,146.00	www.dell.com

The total for the project can obviously vary, depending on the number of required workstations. Also, since Account Manager is scalable, additional hardware can always be purchased at a later date. An estimated total with 5 workstations (considering at least 5 licenses are required for the Oracle Database) would be $13,567.95.

Appendix B.

Timeline

ID	Task Name	Start	Finish	Duration
1	Reasearch	1/6/2003	3/12/2003	9.6w
2	Design and Development	1/6/2003	2/16/2004	58.2w
3	Testing	1/6/2003	2/23/2004	59.2w
4	Write Proposal	2/17/2003	3/12/2003	3.6w
5	Present Proposal	3/12/2003	3/12/2003	.2w
6	Implement Linux OS and Oracle DB Server	3/20/2003	3/31/2003	1.6w
7	Present Prototype	8/25/2003	8/25/2003	.2w
8	Present Final Project	3/1/2004	3/1/2004	.2w