5-1-2005

Implementation of Wireless Connection between Personal Digital Assistant (PDA) and Relational Database.

Wantit Tangrugsasut

Implementation of Wireless Connection between Personal Digital Assistant (PDA) and Relational Database

A Thesis-Equivalent Project
Presented to the
College of Information Science & Technology
Department of Computer Science
and the
Faculty of the Graduate College
University of Nebraska
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Computer Science
University of Nebraska at Omaha

by

Wantit Tangrugsasut

May 2005

THESIS-EQUIVALENT PROJECT

ACCEPTANCE

Acceptance for the faculty of the Graduate College,
University of Nebraska, in partial fulfillmcnt of the
requirements for the degree (Master of Computer Science),
University of Nebraska at Omaha.

Committee

4-21-2005

Peter Wolcott 4/21/05

Mansour

Chairperson

Date 4-21-2005

Implementation of Wireless Connection between Personal Digital Assistant (PDA) and Relational Database

Wantit Tangrugsasut, MS

University of Nebraska at Omaha, 2005

Advisor: Dr. Zhengxin Chen

ABSTRACT

The significant number of mobile devices such as cellular phones, smart phones and PDA usages are increasing rapidly in recent year. Popularity of wireless connection along with the Personal Digital Assistant (PDA); Pocket PC, Palm and Windows CE have been growing significantly as well. Applications have been designed and developed for mobile devices base on the personal and business usage purposes. These applications can be define as *mobile applications.* Even though, the number Pocket PC, Palm or other handheld devices usage are growing, the design and implementation of mobile applications for these devices is still currently a novelty for most researchers, developers and programmers.

In this project, I design, implement and develop a mobile application, which allows users to wirelessly connect SQL server on Pocket PC to SQL server on main database server and exchange data between them. The design of this application involves in the exploring the Pocket PC, which has limited memory, storage space, and no physical keyboard. The design also focuses on developing the user-friendly application interface and efficiently function to exchange data between Pocket PC and data server.

The implementation is based on database synchronization technology called Remote Data Access (RDA). RDA allows mobile application to use SQL statement and Internet-based connectivity for transferring information between two SQL servers on different physical locations. The development of this mobile application used the combination of available technology and commercial products such as Microsoft .Net Technology, SQL database management server and Verizon wireless network infrastructure. This mobile application grants users the ability to communicate with database server remotely wherever within the Verizon Wireless network.

ACKNOWLEDGEMENT

This project is the requirement for the degree of Master of Science in Computer Science, University of Nebraska at Omaha.

I would like to thank my thesis advisor, Prof. Zhengxin Chen, Ph.D. for his guidance, constant support throughout my thesis work and for very useful discussions during the writing of this thesis. I fully appreciate his time, patience, and enthusiasm for this work.

In addition, I would like to thank Prof. Mansour Zand, Ph.D. and Prof. Peter Wolcott, Ph.D. for serving as the project committee members.

This thesis is dedicated to my family members and especially to my grandfather who passed away in March 2004. I thank for their understanding and support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Over recent years, the market for wireless communications has been tremendous growth. Wireless technology has been developed and improved, so now it is capable of reaching virtually every location on the face of the earth. Hundreds of millions of people exchange information every day using pagers, cellular telephones and other wireless equipments. With tremendous success of wireless telephony and messaging services, wireless communication is beginning to be applied to the realm of personal and business computing. No longer bound by the harnesses of wired networks, people will be able to access and share information on a global scale nearly anywhere they venture. Portable or laptop computer has become important equipment in wireless world. Because of its portability, it allows people to send and receive information from remote site while they cannot connect to wired networks. Laptop computers are much more affordable now compare to several years ago and new technologies allow emne to have the capability just like desktop computers and make their size even smaller. However, instead of carrying around a notebook computer, more and more people start using Personal Digital Assistant (PDA) or handheld devices.

Mobile devices, such as laptop computers, Pocket PCs, cellular phones, etc., are now easily affordable, and are becoming more popular in everyday life [Johnson 1996].

In this project, I have designed and implemented mobile application for handheld devices, which allow users to have wireless connection from their Pocket PC to relational database server. The designing and implementing of this application use the combination of commercial products and technologies available in the today world market. Before we start looking at the design and implementation, I will introduce the basic concepts and knowledges that used in this project first.

1.1 Handheld/Mobile Device

If you try to look for the definition of handheld or mobile devices, you will come up with lots of definitions depend on the type of devices and their functionalities, which differ from one to another. However, in this project, I define handheld or mobile devices as any devices that fit in the palm of your hand and come with small touch-screen as input method. Some of these devices equipped with a keyboard and some are keyboardless. Wireless seems to be another functionality you can find in the newer model of handheld devices these days and some even come with telephony characteristics. However, the important feature, which all of these devices have in common, is computing capability by using handheld device operating systems.

During the 1990s, several operating systems or platforms for handheld devices have been developed. The following are some of the leading operating systems, which use widely in today world

- Windows Mobile CE – Pocket PC
- Symbian OS
- Palm OS
- Java 2 Platform, Micro Edition
- BlackBerry

These operating systems are designed specifically for lower processor performance (speed) compare to desktop or laptop PC and they are also designed for RAM limitation, which limit the amount of memory and storage the devices can use.

In this project I have selected Microsoft Mobile Pocket PC operating system, which is an upgraded version of Windows CE operating system version 3.0 manufacture by Microsoft Corporation. There are a lot of debate about the operating systems you can choose for application development so, I came up with the list of advantages and disadvantages, which help me make the decision on the mobile platform selection.

Pocket PC has the following advantages:

- Better performance: faster processor and more memory
- Pocket PC multitasks: Pocket PC can open more than one application at a time
- Pocket PC has more storage compare to other devices
- Pocket PC offered better resolution on the display: most Pocket PCs support a resolution of 240x320 pixels at 16-bit color

It also has the following disadvantages:

- Pocket PC require a speedy processor and more memory to run efficiently compare to other mobile OS
- More expensive than devices with other Mobile OS

Even though, Pocket PC seems to have more advantages than other mobile operating systems, it still has several limitations on mobile devices itself. Due to these limitations (limited computational power, storage, screen size, etc.), the applications, run on mobile operating systems, need to be carefully design, implement and testing. In the next section I will discuss more detail about mobile application and how we can categorize them by looking at their architectural perspective.

1.2 Mobile Application

Mobile applications are any applications that run on handheld devices like PDA, smart phone, Pocket PC or similar devices. These devices are operated by mobile operating system. As I had mentioned before in the last section, there are several limitations in mobile operating system on handheld devices. With these existing limitations, we can claim that handheld devices should not be considered general-purpose computers. So, we cannot expect mobile applications to run complex simulations, compile or execute huge software systems. Even though handheld devices will become increasingly powerful, they still cannot match the computational power and facilities available on typical personal computers.

Most of mobile applications are using client-server architecture, where the mobile client must be fully usable in stand-alone mode, caching data for upload and downloading updates when wireless connection is available. The idea of this design is simple. Clients on handheld devices request information from a server device. The server typically responds with the requested information.

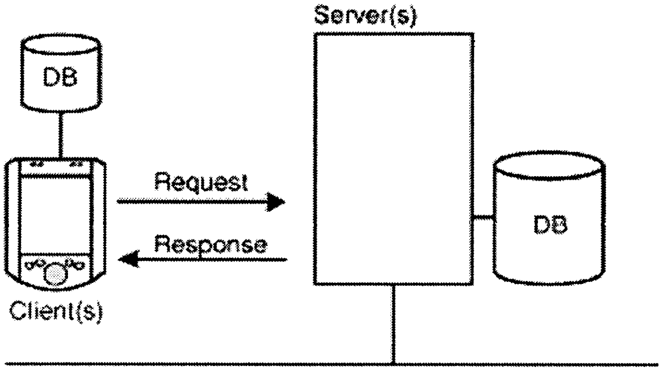


Figure 1.1: Client-Server Architecture for Mobile Application

In client-server application architecture, wireless mobile application can be categorized into several groups depend on the type of connection between application (client) and the back-end (server)

- Stand-alone mobile Application: No connection between mobile application and the back-end server
- Mobile application with connection to the back-end through wireless personal area network
- Mobile application with connection to the back-end through wireless wide area network

1.2.1 Stand-alone mobile Application

Stand-alone mobile application is the application that does not have any connection back to the server. Typically, stand-alone applications are used in situations where communication between client and server cannot be guaranteed so instead of transfer data back and forth between client and server, applications and data are stored on the devices. This type of application depends heavily on the operating system and hardware. The coding can be difficult to release and distribute. The developer may also have to support multiple code versions over multiple devices. Because of the lack of connection between client and server, I cannot use this approach because it does not match the project objective.

1.2.2 Mobile application with connection to the back-end through wireless personal area network

The newest and perhaps most intriguing wireless network is the wireless personal area network (WPAN). A WPAN (wireless personal area network) is a personal area network for interconnecting devices centered around an individual person's workspace; call the "personal operating space" (POS). It is formed by wireless communications among handheld devices in a small physical space, about 10 meters. The device will establish the connection with other devices, which have WPAN capabilities. A key feature in a WPAN is that devices automatically detect, and connect with each other. As the user move around the area with wireless signal, the devices will try to acquire new connection. The connection will be dropped if the devices are out-of-range. WPANs operate at 2.4 GHz frequencies in digital modes to facilitate seamless operation among home or business devices and systems. These devices include Pocket PCs, laptops, cell phones, pagers, printers, or even watches and headphones.

There are several protocols widely used in today wireless market. One of the most famous protocols in WPANs, based on emerging technologies, is BlueTooth. BlueTooth was adapted from IEEE 802.15, defined by IEEE. IEEE 802.15 is a wireless standard, which has characteristic of working well in short-range and small networks within a

Personal Operating Space. This technology is a built-in 2-way radio for short-distance chip set, which include in BlueTooth enabled device. When these devices are turned on, they automatically search for other BlueTooth devices in a relatively small geographical space. If one is found, communications are established. If one of the BlueTooth devices is connected directly or indirectly to the Internet, Internet data is available to the other devices also.

Mobile applications on the handheld device use WPAN connection as a bridge to connect back to the back-end server to upload information directly or indirectly. Because of the range limitation, about 10 meters, in connection, I did not use this approach to complete this project. However, this will be the best approach for "Wireless Local Area Network connection" type project, which requires only short-range connection between handheld device and the back-end server.

1.2.3 Mobile application with connection to the back-end through wireless wide area network

A wireless wide area network (WWAN) is a computer network that extends over large geographical areas using radio, satellite and mobile phone technologies to compete with traditional systems of cable-based networking. The purpose is the same as a Wireless Local Area Network; to share information in real time wirelessly, but because of the greater distances involved, additional hardware, software, protocols, and services are required.

Mobile application with wireless wide area network connection can be categorized into two different types.

- Wireless Internet Mobile Application
- Smart Client Mobile Application

1.2.3.1 Wireless Internet Mobile Application

Wireless Internet Mobile Application use architecture similar to the wired Internet application, which use web pages as application interface and execute program using Internet browser. Web Interface for mobile application is designed a little bit different than normal web application because the resources on handheld devices are limited. The applications themselves need to be smaller and less resource-intense usages. Because of this complication, the Wireless Application Protocol was introduced. The Wireless Application Protocol (WAP) is a specification for a set of communication protocols to standardize wireless content delivery. It was originally developed by a group of vendors in 1997, which included Nokia, Ericsson, Motorola and others. The Wireless Application Protocol uses browser on handheld device called micro-browsers. These browsers come with small file sizes that can accommodate the low memory constraints of handheld devices and the low-bandwidth constraints of a wireless-handheld network. When the users use WAP browser attempting to connect to web site to use mobile application, the handheld device must first find a WAP gateway. The WAP gateway then optimizes the content for wireless mobile applications. Although WAP supports HTML and XML, the WML language (an XML application) is spccifically dcviscd for small scrccns and onc-hand navigation without a keyboard. WML is scalable from two-line text that displays through graphic screens found on items such as smart phones and handheld devices.

WAP also supports WML Script. It is similar to JavaScript, but makes minimal demands on memory and CPU power because it does not contain many of the unnecessary functions found in other scripting languages.

Wireless Internet Mobile Application has the following advantages:

- No software deployment required
- Application upgrade is easier
- Real-time data

But, it also has the following disadvantages:

- Inability to function offline
- Limited user interface
- Security: Wireless Internet Mobile Application usually available for any devices with Internet Browser so anybody can access to the application. To avoid security issue, the addition of security system is required.
- Usage cost: It is expensive to use Wireless Internet Mobile Application since network carriers charge by amount of data application transfer (bytes)

I have decided not to use this approach in this project due to the fact that it is unable to function offline. Users have to work online while using Wireless Internet Mobile Application. Otherwise, the applications will not work until the connection is established.

1.2.3.2 Smart Client Mobile Application

Smart clients are the next generation of mobile applications built on the .NET CE Framework that integrates with Web services. They enable users to work with mobile applications offline or online across mobile environments. However, smart client solutions are not depend on the availability of a network connection. This is particularly important since wireless network coverage is often unreliable and too slow. Smart client take full advantage of information exposed by web services. The applications are easily deployed and managed. It also provides a smart and rich interactive experience by leveraging local resources and intelligently connecting to distribute data sources. If we look at smart clients architecture, we can describe the smart clients as the combination between stand-alone and Wireless Internet mobile application. By using "Push" and "Pull" technology, the core functionality in mobile applications, it gives users ability to transfer data between handheld device and data server through the Hypertext Transfer Protocol (HTTP). Even though smart clients have a high degree of flexibility, they also have disadvantages. These are some of advantages and disadvantages of smart clients.

Smart Client Mobile Application has the following advantages:

- Utilizes local resources: smart client take advantage of the local CPU, memory storage, and any local devices connected to the client, such as a telephone, barcode, and so on. In some situation, it takes advantage of local software or any installed applications that interact with it also.
- Offline capable: Because smart clients are running on the local machine, one of the key benefits that smart client applications offer is that they can be made to work even when the user is not connected.
- Rich user interface: The interface can be designed much more complex using smart clients because application is stored locally on the handheld devices.

- Reduce connection cost: Since application does not have to stay connect all the time, it can help reducing amount of data transfer between client and server.

It also has the following disadvantages:

- Software deployment required
- More complicate to upgrade application
- Use more devices limited resource

Smart clients give much more flexibility to developing and implement the mobile application. Therefore, with the benefits of rich user interface design and offline functionality, smart clients works perfectly with handheld device like Pocket PC. These are the reasons I have decided to use smart client solution for my mobile application implementation and development.

1.3 Purposes and Objectives of the Project

This project focuses on the design and implementation of a smart client mobile application, which allows users to wirelessly connect from their handheld devices (Pocket PC) to the database server and exchange data between SQL server on Pocket PC and main data server. In order to achieve this goal, we have used the following simplified scenario to focus our study; users are the buyers who want to order some products directly from their Pocket PCs at a remote site. The buyers need an application that allows them to add, delete, and modify the order information while the device is offline. After completing the order form, the buyers can send the order(s) information back to the data server at the main office by using function inside the application.

The application needs to be carefully designed, implemented, and tested. The design involves in the exploring the possibility of mobile devices (Pocket PC) in great detail. It is a challenge because of smaller screen size, limited RAM and storage space, and no physical keyboard on handheld devices, which present different design constraints than designing application for a desktop PC. The design also focuses on the application interface. The primary goal about application interface is to create the application that has a user-friendly characteristic as much as possible. In other words, we want to make it simple for user to fill in the information and invoke the transfer information function easily and efficiently.

The implementation is based on commercially available products. By combining Microsoft Visual Basic .Net, Microsoft .Net compact Framework technologies, SQL database server and Verizon wireless service, this mobile application gives users the ability to exchange data with database server remotely.

To test the application, I have developed a web interface, which connects directly to the database server. This web interface will display real time order, customer and item information from the server instantly after receiving data from the mobile application.

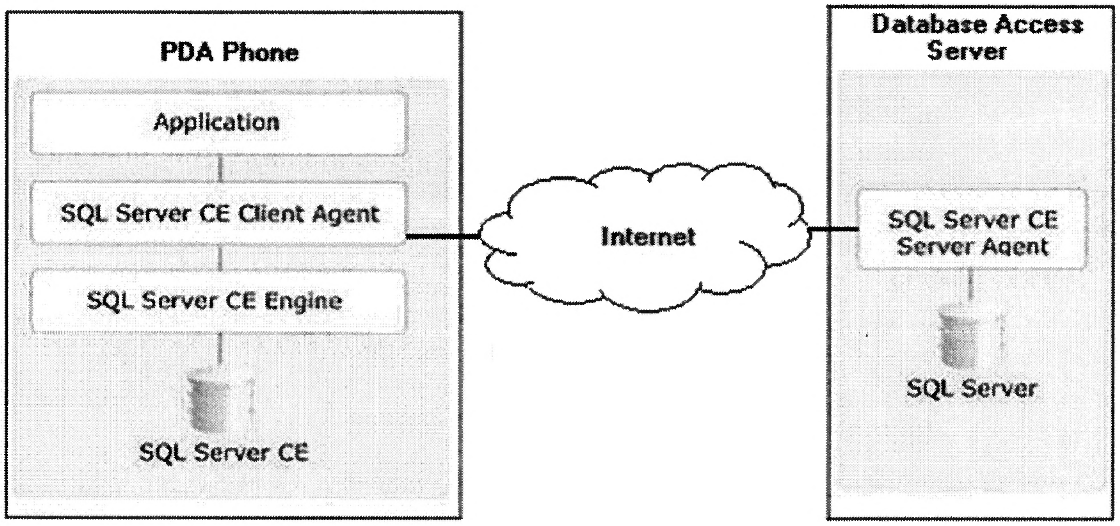


Figure 1.2: Objective Overview

Figure1.2 shows the objective overview for this project.

To be more specific, the objectives of this project are to:

- Explore a methodology of using commercially available products for connecting handheld devices to main SQL database server.
- Design efficient mobile application on handheld devices that allow users to have wireless connection with database server.
- Develop and implement the mobile based on commercially available products and services.
- Test the application is operating efficiently and effectively
- Demonstrate the application functionality

1.4 Organization of this Documentation

This project documentation consists of six chapters. The first chapter includes introduction and discussion about handheld device, mobile application and the objective of this project.

Chapter 2 describes equipments, softwares, and service requirement. In this chapter, I have listed all necessary components used in this project. It includes the description and information about those equipments, softwares, and services individually.

The database structure and application design are described in chapter 3. This chapter contains a great detail of database structure including table structure and relationship among them. The application design section contains a step-by-step layout of the application using flow charts and detailed description of application design goal.

Chapter 4 discuss about the methodology use in this project. It is the review of available technologies, which use and/or can be use in this particular type of project. There are two main parts in that section, Mobile Application Development and Database synchronization. Mobile application development section includes the methods and technologies I used for the project implementation and database synchronization section describes ways to synchronize two or more physical databases.

Application Implementation and Coding are discussed in chapter 5. Application implementation section includes some specific setup and configuration for this project and the coding part contains the detail of special functions and codes used in this development.

Chapter 6 is focusing on Application Testing. This chapter includes the way of testing the data transferring between Pocket PC and main database server. It also includes example of ASP web page and their source code.

Chapter 7 is the conclusion for this project. For more detail, Appendix A (User's Manual) and Appendix B (Source Code) can be found after Bibliography section.

CHAPTER 2: EQUIPMENT, SOFTWARE, SERVICE REQUIREMENT

2.1 Equipment

2.1.1 Personal Digital Assistant (PDA)

I have selected Samsung SPH-i700 to be the handheld device for this project implementation. Samsung SPH-i700 is a Pocket PC 2002 Phone Edition, which run on Intel 300 MHz Intel X Scale PXA250. The i700 also comes with 64MB of RAM and it operates on Windows Mobile for Pocket PCs 2002. It supports CDMA (CDMA2000 1xRTT) wireless protocol with Dual band/single mode, CDMA 800 MHz and 1900 MHz.

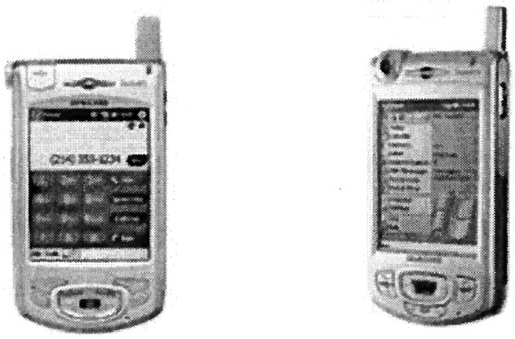


Figure 2.1: Samsung SPH-i700

2.2 Software

2.2.1 Microsoft Visual Basic .Net and Microsoft® .NET Compact Framework

Microsoft Visual Basic .NET is a programming language I have chosen for this mobile application development. Microsoft Visual Basic .NET is the latest version of the Visual Basic development system. Reengineered to be a first-class language on the Microsoft .NET Framework, it provides the productive route for developers to build Windows–based applications, next generation XML Web services, thin-client Web applications, and software for Mobile devices. Visual Basic .NET also allows optional use of new language features, such as inheritance, method overloading, structured exception handling, and free threading which make Visual Basic an object-oriented programming language. Visual Basic .NET fully integrates with the .NET Compact Framework and the Common Language Runtime, which together provides language interoperability, simplified deployment and enhanced security.

Microsoft® .NET Compact Framework is a smart device development platform for the Microsoft .NET initiative. The .NET Compact Framework offer managed code and XML Web services to smart devices, and it enables the execution of secure, downloadable applications on devices such as Pocket PC, mobile phones, and Smartphones.

Because the .NET Compact Framework is a subset of the desktop .NET Framework, developers can easily reuse existing programming skills and existing codes throughout the device, desktop, and server environments.

Additional discussion on Microsoft Visual Basic .Net and Microsoft® .NET Compact Framework will be provided in Mobile Application Development Methodology section.

2.2.2 Microsoft SQL Server 2000 Windows® CE Edition

Microsoft SQL Server 2000 Windows CE Edition (SQL Server CE) is an enterprise-level database management system that I use for store data on handheld device. These data is retrieved and stored directly by the mobile application until send back to main database server. SQL Server CE is the compact database for rapidly developing applications that extend enterprise data management capabilities to mobile devices. It is supporting Structured Query Language (SQL) syntax and providing a development model consistent with SQL Server. The SQL Server CE engine exposes an essential set of relational database features, such as an optimizing query processor and support for transactions and assorted data types, while maintaining a compact footprint that preserves precious system resources.

This project uses SQL Server CE version 2.0 which designed to integrate with the Microsoft .NET Compact Framework by means of Microsoft Visual Basic .NET, simplifying database application development for smart devices. Using the new SQL Server CE data provider to manage code, mobile application can be built highly extensible with offline data management capability for disconnected scenarios.

2.2.3 Microsoft SQL Server 2000 Personal Edition

Microsoft SQL Server 2000 is the complete database and analysis offering for delivering the next generation of scalable e-commerce, line-of-business and data warehousing solutions.

It offers fully Web-Enabled Query, analyze and manipulate data over the Web. Access data securely from a browser, through firewalls, and perform fast full-text searches of formatted documents. It divides database workload to achieve scale-out of applications. Server operating system support up to 32 CPUs and 64 GB of RAM. It performs sophisticated data mining on customer and financial data. Reduce development time with the integrated T-SQL debugger, and allow user to develop functions that can be reused in different applications.

I use this SQL server for storing data on the main database server. These data are the order and customer information, which received from the mobile application on user's Pocket PC. These data also can be reviewed by using web site, which I have created for testing purposes, directly connect to SQL server.

2.2.4 Microsoft ActiveSync 3.7.1

Microsoft ActiveSync 3.7.1 is the synchronization software for Windows Mobile-based Pocket PCs and Smartphones with personal computer. ActiveSync automatically detects the speed of the selected port and updates PC serial port to match it. It also automatically detects and configures the port that is used to connect a Pocket PC to personal computer, whether a serial, USB, or infrared port is selected. In this project, ActiveSync is used in the process of deploying the mobile application from personal computer to Pocket PC.

2.2.5 Internet Information Services (IIS)

Internet Information Services (IIS) is web server, come with Microsoft Windows Operating System (only selected version). IIS provides a highly reliable, manageable, and scalable Web application infrastructure and helps organizations increase Web site and application availability. I use IIS as a web server in this project only for testing purposes. The objective of IIS is to help verifying information that received from Pocket PC in the real time environment by display that information on ASP web pages.

2.3 Service

2.3.1 Verizon Wireless Service

Verizon Wireless service is used as the wireless data transfer provider for the project. Verizon Wireless service offers data transfer plan for wireless access without a separate ISP account. The data transfer rates speed may vary depend on the availability and location. However, Verizon indicates that the minimum transfer rate for data exchange is 50 Kilo Bit per Second (Kbps) and with bursts up to or beyond 144 Kbps.

Verizon Wireless infrastructure uses CDMA standard. The CDMA (Code Division Multiple Access) is the 2G digital cellular standard in the United States. Several calls can share a 1.25 MHz channel within a specific frequency of the radio spectrum simultaneously by assigning a unique coding (Walsh code) to each respective signal. The standard operates on the IP protocol packet switch system, which provides a constant "on" connection. It also relies upon the base station transmitter to transfer while the handset telephone is mobile. The present utilization is for digital signal telephone PCS (Personal Communication Services) on a 1.9 GHz radio frequency. There are two standards of CDMA: IS-95A has a data transfer rate of 14.4 Kbps, while IS-95B is capable of data transfer of 115 Kbps.

Verizon Wireless uses 3G standard, which is also known as CDMA-2000 in some locations. The CDMA-2000 standard was developed by Qualcomm in the United Statcs. It provides data transfer initially at 14.4kbps, with a target rate of 153 kbps, eventually rising to 2 to 2.5 Mbps on a 1.25 MHz channel.

CHAPTER 3: METHODOLOGY

In this section, I have included two main topics. The first is the methodology of mobile application development and the second is methodology of database synchronization.

3.1 Mobile Application Development

In this project, I have selected Microsoft Visual Basic .NET and Microsoft .NET Compact Framework as the tools for mobile application development. Microsoft Visual Basic .NET, the next generation of Visual Basic, is designed to be the most productive tool for creating .NET applications, including Windows applications, Web applications and mobile application.

Visual Basic.NET provides productivity in several principal advantages for mobile application developer. The first advantage is verbosity. Visual Basic.NET allows developer to use English-like syntax when programming and coding the application. It makes source codes easier to read and understand. Visual Basic.NET also provides the easy way to deploy mobile application through Microsoft ActiveSync. After setup the connection between Pocket PC and Visual Basic .NET, Microsoft ActiveSync will copy executable program every time mobile application developer compiling the program.

The easiest way to develop and test a Smart Device Application is to use an emulator. Emulator for the Pocket PC 2002 platform is offered by Visual Basic .NET to help in testing and executing program without the actual handheld device. I use the emulator quite often in this project. It makes the development easier and more convenient. According to the product documentation, each emulator takes up an additional 64MB of

RAM from application development server so at least 256 MB memory is required. They also support two serial ports and one parallel port, which the developers can map to these ports directly from application development server.

Microsoft Visual Basic .NET uses .NET Compact Framework technology, which is a subset of the .NET Framework. Microsoft .NET Compact Framework allows programmers to have the same flexibility in using the languages familiar to .NET Framework and reusing the knowledge of the .NET Class library. However, do note that not all of the classes and methods in the .NET Framework are supported in the .NET Compact Framework Class library.

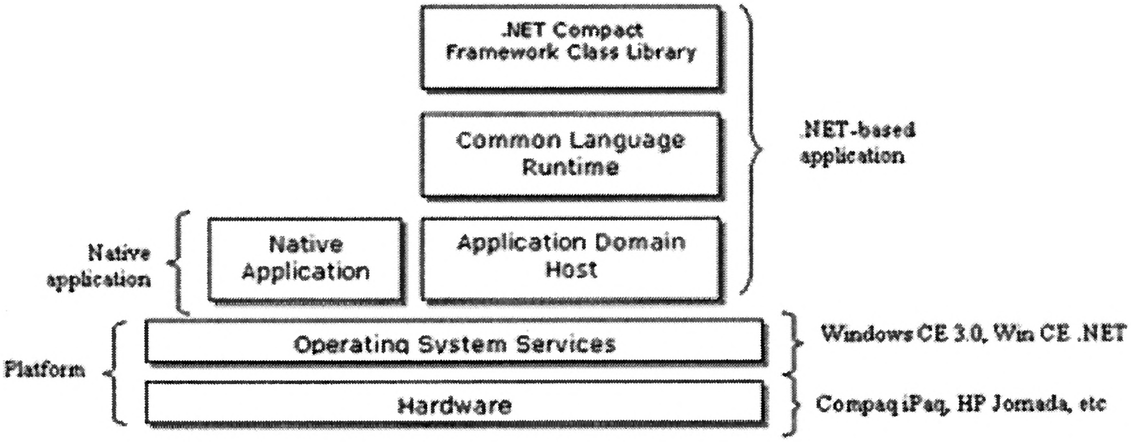


Figure 3.1: Components in the .NET Compact Framework

Figure 3.1 shows the components in the .NET Compact Framework. The .NET Compact Framework contains two elements: a unique class library and the CLR. The class library has been pared down to accommodate the limitations of most mobile devices. It also incorporates classes that take advantage of the extra functionality afforded by the devices.

The CLR has multiple functions: garbage collecting, memory management and Just-In-Time (JIT) compilation during program execution. Also, it performs essential functions such as security and error checking.

The .NET Compact Framework supports two languages for mobile applications: C# and Visual Basic .NET. Visual Studio also supports Device Kits (also known as Device SDKs), which allows you to add in other devices as they appear on the market.

Visual Basic .Net and .Net Compact Framework are a perfect choice for mobile application development. They also make data synchronization easier and more efficient. The next section will cover the methodology of database synchronization and the data synchronization method used in this project.

3.2 Database Synchronization

This project uses two separate sets of tables to store data in two separate physical locations. The first local is on main database server and the other is on user's Pocket PC. These two sets of table have identical database structure and need to be able to communicate to each other. In this section, I will discuss about the connection and interaction between the two.

SQL Server CE 2.0 provides two primary ways of connecting to back-end SQL Server databases.

- Remote Data Access (RDA)

- Merge Replication

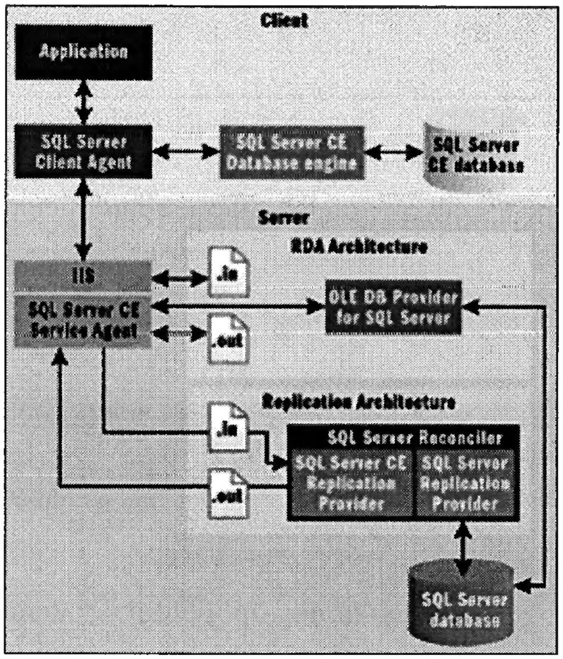


Figure 3.2: Connectivity Options

Figure 3.2 shows the flow of database synchronization options between SQL server and SQL server CE. AS you see in the figure, the application is connected to SQL server CE database through SQL server client agent and database engine. This connection is locally established in Pocket PC by .Net Compact Framework. SQL server client agent is also used for communicating with SQL server agent. The two agents communicate to each other to perform database synchronization.

3.2.1 Remote Data Access (RDA)

RDA provides a simple way for a mobile application to access SQL Server data located in a remote SQL Server. RDA can be used whether the handheld device is continuously connected or intermittently connected to the SQL Server system. It supports connectivity from SQL CE to SQL Server 7.0 or SQL Server 2000 databases.

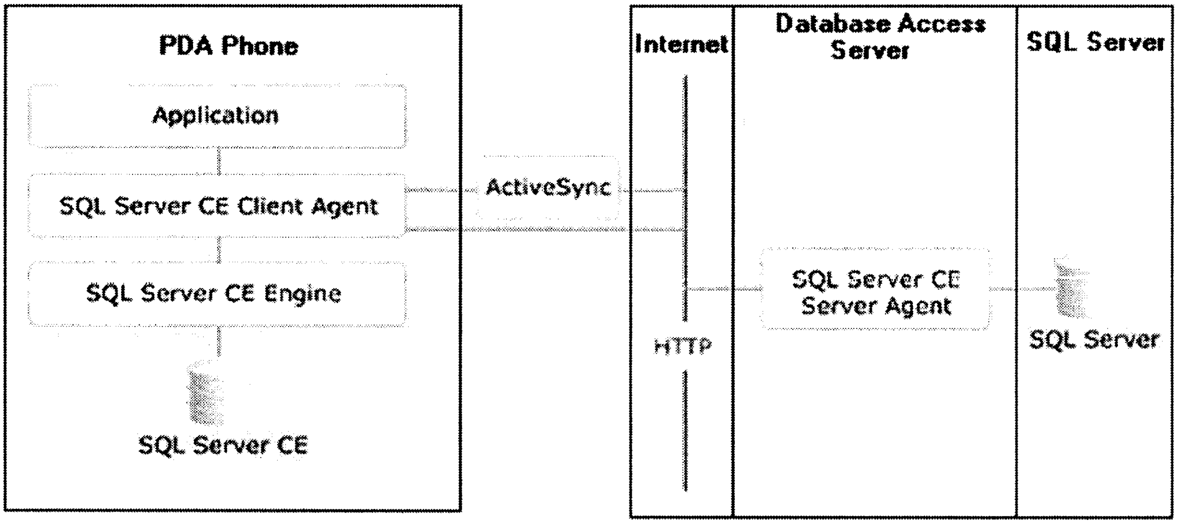


Figure 3.3: RDA Connection

I have selected RDA as the connectivity type. RDA provides a simple way for a mobile application to pull data from a remote SQL Server database table and store it in a local device database table. The application can then read and update the local database table. SQL Server CE can optionally track all changes that are made to the local table. Using this information, the application can later push the changed records from the local table back to the SQL Server table.

The smart client mobile applications can also use RDA to submit SQL statements to be executed on a remote SQL Server database. For example, an application could submit SQL statements that insert, update, or delete records to a remote SQL Server table. RDA is Internet-based so SQL Server CE can communicate with main SQL Server databases through IIS. By connecting via IIS, RDA takes advantage of IIS authentication and authorization services, which supports anonymous and basic authentication as well as

Secure Sockets Layer (SSL) encryption. In addition, since the communication protocol is HTTP, the machine running SQL Server can be located behind a firewall and can be accessed through a publishing rule using Proxy Server.

To reduce the amount of transmitted data, RDA uses compression. This makes RDA well suited to wireless transports. Optionally, encryption can be used to help safeguard sensitive user data. RDA also has a mechanism to handle communication failures. If a failure should occur, retransmission will resume from the last successfully transmitted message buffer. RDA control provides programmatic access to a SQL Server 2000. RDA access is provided by the combination of Visual Basic .NET and compact Framework which both are used in this project.

3.2.2 Merge Replication

Merge replication is the other type of database synchronization offered by SQL CE server. Merge replication in SQL CE server is based on SQL Server 2000 merge replication and is ideally suited to handheld and portable devices because it enables autonomous data updates on the devices and the server. The data on the device and the server can later be synchronized when a connection is re-established.

Usage scenarios for merge replication include read-only replication; data capture and upload; and replicate, update, and synchronize. Most applications will use the combination of these replication alternatives. For this project, mobile application might use read-only replication to download item table to Pocket PC, relying on data capture and upload to Pocket PC and easily upload them back to a server. Therefore, the mobile application might use replicate, update, and synchronize to download customer and orders information, enabling information to be updated on the device. The resulting data could then be uploaded to the server.

Merge Replication can be used in this project. However, I did not use this approach because of two particular reasons. The first reason is that merge replication on SQL CE server is more complex to setup than RDA. In merge replication, the setup and configuration has to be done in both side of SQL database server while RDA can be setup and used within the mobile application program coding. The second reason, I have chosen RDA over merge replication, is that merge replication requires more computing and storage resources. If Pocket PC has no extra storage card installed, then the memory requirements of SQL Server CE and its data could increase more chance of failure or generate unknown problems.

CHAPTER 4: DATABASE STRUCTURE AND APPLICATION DESIGN

It is very important that application design must be fully defined and the structure of the database must be decided before any application implementation can take place. In this project, the designing can be separated into three main areas. They are database relationship, database structure, and designing of mobile application. These three areas must be design thoroughly to make sure that they work as expected.

4.1 Database Relationship

I start database relationship design by using the entity-relationship (ER) data model to map the meaning of database relationship onto a conceptual schema.

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. [Chapple 2003]

The following is ER diagram for this project.

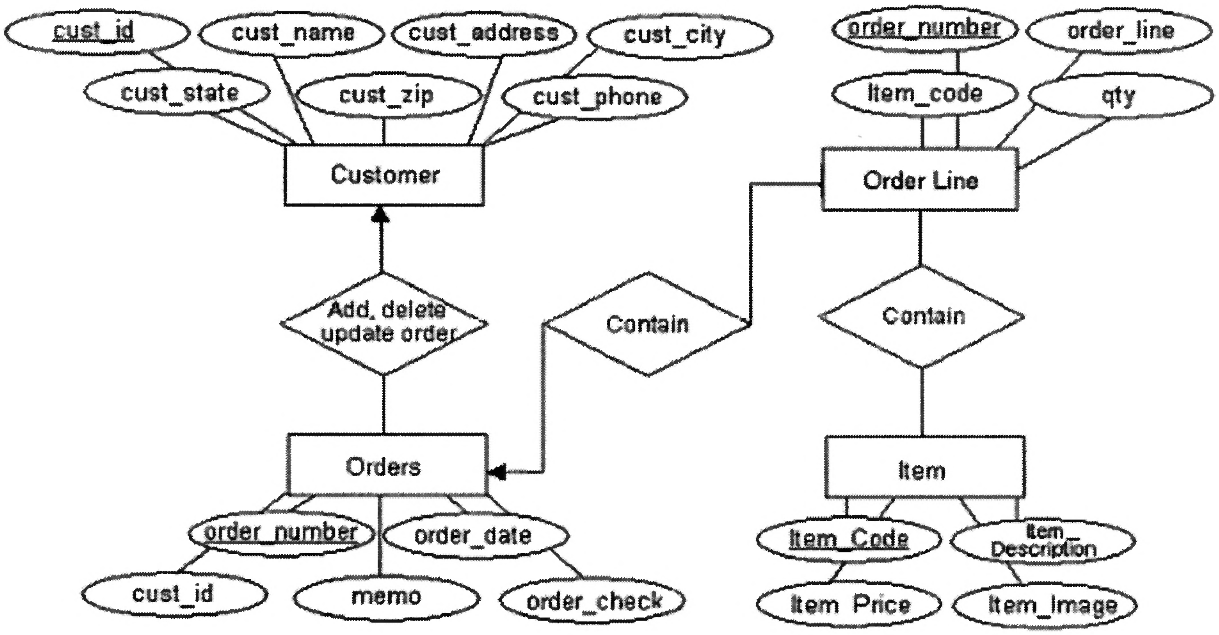
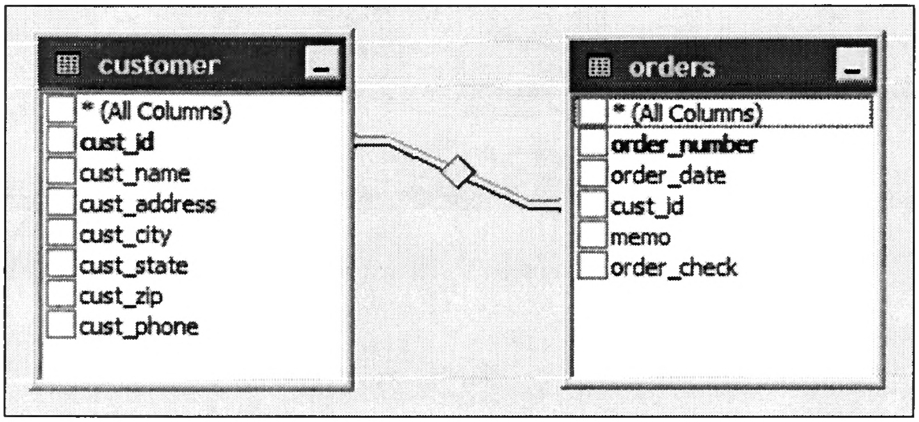


Figure 4.1: ER Diagram for the Project Database Structure

4.1.1 Customer and Orders tables

Customers are the people who add, delete, and update orders. The relationship between customer table and orders table join by cust_id in customer table and cust_id in orders table. It is one to many (1:N) relationship. That means customer can order as many as they want but an order only belong to one customer.



customer

□ * (All Columns)
□ cust_id
□ cust_name
□ cust_address
□ cust_city
□ cust_state
□ cust_zip
□ cust_phone

orders

□ * (All Columns)
□ order_number
□ order_date
□ cust_id
□ memo
□ order_check

Figure 4.2: Customer table and Orders table relationship

4.1.2 Orders and Orderline tables

In this mobile application, the customer is allowed to create orders that have more than one item. Orderline is the table that contains the detail, item information and item quantity, for each order. For example, order-number "101" contains 5 items so order-number "101" will have 5 order lines. The relationship between orders table and orderline table join by order_number in orders table and order_number in orderline table. It is one to many (1:N) relationship which represents one order can contain multiple order lines but each order line only belongs to single order.

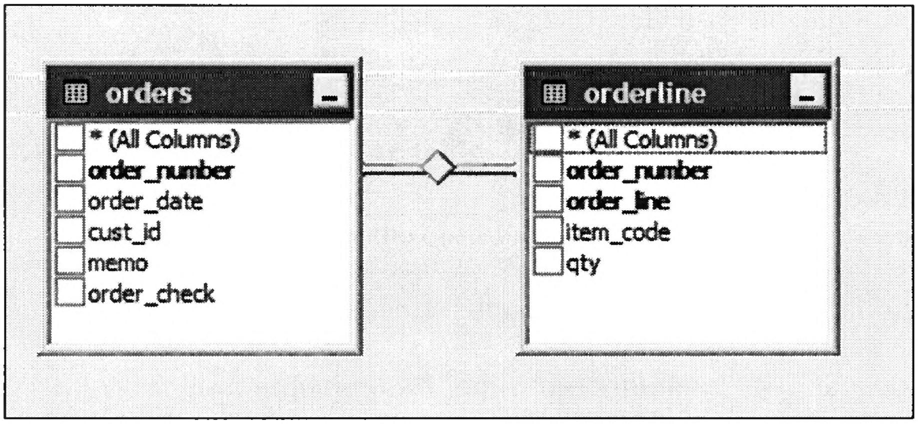


Figure 4.3:Orders table and Orderline table relationship

4.1.3 Orderline and Item tables

As I mentioned before in the last section, orderline is the table that contains the detail, item information and item quantity, for each order. The item information in orderline table is item code. The orderline table and item table are related by item_code in orderline table and item_code in item table. It is one to many (1:N) relationship. It mean that one order line can link the given item code to item table for more information about item itself.

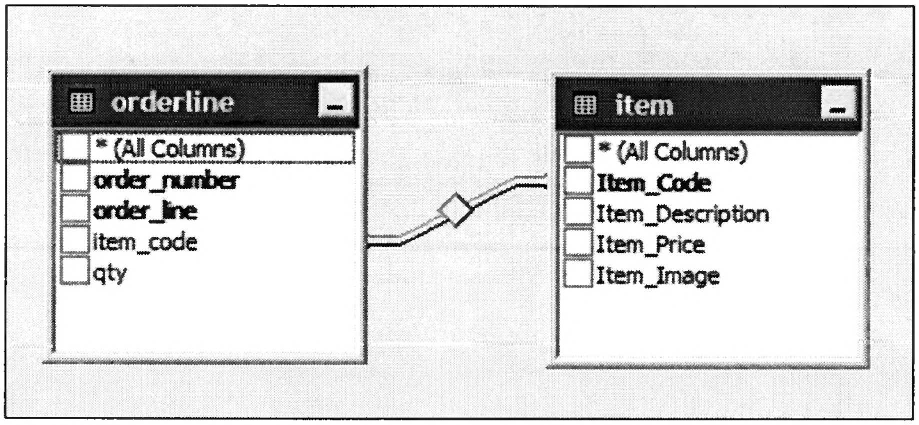


Figure 4.4 :Orderline table and Item table relationship

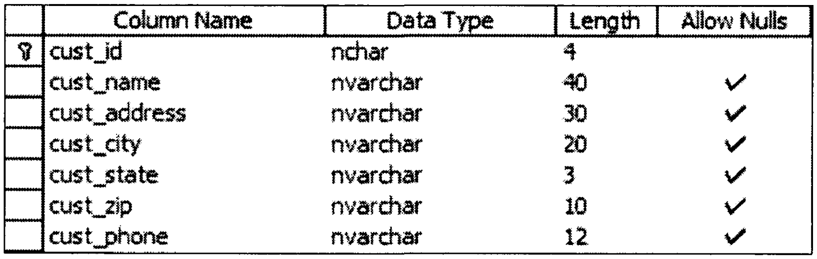
4.2 Database Structure

In the project, all information is stored in Microsoft SQL database. On the database server, the information store in Microsoft SQL Server 2000 Personal Edition and on Pocket PC, the information store in Microsoft SQL Server 2000 Windows® CE Edition. These two locations contain two separate set of table however they are identical database structure. The database structure consists of 4 tables.

- Customer
- Items
- Orders
- Order Line (orderline)

4.2.1 Customer table

Customer table contains information about customers. It contains 7 different fields.



	Column Name	Data Type	Length	Allow Nulls
🔑	cust_id	nchar	4	
	cust_name	nvarchar	40	✓
	cust_address	nvarchar	30	✓
	cust_city	nvarchar	20	✓
	cust_state	nvarchar	3	✓
	cust_zip	nvarchar	10	✓
	cust_phone	nvarchar	12	✓

Figure 4.5: Customer table design

Cust_id: Customer identification number: This number is automatically generated by the application. This field can contain up to 4 characters and it does not allow null value. It usually contains 4-digit number.

Cust_name: Customer name: This field contains customer's first name and last name. This field can contain up to 40 characters.

Cust_address: Customer street address: This field contains customer's street number, name and apartment number if applicant. This field can contain up to 30 characters.

Cust_city: Customer city: This field contains city where customer's street address is in.

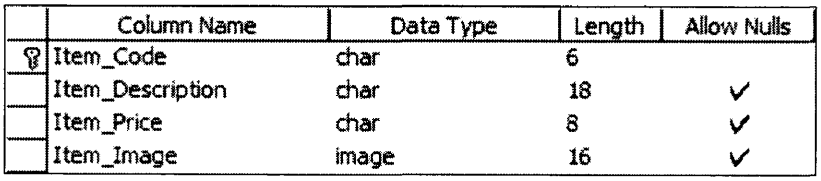
Cust_state: Customer state: This field contains state where customer's city is in. This field contain state abbreviation only, not a whole state name.

Cust_zip: Customer zip code: This field contains zip code where customer's address is in. This field can contain up to 10 characters. However, it usually contains 5-digit number.

Cust_phone: Customer phone number: This field contains customer's telephone number. This field can contain up to 12 characters. It usually contains an area code and a telephone number.

4.2.2 Item table

Item table contain information about customers. It contains 4 different fields.



	Column Name	Data Type	Length	Allow Nulls
🔑	Item_Code	char	6	
	Item_Description	char	18	✓
	Item_Price	char	8	✓
	Item_Image	image	16	✓

Figure 4.6: Item table design

Item_Code: Item identification number or code: This number is automatically generated by application. This field can contain up to 6 characters and it does not allow null value. It usually contains 5-digit number.

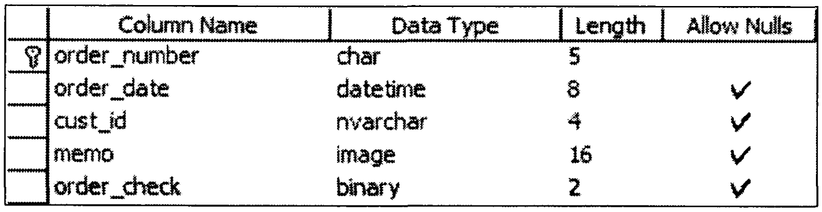
Item_Description: Item description: This field contains item description including type of item, brand and model. This field can contain up to 18 characters.

Item_Price: Item price: This field contains item price in dollars with two decimal.

Item_Image: Item Image: This field contains item image in binary large object (BLOB) format. BLOB is a sequence of bytes with a size ranging from 0 bytes to 2 gigabytes store in relational database. It is representing of an object, such as audio, image, video, etc. data, by using binary stream of data.

4.2.3 Orders table

Orders table contain information about customers. It contains 5 different fields.



	Column Name	Data Type	Length	Allow Nulls
🔑	order_number	char	5	
	order_date	datetime	8	✓
	cust_id	nvarchar	4	✓
	memo	image	16	✓
	order_check	binary	2	✓

Figure 4.7: Orders table design

Order_number: Order number: This number is automatically generated by the application. This field can contain up to 5 characters.

Order_date: Order date: This field contains the date which order is sent back to the data server. It is filled in automatically generated by the application.

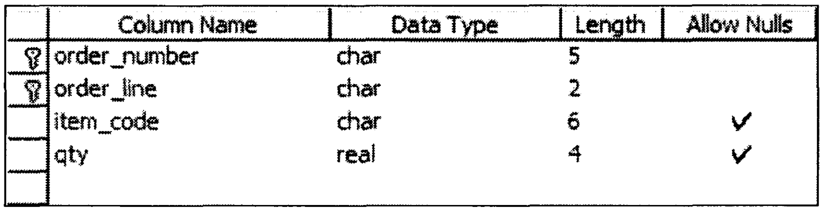
Cust_id: Customer name: This field contains customer's identification number of the customer who creates the order.

Memo: Order special note. This field allows customer to write special note that attach to each order as image file. The image is store in a binary large object (BLOB) format.

Order_check: Order check: This field contains information regarding the order status. "Null" represents the new order that has not yet been sent out to the data server and "1" represents the order that has already been sent out to the data server.

4.2.4 Order Line table

Order Line table contain information about customers. It contains 4 different fields.



	Column Name	Data Type	Length	Allow Nulls
🔑	order_number	char	5	
🔑	order_line	char	2	
	item_code	char	6	✓
	qty	real	4	✓

Figure 4.8: Orderline table design

Order_number: Order number: This number is automatically generated by the application. This field can contain up to 5 characters.

Order_line: Order Line Detail: This field contains line number of detail for particular order number. This field can contain up to 2 characters and it is automatically generated by the application.

Item_code: Item identification number or code: This field contains item code, which order by the customer. This field can contain up to 6 characters and these numbers have to exist in the item table.

Qty (Quantity): Item Quantity: This field contains quantity of an item that customer order. The default value is 1.

The database structure design can be a lot more complicated in large-scale project. However, it is still necessary and recommended because it helps simplifying in application design and reducing difficulty in tracing problems in testing stage.

4.3 Application Design

After I have defined database structure, I can now start the application design. The term *application design* contains merely every single part of development, from data structure layout, flow charts, and entity-relationship diagrams to code layout, documentation, and anything in between. However, in this project the application design contains a step-by-step layout of the application using flow charts and detail descriptions.

The designing of the application starts with the idea of what the application is supposed to do. In this project, the application is supposed to provide user ability to do the following.

- Add/Delete/Update customer information
- View item information
- Add/Delete/Update order information
- Add/Delete item to order line.
- View order detail.
- Send customer and order information back to main database server.

4.3.1 Add/Delete/Update Customer Information

In this application, the users are allowed to add new customers in to the Customer database. They can also modify customer information and delete customer out of the database.

Add Customer: The users are allowed to fill in customer information such as name, address and telephone number. The system will generate and assign a customer identification number to the newly add customer.

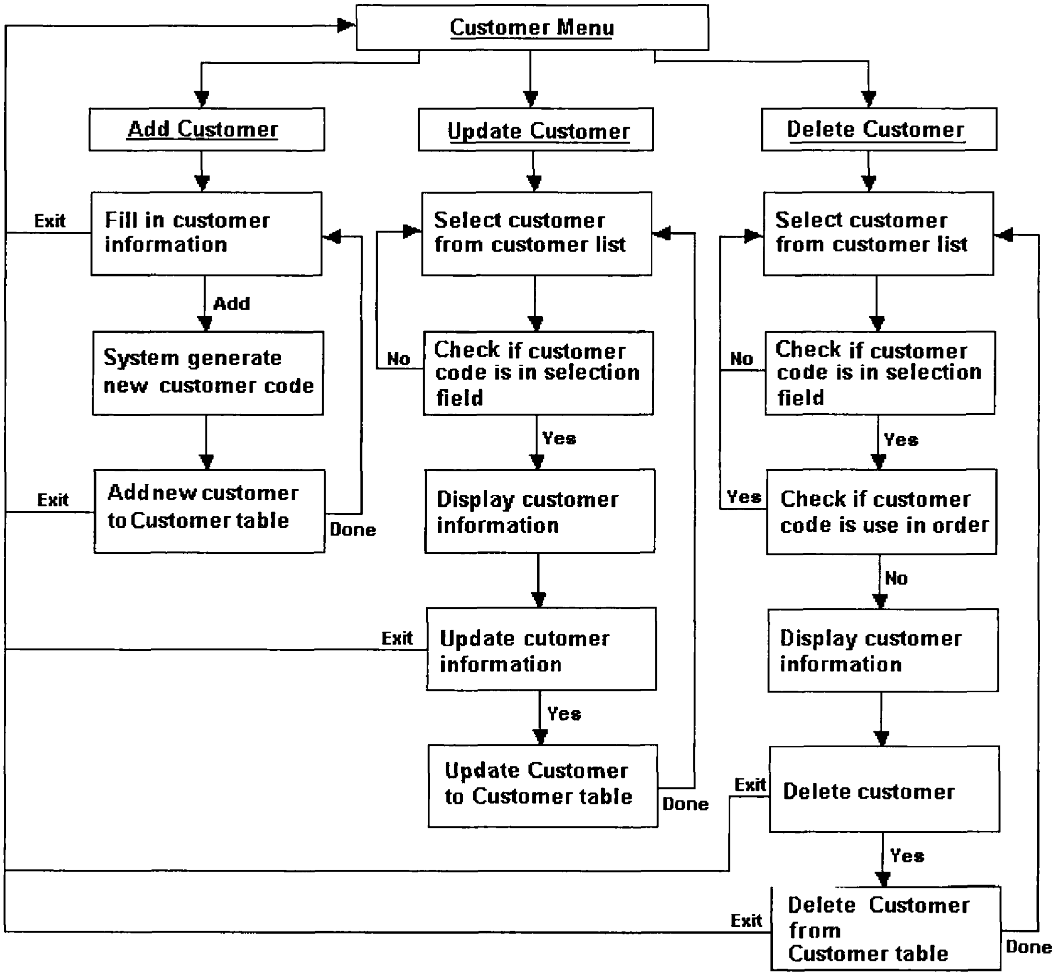

Customer Menu
Add Customer
Update Customer
Delete Customer
Exit
Fill in customer information
Add
System generate new customer code
Exit
Add new customer to Customer table
Done
Select customer from customer list
No
Check if customer code is in selection field
Yes
Display customer information
Exit
Update cutomer information
Yes
Update Customer to Customer table
Done
Select customer from customer list
No
Check if customer code is in selection field
Yes
Yes
Check if customer code is use in order
No
Display customer information
Exit
Delete customer
Yes
Exit
Delete Customer from Customer table
Done

Figure 4.10: Customer Application Design

Update Customer: The users are allowed to update customer information by selecting the customer from the customer list. The current information will display on the screen. At this time, the users can modify customer information and save it. The data will be updated to the Customer table.

Delete Customer: The users are allowed to delete by selecting the customer from the customer list. The current information will display on the screen. At this time, the users can delete customer. The users are only allowed to delete the customer who does not make any orders. If the user try to delete the customer who had already orders, the system will display "The customer cannot be deleted" message.

4.3.2 View Item Information

The users are allowed to view item information such as description, price and also image. By selecting the item code or item description from the list, the item information will display on the screen. Just like in the real system, the user will not allow adding, updating or deleting this information.

4.3.3 Add/Delete/Update Order Information

Add, Delete and Update order information are the main database functions in this application. The modified information will be sent back to the main database server for further process.

Add Order: The users are allowed to create new order by selecting customer from the list. Then they can use the memo box for special instruction. The memo box is a free-hand writing, which allows user to write or draw anything. This memo will be saved as an image file (Binary Large Object) into database. Then the next step is to add item to the order. This will be covered in add item to order line section.

Update Order: The users are allowed to select new customer in the existing order. For example, if user has added the new order and selected the wrong customer, the user can use the update order function to change customer to the right one, instead of delete order and recreate it again. This also allows user to add more item to the order and delete item out of the order. Further information about adding and deleting item will cover in add and delete item to order line section.

Delete Order: The users are allowed to delete order by selecting order number from order number list. The list is showing only the order, which has not been sent to the main database server. The user will not allow deleting the order that has been submitted.

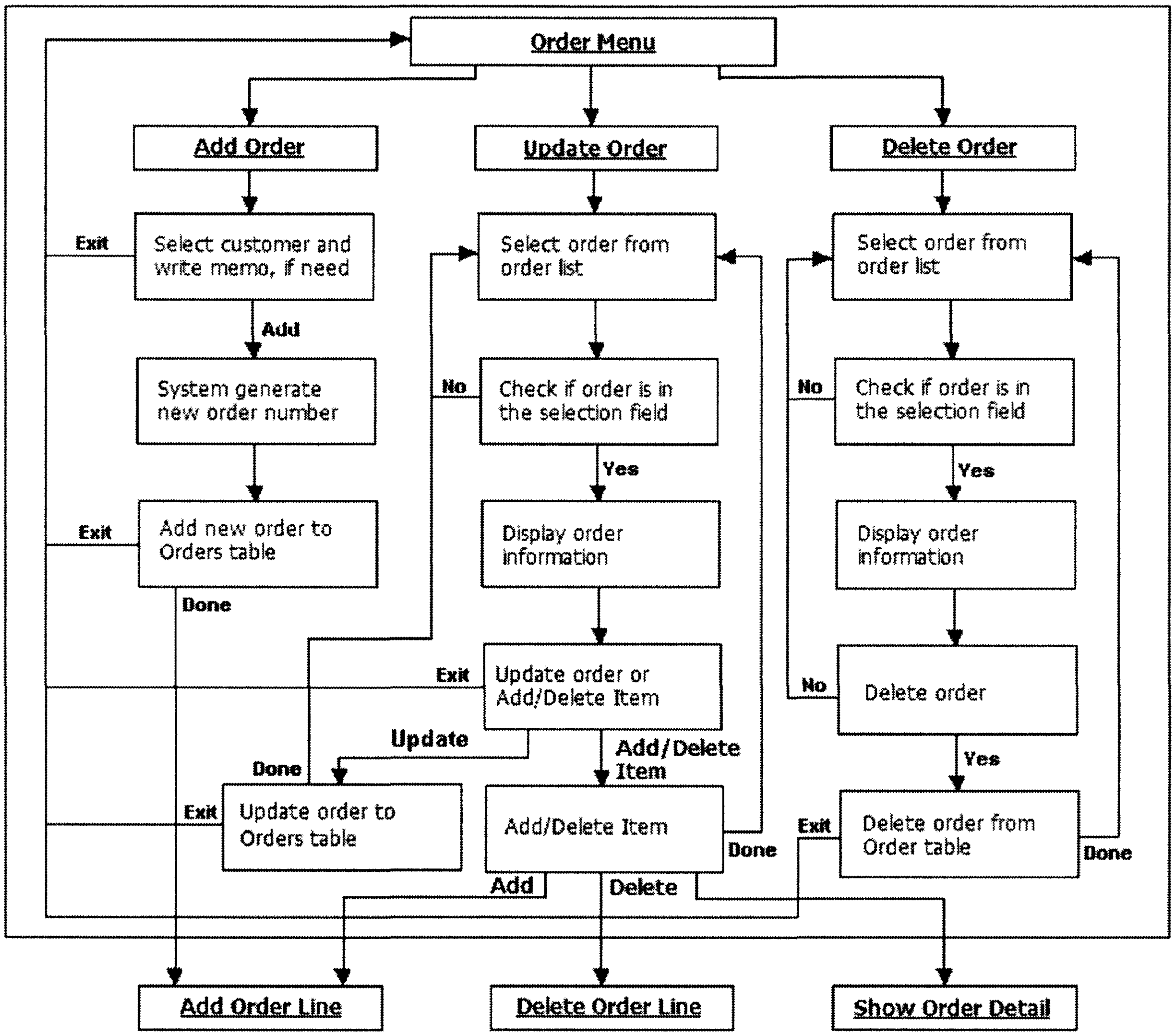

Order Menu
Add Order
Update Order
Delete Order
Exit
Select customer and write memo, if need
Add
System generate new order number
Exit
Add new order to Orders table
Done
Select order from order list
No
Check if order is in the selection field
Yes
Display order information
Exit
Update order or Add/Delete Item
Update
Add/Delete Item
Done
Exit
Update order to Orders table
Add/Delete Item
Done
Add
Delete
Select order from order list
No
Check if order is in the selection field
Yes
Display order information
No
Delete order
Yes
Exit
Delete order from Order table
Done
Add Order Line
Delete Order Line
Show Order Detail

Figure 4.11:Order Application Design

4.3.4 Add/Delete Item to Order Line

After the order is created, the user will have the ability to add and delete item to the order by adding or deleting order line.

Add Item to Order Line: The users are allowed to add item to the order after adding new order or updating order. They are allowed to select item form the item list and fill in the quantity of the item they want to order. The system will generate the order line number and add to the Orderline table. The user can add as many items as they want into one order.

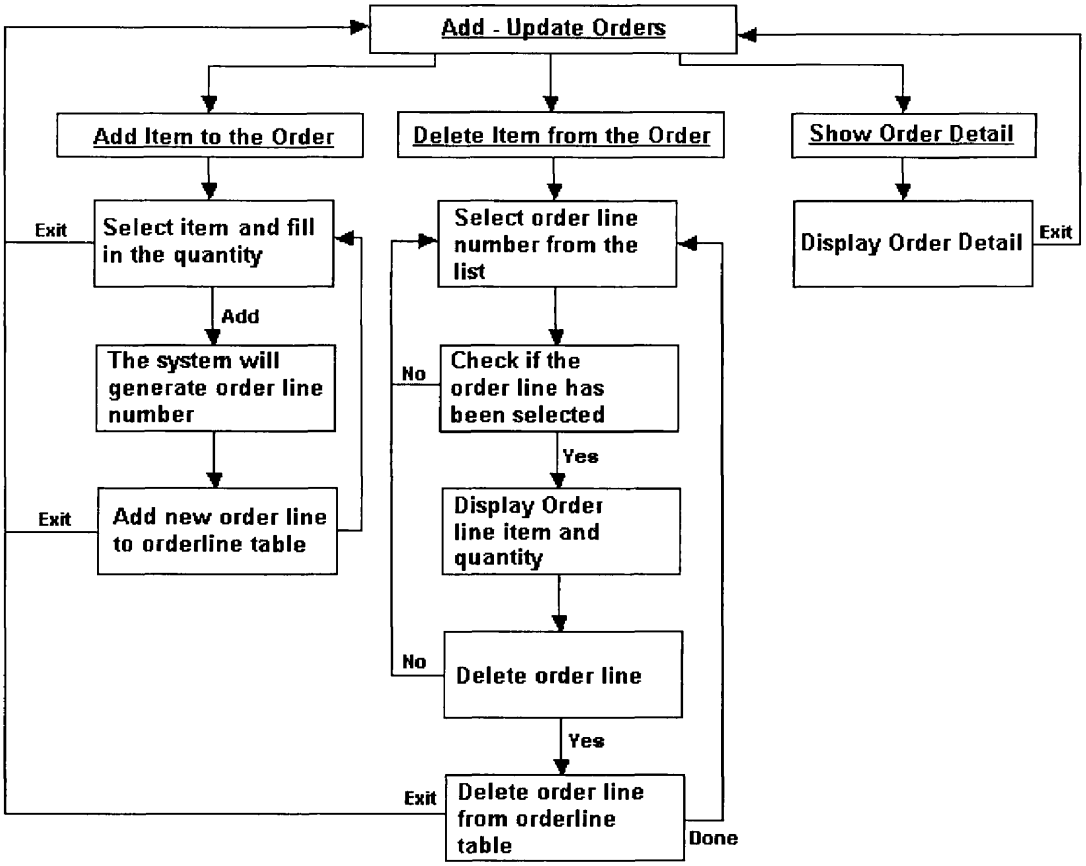

Add - Update Orders
Add Item to the Order
Delete Item from the Order
Show Order Detail
Select item and fill in the quantity
Add
The system will generate order line number
Add new order line to orderline table
Exit
Exit
Select order line number from the list
Check if the order line has been selected
No
Yes
Display Order line item and quantity
Delete order line
No
Yes
Delete order line from orderline table
Exit
Done
Display Order Detail
Exit

Figure 4.12: Order Line Application Design

Delete Item from Order Line: The users are allowed to delete item to the order after updating order. They are allowed to select order line number from the order line number list and then, delete that order line out of the database. The user will not be able to delete the order line of the order that had already been submitted to the main database server.

4.3.5 View Order Detail

The users are also allowed to view order detail after updating order. The order detail screen will show the order number, and every order line that belongs to that order. The order line information includes the item code, item description and quantity of that particular item.

4.3.6 Send Customer and Order Information to Main Database Server

This section will cover only the application design part. The connecting methodology and database synchronization will cover in the later section. In this project, the user can transfer data back to the main database server by one click of the button. The following is the application design steps behind the scene.

- Update order date field
- Send customer, orders, and orderline table
- Update order check field

Update Order Date Field: The system will update the order_date field before sending the orders table back to the main database server. To update this field, the current date will be use as the order date. This will update new order only. The order that had already been submitted before will not get updated.

Send Customer, Orders, and Orderline Table: After updating order_date field, the customer, orders, and orderline table are sent back to the main database server using Remote Data Access (RDA). This will be covered in the later section.

Update Order Check Field: The system will update thc order_check field after sending the orders table back to the main database server. This field is indicating that the particular orders had been sent to the main database server; therefore, the orders cannot be modified and deleted out of the system.

CHAPTER 5: APPLICATION IMPLEMENTATION

Before focusing on the mobile application implementation, I will discuss on how to setup Pocket PC wireless connection first.

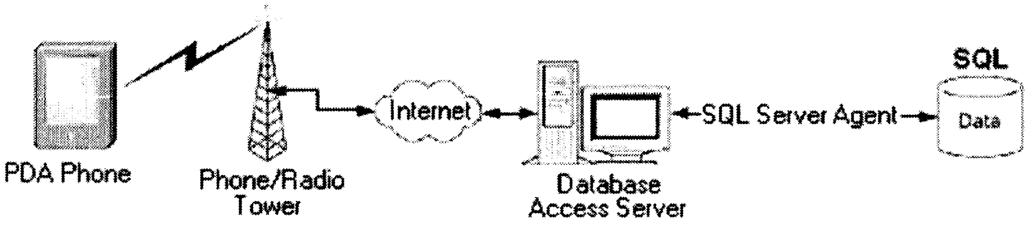


Figure 5.1: Project Overview

5.1 Setup Pocket PC for Wireless Connection

As you can see in the Figure 5.1, PDA or Pocket PC has to be connected to the Internet. There are several ways to complete this connection. In this project, I use mobile phone on the handheld device as a 'modem' to dial in to an Internet. This step requires the Internet Service Provider (ISP) so I have selected Verizon wireless connection as my ISP. The following are steps to setup mobile phone on Pocket PC as modem.

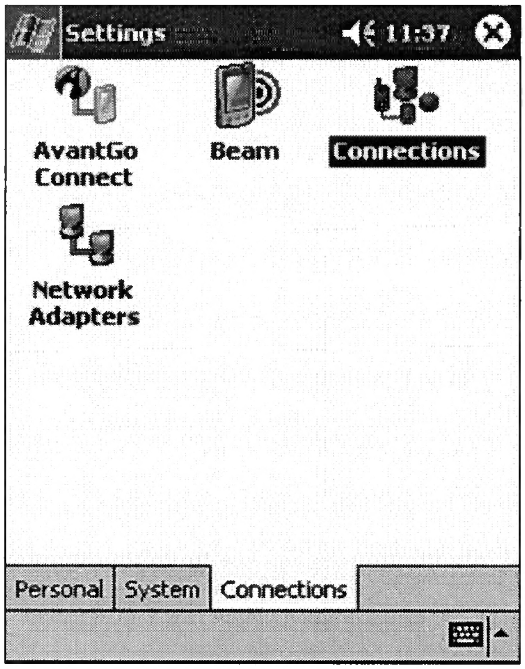
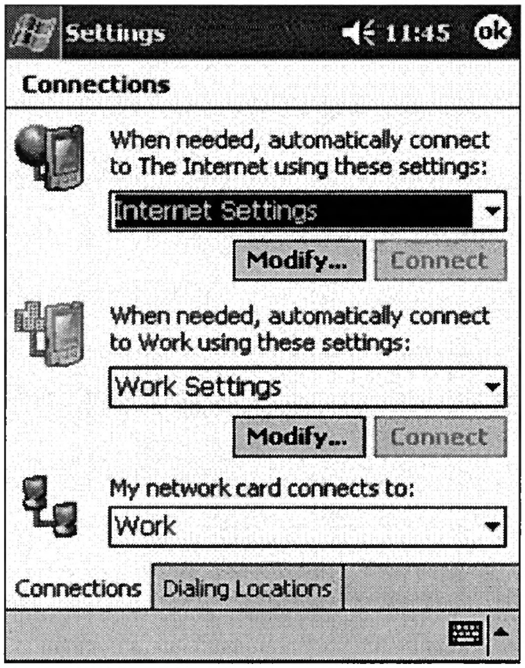


Figure 5.2: Setup Connection on Pocket PC

As see in figure 5.2, On Pocket PC

Tap **Start**, then **Settings**, then the **Connections** tab, then click **Connections** icon



Connections

When needed, automatically connect to The Internet using these settings:

Internet Settings

When needed, automatically connect to Work using these settings:

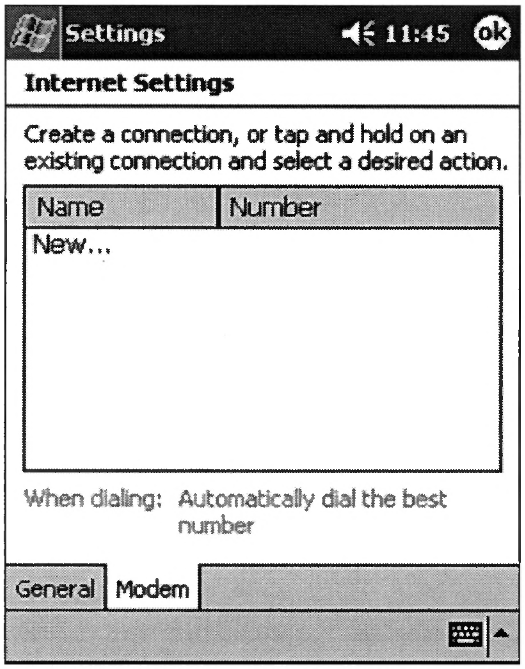
Work Settings

My network card connects to:

Work

Figure 5.3: Internet Setting on Pocket PC

As see in figure 5.3, press the **Modify** button underneath Internet Setting



Internet Settings

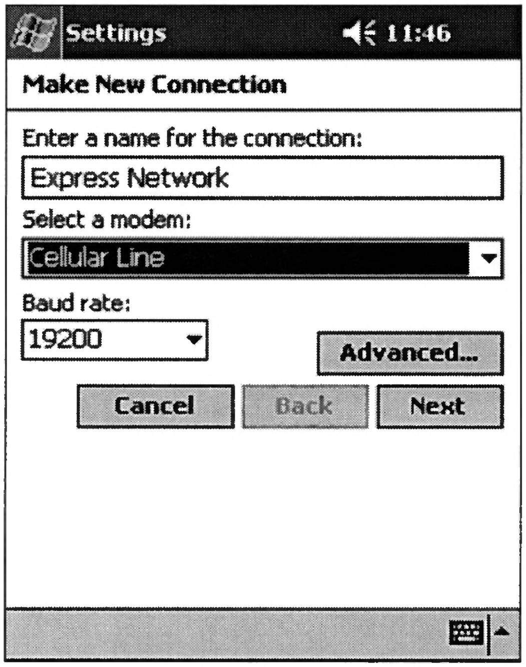
Create a connection, or tap and hold on an existing connection and select a desired action.

Name	Number
New...	

When dialing: Automatically dial the best number

Figure 5.4: Add New Internet Setting

As see in figure 5.4, with the modem tab selected, select **New…**



Make New Connection

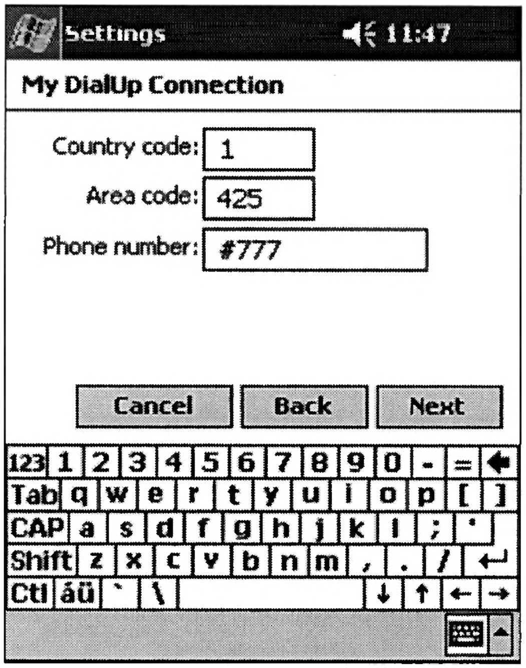
Enter a name for the connection: Express Network

Select a modem: Cellular Line

Baud rate: 19200

Figure 5.5: New Connection Setting

As see in figure 5.5, type a name for the connection. Set baud rate (Default 19200). Then press **Next** button.



Settings 11:47

My DialUp Connection

Country code: 1

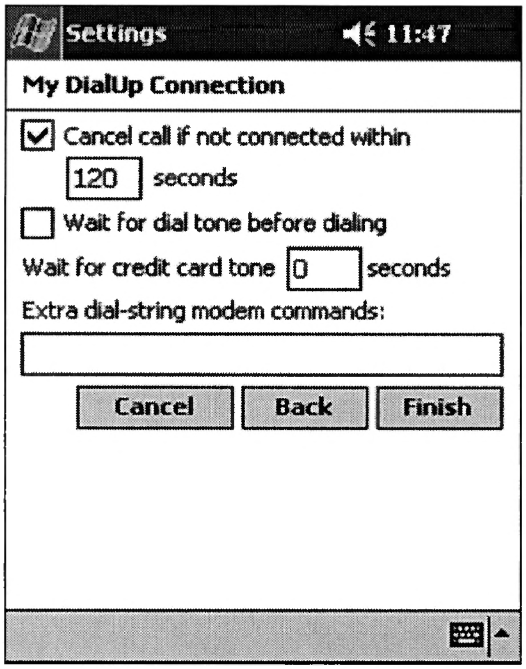
Area code: 425

Phone number: #777

Cancel
Back
Next
123 1 2 3 4 5 6 7 8 9 0 - =
Tab q w e r t y u i o p [ ]
CAP a s d f g h j k l ; '
Shift z x c v b n m , . /
Ctl áü ` \

Figure 5.6: Add Dialup Connection Number

As see in figure 5.6, Type the full number into the Phone number box. Then press **Next** button to add dialup connection number.



My DialUp Connection

☑ Cancel call if not connected within 120 seconds

□ Wait for dial tone before dialing

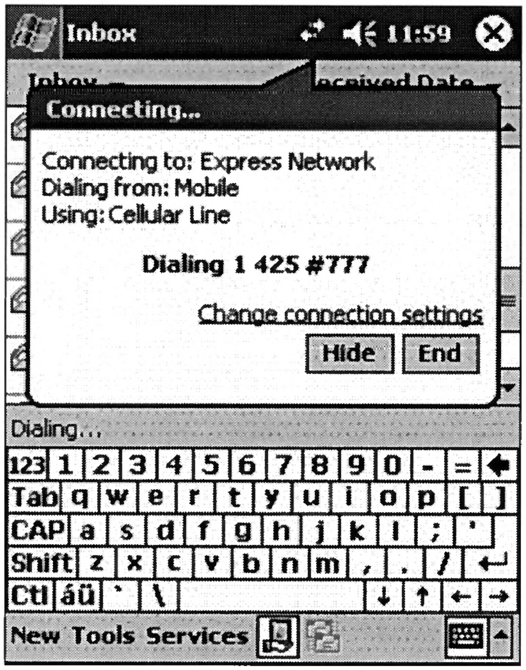
Wait for credit card tone 0 seconds

Extra dial-string modem commands:

Figure 5.7: Dialup Connection Configuration

As see in figure 5.7, remove the Wait for dial tone before dialing Check box and then press **Finish** button.

Figure 5.8 shows the screen when Pocket PC tries to dial up using modem.



Connecting...

Connecting to: Express Network
Dialing from: Mobile
Using: Cellular Line

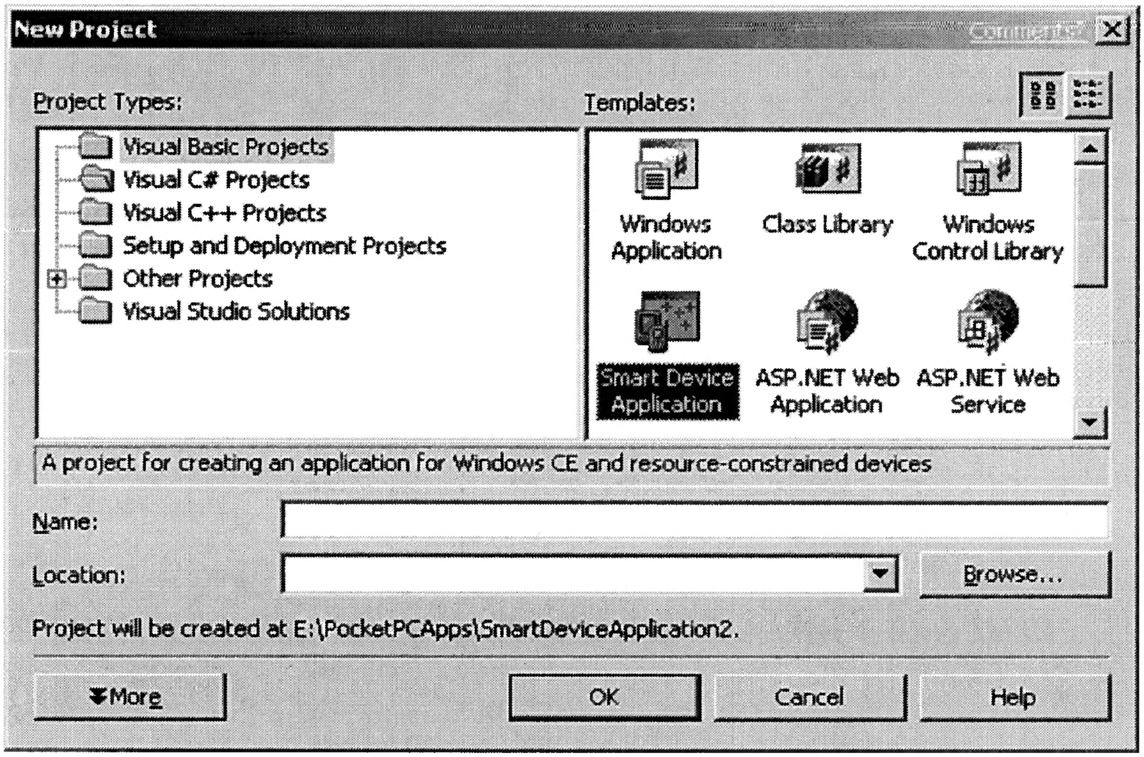
Dialing 1 425 #777

Change connection settings

Figure 5.8: Dial up from Pocket PC

5.2 The Mobile Application Implementation

The mobile application implementation in this project is developed based on .Net technology. Microsoft Visual Basic .NET 2003 comes with Smart Device Extension (SDE) integrated for NET Compact Framework mobile application. It also comes with the Smart Device Application Wizard, which allows developer to create application targeting on either Windows CE .NET devices or Pocket PC. In this project, I used Pocket PC to develop the application. The developers also have the option of creating a WinForms application, a Class Library (assembly), or a Control Library using the Smart Device Application Wizard.



Project Types: Visual Basic Projects

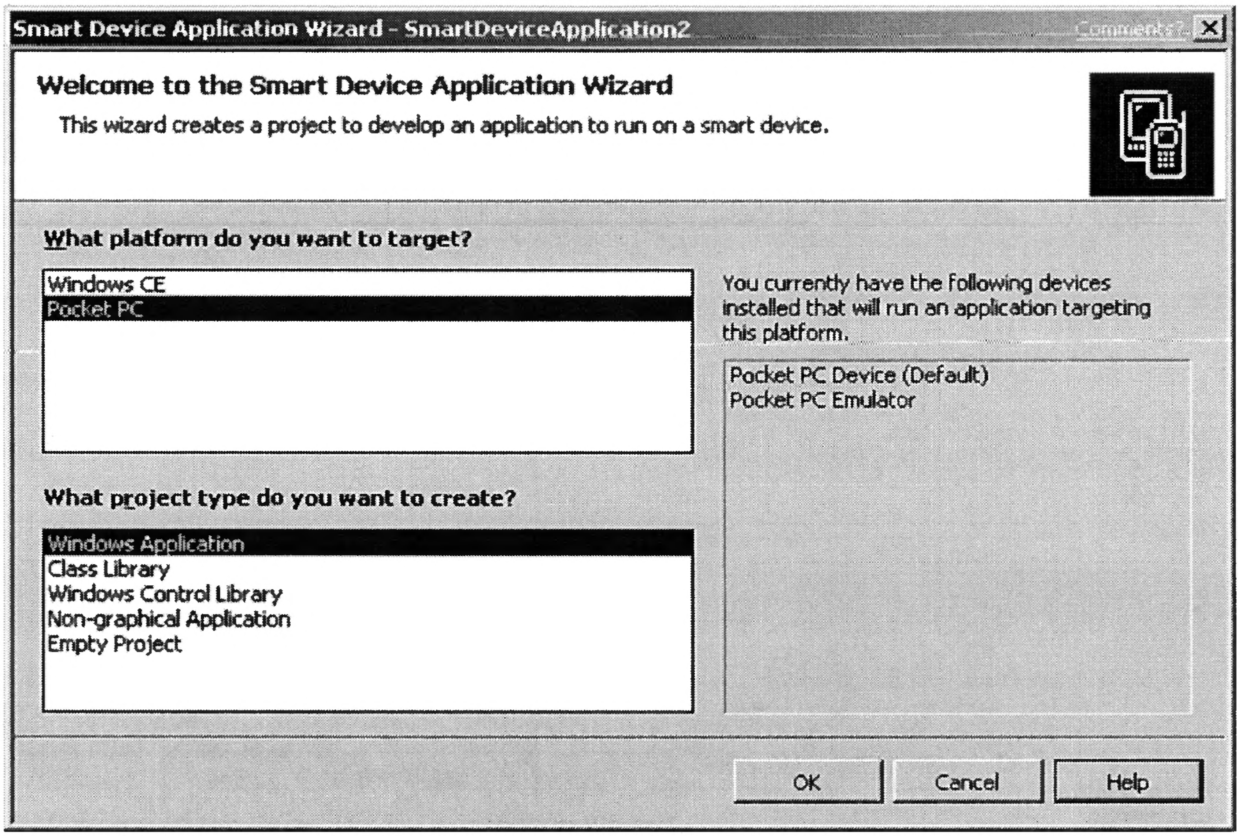
Templates: Smart Device Application

A project for creating an application for Windows CE and resource-constrained devices

Project will be created at E:\PocketPCApps\SmartDeviceApplication2.

Figure 5.9: Create Smart Device Application

Once Microsoft Visual Basic .NET 2003 is installed, I can create smart device application as shown in the following figure 5.9.



Smart Device Application Wizard - SmartDeviceApplication2

Welcome to the Smart Device Application Wizard

This wizard creates a project to develop an application to run on a smart device.

What platform do you want to target?

- Windows CE
- Pocket PC

You currently have the following devices installed that will run an application targeting this platform.

- Pocket PC Device (Default)
- Pocket PC Emulator

What project type do you want to create?

- Windows Application
- Class Library
- Windows Control Library
- Non-graphical Application
- Empty Project

Figure 5.10: Selecting Platform

I have chosen the Pocket PC platform as see in figure 5.10. Visual Basic will have a Windows Form ready to be used. This is very similar to a regular Windows Form, with the differences being a smaller Controls library to choose from, the "Device Controls", as well as the smaller form size. Pocket PCs typically use a 240x320 resolution. The blank smart device form is shown in figure 5.11.

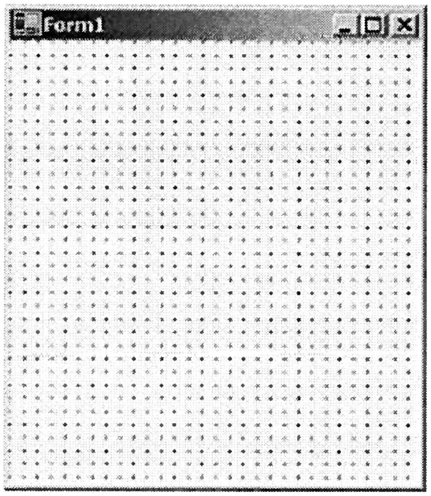


Figure 5.11: Smart Device Form

There are total of fifteen forms in the mobile application implementation. All of their detailed functionality is included in Appendix A (User's Manual). This next section will cover the special function of the smart device application, which I use in this project.

Drop-down Combo Box

I used a lot of drop-down combo box in this project. Drop-down combo boxes are one of the most useful controls on forms. I use them as unbound controls for locating records based on the selection in the drop down list. To limit the application user selection, I typically place these combo boxes in the header section of a form to narrow the list for records based on one or more criteria. However, as bound controls, combo box serves mainly as lookup lists. The text box portion of the control is inherently linked with the drop-down portion. The user makes a selection from the list and the text box immediately updates with that value. The following is the portion of the code to set value in a drop-down box.

```
Dim cmd As SqlCeCommand = cn.CreateCommand()
Dim reader As SqlCeDataReader
cmd.CommandText = "SELECT * FROM customer"
reader = cmd.ExecuteReader
While reader.Read
custid.Items.Add(reader.Item("cust_id"))
End While
```

This particular code reads a customer id from the Customer table and adds the result into custid combo box. The result is shown as in the Figure 5.12.

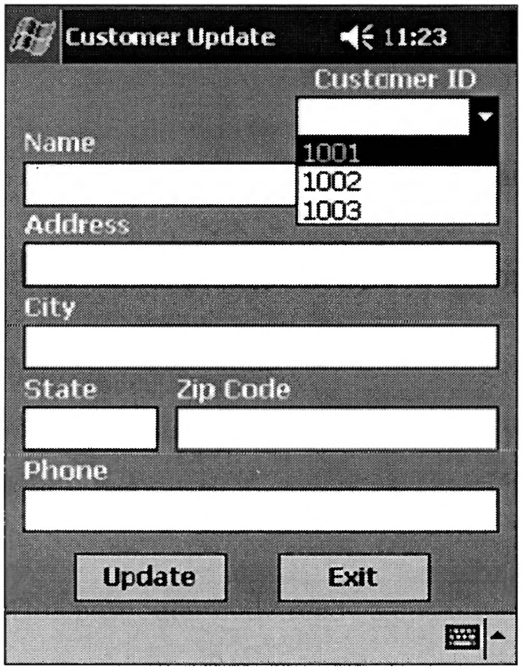


Figure 5.12: Customer Drop Down Combo Box

Soft Input Panel (SIP)

One of the major disadvantages on some Pocket PC is the lack of a physical keyboard. In this project, I use the function in .Net technology called the soft input panel (SIP). The soft input panel (SIP) component provide by .NET Compact Framework called

Microsoft.WindowsCE.Forms.InputPanel. Microsoft.WindowsCE.Forms.InputPanel give me the programmatic control of the SIP on Pocket PC. When I created a Windows-based application that targeted the Pocket PC platform, Visual Basic .NET automatically added a MainMenu component to the first form which then contained the lower menu bar with the SIP icon. No programming is required to use the SIP. However, a typical requirement for programming the SIP is to reposition or resize controls on the form when the SIP is enabled and disabled.

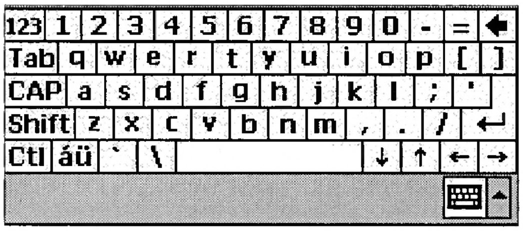

123 1 2 3 4 5 6 7 8 9 0 - = ←
Tab q w e r t y u i o p [ ]
CAP a s d f g h j k l ; '
Shift z x c v b n m , . / ↵
Ctl áü ` \ ↓ ↑ ← →

Figure 5.13: Keyboard for Mobile Application

Signature Capture

I have also use a signature capture box by Hood Canal Mobility. By adding HoodCanalSystems.SignatureCapture class into Visual Basic .Net, the mobile application allows user to save image to SQL Server CE database. This control provides a resizable capture, clear an image, save image as a 16-bit bitmap. The image is converted to a byte array for insertion into a SQL Server CE database as a long binary object and convert back to image for display.

5.3 Special Function and Coding for SQL CE

In this section, I will discuss about some special programming functions and some references in Visual Basic .Net and .Net Compact Framework, which I used in the application implementation.

The most important function I used in this application development is System.Data.SqlServerCE assembly reference. This assembly contains the classes that use to communicate to the SQL Server. In addition, it provides a programmatic access to Microsoft SQL Server 2000 Windows CE Edition (SQL Server CE) databases from a Microsoft Visual Basic .NET application running on a Windows CE .NET-based platform. System.Data.SqlServerCe provides a set of classes designed to expose the functionality of SQL Server CE.

These following are examples of the code under System.Data.SqlServerCE assembly class.

Open the Connection to SQL CE database

```
Dim cn As SqlCeConnection
cn = New SqlCeConnection("Data source=\program files\uno.sdf;Password=")
cn.Open()
```

Add Record

```
Dim cmd As SqlCeCommand = cn.CreateCommand()
cmd.CommandText = "INSERT INTO table_name (field_names)VALUES (‘value’)"
cmd.ExecuteNonQuery()
```

Update Record

```
Dim cmd As SqlCeCommand = cn.CreateCommand()
cmd.CommandText = "Update table_name SET field_name=‘value’”
cmd.ExecuteNonQuery()
```

Delete Record

```
Dim cmd As SqlCcCommand = cn.CreateCommand()
cmd.CommandText = "Delete FROM table_name  WHERE field_name=‘value’”
cmd.ExecuteNonQuery()
```

Drop Table

```
Dim cmd As SqlCeCommand = cn.CreateCommand()
cmd.CommandText = "Drop Table table_name”
cmd.ExecuteNonQuery()
```

Import Table

```
rda = New SqlCeRemoteDataAccess
Dim remoteConnectstring As String = "Provider=SQLOLEDB;Data Source=Server_Name;Initial
Catalog=Database_Name;User Id=sa;Password=sapassword"
rda.InternetLogin = ""
rda.InternetPassword = ""
rda.InternetUrl = "http://70.187.14.200:22/RDA/sscesa20.dll"
rda.LocalConnectionString = "Data Source=\Program Files\uno.sdf;SSCE:Database Password="
rda.Pull("table_name  ", "Select * from table_name  ", remoteConnectstring,
RdaTrackOption.TrackingOnWithIndexes, "customerErrorTable")
```

Export Table

```
Dim rda As SqlCeRemoteDataAccess
Dim URI As New System.Uri("http://70.187.14.200:22/RDA/sscesa20.dll")
Dim connectrequest As Net.WebRequest = Net.WebRequest.Create(URI)
connectrequest.Timeout = Threading.Timeout.Infinite
Dim connectresponse As Net.WebResponse = connectrequest.GetResponse
rda = New SqlCeRemoteDataAccess
```

```
Dim remoteConnectstring As String = "Provider=SQLOLEDB;Data Source=Server_Name;Initial
Catalog=Database_Name;User Id=sa;Password=sapassword"
rda.InternetLogin = ""
rda.InternetPassword = ""
rda.InternetUrl = "http://70.187.14.200:22/RDA/sscesa20.dll"
rda.LocalConnectionString = "Data Source=\Program Files\uno.sdf;SSCE:Database Password="
rda.Push("", remoteConnectstring, RdaBatchOption.BatchingOn)
```

There are lots more Visual Basic .Net code using in this project. Please go to Appendix B (Source Code) section for more detail.

CHAPTER 6: APPLICATION TESTING

The application testing is the next step after the application is completely implemented. As shown in Figure 6.1, once the data has been sent from Pocket PC to database server, the data will be stored in SQL database. To show that the data is transferred from Pocket PC to main database server correctly, I developed the ASP web page to retrieve data, which currently store on SQL data server.

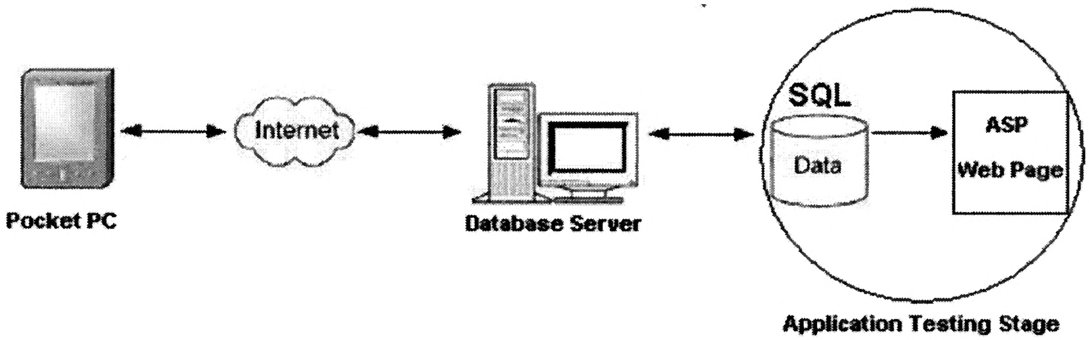


Figure 6.1: Application Testing Stage

In order to use ASP web page, the web server is required. I installed Internet Information Service (IIS) as the web server on the main database server. By using IIS, I can create web page that connect to SQL table and display the current data from database. To connect to web server,

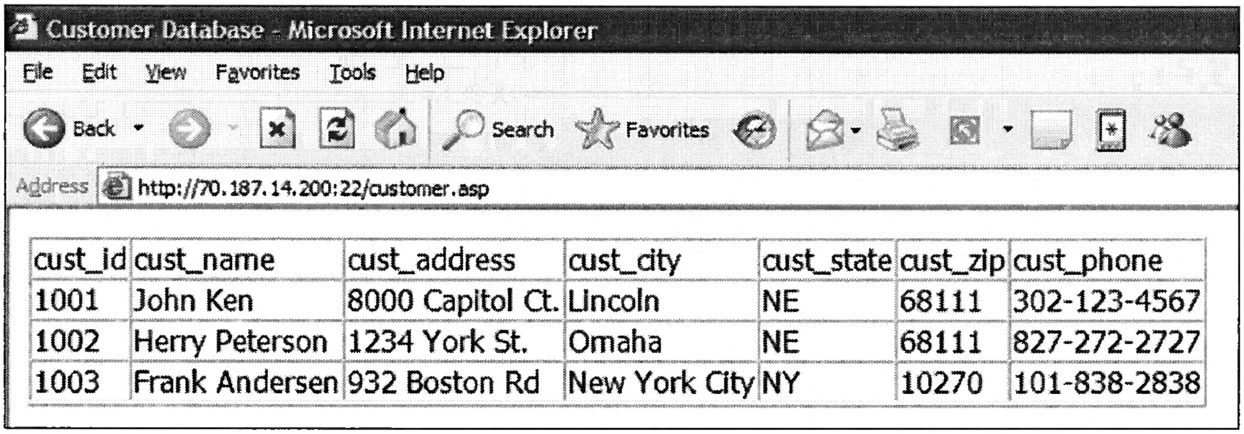
6.1 Customer ASP web page

To verify the customer information, I have developed the customer ASP web page, which retrieves data directly from the Customer table in SQL server. The fields that display on the web interface are

- Cust_id
- Cust_name

- Cust_address
- Cust_city
- Cust_state
- Cust_zip
- Cust_phone

Figure 6.2 shows the example of customer.asp.



Address http://70.187.14.200:22/customer.asp

cust_id	cust_name	cust_address	cust_city	cust_state	cust_zip	cust_phone
1001	John Ken	8000 Capitol Ct.	Lincoln	NE	68111	302-123-4567
1002	Herry Peterson	1234 York St.	Omaha	NE	68111	827-272-2727
1003	Frank Andersen	932 Boston Rd	New York City	NY	10270	101-838-2838

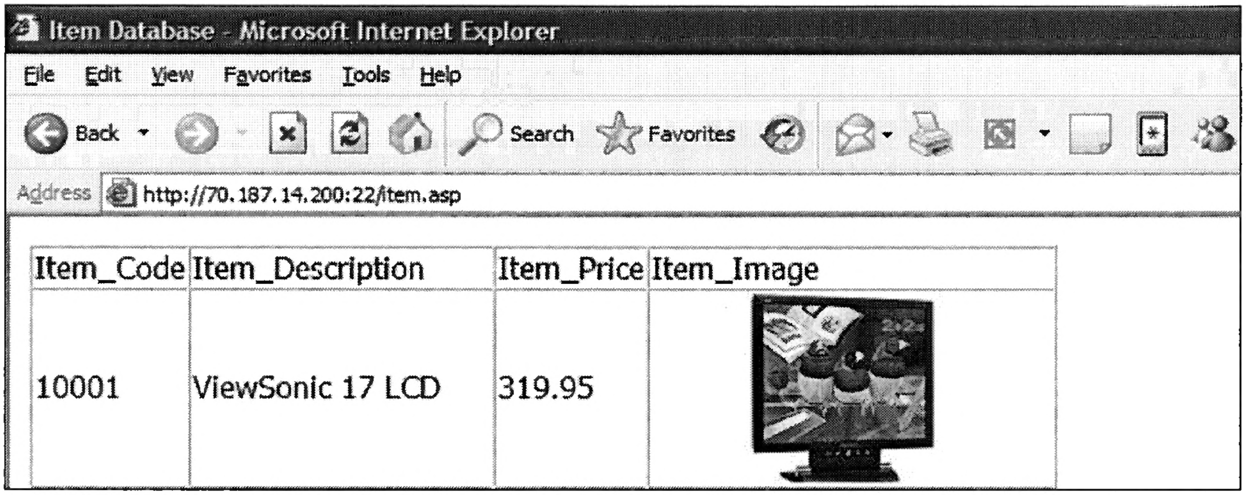
Figure 6.2: Customer ASP Web Page

6.2 Item ASP web page

To display the item information from data server, I have developed the item ASP web page, which retrieves data directly from item table in SQL server. The fields that display on the web interface are

- Item_Code
- Item_Description
- Item_Price
- Item_Image

Figure 6.3 shows the example of item.asp.


Item Database - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address

http://70.187.14.200:22/item.asp

Item_Code	Item_Description	Item_Price	Item_Image
10001	ViewSonic 17 LCD	319.95	

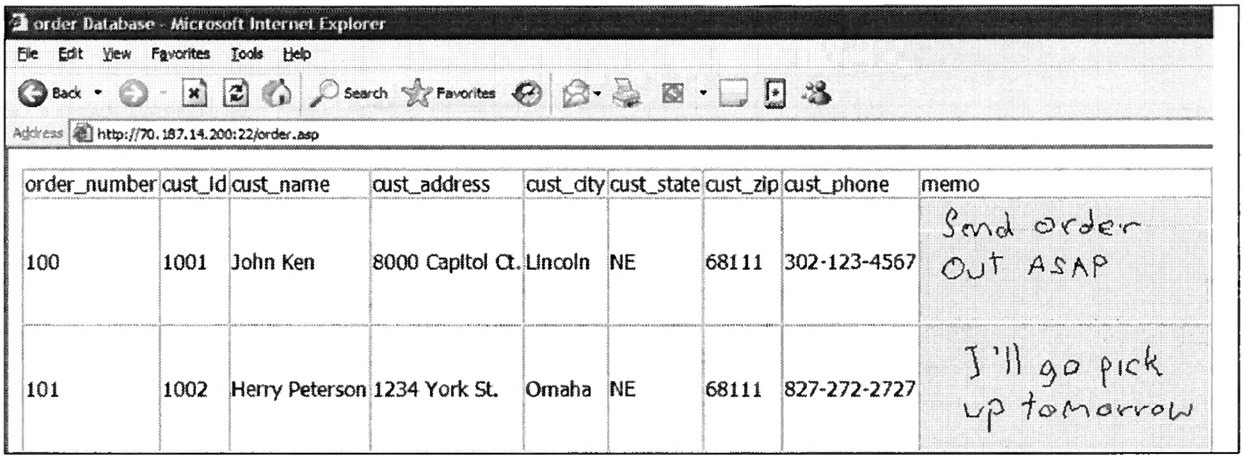
Figure 6.3: Item ASP Web Page

6.3 Orders ASP web page

To verify the order information, I have developed the order ASP web page, which retrieves data directly from order, orderline, and customer tables in SQL server. The fields that display on the web interface are

- Order_number
- Cust_id
- Cust_name
- Cust_address
- Cust_city
- Cust_state
- Cust_zip
- Cust_phone
- Memo

Figure 6.4 shows the example of order.asp.


order Database - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://70.187.14.200:22/order.asp

order_number	cust_id	cust_name	cust_address	cust_city	cust_state	cust_zip	cust_phone	memo
100	1001	John Ken	8000 Capitol Ct.	Lincoln	NE	68111	302-123-4567	Send order Out ASAP
101	1002	Herry Peterson	1234 York St.	Omaha	NE	68111	827-272-2727	I'll go pick up tomorrow

Figure 6.4: Orders ASP Web Page

In this project, users need to use the web server IP address to connect to port 22 of the web server. Therefore, users also need to specific the ASP web page they want to access. For examples, for customer information, users need to access to "http://70.187.14.200:22/customer.asp". The IP address in this project is not static so it can be changed upon Internet Service Provider (ISP).

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

As the numbers of usage handheld devices such as Pocket PC or Palm are increasing, the mobile applications run on those become significantly more important as well.

In this project, I have developed a mobile application using commercially available products to wirelessly connect a handheld device and SQL database server. Therefore, I have explored the methodology of database synchronization technology called Remote Data Access (RDA). RDA allows mobile application to use SQL statement and Internet-based connectivity for transferring information between two SQL servers on different physical locations. By using the combination of these commercial products such as Microsoft Visual Basic .Net, Microsoft SQL server, Verizon wireless network and RDA technology, I have successfully designed and implemented mobile application that allows users to wirelessly exchange and transfer data between Pocket PC to SQL database server.

The completion of this design and implementation in this project does not just proof that it is possible to wirelessly connect the handheld devices with the data server. It also contributes the idea of the possibility using this technology for future expansion work. For example, some businesses such as department stores or even grocery stores can use this technology to distribute their product catalogs to any Pocket PC users so they can order products directly from their PDA. Without the need of the actual computers, Pocket PC users can shop from anywhere they want.

In addition, this technology also can be used for increasing the security standpoint. For example, we can develop application for any credit payment with the signature verification capability directly from a Pocket PC. With the free-handwriting function, the application can now send the handwriting image to verify the signature of the credit card users.

As you can see, this technology can be used in various ways. This idea of data exchange is not just for the Pocket PC, in the future, you will see more mobile application usages on other wireless devices such as cellular phone or smart phone as well. I believe with this technology, wireless devices perspective will be changed. The idea of improving the design and implementation of mobile applications will become more important and hopefully we will have more mobile applications with efficiently and effectively data exchange in the future soon.

7.2 Future Work

The mobile application, which I have developed in this project, is designed for only a single-user environment. The design and implementation do not include any mechanisms that can handle issues in a multi-users environment. Some problems might occur when multiple users are accessed and used this mobile application at the same time. Assuming, the first client and the second client are reading the same customer record from the Customer table simultaneously, and they both decided to change the data, and tried to write their modified data back to the database server. The problem occurs when one of the data will be lost because the other overwrites it. This problem is known as a lost update problem.

Lost update problem occurs when two transactions are being updated approximately at the same time. For example, transactions T1 and T2 are trying to read and modify the value of X but each transaction is using a different method of a calculation to modify the value. This result in the transactions each containing a different value for X. T1 writes the value it holds for X to the database after it is read but before it is written by T2. T2 then writes the value it holds for X to the database, overwriting the value written by T1. The value written by T1 is lost.

The lost update is one of the problems, which motivates the need for concurrency protocols, which refers to the activity of coordinating concurrent accesses to a database in a multi-users database management. The main objective of concurrency control is to prevent database updates performed by one user from interfering with database retrievals and updates performed by another.

In the last few decades, the concurrency control issue have been studied and categorized into to groups depend on types of database management system, such as centralized database system, distributed database systems, or databases in mobile computing environment. The problem examined in this project shares similar considerations with concurrency control problem in distributed database management system. However, there are some characteristic features that make a wireless network with mobile clients distributed database differs from distributed database management system. These are:

- **Asymmetry in the communications**: The bandwidth in the downstream direction (servers-to-clients) is much greater than that in the upstream direction (clients-to-servers).
- **Frequent disconnections**: Mobile clients do not stay connected to the network continuously
- **Power limitations**: Some of the handheld devices are severely limited by the amount of energy they can use before the batteries have to be recharged.
- **Screen size**: The handheld devices, such as the Personal Digital Assistants or Pocket PC come with very small touch-screens.

The concurrency control problem is exacerbated in wireless network with mobile clients distributed database system because users may access data stored in many different devices, and a concurrency control mechanism cannot know about transactions that occur on every devices instantaneously.

One of the mechanisms, which introduces by Daniel Barbara [28] to solve the concurrency control problem in mobile-client of distributed database, is called "Guarantee Session". A session is defined as a sequence of read and writes operations performed during the execution of an application by a client. The guarantees provided by their method are:

- **Read Your Writes**: Any read operation in the session must reflect the values established by previous writes in that session.

- **Monotonic Reads**: Successive reads reflect a non-decreasing set of writes.
- **Writes Follow Reads**: Writes are propagated after the reads on which they depend.
- **Monotonic Writes**: Writes are propagated after writes that logically preceded them.

These guarantees are enforced by having a session manager that serializes read and write operations and by assigning unique identifiers to each write. A server must be willing to return information about the unique write identifier and the set of write identifiers for writes that are relevant to a given read. The session manager maintains a read-set of identifiers for the writes relevant to session reads and a write-set of identifiers for those writes performed in the session. For instance, the Read Your Writes guarantee is enforced by following two steps. Whenever a write is accepted by a server, its identifier is added to the session's write-set. Before a read is processed in a server *S* at a time *t*, the session manager must check that the write-set is a subset of the sequence of writes received by *S* before *t*.

As for the product I used, SQL CE uses optimistic concurrency control method in Remote Data Access (RDA) called tracked Pull and Push methods. Optimistic Concurrency Control, also known as validation or certification techniques, is the concurrency control that does check the information while a transaction is processing. The key is that a transaction makes no changes directly to the database. All changes are made on copies of the records seen only by that transaction. Before the changes are made to the database, the system checks to see if it can assure serialization. If it is not serialization, the transaction is aborted. The advantage of this method is that there is no overhead during the transaction. If there are not many other transactions, no extra work or check may need to be done.

In Remote Data Access without tracked Pull and Push methods, SQL Server does not keep pulled records locked. When the application calls Push, the changes made to the local SQL Server CE database are unconditionally applied to the SQL Server database. This might cause changes made by other users of the SQL Server database to be lost.

In Remote Data Access with tracked Pull and Push methods, a record is left unlocked until the time comes to update or delete it. At that point, the record is reread and checked to see if it has been changed since it was last read. If the record has changed, the update or delete fails and must be tried again. To determine whether a record has been changed, its new version is checked against a cached version of the record. This checking can be based on the record version timestamp column, which is a part of tracked Pull and Push methods configuration.

The tracked Pull and Push methods does not configure in this implementation. To configure this feather, redesign the databases to include a timestamp field is needed. The code also has to be modified to record timestamp for each pull and push transaction and use those timestamp as a concurrency control mechanism. If concurrency control mechanism is configured correctly in this project, I am sure that this mobile application will become much more efficiently and effectively with multi-users environment.

BIBLIOGRAPHY

[1] Laberge, Robert and Vujosevic, Srdjan Building PDA Databases for Wireless and Mobile Development, John Wiley & Sons, 2002

[2] Stallings, William Wireless Communications & Networks, Prentice Hall, 2001

[3] Tiffany, Rob The Definitive Guide to the .NET Compact Framework, Apress, 2003

[4] Yan, Jinwei Query optimizing in a mobile environment, Thesis (M.S.) University of Nebraska at Omaha, 2001

[5] Lin, Yi-Bing and Chlamtac, Imrich Wireless and mobile network architectures, Wiley computer publishing, 2001

[6] Dollard, Kathleen VB.NET Provides Power and Productivity, Visual Studio Magazine, 2003

[7] Ganguly, S. & Alonso, R. Query Optimization in Mobile Environment, Department of Computer Sciences, Rutgers University, New Brunswick, NJ

[8] Tiffany, Rob SQL Server CE Database Development with the .NET Compact Framework, Apress, 2003

[9] Chan, Wesley Project Voyager: Building an Internet Presence for People, Places, and Things, Massachusetts Institute of Technology, Massachusetts, May 2001

[10] http://msdn.microsoft.com

[11] Imielinski, T. and Badrinath, B.R. Data Management for Mobile Computing, Rutgers University, New Brunswick, NJ

[12] Duchamp, D. Issues in Wireless Mobile Computing, Columbia University, New York, NY

[13] Lee, Wei-Meng Developing Pocket PC Apps with SQL Server CE, http://www.ondotnet.com, 2002

[14] Acharya, A., Badrinath, B.R. and Imielinski, T. Checkpointing Distributed Applications on Mobile Computers, Department of Computer Science, Rutgers University, New Brunswick, NJ

[15] Imielinski, T. and Badrinath, B.R. Mobile Wireless Computing: Solutions and Challenges in Data Management, Department of Computer Science, Rutgers University, New Brunswick, NJ

[16] Lavin, P. Application Deployment and Integration Research Paper, Ruscombe Parke, Twyford, UK

[17] Cho, Jin-Hee Design, Implementation and Analysis of Wireless Ad Hoc Messenger, Department of Computer Science, Virginia Polytechnic Institute and University, Falls Church, VA

[18] Forsberg, Christian Pocket PC Development in the Enterprise-Mobile Solution with Visual Basic and .NET, Addison-Wesley, 2002

[19] Lee, Wei-Meng Using Remote Data Access with SQL Server CE 2.0, http://www.ondotnet.com, 2003

[20] Kunz, Thomas An Architecture for Adaptive Mobile Applications, Department of Systems and Computer Engineering, Carleton University

[21] Beaulieu, Mark Wireless Internet: Applications and Architecture, Addison-Wesley, 2002

[22] Tamleht, Carl Product Data Management Information in PDA Browsers, Thesis (M.S.) Department of Numerical Analysis and Computer Science, Kungl Tekniska Högskolan, Sweden, 2002

[23] Tai, Chia-Yu Temples of Taipei a Handheld Tour Guild for Pocket PC, Michigan State University, 2003

[24] Milroy, Steve Pocket PC and Windows Smartphone, Windows IT Pro Magazine, 2003

[25] Bernstein, Philip and Goodman, Nathan Concurrency Control in Distributed Database Systems, Cambridge, MA

[26] Silberschatz, Abraham and Korth, Henry Database System Concepts, WCB/McGraw-Hill, 1998

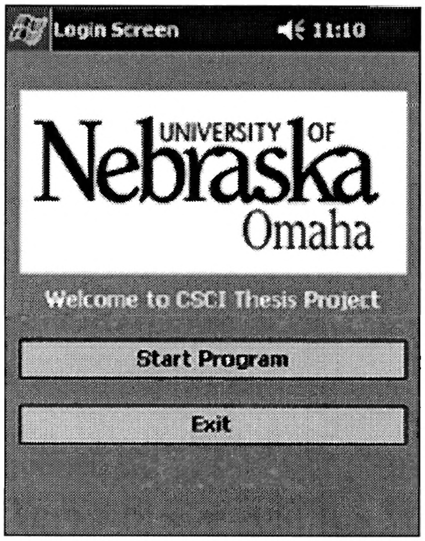
[27] Elmasri, Ramez and Navathe, Shamkant Fundamentals of Database Systems, Addison-Wesley, 1994

[28] Barbara, Daniel Mobile Computing and databases-A Survey, IEEE Transactions on Knowledge and Data Engineering, Vol11, No.1, 1999

APPENDIX A – USER'S MANUAL

Login Screen

The application starts with the Login screen. See figure A-1 for Login screen.



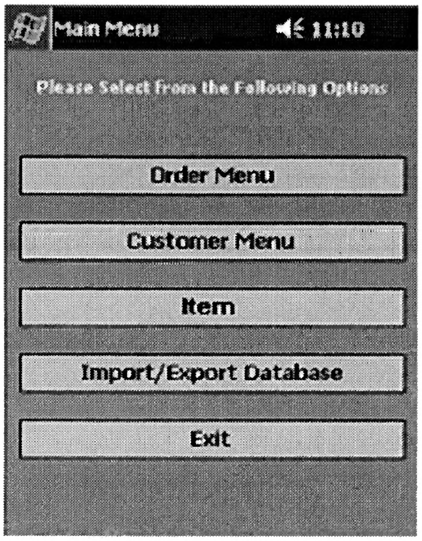
Welcome to CSCI Thesis Project

Figure A-1: Login Screen

To start the application, press "Start Program" button. This will take users to Main Menu screen. To quit the application, press "Exit" button.

Main Menu

The main menu is used for directing users to the submenu for specific tasks in this application. See figure A-2 for Main Menu screen.



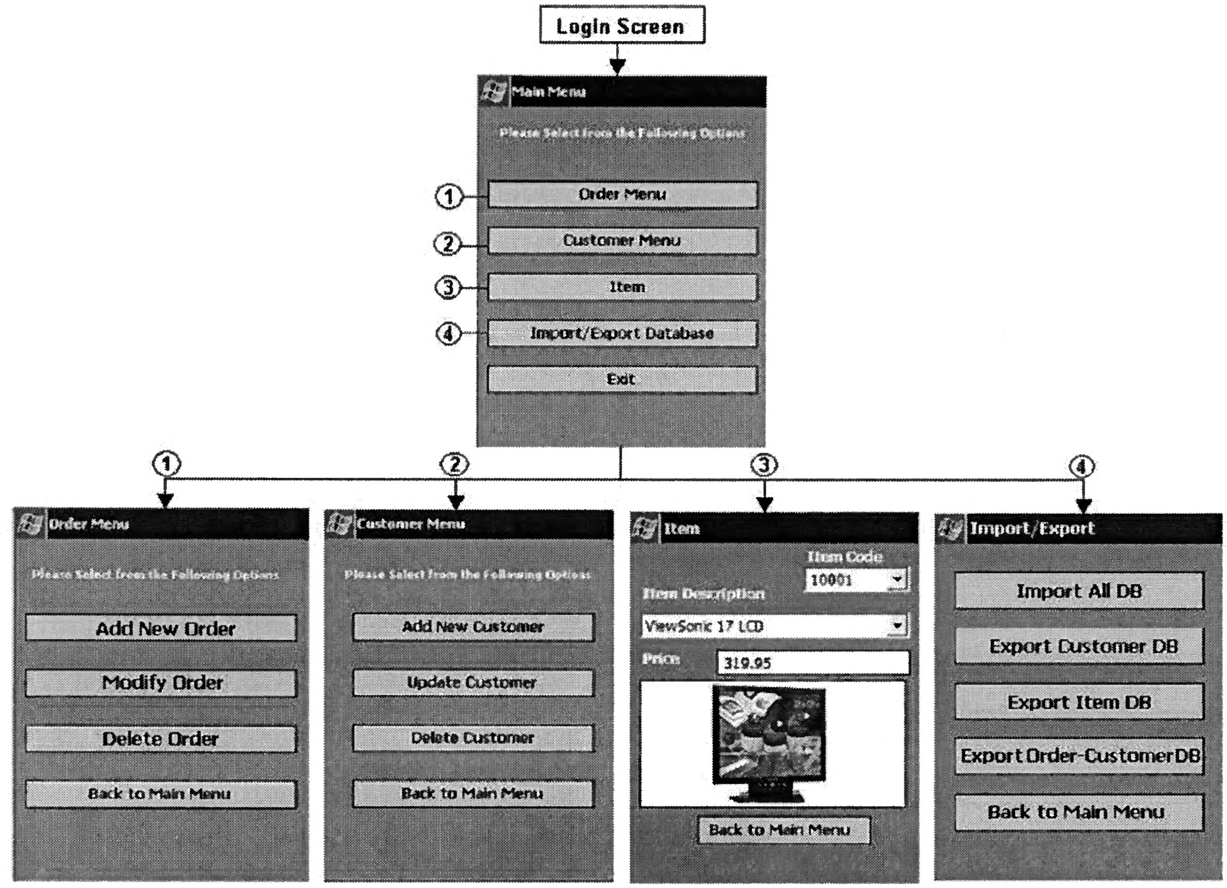
Please Select from the Following Options

Figure A-2: Main Menu

The main menu options are as following:

- Order Menu: to add, modify, delete order information.
- Customer Menu: to add, update, delete customer information.
- Item: to view item information.
- Import/Export Database: to import and export information to main database server.
- Exit: to exit the application

Figure A-3 shows the workflow diagram for this application start from Main Menu screen. This diagram contains the screen, which will display as a result of pressing the button on the main menu.



Item Code 10001

Item Description ViewSonic 17 LCD

Price 319.95

Figure A-3: Workflow from Main Menu

1. Order Menu

As shown in figure A-4, the Order Menu screen is used for directing users to add, modify, and delete order function in this application.

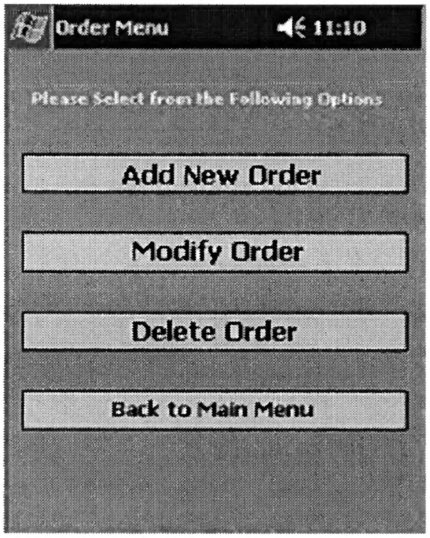
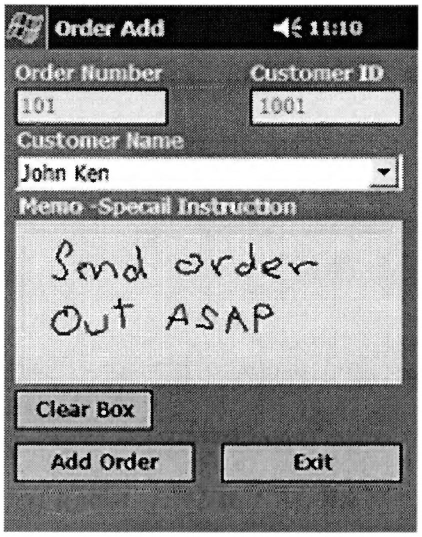


Figure A-4: Order Menu

1.1 Add New Order

The Order Add screen allows users to add new order to the application.



Order Number: 101

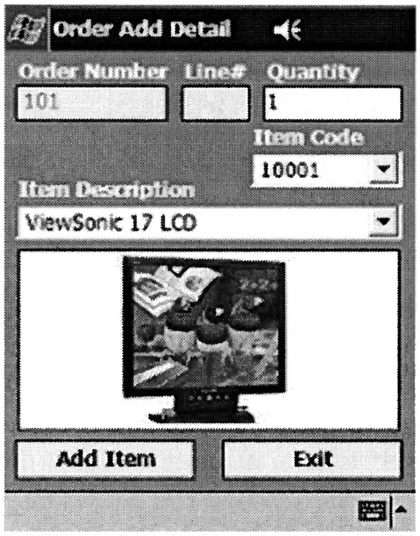
Customer ID: 1001

Customer Name: John Ken

Memo -Specail Instruction: Send order Out ASAP

Figure A-5: Order Add

As shown in figure A-5, to add new order, the users have to select customer from Customer Name drop-down box. Memo box is free-handwriting box allow users to right any special instruction for this specific order. Writing special instruction into memo box is optional. The users have to press “Add Order” button to add the order. The order number will be automatically generated by the application and the application will display Order Add Detail screen.



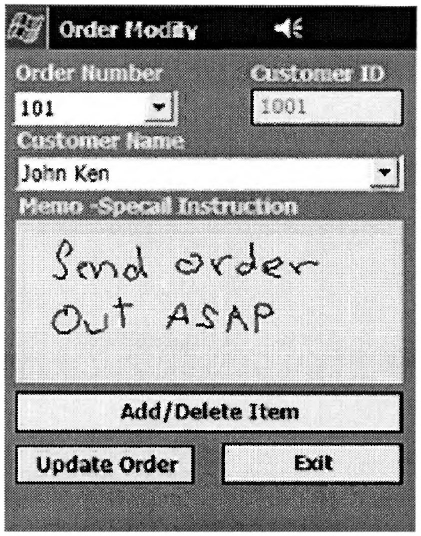
Order Number 101
Line#
Quantity 1
Item Code 10001
Item Description ViewSonic 17 LCD

Figure A-6: Order Add Detail

As shown in figure A-6: Order Add Detail screen, the users are allowed to add item(s) to the order by selecting Item Code or Item Description drop-down box and type in the quantity of the item. Then, press “Add Item” button to add the item to the order. To exit the Order Add Detail screen, press “Exit” button.

1.2 Modify Order

As shown in figure A-7, the Order Modify screen allows user to change customer on the specific order. First of all, the users have to select the order, which need to be modified from Order Number drop-down box. After the order has been selected, the application will display the current order information on the screen. The users will then be allowed to change customer by selecting the new customer name from the drop-down box.


Order Modify

Order Number: 101

Customer ID: 1001

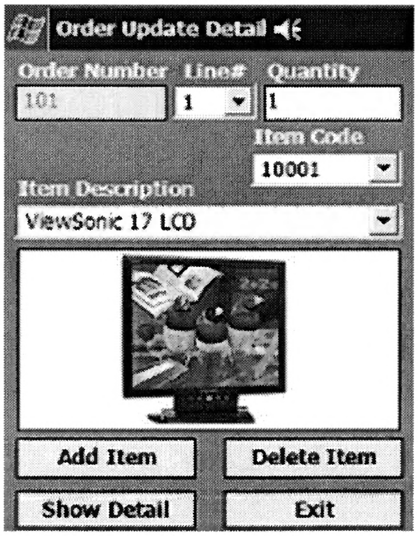
Customer Name: John Ken

Memo -Specail Instruction:

Send order
Out ASAP

Figure A-7: Order Modify

To modify the order information, press “Update Order” button. The users are also allowed to add item to or delete item from the order. By simply press “Add/Delete Item” Button, the application will display Order Update Detail screen.


Order Update Detail

Order Number: 101

Line#: 1

Quantity: 1

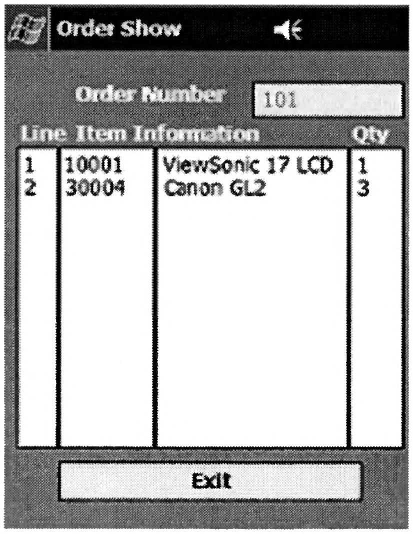
Item Code: 10001

Item Description: ViewSonic 17 LCD

Figure A-8: Order Update Detail

As shown in figure A-8, The Order Update Detail screen allow users to add item to the order by selecting Item Code or Item Description from drop-down box, typing in the quantity and then pressing "Add Item" button. The users also are allowed to delete item from the order by selecting the order line number from Line# drop-down box and simply pressing "Delete Item" button.

As shown in figure A-9, to show all item(s) in the order, the users can use "Show Detail" button.



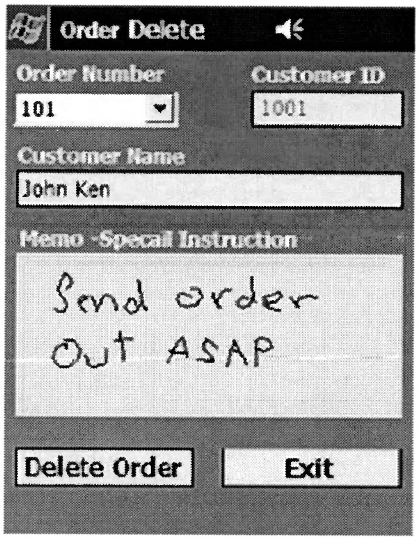
Order Number 101

Line	Item Information		Qty
1	10001	ViewSonic 17 LCD	1
2	30004	Canon GL2	3

Figure A-9: Order Show

1.3 Delete Order

As shown in figure A-10, the Delete Order screen allows the users to delete order by selecting order number from order number drop-down box. The application only allows deleting the order, which has not been sent to main database server.



Order Number: 101

Customer ID: 1001

Customer Name: John Ken

Memo -Specail Instruction: Send order Out ASAP

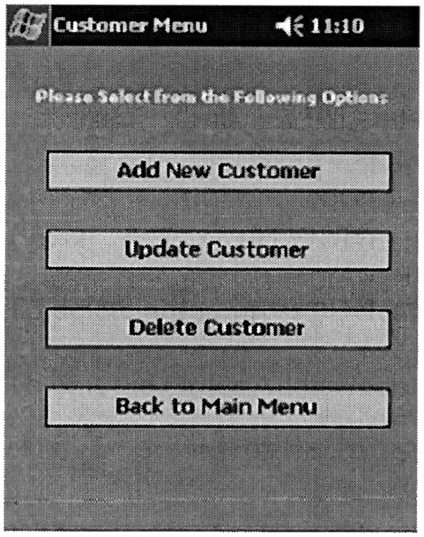
Figure A-10: Order Delete

To delete order, press “Delete Order” button.

To exit the Order Add screen, press “Exit” button.

2. Customer Menu

As shown in figure A-11, the Customer Menu screen is used for directing users to add, update, and delete customer function in this application.



Please Select from the Following Options:

Figure A-11: Customer Menu

The Customer Menu Options are as following:

- Add New Customer
- Update Customer
- Delete Customer
- Back to Main Menu: Exist Customer Menu screen

2.1 Add New Customer

As shown in figure A-12, the Add New Customer screen allows users to add new customer by fill in customer information in customer add screen.

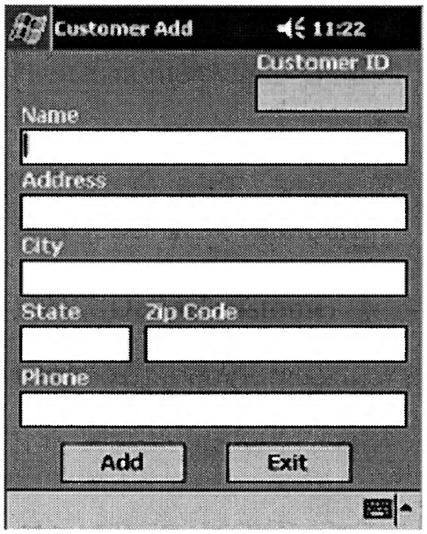


Figure A-12: Customer Add

To add customer, press “Add Customer” button. The customer number will be automatically generated by the application.

To exit the Customer Add screen, press “Exit” button.

2.2 Update Customer

As shown in figure A-13, the Update Customer allows user to change customer information. First of all, the users have to select the customer ID, which need to be updated from Customer ID drop-down box. After the customer has been selected, the application will display the current customer information on the screen. The users will then be allowed to change customer information by typing the new customer information.

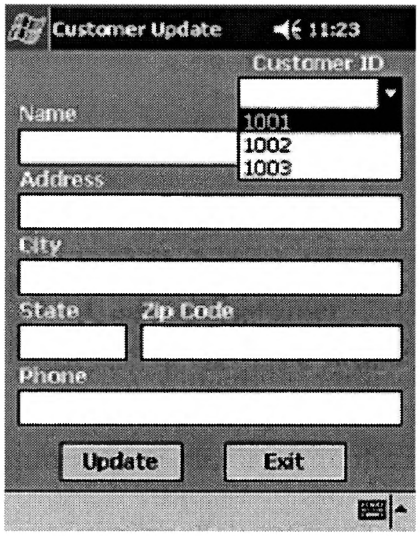


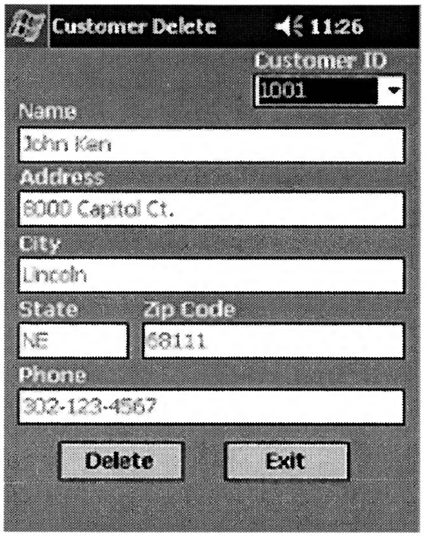
Figure A-13: Customer Update

To update the customer information, press "Update" button.

To exit the "Customer Update" screen, press "Exit" button.

2.3 Delete Order

As shown in figure A-14, the Delete Order screen allows users to delete customer by selecting the customer ID, which need to be deleted, from Customer ID drop-down box. The application only allows deleting customer whose name does not belong in any order.



Customer ID: 1001

Name: John Ken

Address: 8000 Capitol Ct.

City: Lincoln

State: NE

Zip Code: 68111

Phone: 302-123-4567

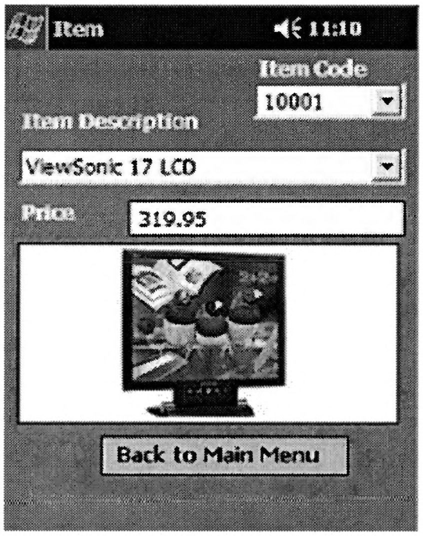
Figure A-14: Customer Delete

To delete order, press “Delete Order” button.

To exit the Order Add screen, press “Exit” button.

3. Item

As shown in figure A-15, the Item screen allows users to view the item image and item price by selecting item code or item description for drop-down box.



Item Code: 10001

Item Description: ViewSonic 17 LCD

Price: 319.95

Figure A-15: Item

To exit the "Item" screen, press "Back to Main Menu" button.

4. Import/Export Database

Import/Export Database button from the main menu will direct users to import export function, which allows them to transfer information between their Pocket PC and main database server.

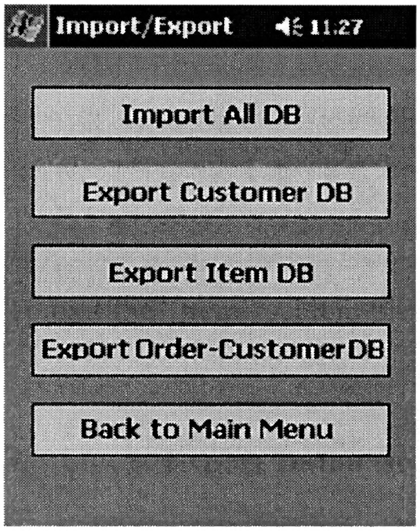


Figure A-16: Import/Export

As shown in figure A-16: Import/Export screen, the pocket PC will automatically dial up for the connection once the user press any import or export button.

The import/export options are as following:

- Import All DB: import customer, item, order, and orderline table from main database server.
- Export Customer DB: export customer database from pocket pc back to main database server.

- Export Item DB: export item database from pocket pc back to main database server
- Export Order-Customer DB: export order, orderline and customer database from pocket pc back to main database server
- Back to Main Menu: exit back to the Main Menu screen

APPENDIX B – SOURCE CODE

```
Imports System.Reflection
Imports System.Runtime.InteropServices
Imports System.Data.SqlServerCE
Imports System.IO
Imports System.Data
Imports System.Data.Common
Imports System.Data.SqlTypes
Imports System.Drawing.Imaging
Imports System.Drawing
Imports System.Windows.Forms

<Assembly: AssemblyTitle("")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>

<Assembly: AssemblyVersion("1.0.*")>
```

```
Public Class Login_Scr

Private Sub login_button_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles login_button.Click
        Dim nForm As Main_scr
        Dim oForm As Login_Scr
         oForm = New Login_Scr
            oForm.Close()
            oForm = Nothing
            nForm = New Main_scr
            nForm.ShowDialog()
            nForm = Nothing
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Application.Exit()
End Sub

End Class
```

```
Public Class Main_scr

Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
    Button4.Enabled = False
    Dim oForm As Main_scr
    Dim nForm As Cust_Menu_scr
    oForm = New Main_scr
    Button4.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New Cust_Menu_scr
    nForm.ShowDialog()
    nForm = Nothing
End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
    Button5.Enabled = False
    Dim oForm As Main_scr
    Dim nForm As item_scr
    oForm = New Main_scr
    Button5.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New item_scr
    nForm.ShowDialog()
    nForm = Nothing
End Sub
```

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
   Application.Exit()
End Sub


Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
       Button1.Enabled = False
       Dim oForm As Main_scr
       Dim nForm As Order_Menu_Scr
       oForm = New Main_scr
       Button1.Enabled = True
       oForm.Close()
       oForm = Nothing
       nForm = New Order_Menu_Scr
       nForm.ShowDialog()
       nForm = Nothing
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
       Button3.Enabled = False
       Dim oForm As Main_scr
       Dim nForm As Connect
       oForm = New Main_scr
       Button3.Enabled = True
       oForm.Close()
       oForm = Nothing
       nForm = New Connect
       nForm.ShowDialog()
```

```
        nForm = Nothing
    End Sub


    End Class
```

```
Public Class Cust_Menu_scr

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Button1.Enabled = False
    Dim nForm As Cust_Add
    Dim oForm As Cust_Menu_scr
    oForm = New Cust_Menu_scr
    Button1.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New Cust_Add
    nForm.ShowDialog()
    nForm = Nothing
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Button2.Enabled = False
    Dim oForm As Cust_Menu_scr
    Dim nForm As Cust_Update
    oForm = New Cust_Menu_scr
    Button2.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New Cust_Update
    nForm.ShowDialog()
    nForm = Nothing
End Sub
```

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
    Application.Exit()
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
    Button3.Enabled = False
    Dim oForm As Cust_Menu_scr
    Dim nForm As Cust_Delete
    oForm = New Cust_Menu_scr
    Button3.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New Cust_Delete
    nForm.ShowDialog()
    nForm = Nothing
End Sub

End Class
```

```
Public Class Cust_Add

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim cn As SqlCeConnection
    Try
      Button1.Enabled = False
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
        Dim cmd As SqlCeCommand = cn.CreateCommand()
      cmd.CommandText = "SELECT MAX(cust_id) as cust_id FROM customer"
      Dim reader As SqlCeDataReader
      reader = cmd.ExecuteReader
      While reader.Read()
        If IsDBNull(reader.Item("cust_id")) Then
          custid.Text = "1001"
        Else
          custid.Text = reader.Item("cust_id") + 1
        End If
      End While
      cmd.CommandText = "INSERT INTO customer (cust_id, cust_name,
cust_address, cust_city, cust_state, cust_zip, cust_phone)VALUES (N'" & custid.Text &
"',N'" & custname.Text & "',N'" & custaddress.Text & "',N'" & custcity.Text & "',N'" &
custstate.Text & "',N'" & custzip.Text & "',N'" & custphone.Text & "')"
      cmd.ExecuteNonQuery()
      If cn.State <> ConnectionState.Closed Then
        cn.Close()
      End If
      custid.Text = ""
      custname.Text = ""
```

```
            custaddress.Text = ""
            custcity.Text = ""
            custstate.Text = ""
            custzip.Text = ""
            custphone.Text = ""
        Catch sqlex As SqlCeException
            Dim sqlError As SqlCeError
            For Each sqlError In sqlex.Errors
                MessageBox.Show(sqlError.Message)
            Next
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        Finally
        End Try
        MessageBox.Show("The new customer has been added successfully.")
        Button1.Enabled = True
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button2.Click
        Application.Exit()
    End Sub

    End Class
```

```
Public Class Cust_Update

Private Sub Cust_Update_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Try
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        Dim reader As SqlCeDataReader
        cmd.CommandText = "SELECT * FROM customer"
        reader = cmd.ExecuteReader
        While reader.Read
            custid.Items.Add(reader.Item("cust_id"))
    End While
    Catch sqlex As SqlCeException
        Dim sqlError As SqlCeError
        For Each sqlError In sqlex.Errors
            MessageBox.Show(sqlError.Message)
        Next
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    Finally
    End Try
End Sub

Private Sub custid_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles custid.SelectedIndexChanged
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Try
```

```
            cmd.CommandText = "SELECT * FROM customer WHERE cust_id= '" &
custid.Items(custid.SelectedIndex) & "'"
            Dim reader As SqlCeDataReader
            reader = cmd.ExecuteReader
            While reader.Read()
                custname.Text = reader.Item("cust_name")
                custaddress.Text = reader.Item("cust_address")
                custcity.Text = reader.Item("cust_city")
                custstate.Text = reader.Item("cust_state")
                custzip.Text = reader.Item("cust_zip")
                custphone.Text = reader.Item("cust_phone")
        End While
        Catch sqlex As SqlCeException
            Dim sqlError As SqlCeError
            For Each sqlError In sqlex.Errors
                MessageBox.Show(sqlError.Message)
            Next
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        Finally
        End Try
End Sub


Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        If cn.State <> ConnectionState.Closed Then
            cn.Close()
        End If
        Application.Exit()
    End Sub
```

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Try
      cmd.CommandText = "Update customer SET cust_name=N'" & custname.Text &
"',cust_address=N'" & custaddress.Text & "',cust_city=N'" & custcity.Text &
"',cust_State=N'" & custstate.Text & "',cust_zip=N'" & custzip.Text & "',cust_phone=N'"
& custphone.Text & "' WHERE cust_id= N'" & custid.Items(custid.SelectedIndex) & "'"
     cmd.ExecuteNonQuery()
    Catch sqlex As SqlCeException
      Dim sqlError As SqlCeError
      For Each sqlError In sqlex.Errors
        MessageBox.Show(sqlError.Message)
      Next
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    Finally
    End Try
    MessageBox.Show("Update Finished Successfully")
End Sub

End Class
```

```
Public Class Cust_Delete

Private Sub Cust_Delete_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT cust_id FROM customer"
    reader = cmd.ExecuteReader
    While reader.Read
        custid.Items.Add(reader.Item("cust_id"))
    End While
End Sub

Private Sub custid_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles custid.SelectedIndexChanged
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    cmd.CommandText = "SELECT * FROM customer WHERE cust_id= '" &
custid.Items(custid.SelectedIndex) & "'"
    Dim reader As SqlCeDataReader
    reader = cmd.ExecuteReader
    While reader.Read()
        custname.Text = reader.Item("cust_name")
        custaddress.Text = reader.Item("cust_address")
        custcity.Text = reader.Item("cust_city")
        custstate.Text = reader.Item("cust_state")
        custzip.Text = reader.Item("cust_zip")
        custphone.Text = reader.Item("cust_phone")
    End While
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    If cn.State <> ConnectionState.Closed Then
      cn.Close()
    End If
    Application.Exit()
  End Sub

Private Sub DeleteButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DeleteButton.Click
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
      cmd.CommandText = "SELECT COUNT(cust_id) AS cust_id FROM orders
WHERE cust_id = N'" & custid.Items(custid.SelectedIndex) & "'"
      reader = cmd.ExecuteReader
      While reader.Read()
          counttemp.Text = reader.Item("cust_id")
      End While
    If counttemp.Text = 0 Then
      cmd.CommandText = "Delete FROM customer  WHERE cust_id= N'" &
custid.Items(custid.SelectedIndex) & "'"
      cmd.ExecuteNonQuery()
      custid.Items.Clear()
      cmd.CommandText = "SELECT cust_id FROM customer"
      reader = cmd.ExecuteReader
      While reader.Read
        custid.Items.Add(reader.Item("cust_id"))
      End While
      custname.Text = ""
```

```
            custaddress.Text = ""
            custcity.Text = ""
            custstate.Text = ""
            custzip.Text = ""
            custphone.Text = ""
            MessageBox.Show("The customer has been deleted successfully.")
        Else
            MessageBox.Show("The customer cannot be deleted.")
        End If
    End Sub

    End Class
```

```
Public Class Item_scr

Private Sub Item_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT * FROM item"
    reader = cmd.ExecuteReader
    While reader.Read
      itemcode.Items.Add(reader.Item("Item_code"))
      itemdesc.Items.Add(reader.Item("Item_Description"))
    End While
End Sub

Private Sub itemdesc_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles itemdesc.SelectedIndexChanged
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    cmd.CommandText = "SELECT * FROM item WHERE Item_Description= '" &
itemdesc.Items(itemdesc.SelectedIndex) & "'"
    Dim reader As SqlCeDataReader
    reader = cmd.ExecuteReader
    While reader.Read()
      itemprice.Text = reader.Item("Item_Price")
      itemcode.Text = reader.Item("Item_code")
    End While
End Sub
```

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    If cn.State <> ConnectionState.Closed Then
        cn.Close()
    End If
    Application.Exit()
End Sub

Private Sub itemid_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles itemcode.SelectedIndexChanged
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    cmd.CommandText = "SELECT * FROM item WHERE Item_code= '" &
itemcode.Items(itemcode.SelectedIndex) & "'"
    Dim reader As SqlCeDataReader
    reader = cmd.ExecuteReader
    While reader.Read()
        itemprice.Text = reader.Item("Item_Price")
        itemdesc.Text = reader.Item("Item_description")
    End While
    cmd.CommandText = "SELECT Item_Image FROM item WHERE Item_code= '"
& itemcode.Items(itemcode.SelectedIndex) & "'"
    Dim value As Byte()
    Try
        reader = cmd.ExecuteReader()
    Catch ex As SqlCeException
        MessageBox.Show(ex.ToString(), "DB operation failed")
        Exit Sub
    End Try
    If Not reader.Read() Then
        MessageBox.Show("No records in the database")
        Exit Sub
    End If
    value = CType(reader("Item_Image"), Byte())
    Dim ms As MemoryStream = New MemoryStream(value)
    PictureBox1.Image = New Bitmap(ms)
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Dim fs As FileStream
    Try
      fs = File.Open("\program files\uno_project\30005.jpg", FileMode.Open)
    Catch ex As Exception
      MessageBox.Show(ex.ToString(), "Failed to open the image")
      Exit Sub
    End Try
    Dim imageData(fs.Length) As Byte
    fs.Read(imageData, 0, imageData.Length)
    fs.Close()
    Try
      Dim cmd As SqlCeCommand = cn.CreateCommand()
      cmd.CommandText = "insert into item (Item_Code,Item_Description,Item_Price,
item_Image) values (?,?,?,?)"
      cmd.Parameters.Add("Item_Code", "30005")
      cmd.Parameters.Add("Item_Description", "Sony DCR-TRV260")
      cmd.Parameters.Add("Item_Price", "349.95")
      Dim param As SqlCeParameter = cmd.Parameters.Add("Item_Image",
SqlDbType.Image)
      param.Value = imageData
      cmd.ExecuteNonQuery()
    Catch sqex As SqlCeException
      MessageBox.Show(sqex.ToString(), "DB operation failed")
```

```
        End Try
        MessageBox.Show("done")
    End Sub

End Class
```

```
Public Class Order_Menu_Scr

Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
    Application.Exit()
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Button1.Enabled = False
    Dim oForm As Order_Menu_Scr
    Dim nForm As Order_Add
    oForm = New Order_Menu_Scr
    Button1.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New Order_Add
    nForm.ShowDialog()
    nForm = Nothing
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
    Button3.Enabled = False
    Dim oForm As Order_Menu_Scr
    Dim nForm As Order_Delete
    oForm = New Order_Menu_Scr
    Button3.Enabled = True
    oForm.Close()
    oForm = Nothing
    nForm = New Order_Delete
```

```
        nForm.ShowDialog()
        nForm = Nothing
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button2.Click
        Button2.Enabled = False
        Dim oForm As Order_Menu_Scr
        Dim nForm As Order_Update
        oForm = New Order_Menu_Scr
        Button2.Enabled = True
        oForm.Close()
        oForm = Nothing
        nForm = New Order_Update
        nForm.ShowDialog()
        nForm = Nothing
    End Sub

    End Class
```

```
Public Class Order_Add

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
    If cn.State <> ConnectionState.Closed Then
        cn.Close()
    End If
    Application.Exit()
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
    sc.Clear()
End Sub

Private Sub Order_Add_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT cust_id, cust_name FROM customer"
    reader = cmd.ExecuteReader
    While reader.Read
        custname.Items.Add(reader.Item("cust_name"))
    End While
End Sub

Private Sub custname2_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles custname.SelectedIndexChanged
```

```
      If cn.State <> ConnectionState.Open Then
         cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
         cn.Open()
      End If
      Dim cmd As SqlCeCommand = cn.CreateCommand()
      Try
         cmd.CommandText = "SELECT * FROM customer WHERE cust_name= '" &
custname.Items(custname.SelectedIndex) & "'"
         Dim reader As SqlCeDataReader
         reader = cmd.ExecuteReader
         While reader.Read()
            custid.Text = reader.Item("cust_id")
         End While
      Catch sqlex As SqlCeException
         Dim sqlError As SqlCeError
         For Each sqlError In sqlex.Errors
            MessageBox.Show(sqlError.Message)
         Next
      Catch ex As Exception
         MessageBox.Show(ex.Message)
      Finally
      End Try
End Sub

Private Sub Add_Button_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Add_Button.Click
      If custid.Text <> "" Then
         If cn.State <> ConnectionState.Open Then
            cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
```

```
       cn.Open()
    End If
    Dim BitmapBytes As Byte()
    Try
       Dim cmd As SqlCeCommand = cn.CreateCommand()
       cmd.CommandText = "SELECT MAX(order_number) as ordernumber FROM
orders"
       Dim reader As SqlCeDataReader
       reader = cmd.ExecuteReader
       While reader.Read()
          If IsDBNull(reader.Item("ordernumber")) Then
             ordernum.Text = "100"
          Else
             ordernum.Text = reader.Item("ordernumber") + 1
          End If
       End While
       Add_Button.Enabled = False
       BitmapBytes = sc.SaveToByteArray()
       cmd.CommandText = "insert into orders (order_number,cust_id,memo) values
(?,?,?)"
       cmd.Parameters.Add("order_number", ordernum.Text)
       cmd.Parameters.Add("cust_id", custid.Text)
       cmd.Parameters.Add("memo", BitmapBytes)
       cmd.ExecuteNonQuery()
    Catch ex As SqlCeException
       MessageBox.Show(ex.ToString(), "DB operation failed")
       Exit Sub
    End Try
    If cn.State <> ConnectionState.Closed Then
       cn.Close()
    End If
```

```
      Dim oForm As Order_Add
      Dim nForm As Order_Add_Detail
      oForm = New Order_Add
      oForm.Close()
      oForm = Nothing
      nForm = New Order_Add_Detail
      nForm.ShowDialog()
      nForm = Nothing
      Add_Button.Visible = False
      Up_Button.Visible = True
    Else
      MessageBox.Show("Please select customer before add order")
    End If
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Up_Button.Click
    If custid.Text <> "" Then
      If cn.State <> ConnectionState.Open Then
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
      End If
      Dim BitmapBytes As Byte()
      Try
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        Up_Button.Enabled = False
        cmd.CommandText = "Delete FROM orders WHERE order_number= '" &
ordernum.Text & "'"
        cmd.ExecuteNonQuery()
        BitmapBytes = sc.SaveToByteArray()
```

```
                cmd.CommandText = "insert into orders (order_number,cust_id,memo) values
(?,?,?)"
                cmd.Parameters.Add("order_number", ordernum.Text)
                cmd.Parameters.Add("cust_id", custid.Text)
                cmd.Parameters.Add("memo", BitmapBytes)
                cmd.ExecuteNonQuery()
                cn.Close()
            Catch ex As SqlCeException
                MessageBox.Show(ex.ToString(), "DB operation failed")
                Exit Sub
            End Try
            Up_Button.Enabled = True
            MessageBox.Show("Update order finished sucessfully")
        Else
            MessageBox.Show("Please select customer before add order")
        End If
End Sub

End Class
```

```
Public Class Order_Delete

Private Sub Order_Delete_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT order_number FROM orders WHERE order_check
IS NULL"
    reader = cmd.ExecuteReader
    While reader.Read
      ordernum.Items.Add(reader.Item("order_number"))
    End While
End Sub

Private Sub ordernum_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ordernum.SelectedIndexChanged
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    cmd.CommandText = "SELECT customer.cust_name AS cust_name,
orders.order_number AS order_numbe, orders.cust_id AS cust_id FROM  orders INNER
JOIN customer ON orders.cust_id = customer.cust_id WHERE orders.order_number = '"
& ordernum.Items(ordernum.SelectedIndex) & "'"
    Dim reader As SqlCeDataReader
    reader = cmd.ExecuteReader
    While reader.Read()
      custid.Text = reader.Item("cust_id")
      custname.Text = reader.Item("cust_name")
    End While
    cmd.CommandText = "SELECT memo FROM orders WHERE
orders.order_number = '" & ordernum.Items(ordernum.SelectedIndex) & "'"
    Dim BitmapBytes As Byte()
    Try
      reader = cmd.ExecuteReader()
    Catch ex As SqlCeException
      MessageBox.Show(ex.ToString(), "DB operation failed")
      Exit Sub
    End Try
    If Not reader.Read() Then
      MessageBox.Show("No records in the database")
      Exit Sub
    End If
    BitmapBytes = CType(reader("memo"), Byte())
    Dim ms As MemoryStream = New MemoryStream(BitmapBytes)
    sc.Image = New Bitmap(ms)
End Sub

Private Sub button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
    If cn.State <> ConnectionState.Closed Then
      cn.Close()
    End If
    Application.Exit()
End Sub

Private Sub Delete_Button_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Delete_Button.Click
    If ordernum.Text <> "" Then
      Dim cmd As SqlCeCommand = cn.CreateCommand()
      Try
```

```
        cmd.CommandText = "Delete FROM orders  WHERE order_number= N'" &
ordernum.Items(ordernum.SelectedIndex) & "'"
        cmd.ExecuteNonQuery()
        cmd.CommandText = "Delete FROM orderline  WHERE order_number= N'"
& ordernum.Items(ordernum.SelectedIndex) & "'"
        cmd.ExecuteNonQuery()
      Catch sqlex As SqlCeException
        Dim sqlError As SqlCeError
        For Each sqlError In sqlex.Errors
          MessageBox.Show(sqlError.Message)
        Next
      Catch ex As Exception
        MessageBox.Show(ex.Message)
      Finally
      End Try
      ordernum.Items.Clear()
      Dim reader As SqlCeDataReader
      cmd.CommandText = "SELECT order_number FROM orders"

      reader = cmd.ExecuteReader
      While reader.Read
        ordernum.Items.Add(reader.Item("order_number"))
      End While
      custname.Text = ""
      custid.Text = ""
      sc.Image = Nothing
      MessageBox.Show("Delete order finished successfully")
    Else
      MessageBox.Show("Please select order before delete order")
    End If
End Sub
End Class
```

```
Public Class Order_Update

Private Sub custname_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles custname.SelectedIndexChanged
    If cn.State <> ConnectionState.Open Then
      cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
      cn.Open()
    End If
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Try
      cmd.CommandText = "SELECT * FROM customer WHERE cust_name= '" &
custname.Items(custname.SelectedIndex) & "'"
      Dim reader As SqlCeDataReader
      reader = cmd.ExecuteReader
      While reader.Read()
        custid.Text = reader.Item("cust_id")
      End While
    Catch sqlex As SqlCeException
      Dim sqlError As SqlCeError
      For Each sqlError In sqlex.Errors
        MessageBox.Show(sqlError.Message)
      Next
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    Finally
    End Try
End Sub

Private Sub ordernum_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ordernum.SelectedIndexChanged
```

```
    If cn.State <> ConnectionState.Open Then
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
    End If
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    cmd.CommandText = "SELECT customer.cust_name AS cust_name,
orders.order_number AS order_number, orders.cust_id AS cust_id FROM  orders
INNER JOIN customer ON orders.cust_id = customer.cust_id WHERE
orders.order_number = '" & ordernum.Items(ordernum.SelectedIndex) & "'"
    Dim reader As SqlCeDataReader
    reader = cmd.ExecuteReader
    While reader.Read()
        custid.Text = reader.Item("cust_id")
        custname.Text = reader.Item("cust_name")
    End While
    cmd.CommandText = "SELECT memo FROM orders WHERE order_number = '"
& ordernum.Items(ordernum.SelectedIndex) & "'"
    Dim BitmapBytes As Byte()
    Try
        reader = cmd.ExecuteReader()
    Catch ex As SqlCeException
        MessageBox.Show(ex.ToString(), "DB operation failed")
        Exit Sub
    End Try
    If Not reader.Read() Then
        MessageBox.Show("No records in the database")
        Exit Sub
    End If
    BitmapBytes = CType(reader("memo"), Byte())
    Dim ms As MemoryStream = New MemoryStream(BitmapBytes)
```

```
        sc.Image = New Bitmap(ms)
    End Sub

    Private Sub Up_Button_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Up_Button.Click
        If custid.Text <> "" And ordernum.Text <> "" Then
            If cn.State <> ConnectionState.Open Then
                cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
                cn.Open()
            End If
            Try
                Dim cmd As SqlCeCommand = cn.CreateCommand()
                Up_Button.Enabled = False
                cmd.CommandText = "Update orders SET cust_id='" & custid.Text & "'
WHERE order_number= N'" & ordernum.Items(ordernum.SelectedIndex) & "'"
                cmd.ExecuteNonQuery()
            Catch ex As SqlCeException
                MessageBox.Show(ex.ToString(), "DB operation failed")
                Exit Sub
            End Try
            Up_Button.Enabled = True
            MessageBox.Show("Update order finished successfully")
        Else
            MessageBox.Show("Please select order and customer before update order")
        End If
    End Sub

    Private Sub button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
        If cn.State <> ConnectionState.Closed Then
        cn.Close()
    End If
    Application.Exit()
End Sub

Private Sub Order_Update_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT order_number FROM orders WHERE order_check
IS NULL"
    reader = cmd.ExecuteReader
    While reader.Read
        ordernum.Items.Add(reader.Item("order_number"))
    End While
    cmd.CommandText = "SELECT cust_name FROM customer"
    reader = cmd.ExecuteReader
    While reader.Read
        custname.Items.Add(reader.Item("cust_name"))
    End While
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    If ordernum.Text <> "" Then
        Button2.Enabled = False
        If cn.State <> ConnectionState.Closed Then
            cn.Close()
            End If
            Dim oForm As Order_Update
            Dim nForm As Order_Update_Detail
            oForm = New Order_Update
            Button2.Enabled = True
            oForm.Close()
            oForm = Nothing
            nForm = New Order_Update_Detail
            nForm.ordernum.Text = ordernum.Items(ordernum.SelectedIndex)
            nForm.ShowDialog()
            nForm = Nothing
        Else
            MessageBox.Show("Please select order before add/delete item")
        End If
    End Sub

End Class
```

```
Public Class Order_Show

Private Sub Order_Show_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT orderline.order_line, orderline.item_code AS
item_code, orderline.qty AS qty, item.item_description AS item_description FROM
orderline INNER JOIN item ON orderline.item_code = item.item_code WHERE
orderline.order_number = "'" & ordernum.Text & "'"
    reader = cmd.ExecuteReader
    listbox.Items.Clear()
    While reader.Read
      Listbox.Items.Add(reader.Item("order_line"))
      ListBox2.Items.Add(reader.Item("item_code"))
      ListBox3.Items.Add(reader.Item("item_description"))
      ListBox4.Items.Add(reader.Item("qty"))
    End While
End Sub

Private Sub button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
    If cn.State <> ConnectionState.Closed Then
      cn.Close()
    End If
    Application.Exit()
End Sub
End Class
```

```
Public Class Connect

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim cn As SqlCeConnection
    Dim rda As SqlCeRemoteDataAccess
    Dim sqlengine As SqlCeEngine
    Try
      Button1.Enabled = False
      If (Not File.Exists("\program files\uno_project\uno.sdf")) Then
        sqlengine = New SqlCeEngine
        sqlengine.LocalConnectionString = "Data source=\program
files\uno_project\uno.sdf;Password="
        sqlengine.CreateDatabase()
        sqlengine.Dispose()
      Else
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        cmd.CommandText = "DROP TABLE Customer"
        cmd.ExecuteNonQuery()
        If cn.State <> ConnectionState.Closed Then
          cn.Close()
        End If
      End If
      rda = New SqlCeRemoteDataAccess
      Dim remoteConnectstring As String = "Provider=SQLOLEDB;Data
Source=PUI;Initial Catalog=UNO;User Id=sa;Password=sapassword"
      rda.InternetLogin = ""
      rda.InternetPassword = ""
      rda.InternetUrl = "http://70.187.14.200:22/RDA/sscesa20.dll"
      rda.LocalConnectionString = "Data Source=\Program
Files\UNO_Project\uno.sdf;SSCE:Database Password="
      rda.Pull("customer", "Select * from customer", remoteConnectstring,
RdaTrackOption.TrackingOnWithIndexes, "customerErrorTable")
      rda.Pull("item", "Select * from item", remoteConnectstring,
RdaTrackOption.TrackingOnWithIndexes, "itemErrorTable")
      rda.Pull("orders", "Select * from orders", remoteConnectstring,
RdaTrackOption.TrackingOnWithIndexes, "ordersErrorTable")
      rda.Pull("orderline", "Select * from orderline", remoteConnectstring,
RdaTrackOption.TrackingOnWithIndexes, "orderlineErrorTable")
    Catch sqlex As SqlCeException
      Dim sqlError As SqlCeError
      For Each sqlError In sqlex.Errors
        MessageBox.Show(sqlError.Message)
      Next
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    Finally
      rda.Dispose()
      Button1.Enabled = True
    End Try
    MessageBox.Show("Finished Import Database")
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Dim rda As SqlCeRemoteDataAccess
    Try
      Button2.Enabled = False
      Dim URI As New System.Uri("http://70.187.14.200:22/RDA/sscesa20.dll")
      Dim connectrequest As Net.WebRequest = Net.WebRequest.Create(URI)
      connectrequest.Timeout = Threading.Timeout.Infinite
      Dim connectresponse As Net.WebResponse = connectrequest.GetResponse
      rda = New SqlCeRemoteDataAccess
      Dim remoteConnectstring As String = "Provider=SQLOLEDB;Data
Source=PUI;Initial Catalog=UNO;User Id=sa;Password=sapassword"
      rda.InternetLogin = ""
      rda.InternetPassword = ""
      rda.InternetUrl = "http://70.187.14.200:22/RDA/sscesa20.dll"
      rda.LocalConnectionString = "Data Source=\Program
Files\UNO_Project\uno.sdf;SSCE:Database Password="
      rda.Push("customer", remoteConnectstring, RdaBatchOption.BatchingOn)
    Catch sqlex As SqlCeException
      Dim sqlError As SqlCeError
      For Each sqlError In sqlex.Errors
        MessageBox.Show(sqlError.Message)
      Next
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    Finally
      rda.Dispose()
      Button2.Enabled = True
    End Try
    MessageBox.Show("Finished Export Database")
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
    Application.Exit()
End Sub
```

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
    Dim rda As SqlCeRemoteDataAccess
    Try
      Button5.Enabled = False
      Dim URI As New System.Uri("http://70.187.14.200:22/RDA/sscesa20.dll")
      Dim connectrequest As Net.WebRequest = Net.WebRequest.Create(URI)
      connectrequest.Timeout = Threading.Timeout.Infinite
      Dim connectresponse As Net.WebResponse = connectrequest.GetResponse
      rda = New SqlCeRemoteDataAccess
      Dim remoteConnectstring As String = "Provider=SQLOLEDB;Data
Source=PUI;Initial Catalog=UNO;User Id=sa;Password=sapassword"
      rda.InternetLogin = ""
      rda.InternetPassword = ""
      rda.InternetUrl = "http://70.187.14.200:22/RDA/sscesa20.dll"
      rda.LocalConnectionString = "Data Source=\Program
Files\UNO_Project\uno.sdf;SSCE:Database Password="
      rda.Push("item", remoteConnectstring, RdaBatchOption.BatchingOn)
    Catch sqlex As SqlCeException
      Dim sqlError As SqlCeError
      For Each sqlError In sqlex.Errors
        MessageBox.Show(sqlError.Message)
      Next
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    Finally
      rda.Dispose()
      Button5.Enabled = True
    End Try
    MessageBox.Show("Finished Export Database")
End Sub
```

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
    Dim cn As SqlCeConnection
    Dim cn2 As SqlCeConnection
    Dim rda As SqlCeRemoteDataAccess
    Button4.Enabled = False
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    cmd.CommandText = "Update orders SET order_date='" & Today & "' WHERE
order_check IS NULL"
    cmd.ExecuteNonQuery()
    If cn.State <> ConnectionState.Closed Then
      cn.Close()
    End If
    Try
      Dim URI As New System.Uri("http://70.187.14.200:22/RDA/sscesa20.dll")
      Dim connectrequest As Net.WebRequest = Net.WebRequest.Create(URI)
      connectrequest.Timeout = Threading.Timeout.Infinite
      Dim connectresponse As Net.WebResponse = connectrequest.GetResponse
      rda = New SqlCeRemoteDataAccess
      Dim remoteConnectstring As String = "Provider=SQLOLEDB;Data
Source=PUI;Initial Catalog=UNO;User Id=sa;Password=sapassword"
      rda.InternetLogin = ""
      rda.InternetPassword = ""
      rda.InternetUrl = "http://70.187.14.200:22/RDA/sscesa20.dll"
      rda.LocalConnectionString = "Data Source=\Program
Files\UNO_Project\uno.sdf;SSCE:Database Password="
      rda.Push("orders", remoteConnectstring, RdaBatchOption.BatchingOn)
      rda.Push("orderline", remoteConnectstring, RdaBatchOption.BatchingOn)
      rda.Push("customer", remoteConnectstring, RdaBatchOption.BatchingOn)
    Catch sqlex As SqlCeException
      Dim sqlError As SqlCeError
      For Each sqlError In sqlex.Errors
        MessageBox.Show(sqlError.Message)
      Next
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    Finally
      rda.Dispose()
      cn2 = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
      cn2.Open()
      Dim cmd2 As SqlCeCommand = cn2.CreateCommand()
      cmd2.CommandText = "Update orders SET order_check = '1'"
      cmd2.ExecuteNonQuery()
      If cn2.State <> ConnectionState.Closed Then
        cn2.Close()
      End If
      Button4.Enabled = True
    End Try
    MessageBox.Show("Finished Export Database")
End Sub

End Class
```

```
Public Class Order_Add_Detail

Private Sub button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
    If cn.State <> ConnectionState.Closed Then
      cn.Close()
    End If
    Application.Exit()
End Sub

Private Sub itemcode_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles itemcode.SelectedIndexChanged
    If itemcode.Text <> "" Then
      If cn.State <> ConnectionState.Open Then
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
      End If
      Dim cmd As SqlCeCommand = cn.CreateCommand()
      Dim reader As SqlCeDataReader
      cmd.CommandText = "SELECT * FROM item WHERE Item_code= '" &
itemcode.Items(itemcode.SelectedIndex) & "'"
      reader = cmd.ExecuteReader
      While reader.Read()
        itemdesc.Text = reader.Item("Item_description")
      End While
      cmd.CommandText = "SELECT Item_Image FROM item WHERE Item_code= '"
& itemcode.Items(itemcode.SelectedIndex) & "'"
      Dim value As Byte()
      Try
        reader = cmd.ExecuteReader()
      Catch ex As SqlCeException
        MessageBox.Show(ex.ToString(), "DB operation failed")
        Exit Sub
      End Try
      If Not reader.Read() Then
        MessageBox.Show("No records in the database")
        Exit Sub
      End If
      value = CType(reader("Item_Image"), Byte())
      Dim ms As MemoryStream = New MemoryStream(value)
      picturebox1.Image = New Bitmap(ms)
    End If
End Sub

Private Sub itemdesc_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles itemdesc.SelectedIndexChanged
    If itemdesc.Text <> "" Then
      If cn.State <> ConnectionState.Open Then
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
      End If
      Dim cmd As SqlCeCommand = cn.CreateCommand()
      cmd.CommandText = "SELECT * FROM item WHERE Item_Description= '" &
itemdesc.Items(itemdesc.SelectedIndex) & "'"
      Dim reader As SqlCeDataReader
      reader = cmd.ExecuteReader
      While reader.Read()
        itemcode.Text = reader.Item("Item_code")
      End While
    End If
End Sub

Private Sub Order_Add_Detail_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        cmd.CommandText = "SELECT MAX(order_number) as ordernumber FROM
orders"
        Dim reader As SqlCeDataReader
        reader = cmd.ExecuteReader
        While reader.Read()
            ordernum.Text = reader.Item("ordernumber")
        End While
        cmd.CommandText = "SELECT Item_Code, Item_Description FROM item"
        reader = cmd.ExecuteReader
        While reader.Read
            itemcode.Items.Add(reader.Item("Item_code"))
            itemdesc.Items.Add(reader.Item("Item_Description"))
        End While
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Add_Button.Click
        If itemcode.Text <> "" And qty.Text <> "" Then
            If cn.State <> ConnectionState.Open Then
                cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
                cn.Open()
            End If
        Try
            Dim cmd As SqlCeCommand = cn.CreateCommand()
            cmd.CommandText = "SELECT MAX(order_line) as order_line FROM
orderline WHERE order_number = "'" & ordernum.Text & "'"
            Dim reader As SqlCeDataReader
            reader = cmd.ExecuteReader
            While reader.Read()
                If IsDBNull(reader.Item("order_line")) Then
                    orderline.Text = "1"
                Else
                    orderline.Text = reader.Item("order_line") + 1
                End If
            End While
            Add_Button.Enabled = False
            cmd.CommandText = "insert into orderline
(order_number,order_line,item_code,qty) values (?,?,?,?)"
            cmd.Parameters.Add("order_number", ordernum.Text)
            cmd.Parameters.Add("order_line", orderline.Text)
            cmd.Parameters.Add("item_code", itemcode.Text)
            cmd.Parameters.Add("qty", qty.Text)
            cmd.ExecuteNonQuery()
            MessageBox.Show("The item has been added to the order")
            Add_Button.Enabled = True
        Catch ex As SqlCeException
            MessageBox.Show(ex.ToString(), "DB operation failed")
            Exit Sub
        End Try
        itemcode.Text = Nothing
        itemdesc.Text = Nothing
        qty.Text = "1"
        orderline.Text = ""
        picturebox1.Image = Nothing
    Else
        MessageBox.Show("Please select item and enter quantity before add the item")
    End If
End Sub

End Class
```

```
Public Class Order_Update_Detail

Private Sub button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
    If cn.State <> ConnectionState.Closed Then
      cn.Close()
    End If
    Application.Exit()
End Sub

Private Sub Order_Update_Detail_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
    cn.Open()
    Dim cmd As SqlCeCommand = cn.CreateCommand()
    Dim reader As SqlCeDataReader
    cmd.CommandText = "SELECT order_line FROM orderline WHERE
order_number= N'" & ordernum.Text & "'"
    reader = cmd.ExecuteReader
    While reader.Read
      orderline.Items.Add(reader.Item("order_line"))
    End While
    cmd.CommandText = "SELECT * FROM item"
    reader = cmd.ExecuteReader
    While reader.Read
      itemcode.Items.Add(reader.Item("Item_code"))
      itemdesc.Items.Add(reader.Item("Item_Description"))
    End While
End Sub
```

```
Private Sub Add_Button_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Add_Button.Click
    If itemcode.Text <> "" And qty.Text <> "" Then
        If cn.State <> ConnectionState.Open Then
            cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
            cn.Open()
        End If
        Try
            Dim cmd As SqlCeCommand = cn.CreateCommand()
            cmd.CommandText = "SELECT MAX(order_line) as order_line FROM
orderline WHERE order_number = '" & ordernum.Text & "'"
            Dim reader As SqlCeDataReader
            reader = cmd.ExecuteReader
            While reader.Read()
                If IsDBNull(reader.Item("order_line")) Then
                    orderlinetemp.Text = "1"
                    orderline.Items.Add("1")
                Else
                    orderlinetemp.Text = reader.Item("order_line") + 1
                    orderline.Items.Add(reader.Item("order_line") + 1)
                End If
            End While
            Add_Button.Enabled = False
            cmd.CommandText = "insert into orderline
(order_number,order_line,item_code,qty) values (?,?,?,?)"
            cmd.Parameters.Add("order_number", ordernum.Text)
            cmd.Parameters.Add("order_line", orderlinetemp.Text)
            cmd.Parameters.Add("item_code", itemcode.Text)
            cmd.Parameters.Add("qty", qty.Text)
            cmd.ExecuteNonQuery()
            MessageBox.Show("The item has been added to the order")
            Add_Button.Enabled = True
        Catch ex As SqlCeException
            MessageBox.Show(ex.ToString(), "DB operation failed")
            Exit Sub
        End Try
        itemcode.Text = Nothing
        itemdesc.Text = Nothing
        qty.Text = "1"
        orderlinetemp.Text = ""
        orderline.Text = Nothing
        PictureBox1.Image = Nothing
    Else
        MessageBox.Show("Please select item and enter quantity before add the item")
    End If
End Sub

Private Sub orderline_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles orderline.SelectedIndexChanged
    If orderline.Text <> "" Then
        If cn.State <> ConnectionState.Open Then
            cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
            cn.Open()
        End If
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        cmd.CommandText = "SELECT orderline.item_code AS item_code, orderline.qty
AS qty, item.item_description AS item_description FROM  orderline INNER JOIN item
ON orderline.item_code = item.item_code WHERE orderline.order_number = '" &
ordernum.Text & "' and orderline.order_line = '" &
orderline.Items(orderline.SelectedIndex) & "'"
```

```
        Dim reader As SqlCeDataReader
        reader = cmd.ExecuteReader
        While reader.Read()
          itemcode.Text = reader.Item("item_code")
          itemdesc.Text = reader.Item("item_description")
          qty.Text = reader.Item("qty")
        End While
        cmd.CommandText = "SELECT item_image FROM  orderline INNER JOIN
item ON orderline.item_code = item.item_code WHERE orderline.order_number = "' &
ordernum.Text & "' and orderline.order_line = "' &
orderline.Items(orderline.SelectedIndex) & "'"
        Dim BitmapBytes As Byte()
        Try
          reader = cmd.ExecuteReader()
        Catch ex As SqlCeException
          MessageBox.Show(ex.ToString(), "DB operation failed")
          Exit Sub
        End Try
        If Not reader.Read() Then
          MessageBox.Show("No records in the database")
          Exit Sub
        End If
        BitmapBytes = CType(reader("item_image"), Byte())
        Dim ms As MemoryStream = New MemoryStream(BitmapBytes)
        PictureBox1.Image = New Bitmap(ms)
    End If
End Sub

Private Sub itemcode_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles itemcode.SelectedIndexChanged
    If itemcode.Text <> "" Then
```

```
      If cn.State <> ConnectionState.Open Then
        cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
        cn.Open()
      End If
      Dim cmd As SqlCeCommand = cn.CreateCommand()
      Dim reader As SqlCeDataReader
      cmd.CommandText = "SELECT * FROM item WHERE Item_code= '" &
itemcode.Items(itemcode.SelectedIndex) & "'"
      reader = cmd.ExecuteReader
      While reader.Read()
        itemdesc.Text = reader.Item("Item_description")
      End While
      cmd.CommandText = "SELECT Item_Image FROM item WHERE Item_code= '"
& itemcode.Items(itemcode.SelectedIndex) & "'"
      Dim value As Byte()
      Try
        reader = cmd.ExecuteReader()
      Catch ex As SqlCeException
        MessageBox.Show(ex.ToString(), "DB operation failed")
        Exit Sub
      End Try
      If Not reader.Read() Then
        MessageBox.Show("No records in the database")
        Exit Sub
      End If
      value = CType(reader("Item_Image"), Byte())
      Dim ms As MemoryStream = New MemoryStream(value)
      PictureBox1.Image = New Bitmap(ms)
    End If
End Sub
```

```
Private Sub itemdesc_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles itemdesc.SelectedIndexChanged
    If itemdesc.Text <> "" Then
        If cn.State <> ConnectionState.Open Then
            cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
            cn.Open()
        End If
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        cmd.CommandText = "SELECT * FROM item WHERE Item_Description= '" &
itemdesc.Items(itemdesc.SelectedIndex) & "'"
        Dim reader As SqlCeDataReader
        reader = cmd.ExecuteReader
        While reader.Read()
            itemcode.Text = reader.Item("Item_code")
        End While
    End If
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    If orderline.Text <> "" Then
        If cn.State <> ConnectionState.Open Then
            cn = New SqlCeConnection("Data source=\program
files\uno_project\uno.sdf;Password=")
            cn.Open()
        End If
        Dim cmd As SqlCeCommand = cn.CreateCommand()
        Try
        cmd.CommandText = "Delete FROM orderline  WHERE
orderline.order_number = '" & ordernum.Text & "' and orderline.order_line = '" &
orderline.Items(orderline.SelectedIndex) & "'"
        cmd.ExecuteNonQuery()
      Catch sqlex As SqlCeException
        Dim sqlError As SqlCeError
        For Each sqlError In sqlex.Errors
          MessageBox.Show(sqlError.Message)
        Next
      Catch ex As Exception
        MessageBox.Show(ex.Message)
      Finally
      End Try
      orderline.Items.Clear()
      Dim reader As SqlCeDataReader
      cmd.CommandText = "SELECT order_line FROM orderline WHERE
order_number= N'" & ordernum.Text & "'"
      reader = cmd.ExecuteReader
      While reader.Read
        orderline.Items.Add(reader.Item("order_line"))
      End While
      itemcode.Text = Nothing
      itemdesc.Text = Nothing
      qty.Text = "1"
      PictureBox1.Image = Nothing
      MessageBox.Show("The item has been deleted from the order successfully.")
    End If
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
```

```
    If ordernum.Text <> "" Then
      Button3.Enabled = False
      If cn.State <> ConnectionState.Closed Then
        cn.Close()
      End If
      Dim oForm As Order_Update_Detail
      Dim nForm As Order_Show
      oForm = New Order_Update_Detail
      Button3.Enabled = True
      oForm.Close()
      oForm = Nothing
      nForm = New Order_Show
      nForm.ordernum.Text = ordernum.Text
      nForm.ShowDialog()
      nForm = Nothing
    Else
      MessageBox.Show("Please select order before add/delete item")
    End If
End Sub

End Class
```