UNIVERSITY OF HERTFORDSHIRE

Faculty of Engineering & Information Science

BACHELOR OF ENGINEERING DEGREE/DEGREE WITH HONOURS IN ELECTRONIC ENGINEERING

Project Report

SPEAKER RECOGNITION FOR ACCESS CONTROL

George Koutsis

April 2002

ABSTRACT

A User Interface Program was created to join the Smart Card Verification program with the Speaker Recognition. Details on what is a smartcard and how it works can be found in the report as well as how the interface program was created.

ACKNOWLEDGEMENTS

The author would like to thank his project supervisor Dr. A. M. Ariyaeeinia for his continued guidance and support throughout the project. The initial stages of programming required the author to ask experienced people in the field of programming. Although he may not be aware of the influence he had, the author would like to thank Mr. George Tsitouridis.

CONTENTS

CHAPTER 1 – INTRODUCTION

1.1 Introduction to the Project

The whole project was based on investigating and implementing an appropriate interface for a Speech Recognition application. The whole mechanism is triggered with the use of a card reader and the suited smart cards.

1.2 Overview of Speaker Recognition

Speech Recognition is the process where a piece is software is charged with the analysis of someone's voice sample in that way that it can be reused to authenticate a feature sample. The authentication is based on a matching process, usually from patterns in both the voice sample and the one stored in the system. In other words this is a way for users to identify themselves to a security system.

1.3 Applications of Speaker Verification and Motivation

There is a wide range of applications where speaker verification can be used to enhance security with the convenience of voice. Applications in the financial and telecommunications markets support transactions and provide information that is valuable and subject to fraud.

This advanced recognition capability addresses the security needs of the high volume call processing marketplace by enabling Interactive Voice Response (IVR) applications to use voice verification, a biometric, in place of existing means of authentication such as touchtone based passwords or PINs

1.4 Project Aims

The Aim of this project is to develop an Interface between a smart card reader and a Speaker Verification Program. The Interface requires the user to input his personal details and records his voice is a file for future use.

1.5 Project Demands

With no previous knowledge on Programming, the main demand of the author was to research and understand these areas as well as any associated subject material. Other demands included:-

- Gaining a good understanding of the programming language required for designing the interface.
- To acquire some knowledge of the operation and requirements of the speaker Verification Program.
- To learn how the smart card reader, reads and writes on the card and implement it on the interface program.
- To investigate current methods of sound capture and decide upon a suitable technique for system implementation.
- Learn more about databases and SQL.
- Time Management. A time plan was created at the outset of the project to give guidelines of when each part of the project should be completed. Although some features may have changed it was necessary to have an indication of the overall time structure of the project.

1.6 Report Structure

Each chapter in the report has an introductory overview to give the reader an idea of its content. The reader may then decide if the documented material is relevant to them.

CHAPTER 2 – SMART CARD TECHNOLOGIES

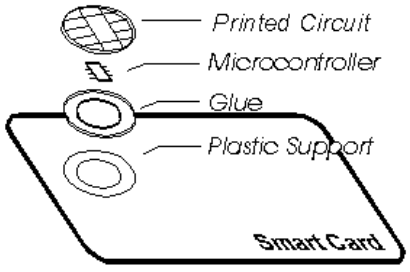
2.1 Overview

The smart card, an intelligent token, is a credit card sized plastic card embedded with an integrated circuit chip. It provides not only memory capacity, but computational capability as well. The self-containment of smart card makes it resistant to attack as it does not need to depend upon potentially vulnerable external resources. Because of this characteristic, smart cards are often used in different applications which require strong security protection and authentication.

2.2 What a Smart Card is

The printed circuit conforms to ISO standard 7816/3 which provides five connection points for power and data. It is hermetically fixed in the recess provided on the card and is burned onto the circuit chip, filled with a conductive material, and sealed with contacts protruding. The printed circuit protects the circuit chip from mechanical stress and static electricity. Communication with the chip is accomplished through contacts that overlay the printed circuit.

The capability of a smart card is defined by its integrated circuit chip. Typically, an integrated circuit chip consists of a microprocessor, read only memory (ROM), non-static random access memory (RAM) and electrically erasable programmable read only memory (EEPROM) which will retain its state when the power is removed. The current circuit chip is made from silicon which is not bendable and particularly easy to break. Therefore, in order to avoid breakage when the card is bent, the chip is restricted to only a few millimetres in size.



2.3 Smart Card Applications

There are over 300,000,000 GSM mobile telephones with smart cards which contain the mobile phone security and subscription information. The handset is personalized to the individual by inserting the card which contains its phone number on the network, billing information, and frequently call numbers.

Almost every small dish TV satellite receiver uses a smart card as its removable security element and subscription information.

The Financial industry has been quick to adopt smart card technology in various countries around the world. Every French Visa Debit card (over 25,000,000) has a chip in it. In Germany, about 40,000,000 banking cards have been issued. EuroPay, MasterCard, and Visa all have smart card programs for their bank members. In the Portugal and Singapore, the national banking networks have launched electronic purse projects. Proton has worked with its banking partners to issue over 25,000,000 electronic purse cards in several countries.

Various countries with national health care programs have deployed smart card systems. The largest is the German solution which deployed over 80,000,000 cards to every person in Germany and Austria.

There are over 100 countries world wide that have reduced or eliminated coins from the pay phone system by issuing smart cards. Greece, Germany, France, UK, Brazil, Mexico, and China have major programs.

Other applications for smart cards include computer/internet user authentication and non-repudiation, retailer loyalty programs, physical access, resort cards, mass transit, electronic toll, product tracking, national ID, drivers license, pass ports, and the list goes on.

2.4 The Smart Card Reader / Writer (Programmer)

In order to program and read a smart card you need a smart card reader. Smart Card Readers are available that interface to RS232 serial ports, USB ports, PCMCIA slots, floppy disk slots, parallel ports, infrared IRDA ports and Keyboards and keyboard wedge readers. Extensive price and performance differences exist between an industrial strength intelligent reader that supports a wide variety of card protocols and a home style win-card reader that only works with microprocessor cards and performs all processing of the data in the PC

2.5 The Software Programmer

Inside the package that came with the smart card reader was also a CD with some useful utilities to program the smart card. The program the author is using is called Scard Delphi Sample. It enables the author to program specific locations into the smart card's memory thus giving more flexibility.

CHAPTER 3 – SPEAKER VERIFICATION AND SMART CARD INTERFACE

3.1 Overview

For the Speaker Verification to work correctly, it needs 10 recordings of the user's voice. That is the only connection these two programs have between them.

3.2 Advantages and Disadvantages of the Smart Card / Speech Recognition Combination

The advantages of this combination are:

- Increased Security
- Fast and Easy Access
- No need to remember numbers, codes etc.

The disadvantages are:

- Unable to get access due to voice change
- Accidental destruction of smart card

3.3 User Interface Issues

The user Interface was created as simple as possible so that even an amateur PC user will be able to use it. Keeping in mind that it must user friendly the result is very good giving the user the ability to access the menus easily. The boxes contain easy to understand words and the feedback given to the user is adequate. There is always the possibility for improvements which will come with the use of the program, finding any weaknesses and reporting them to the author.

CHAPTER 4 – SOFTWARE INTERFACE DESIGN AND IMPLEMENTATION ANALYSIS

4.1 Overview

Perhaps one of the most important part of the project was the Software Interface. There were great limitations and great considerations to be made in order to achieve something such as the one created and presented.

4.2 Interface Issues, under a Software Design and Implementation view

The Programming Environment, selected, to build the application was Borland (today Inspire) Delphi, which actually is an Object-Oriented Visual Pascal. The selection was done based on the facilities provided along with the advice of Mr. George Tsitouridis for a compact, easy and most complete implementation. As stated above, while dealing with visual programming engines, creating an interface is not hard to achieve. The issues that rise in this specific case had more to do with the nature of the application rather than the programming methods. In other words it was more important to consider how to form the interface rather than how to program it. The reason behind it was the fact that a Card Reader Software is applicable only in specific cases, such as the use of a post with a touch screen along with the card reader as a security control mechanism. Therefore, it is obvious that the Interface suitable for cases like this.

4.3 Interface Design

As was stated above, the driving reasoning, behind it, is to create an Interface as simple as possible, without many rising windows, along with a sense of completeness. That is more to fit and presented on the screen at the some, though, level. This means that the author built something that resembles a serial sequence of events within the same window, which makes it much more easy and fast to use and operate, without confusing the user, or wasting his or hers time.

4.4 Interface Implementation / Discussion / Methods and Tools

As soon as all the design issues for the User Interface (UI) had been taken care, the implementation process did not take too long to complete. As it is obvious the main obstacles faced at this part of the project was to decide on the approach to be used in order not to built the interface rather than creating a suitable functionality.

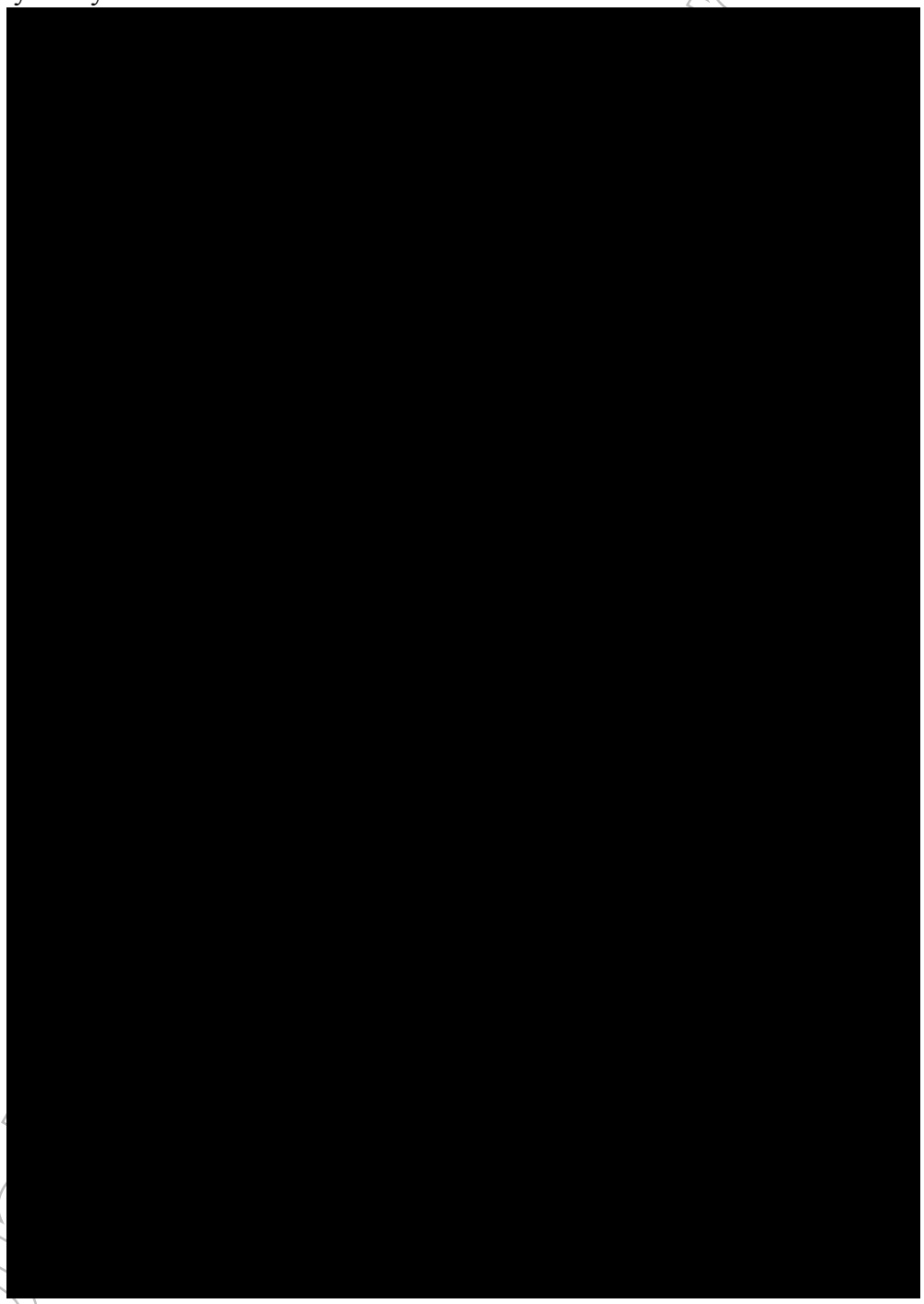
The programming technique, used to achieve such a result, was to use auto hide menus. This means that all the menu, bars, voice sampling controls, etc. are located on the same form and thus on the same windows, but not visible to the user, until a special trigger is activated that will bring them up. This design, though quite hard to implement, as far as the visual part in concerned (because all the visual components – VCL in Delphi – are condense), is extremely efficient. In more detail:

- **It is easier for the user to understand and use it:**
  Since there are actually only 2 main screens and by using a printed –on the screen - list of the actions taken, while in operation, simplifies it in such a way, that even the most computer-naive can work it out, just by reading the on-screen instruction. For this reason the instruction are as simplified, condense and non-technical as possible.
- **It is faster for the user to navigate through:**
  As it is already explained above the application is based only on 2 screens, one screen for each registered and non-registered users. Each screen has certain states according to the whole application state. This makes extremely easy and more important very fast for the users to interact with the system.
- **It is suitable for usage at special cases, such as where mouse is absent such as a Kiosk:**
  The main reason why this design approach was followed is that applications ability to be used in special cases. An example would be a security kiosk with a touch screen, an alphanumeric only keyboard and no mouse. In such cases navigation and usage can be proven to be quite problematic, especially when the UI is too complicated.
- **The memory footprint of the application is minimized:**
  This design approach, though, has its beneficial impact on the whole system as well. The resources that the complier would utilize for the application are minimized as there is no need for extra Object (window) creation in memory during run-time, not to mention that part of the initiated components are not activated instantly, freeing even more RAM.
- **Code is optimized due to the luck of reference to Objects in Units located outside the main one:**
  Again here due to the luck of more than two main Units there are only a few references to objects found in those two. The rest is references from internal to other internal objects, which minimizes resources allocation (discussed in more detail at section 6.3).
- **Homogeneous UI (User Interface):**
  Exactly because of the use of just one form / window the appearance details had to be set only once, not to mention that due to inheritance and the fact that groups of VCL were used, minimized the effort of applying

these details to each visual component. This procedure increased dramatically the User Interface consistency in such a degree that appears to be no differences in the drawing philosophy of each VCL or group of VCLs.

- **Easier and faster to implement:**
  As a result of all the above mention and analyzed, it is more than clear that the whole procedure of modeling and coding the UI was greatly simplified and strait forward leaving almost no space for mistakes and misunderstandings.

Here is the complete flow chart of the User Interface based on the actions taken by the system:



4.5 Conclusions

The Interface was the rather "fun" part where the author had many ideas of what the ideal interface would be according to him. Borland Delphi gives a lot of choices and solutions on how to create the dialogs between the pc and the user. After having understood exactly how the program should be the design was quite easy. With the help of some books on programming the result is pretty good with clear buttons and menus, giving a professional but fairly easy to use interface. The whole interface was created in such a way that it can be used in many different locations and cases (thus not being just a PC) providing in that way greater usability.

CHAPTER 5 – SOFTWARE ENGINE DESIGN AND IMPLEMENTATION ANALYSIS

5.1 Overview

We shall move now, on discussing further technical details of the software application. It is too important to analyze the background engine of the whole system, as it is responsible for the whole functional behavior.

5.2 Software Engine Issues

While designing the core engine of the application, no serious obstacles arose; there were huge problem with the coding procedure itself. The reason for that was mainly the author's luck of knowledge in Visual Object-Oriented Programming and furthermore the importing procedure into Borland Delphi of a VCL, so that the card reader would become available and accessible to the programming environment. This is actually the part where Mr. George Tsitouridis provided extreme support in many different levels. He taught me the basic principals of Visual OO Programming and provided the author with enough education material, such as books and magazines, to get started and actually developing a quite extensive knowledge on it.

5.3 Software Engine Design

Designing the core of the application was one of the most difficult and challenging parts, as it gave the feeling of using a tool without knowing its abilities or its limitation. Finally the following approach was adopted:

The program will have only two main units. One would be the form / window – the User Interface along with its UI specific code, which would make calls on the second unit which is all the code for processing data along with all the subroutines used to control the Database.

Of course the third and the last level, is the database itself which is used to store user personal data, which are processed and compared with that located on the Smart Cards.

5.4 Software Engine Implementation / Discussion / Methods and Tools

Let's start analyzing the application core by levels. The first and lowest level is the Database level. This is implemented using the Delphi's built in PARADOX Database support. It contains essential user information (such as : Last Name, First Name, Father's Name, Mother's Name, Date Of Birth, Registration ID, as long as other control fields that indicate whether a user is deleted from the system or not and whether a user, even a registered, is suppose to use the system – have clearance).

On the second level there is the database control software, along with the appropriate processing routines. In order, though, to implement access on the database, there were two options:

- either to use Delphi's procedural language or
- use a non procedural query language like SQL

The obvious choice was the second one. Though much more difficult to implement, not to mention complicated, it is extremely flexible. At this point another effort was done by the author and that was the process of understanding and learning SQL. Finally there were some SQL queries created to interrogate the database, which were also embedded into Delphi, so that variable can be used instead of constants.

Apart from the third level which is the User Interface, the second one has another part that is charged with the manipulation of data. This was the most difficult of all to implement. The first great obstacle came form the card reader itself. It seemed there was no way of actually having Delphi "see" through the COM port, the Smart Card Reader at the other end. The reason as it was proven later with the help of Mr. George Tsitouridis, is that the specific hardware component uses a special Delphi VCL in order to be integrated into the programming language.

The problems with the Card Reader did not come to an end after the installation of this piece of software. More essential action were about to be taken in order to achieve the communication between the software application and the hardware. The total luck of user manual forced the author to actually decompose example software given in source code, in order to obtain vital information on the programming of the Smart Card Reader VCL.

As soon as this was over and other problem come. That was the sound supporting VCL components in Delphi. What was required was, simply, a facility to record stereo sound by just clicking a button on the interface and of course playing it back. Unfortunately the Delphi built in Media Player API interface was too complicated and in some case not compatible. There the problem solved by downloading from the Internet a special VCL component that would do just that.

Some furthermore technical details can be found in Chapter 6.

5.5 Conclusions

The author chose Delphi because it was easier to implement with it, the smart card reader programmer. The Delphi component, regarding the reader, supplied within the Card Reader CD, was installed after an enormous effort to access the Smart Card Reader though the Programming Language. The most difficult part was the creation of the PARADOX database, not just the table itself, but mainly the whole set and especially the Delphi embedded SQL queries. Finally great consideration must be given in the author's first attempt to search the Internet for freeware VCL and use them within the application as a way of substituting the complex, built it, equivalent component.

CHAPTER 6 – CONCLUSIONS AND FURTHER WORK

6.1 Project Overview

The combination of a smart card with a speech verification programs proves useful especially where we don't want unauthorized access. It provides a high security with the benefit of a unique key which is the user's voice. That means that nobody unless they have their voices recorded can gain access to the system. This technique is widely used many years now with success and is going to be one of the most secure ways to log on to the future computer (and not just only) systems.

6.2 Final Conclusions

This project was a good chance for the author to improve his skills in programming, time management and gathering information. After having spent many hours on the project and reading articles about programming one thing is sure. There is no secure way to gain access to a system. There will always be a way to break it or bypass it through a small mistake in the code. So this makes the need for code improvement definite.

6.3 Further Work

As soon as the project was complete there was more time available and therefore some further steps were taken. Those were the code optimization, a way of enforcing stronger security on the system and finally the attempt to integrate the application with a Speech Recognition Software. In detail:

- **Code Optimization:**

  Having studying in more detail Borland Delphi, the author was able to isolate certain parts in the program code he wrote, that if they were rewritten the application would operate more efficiently (in any aspect).

- **Increase Security Mechanism:**

  The mechanism that was used was quite simple. First of all the system is able to accept only a specific kind of Smart Cards. In more detail it is checking for I2C with 2K memory size. If something else is given, then it will be rejected. Then it is seeking for a secret pass code write in the last 11 characters. That code is **CSRS640870C**. If it is not found then it mean that the card was not issued by the organization using the system and therefore it is again rejected.

- **Integrate with Speech Recognition Software:**

  The author feels confident to state that though a Speech Recognition Software was given, in order to be integrated with the Smart Card Reader software that was impossible due to technical reasons. In detail the software packet called "RTSVS", by Mr. Johann Siaw, was supposed to operate as a background voice sample matching engine, but instead it came out to be a fully compiled stand alone software, to which the author could do nothing, not even study, due to its nature. Therefore the integration never took place, as it is impossible to merge to already compiled executable files.

CHAPTER 7 – CARD READER IN ACTION

7.1 Installation

The installation of the program is pretty much like any other software program.

During the installation the system administrator has the option of entering his personal details: Name and company. After that the install program will display the installation path on the computer.

A status bar will show the installation progress so that the Administrator can see when then installation is completed.

After the completion a shortcut of the program is placed on the desktop giving fast access to the program.

7.2 Execution

After the correct connection of the smart card reader on the RS232 port the program will start prompting the user to insert his card in the reader.

If the card reader is not properly connected an error message will show.

When a new user inserts his card in the card reader the system checks its database and prompts the new user to insert his details in the database.

After the new user has completed his registration the recording phase will begin where the user will be asked to say a specific sentence ten times. These recording as saved in the hard drive as PCM audio files.
After the recording has ended the user doesn't have clearance on the system unless the system administrator checks that everything is ok with the account. This is a small measure to prevent unauthorized personnel registering with the

system and gaining access. So if somebody registers with a stolen smart card, formatted to work in the system, will not be able to use it.

The next time the registered user inserts his card in the reader the system will recognize him and if the system administrator gave him clearance, he/she will be able to use the system.

If not ……..

BIBLIOGRAPHY

1) CONNOLLY, T., BEGG, C. (1998) *Database Systems*. Second Edition.: Addison – Wesley

2) OSIER, D.,GRODMAN, S.,BATSON, S. (1997) *Delphi 3 in 14 Days*. First Edition. Indiana: SAMS

3) DFSStatusBar (1999) Delphi Free Stuff URL: http://www.delphifreestuff.com

4) CANTU, M. (1999) *Mastering Delphi 5*. First Edition. Alameda: SYBEX

5) DOUG BELL, IAN MORREY, JOHN PUGH (1992) Software Engineering. Second Edition. : Prentice Hall

6) NotifyIcon (2000) Tanis. Denmark URL: http://www.tanis.dk

7) KOFFMAN, EB. (1994) *Pascal*. Fifth Edition.: Addison – Wesley

8) ENTSMINGER, G. (1996) *The way of Delphi*. First Edition. New Jersey: Prentice Hall PTR

9) UmValuEdit, INETDetector (2000) UtilMind. URL: http://www.utilmind.com

APPENDIX A – SOURCE CODE

CardReader.dpr:

```
program CardReader;

uses
  Forms,
  Main in 'Main.pas' {MainF},
  DBComp in 'DBComp.pas' {UsersDS: TDataModule};

{$R *.RES}

begin
  Application.Initialize;
  Application.Title := 'Card - Speaker Recognition System';
  Application.CreateForm(TMainF, MainF);
  Application.CreateForm(TUsersDS, UsersDS);
  Application.Run;
end.
```

DBComp.dpr:

```
object UsersDS: TUsersDS
  OldCreateOrder = False
  Left = 237
  Top = 233
  Height = 640
  Width = 817
  object Users: TTable
    Active = True
    IndexFieldNames = 'ID'
    TableName = 'Users.db'
    Left = 48
    Top = 88
  end
  object DS1: TDataSource
    DataSet = Users
    Left = 96
    Top = 88
  end
  object SearchUser: TQuery
    DataSource = DS1
    SQL.Strings = (
      'SELECT Name, Clear '
      'FROM users '
      'WHERE ID=:IDI;')
    Left = 144
    Top = 88
    ParamData = <
      item
        DataType = ftUnknown
        Name = 'IDI'
        ParamType = ptUnknown
      end>
  end
  object DS2: TDataSource
    AutoEdit = False
    DataSet = SearchUser
```

```
  Left = 200
  Top = 88
end
object SearchDel: TQuery
  DataSource = DS1
  SQL.Strings = (
    'SELECT ID'
    'FROM users '
    'WHERE Del ='#39'Y'#39';')
  Left = 144
  Top = 144
end
object DS3: TDataSource
  AutoEdit = False
  DataSet = SearchDel
  Left = 200
  Top = 144
end
object Total: TQuery
  DataSource = DS1
  SQL.Strings = (
    'SELECT COUNT(*) As sinolo'
    'FROM users;')
  Left = 144
  Top = 200
end
object DS4: TDataSource
  AutoEdit = False
  DataSet = Total
  Left = 200
  Top = 200
end
object Insert: TQuery
  DataSource = DS1
  ParamCheck = False
  SQL.Strings = (
```

```
      'INSERT INTO users VALUES (:IDI, :NAM, :MNAM, :FNAM, :DB, '#39'N'#39',
'#39'N' +
      #39');')
    UniDirectional = True
    UpdateMode = upWhereChanged
    Left = 144
    Top = 256
    ParamData = <
      item
        DataType = ftUnknown
        Name = 'IDI'
        ParamType = ptUnknown
      end
      item
        DataType = ftUnknown
        Name = 'NAM'
        ParamType = ptUnknown
      end
      item
        DataType = ftUnknown
        Name = 'MNAM'
        ParamType = ptUnknown
      end
      item
        DataType = ftUnknown
        Name = 'FNAM'
        ParamType = ptUnknown
      end
      item
        DataType = ftUnknown
        Name = 'DB'
        ParamType = ptUnknown
      end>
  end
  object Delete: TQuery
    SQL.Strings = (
      'DELETE FROM users'
```

```
    'WHERE ID=:IDI;')
  Left = 144
  Top = 312
  ParamData = <
    item
      DataType = ftUnknown
      Name = 'IDI'
      ParamType = ptUnknown
    end>
 end
end
```

Main.dpr:

```
object MainF: TMainF
  Left = 476
  Top = 139
  ActiveControl = CloseB
  BorderIcons = []
  BorderStyle = bsSingle
  Caption = 'Card - Speaker Recognition System'
  ClientHeight = 501
  ClientWidth = 535
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  Icon.Data = {
    0000010001002020100001000400E802000016000000280000002000000040000
    0000010004000000000000000000000000000000000000000000000000000000
    0000000080000080000000808000800000008000800080800000C0C0C0008080
    80000000FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF000000
    0000000000000000000000000000000000000000000770000000000000000000
    00000000F787000000000000000000000770080787000000000000000000000
    088777008787000000000000000000008F7878F778700000000000080000000
    08F78708F787000000000008B33000000007870007870000000000800BB800000
    0087870087870000000833BB80000008F77878F7787000000008BBB80300000
    08F7878F7787000000800BB80300000000078708F78700000833BB8030000000
    00878708F78800008BBBB803000000008F77878F77F800008BBB803000000000
    8F7788F77F778000BBB803000000000008F78F77F000000BBB80300000000000
    08F8F7700888888BB803000000000000008F8F77088BBBBBBB8030000000000008
    F78F708BBBBBBBB7B30000000000008F778F70BBBB7B7B7B73000000000008F7
    7F8F0BBBB0B0B7B7B300000000008F77F78F8BBBBB0B0B7B7300000000008F77
    7F788BBBBBB0B0B7B300000000008F77F7008BBB880B0B0B7300000000008F77
    708F8BB00770B0B73000000000008F77F0878BB00070BBB730000000000008F7
    708808B00800BB73000000000000008F77000788000BB3300000000000000008
    FFF0878077773000000000000000000088088088088800000000000000000000
    0000800080000000000000000000000000000000000000000000000000000FFF9
```

```
  FFFFFFF0FFFFFE607FFFFC007FFFF8007FFFF8007FF1F8007FE0FC107F80F800
  7F00F0007F00F8007C01FC007803F8007007F000700FF000201FF800003FF800
  007FF00000FFE00001FFC00001FF800001FF000001FF000001FF000001FF0008
  03FF000403FF800207FFC0000FFFE0001FFFF0107FFFFE73FFFFFF07FFFF}
OldCreateOrder = False
Position = poDesktopCenter
PixelsPerInch = 96
TextHeight = 13
object P1: TPanel
  Left = 0
  Top = 0
  Width = 535
  Height = 460
  Align = alClient
  TabOrder = 0
  object L2: TStaticText
    Left = 1
    Top = 34
    Width = 533
    Height = 31
    Align = alTop
    Alignment = taCenter
    AutoSize = False
    BorderStyle = sbsSunken
    Font.Charset = GREEK_CHARSET
    Font.Color = clWindowText
    Font.Height = -19
    Font.Name = 'Tahoma'
    Font.Style = [fsBold]
    ParentFont = False
    TabOrder = 0
  end
  object ME: TMemo
    Left = 1
    Top = 65
    Width = 343
    Height = 394
```

```
  TabStop = False
  Align = alClient
  ReadOnly = True
  TabOrder = 1
end
object L1: TStaticText
  Left = 1
  Top = 1
  Width = 533
  Height = 33
  Align = alTop
  Alignment = taCenter
  Caption = 'Welcome to the University of Hertfordshire'
  Font.Charset = GREEK_CHARSET
  Font.Color = clTeal
  Font.Height = -24
  Font.Name = 'Tahoma'
  Font.Style = [fsBold]
  ParentFont = False
  TabOrder = 2
end
object P3: TPanel
  Left = 344
  Top = 65
  Width = 190
  Height = 394
  Align = alRight
  BevelOuter = bvNone
  BorderWidth = 1
  TabOrder = 3
  Visible = False
  object GBB: TGroupBox
    Left = 1
    Top = 248
    Width = 188
    Height = 145
    Align = alBottom
```

```
Caption = ' Voice Sampling '
TabOrder = 1
Visible = False
object L3: TLabel
  Left = 24
  Top = 80
  Width = 92
  Height = 13
  Caption = 'Number of attempts'
end
object STAR: TButton
  Left = 8
  Top = 24
  Width = 169
  Height = 25
  Caption = 'Start Voice Sampling'
  TabOrder = 0
  OnClick = STARClick
end
object STOR: TButton
  Left = 8
  Top = 48
  Width = 169
  Height = 25
  Caption = 'Strop Voice Sampling'
  TabOrder = 1
  OnClick = STORClick
end
object Reseq: TButton
  Left = 8
  Top = 112
  Width = 169
  Height = 25
  Caption = 'Restart Sequence'
  TabOrder = 2
  OnClick = ReseqClick
end
```

```
  object ED5: TEdit
    Left = 120
    Top = 80
    Width = 41
    Height = 19
    BevelInner = bvNone
    BevelOuter = bvNone
    Ctl3D = False
    ParentCtl3D = False
    ReadOnly = True
    TabOrder = 3
  end
end
object GBC: TGroupBox
  Left = 1
  Top = 1
  Width = 188
  Height = 88
  Align = alTop
  Caption = ' Voice Sampling '
  TabOrder = 2
  Visible = False
  object SAV: TButton
    Left = 8
    Top = 24
    Width = 169
    Height = 25
    Caption = 'Start Voice Sampling'
    TabOrder = 0
    OnClick = SAVClick
  end
  object SOV: TButton
    Left = 8
    Top = 48
    Width = 169
    Height = 25
    Caption = 'Strop Voice Sampling'
```

```
    TabOrder = 1
    OnClick = SOVClick
  end
end
object GBA: TGroupBox
  Left = 1
  Top = 89
  Width = 188
  Height = 159
  Align = alClient
  Caption = ' Personal Data '
  TabOrder = 0
  Visible = False
  object GoOnB: TButton
    Left = 102
    Top = 216
    Width = 75
    Height = 25
    Caption = 'Proceed'
    TabOrder = 4
    Visible = False
    OnClick = GoOnBClick
  end
  object GB4: TGroupBox
    Left = 2
    Top = 162
    Width = 184
    Height = 49
    Align = alTop
    Caption = ' Date of birth '
    TabOrder = 3
    Visible = False
    object Cala: TDateTimePicker
      Left = 8
      Top = 16
      Width = 169
      Height = 21
```

```
    CalAlignment = dtaLeft
    Date = 21916.1567955903
    Time = 21916.1567955903
    DateFormat = dfShort
    DateMode = dmUpDown
    Kind = dtkDate
    ParseInput = False
    TabOrder = 0
    OnChange = CalaChange
  end
end
object GB3: TGroupBox
  Left = 2
  Top = 113
  Width = 184
  Height = 49
  Align = alTop
  Caption = ' Father'#39's Name '
  TabOrder = 2
  Visible = False
  object ED3: TEdit
    Left = 8
    Top = 16
    Width = 169
    Height = 21
    TabOrder = 0
    OnChange = ED3Change
  end
end
object GB2: TGroupBox
  Left = 2
  Top = 64
  Width = 184
  Height = 49
  Align = alTop
  Caption = ' Mother'#39's Name '
  TabOrder = 1
```

```
      Visible = False
      object ED2: TEdit
        Left = 8
        Top = 16
        Width = 169
        Height = 21
        TabOrder = 0
        OnChange = ED2Change
      end
    end
    object GB1: TGroupBox
      Left = 2
      Top = 15
      Width = 184
      Height = 49
      Align = alTop
      Caption = ' Full Name '
      TabOrder = 0
      Visible = False
      object ED1: TEdit
        Left = 8
        Top = 16
        Width = 169
        Height = 21
        TabOrder = 0
        OnChange = ED1Change
      end
    end
  end
 end
end
object P2: TPanel
  Left = 0
  Top = 460
  Width = 535
  Height = 41
  Align = alBottom
```

```
  TabOrder = 1
  object CloseB: TButton
    Left = 456
    Top = 8
    Width = 75
    Height = 25
    Caption = 'Close'
    TabOrder = 0
    OnClick = Button1Click
  end
  object PB: TProgressBar
    Left = 8
    Top = 8
    Width = 201
    Height = 23
    Min = 0
    Max = 100
    Smooth = True
    TabOrder = 1
  end
end
object SC: TSmartCard
  Active = True
  AutoUnlock = False
  Language = lngEnglish
  LanguageText.Strings = (
    'CrdDetect=Detecting card...'
    'CrdInsert=Please insert a smartcard...'
    'CrdInvalid=Invalid / unknown smartcard!'
    'CrdLocked=Card is locked by '#39'#1'#39
    'CrdReady=Card ok'
    'CrdValid=Card activated and ready'
    'Progress=Operation in progress #1'
    'Search=Looking for a terminal at #1 - #2'
    'SearchFail=No terminal found at #1!'
    'SearchOK=Terminal OK at #1'
    'TerminalErr=Please check terminal settings, no access!'
```

```
    'TerminalTyp=Wrong terminal type!')
  StatusText = 'Please check terminal settings, no access!'
  OnCardDetect = SCCardDetect
  OnCardActive = SCCardActive
  OnCardInvalid = SCCardInvalid
  OnCardWait = SCCardWait
  OnDeviceError = SCDeviceError
  OnProgress = SCProgress
  Left = 72
end
object WinXP1: TWinXP
  Left = 40
end
object AU: TAudio
  Player.BitsPerSample = _16
  Player.Channels = Stereo
  Player.SampleRate = 44000
  Recorder.BitsPerSample = _16
  Recorder.Channels = Stereo
  Recorder.SampleRate = 44000
  Recorder.TrigLevel = 0
  Recorder.Triggered = False
  Version = '4.1 (32bit)'
  OnPlayed = AUPlayed
  Left = 104
end
object IL: TImageList
  Left = 136
  Bitmap = {
    494C010103000400040010001000FFFFFFFFFF10FFFFFFFFFFFFFFFF424D3600
    0000000000003600000028000000400000001000000001002000000000000010
    00000000000000000000000000000000000000FFFFFF00E4E9FB00FFFFFF00FFFF
    FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
    FF00FFFFFF00FFFFFF00E7EBFC00FFFFFF0000000000000000000FCFCFC00E8E8
    E900E7E8E800FAFAFB0000000000000000000000000000000000F8F8FA00E3E3
    E400E3E4E500FBFBFC0000000000000000000FFFFFF00FFFFFF00FFFFFF00FFFF
    FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
```

FF00FFFFFF00FFFFFF00FFFFFF0000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF0085B785000F6F0F001674
16001A761A001A761A0018781800177917001 37D13000D7F0D000A7E0A00077C
0700027B0200007000007FB07F00FFFFFF0000000000FEFEFE00E2E2E3006877
8200776B6700C2C0C200F7F7F70000000000000000F5F5F500C1BEBE00496B
80007C747200E0E1E100FEFEFE0000000000C0C8EA000B2DB3001333B4001938
B8001B3AB8001939B8001839B9001537B8001237BD000D34C0000A31BE00062E
BF00022BC0000026B600001D9400FFFFFF0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF00118311001F8C1F002A91
2A002F932F002E942E002C962C00299A2900239E23001CA31C0015A415000DA4
0D00059F050001910100006F0000FFFFFF0000000000FEFEFE0076A4BE0028D7
FF000074A7005C4F4800B5B1B000F1F1F100F1F0F100B1AEAE00237BA70095F1
FF003A617800DBDBDC00FEFEFE0000000000072FCF00183ED7002449D9002C4F
DB002F52DC002C51DC002B52DD002750DD00224FDF001B4BE1001447E1000C42
E200053AE0000032D8000026B700FFFFFF0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF00198D19002C962C00379C
37003D9F3D003C9F3C0039A13900A3D6A300FFFFFF0024AF24001CB11C0013B2
13000AAD0A00049F040002790200FFFFFF00000000000000000E0E6EA00E9F2
F60001CCFF000F93C0004E4440009994930090909 10063A1C0002AD5FF0038D7
F60089827F00F1F1F1000000000000000000B35DC002349E0003154E1003A5C
E3003D5EE4004667E600C1CCF700FFFFFF00E5EAFC007595F1001B51E900124A
E9000A43E8000339E100012BC000FFFFFF0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF0022912200389C380043A2
430048A4480045A5450042A64200FFFFFF00FFFFFF00FFFFFF0021B5210018B6
18000EB10E0008A30800057E0500FFFFFF00000000000000000FBFCFC00076A
9B0017D3FF0015D3FF003BCCEB002C495A00CDE1EB004BE4FF0008CFFF000069
9B00C0BDBD00FBFBFC0000000000000000001039DD002E52E1003D5EE4004665
E500BAC6F600FFFFFF0093A8F1003A63E700597EEC00E4EBFC00FFFFFF001750
EA000E48EA00073DE200042EC200FFFFFF0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF002C962C0042A042004CA5
4C004FA74F004CA74C0046A7460040AA4000FFFFFF00FFFFFF00FFFFFF001AB3
1A0014AF14000FA30F000B800B00FFFFFF0000000000000000000000000096B8

CC00EBFBFF0002CCFF0035DEFF006EF0FF0097FEFF005DEBFF0024D8FF004C67
7700E5E4E500000000000000000000000000153EDE00385AE3004665E50090A2
EF00FFFFFF00496AE6004468E7003D66E7003562E8002B5CE9007496F100FFFF
FF00134BE9000E42E1000A32C200FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF00359A35004BA54B0052A8
520053A953004EA84E0049A7490041A8410038AA3800FFFFFF00FFFFFF00FFFF
FF0019AC190018A2180012821200FFFFFF00000000000000000FAFAFA00D9D9
DB005BA9D0000ACFFF0020D7FF0053E8FF0089FAFF007BF5FF0021A4CC007066
6400D8D7D800F8F8FA0000000000000000001C42DF004261E4004E6BE600FFFF
FF00516EE6004C6CE6004669E700FFFFFF003661E7002C5CE8002457E900E3E9
FC006E8FF0001647E1001037C300FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF003F9F3F0053A95300FFFF
FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
FF00FFFFFF001F9E1F0018811800FFFFFF00FDFDFD00EEEEEE00BDB9B9003D6C
87000BC1F0001CD5FF0009CFFF003BE0FF0072F2FF0093FDFF004BD7F0002371
98006C625D00B7B5B500EBEBEC00FCFCFD002248DF004B69E5005470E700FFFF
FF00536FE7004D6BE6004668E600FFFFFF00355EE6002B59E7002353E600476F
EB00F1F4FD001D4BE000163BC100FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF0045A245005AAC5A00FFFF
FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
FF00FFFFFF00259A25001D7F1D00FFFFFF00E0E0E0007F8588000F8AB80019D4
FF0058EAFF002DDBFF0003CDFF0027D9FF005AEAFF008FFCFF0071F2FF0034DD
FF000085B800434548008C848300DEDDDE00284DE000526EE7005973E800FFFF
FF00546FE7004D6AE6004565E500FFFFFF00345BE4002B55E5002450E5001E4D
E500FFFFFF00244EDF001B3DC000FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF004FA74F0063B1630061AF
610059AB590051A6510048A248003F9F3F00369C3600FFFFFF00FFFFFF00FFFF
FF00269926002A972A00217E2100FFFFFF003079A30000C9FC0073F3FF0095FE
FF006AF0FF0040E2FF0015D3FF000FD1FF0044E3FF007AF5FF0089FAFF004CE6
FF0011D2FF001DD3FC001E648C00A09E9F002F52E1005D77E800607AE800FFFF
FF00556FE7004D69E5004564E400FFFFFF003458E3002B53E300264FE300748F
ED00C8D2F8002950DD001F40BF00FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000FFFFFF0053A953006CB66C0068B4
68005EAD5E0054A854004CA34C00429F4200FFFFFF00FFFFFF00FFFFFF002997
29002B982B002D952D00237E2300FFFFFF00076D9E00C9E3EE0084B9D10061B2
CC004097BB000D7BAA0056E1FF0001CCFF002DDBFF00117CAA004499BB0077B1
CC00AACCDD00D4E5EE000877A700C1C1C1003255E2006780E9006881E900F5F7
FE00A1AEF200506BE6004764E400FFFFFF003658E2002E53E2002A50E200FFFF
FF003659E4002D52DD002242BF00FFFFFF0000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000FFFFFF005EAF5E007ABD7A0070B8
700063B063005AAB5A0052A65200FFFFFF00FFFFFF00FFFFFF00339933003099
3000309830002F942F00237D2300FFFFFF0000000000000000000000000000
000000000000C4DBE800E2FAFF0010D1FF0017D4FF005D656A00E2E1E300F3F3
F400F1F1F200F0EEEE00F0F0F000F8F8FA003759E3007089EB00718AEB006780
EA00FFFFFF009EADF1004B67E5004361E4003B5CE3003456E200FFFFFF00B0BD
F4002D51E2002E51DE002443BF00FFFFFF0000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000FFFFFF006BB56B008DC68D0080C0
80006FB76F0067B2670060AE6000B4D9B400FFFFFF004CA54C0049A4490041A1
41003A9D3A00309530001E7A1E00FFFFFF0000000000000000000000000000
00000000000000000000D4E5EE0022D7FF0002AADD008B878600F4F4F4000000
00000000000000000000000000000000003E5EE3007F96ED008398ED00718A
EB00657EE900F5F6FE00FFFFFF00FFFFFF00FFFFFF00FFFFFF0091A4F000385A
E3003457E2003053DD002140BD00FFFFFF0000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000FFFFFF0077BB77009DCF9D008CC6
8C0079BC790070B8700069B4690065B2650062B062005DAE5D0056AB56004EA7
4E0041A141002F942F0019771900FFFFFF0000000000000000000000000000
00000000000000000004EA0C90034DDFF00016FA100B3AFAE00FBFBFB000000
00000000000000000000000000000004867E50094A6F00096A8F0008095
ED00728BEB006B84EA00657EE800617BE8005E78E8005A75E8005370E6004B6A
E5004061E4003154DD001B3BBC00FFFFFF0000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000FFFFFF00B1D8B10076BB760067B3
67005BAD5B0054A954004FA74F004AA44A004BA54B0046A346003FA03F003B9E
3B0031983100238C23008ABB8A00E5EAFB0000000000000000000000000000
00000000000000000002C7AA80062E7FF0027678D00D0D0D000FEFEFE000000
00000000000000000000000000000000005D78E8008FA2EF0092A5F0007B92

EC00718AEB006781E900647EE900607AE8005E78E8005874E7005370E6004A69
E5003D5EE4002C50DB001636B900FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFFFFF00FFFFFF00FFFFFF00FFFF
FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF0000000000000000000000000000000
00000000000000000000D3E4ED00046D9E00A2ADB400F3F3F300000000000000
0000000000000000000000000000000000FFFFFF00607BE8004261E4003859
E3003557E2003053E2003254E2003154E2002B4EE100284DE1002349DF002449
E0001B42DF00133AD500B2BCE600FFFFFF0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000424D3E000000000003E000000
28000000400000001000000001000100000000008000000000000000000000
0000000000000000000000FFFFFF000000C3C3000100000000818100000000
0000800100000000000C003000000000000C003000000000000E00700000000
0000C00300000000000000000000000000000000000000000000000000000000
00000000000000000000F800000000000000FC1F000000000000FC1F00000000
0000FC1F000000000000FC3F0000000000000000000000000000000000000000
000000000000}
  end
end

DBComp.pas:

```
unit DBComp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Db, DBTables;
type
  TUsersDS = class(TDataModule)
    Users: TTable;
    DS1: TDataSource;
    SearchUser: TQuery;
    DS2: TDataSource;
    SearchDel: TQuery;
    DS3: TDataSource;
    Total: TQuery;
    DS4: TDataSource;
    Insert: TQuery;
    Delete: TQuery;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  UsersDS: TUsersDS;

implementation

{$R *.DFM}
end.
```

Main.pas:

```
unit Main;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  SCardC32, ComCtrls, ExtCtrls, StdCtrls, Buttons, Db, DBTables, Grids,
  DBGrids, Mask, DBCtrls, Gauges, WinXP, Audio, ImgList;

type
  TMainF = class(TForm)
    SC: TSmartCard;
    P1: TPanel;
    P2: TPanel;
    L2: TStaticText;
    ME: TMemo;
    CloseB: TButton;
    L1: TStaticText;
    PB: TProgressBar;
    WinXP1: TWinXP;
    P3: TPanel;
    GB1: TGroupBox;
    GB2: TGroupBox;
    GB3: TGroupBox;
    GB4: TGroupBox;
    GoOnB: TButton;
    ED1: TEdit;
    ED2: TEdit;
    ED3: TEdit;
    AU: TAudio;
    IL: TImageList;
    GBA: TGroupBox;
    GBB: TGroupBox;
    STAR: TButton;
    STOR: TButton;
    Reseq: TButton;
    L3: TLabel;
    ED5: TEdit;
    GBC: TGroupBox;
    SAV: TButton;
```

```
    SOV: TButton;
    Cala: TDateTimePicker;
    procedure SCCardWait(Sender: TObject; DeviceIndex: Integer);
    procedure SCCardDetect(Sender: TObject; DeviceIndex: Integer);
    procedure SCDeviceError(Sender: TObject; DeviceIndex: Integer);
    procedure Button1Click(Sender: TObject);
    procedure SCCardInvalid(Sender: TObject; DeviceIndex: Integer);
    procedure SCProgress(Sender: TObject; DeviceIndex, Progress: Integer);
    procedure SCCardActive(Sender: TObject; DeviceIndex: Integer);
    procedure ED1Change(Sender: TObject);
    procedure ED2Change(Sender: TObject);
    procedure ED3Change(Sender: TObject);
    procedure GoOnBClick(Sender: TObject);
    procedure STARClick(Sender: TObject);
    procedure STORClick(Sender: TObject);
    procedure AUPlayed(Sender: TObject);
    procedure ReseqClick(Sender: TObject);
    procedure SAVClick(Sender: TObject);
    procedure SOVClick(Sender: TObject);
    procedure CalaChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainF: TMainF;
  Timer: Integer;

implementation

uses DBComp;

var
  CCont : TStringList;
```

```
{$R *.DFM}

procedure TMainF.SCCardWait(Sender: TObject; DeviceIndex: Integer);
begin
  ME.Lines.Clear;
  ED1.Clear;
  ED2.Clear;
  ED3.Clear;
  GB2.Visible:=False;
  GB3.Visible:=False;
  GB4.Visible:=False;
  GoOnB.Visible:=False;
  GBB.Visible:=False;
  P3.Visible:=False;
  P3.Visible:=False;
  PB.Position:=0;
  L2.Caption:='Please insert your Smart Card in the indicated slot';
end;

procedure TMainF.SCCardDetect(Sender: TObject; DeviceIndex: Integer);
begin
  ME.Lines.Clear;
  L2.Caption:='Please wait while your Smart Card is being accessed';
end;

procedure TMainF.SCDeviceError(Sender: TObject; DeviceIndex: Integer);
begin
  ME.Lines.Clear;
  L2.Caption:='Device Error! Please try again later';
end;

procedure TMainF.SCCardInvalid(Sender: TObject; DeviceIndex: Integer);
begin
  ME.Lines.Clear;
  L2.Caption:='Your Smart Card is not valid';
end;
```

```
procedure TMainF.SCProgress(Sender: TObject; DeviceIndex,
  Progress: Integer);
begin
  PB.Position:=Progress;
end;

procedure TMainF.SCCardActive(Sender: TObject; DeviceIndex: Integer);

var RE: TDBEdit;

begin
  RE:=TDBEdit.Create(MainF);
  {System checks the type of the card}
  try if SC.CardInfo.Strings[2]='Type=I2C 2K' then
       begin
        CCont:=TStringList.Create;
        {System checks if the card belongs to this system, by checking the last 11 chars}
        try SC.ComandList('Card,MemRead,245,11',CCont) finally end;
        if CCont.Strings[0]='CSRS640870C' then
          begin
            CCont.Clear;
            try SC.ComandList('Card,MemRead,0,10',CCont)finally end;
            {if the first 10 chars are different from '999999999' then the card is activated
and the char indicated is the uses' ID}
            if CCont.Strings[0]<>'9999999999' then
              begin

DBComp.UsersDS.SearchUser.ParamByName('IDI').AsString:=CCont.Strings[0];
                DBComp.UsersDS.SearchUser.Open;
                RE.DataSource:=DBComp.UsersDS.DS2;
                RE.DataField:='Name';
                L2.Caption:='Your Smart Card has been successfully accessed';
                ME.Lines.Add('Welcome back '+RE.EditText+'.');
                RE.Clear;
                RE.DataField:='Clear';
                if RE.EditText='Y' then
                  begin
```

```
                ME.Lines.Add('You have clearance to use the system.');
                ME.Lines.Add('-----------------------------------');
                ME.Lines.Add('-> Standing by for speach recognition.');
                P3.Visible:=True;
                GBC.Visible:=True;
                SOV.Enabled:=False;
              end
             else
              begin
                ME.Lines.Add('You do not have clearance to use the system.');
                ME.Lines.Add('-----------------------------------');
                ME.Lines.Add('If you have just register, you need to wait for you account
activation.');
                ME.Lines.Add('Please remove your card and contact reception.');
              end;
           end
         else
           begin
            L2.Caption:='Your Smart Card has been successfully accessed.';
            ME.Lines.Add('Welcome new user!');
            ME.Lines.Add('Standing by to receive your personal information and voice
sample.');
            ME.Lines.Add('-----------------------------------');
            ME.Lines.Add('-> Please Insert your Personal Data.');
            P3.Visible:=True;
            GBA.Visible:=True;
            GB1.Visible:=True;
            {TODO: Give focus on control}
           end;
         RE.Free;
         CCont.Free;
        end
    end
  else
    begin
     L2.Caption:='Your Smart Card is not valid';
    end;
```

```
  finally end;
end;

procedure TMainF.ED1Change(Sender: TObject);
begin
  GB2.Visible:=True;
end;

procedure TMainF.ED2Change(Sender: TObject);
begin
  GB3.Visible:=True;
end;

procedure TMainF.ED3Change(Sender: TObject);
begin
  GB4.Visible:=True;
end;

procedure TMainF.GoOnBClick(Sender: TObject);

var RE: TDBEdit;
    IDI: String;

begin
  if MessageDlg('Are all the information entered corret?', mtConfirmation, [mbYes,
mbNo], 0)=mrYes then
    begin
      GBA.Enabled:=False;
      GBB.Visible:=True;
      STOR.Enabled:=False;
      ME.Lines.Add('-> Please stand by for voice samplnig.');
      ED5.Text:='1';
      Reseq.Enabled:=False;
      RE:=TDBEdit.Create(MainF);
      DBComp.UsersDS.SearchDel.Open;
      RE.DataSource:=DBComp.UsersDS.DS3;
      RE.DataField:='ID';
```

```
    if RE.EditText='' then
     begin
      RE.Clear;
      RE:=TDBEdit.Create(MainF);
      DBComp.UsersDS.Total.Open;
      RE.DataSource:=DBComp.UsersDS.DS4;
      RE.DataField:='sinolo';
      IDI:=IntToStr(StrToInt(RE.EditText)+1);
      if Length(IDI)=1 then IDI:='000000000'+IDI;
      if Length(IDI)=2 then IDI:='00000000'+IDI;
      if Length(IDI)=3 then IDI:='0000000'+IDI;
      if Length(IDI)=4 then IDI:='000000'+IDI;
      if Length(IDI)=5 then IDI:='00000'+IDI;
      if Length(IDI)=6 then IDI:='0000'+IDI;
      if Length(IDI)=7 then IDI:='000'+IDI;
      if Length(IDI)=8 then IDI:='00'+IDI;
      if Length(IDI)=9 then IDI:='0'+IDI;
     end
    else
     begin
      IDI:=RE.EditText;
      DBComp.UsersDS.Delete.ParamByName('IDI').AsString:=IDI;
      DBComp.UsersDS.Delete.ExecSQL;
     end;
    DBComp.UsersDS.Insert.ParamByName('IDI').AsString:=IDI;
    DBComp.UsersDS.Insert.ParamByName('NAM').AsString:=ED1.Text;
    DBComp.UsersDS.Insert.ParamByName('MNAM').AsString:=ED2.Text;
    DBComp.UsersDS.Insert.ParamByName('FNAM').AsString:=ED3.Text;
    DBComp.UsersDS.Insert.ParamByName('DB').AsString:=DateToStr(Cala.Date);
   end
  else
   begin
    ED1.Clear;
    ED2.Clear;
    ED3.Clear;
    GB4.Visible:=False;
    GB3.Visible:=False;
```

```
    GB2.Visible:=False;
    GoOnB.Visible:=False;
   end;
end;

procedure TMainF.STARClick(Sender: TObject);
begin
 Reseq.Enabled:=False;
 STAR.Enabled:=False;
 STOR.Enabled:=True;

AU.Recorder.RecordToFile('samples\'+DBComp.UsersDS.Insert.Params.ParamValues['I
DI']+'_'+ED5.Text+'.wav',NIL,NIL);
 AU.Recorder.Start;
end;

procedure TMainF.STORClick(Sender: TObject);

begin
 STOR.Enabled:=False;
 AU.Recorder.Stop;

AU.Player.PlayFile('samples\'+DBComp.UsersDS.Insert.Params.ParamValues['IDI']+'_'
+ED5.Text+'.wav',0);
end;

procedure TMainF.AUPlayed(Sender: TObject);
begin
 if MessageDlg('Are you satisfied with the sample you provided? If not you have to
reapeat this stage.', mtConfirmation, [mbYes, mbNo], 0)=mrYes then
   begin
    if ED5.Text='10' then
      begin
       ME.Lines.Add('-> Please stand by for System update.');
       DBComp.UsersDS.Insert.ExecSQL;
       ME.Lines.Add('-> Please stand by for Card update.');
```

```
SC.ComandStr(cmCard+cmMemWrite+'0'+','+'10',DBComp.UsersDS.Insert.Params.Par
amValues['IDI']);
        ME.Lines.Add('-> Registration Complete!');
        SC.Active:=False;
        if MessageDlg('Congratulations! You are now registered on the Voice Recognition
System.', mtInformation, [mbYes], 0)=mrYes then
          begin
            GBA.Enabled:=True;
            ME.Lines.Clear;
            ED1.Clear;
            ED2.Clear;
            ED3.Clear;
            GB2.Visible:=False;
            GB3.Visible:=False;
            GB4.Visible:=False;
            GoOnB.Visible:=False;
            GBB.Visible:=False;
            P3.Visible:=False;
            SC.Active:=True;
          end;
       end;
       ED5.Text:=IntToStr(StrToInt(ED5.Text)+1);
       Reseq.Enabled:=True;
    end;
    STAR.Enabled:=True;
    STOR.Enabled:=False;
    Reseq.Enabled:=True;
end;

procedure TMainF.ReseqClick(Sender: TObject);
begin
  ED5.Text:='1';
end;

procedure TMainF.SAVClick(Sender: TObject);
begin
```

```
  SAV.Enabled:=False;
  SOV.Enabled:=True;
  AU.Recorder.RecordToFile('tocompare.wav',NIL,NIL);
  AU.Recorder.Start;
end;

procedure TMainF.SOVClick(Sender: TObject);
begin
  SOV.Enabled:=False;
  AU.Recorder.Stop;
  P3.Visible:=False;
  GBC.Visible:=False;
  SAV.Enabled:=True;
  ME.Lines.Add('-> Matching Sample.');
end;

procedure TMainF.CalaChange(Sender: TObject);
begin
  GoOnB.Visible:=True;
end;

procedure TMainF.Button1Click(Sender: TObject);
begin
  Application.Terminate;
end;

end.
```