

**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**USER DICTIONARY SYSTEM**

**Graduation Project  
COM- 400**

**Student: Mohammad AL-rafati (20011121)**

**Supervisor: Dr Umit Soyer**

**Nicosia - 2006**

## ACKNOWLEDGMENTS

First of all I am happy to complete the task which I had given with blessing of God and also I am grateful to all the people in my life who have, supported me, advised me, taught me and who have always encouraged me to follow my dreams and ambitions. My dearest parents, my friends. They have taught me that no dream is unachievable. As in the words of Walt Disney "If you can dream it, you can do it."

I wish to thank my advisor, Mr. Umit Soyer, for intellectual support, encouragement, and enthusiasm, which made this project possible, and his patience for correcting both my stylistic and scientific errors.

Finally, I wish by this project to be useful for all students, especially Computer Engineering to support our improvement.

And above, I thank God for giving me stamina and courage to achieve my objectives.

To all of them, my love and respect

## **ABSTRACT**

My project is about dictionary system which is providing translation from English To Turkish and from Turkish to English.

This system is designed and developed to support easy and powerful search about the desired word or translation.

I design this system above of the need to develop the performance and the result of searching and finding desired translation of database.

My system consist of main three process which are search from English to Turkish, From Turkish to English, and add new words to database.

The aim at this project is to maintain powerful search with user friendly from to gain easy and helpful translation.

The users can used this system through the internet. Just open the web page and make Download for this system.

I made my project by using Delphi language because is very development and better than any other language. And this language give me the powerful to understand how I can make program using database.

## TABLE OF CONTENTS

<b>ACKNOWLEDGMENT</b>	<b>i</b>
<b>ABSTRACT</b>	<b>ii</b>
<b>TABLE OF CONTENTS</b>	<b>iii</b>
<b>INTRODUCTIONS</b>	<b>vi</b>
<b>CHAPTER 1. USER DICTIONARY SYSTEM</b>	<b>1</b>
1.1 Over view	1
1.2 Software Requirement specifications	1
1.3 What is Dictionary System	1
1.4 Dictionary System Specifications	1
1.5 System Test Description	1
<b>CHAPTER 2. DATABASE. DELPHI LANGUAGE</b>	<b>14</b>
2.1 Introduction	14
2.2 What is Database?	14
2.2.1 Database management System(DBMS)	14
2.2.2 Database Design	15
2.2.3 Creating the Database	15
2.2.4 Database Object	16
2.3 Tables	16
2.3.1 Opening a table	17
2.3.2 Connecting to Database	17
2.4 Summary	17
<b>CHAPTER 3. DELPHI LANGUAGE</b>	<b>19</b>
3.1 History and Introduction	19
3.2 What is Delphi?	20
3.3 Starting a new Application	20
3.4 Setting Property value	20



3.4.1 Form File	21
3.4.2 Unit File	21
3.4.3 Naming Unit And Project Source Code File	23
3.5 Working With Projects	23
3.5.1 What Is A Project	23
3.5.2 Viewing A Project contents	24
3.5.3 Saving Project And Individual Project File	25
3.5.4 Managing Projects	25
3.5.4.1 Selecting Project to Work On	25
3.5.4.2 Searching For Files	25
3.5.4.3 Removing Items in the Project Manager	25
3.5.4.4 Coping in the Project Manager	26
3.5.4.5 Getting Project Path and Unit Information	27
3.5.4.6 Adding Project to the Project Group	27
3.5.5 Sharing Objects	27
3.5.6 Creating Project Group	27
3.5.7 Specifying Default Project, New Form, Main Form	28
3.5.8 Running A Project	28
3.6 Using Object Pascal	28
3.7 Pascal Source File	29

## **CHAPTER 4. APPENDIX 30**

4.1 Personal Data Page	30
4.2 User Menu Page	31
4.3 User Register	32
4.4 Main Menu Page	33
4.5 Translation Page	34
4.5.1 Translation from English to Turkish	34
4.5.2 Translation from Turkish to English	36

4.6 Advanced Search	37
4.7 Administration Menu Page	39
4.8 Administrator Register Page	40
4.9 Administrator Menu Page	41
4.9.1 Translation Page	42
4.9.2 Add New Page	44
4.9.3 Update Page	45
4.9.4 Delete Page	47
4.9.5 Login New manager	49
4.10 Manager Register Page	50
4.11 Manager Edit Menu Page	51
4.11.1 Update Page	52
4.11.2 Add New Page	53
 <b>CONCLUSION</b>	 <b>55</b>
<b>REFERENCES</b>	<b>56</b>

## Introduction

Dictionary System is providing translation from English to Turkish and from Turkish to English.

This system is designed and developed to support easy and powerful search about the desired word or translation.

I designed this system above of the need to develop the performance and the result of searching and finding desired translation of database.

My system consist of main three process which are search from English to Turkish, From Turkish to English, and add new words to database.

The aim at this project is to maintain powerful search with user friendly from to gain easy and helpful translation.

The users can used this system through the internet. Just open the web page and make Download for this system.

I made my project by using Delphi language because is very development and better than any other language. And this language giving me the powerful to understand how I can made program using database.

My First chapter is all about explaining my project .what is Dictionary System and what Dictionary System Specification, and how we can use Dictionary System and the aim of the dictionary system.

Second chapter is talking about the Data base in Delphi The database one or more, large structured sets of persistent data. Usually associated with software to update and query the data. A simple database might be a single file containing many records, each of which contains the same set of fields, where each field is certain

Fixed width. A database is one component of a database management system.

Third chapter is talk about Delphi language Delphi is the premier development tool within Windows computing, and communications is one of the fastest growing segments of the Windows development Platform

Fourth chapter is about the program Appendix.



## **Chapter 1. USER DICTIONARY SYSTEM**

### **1.1 Over view**

Dictionary system need to Delphi program until to work.  
Dictionary system consists of main three processes which are search from English to Turkish, from Turkish to English, and Add new word to database.  
In edition to the user can be using this system through the internet.

### **1.2 Software Requirements Specification**

In my project, I saved my project in one file (folder) by using the Delphi program.

My project need Delphi program to work so you must install Delphi program on your PC. Follow these steps to know how you can use dictionary system.

1. Install Windows 2000 or XP on your PC.
2. Run Delphi 6 or 7 on Your PC.
3. Install the dictionary system on your PC.

If you are running Windows 2000 or XP Professional on your computer you can install Borland Delphi 6 or 7 for free, after that installing dictionary system on your PC then the dictionary system will take Delphi emblem you will see that.

### **1.3 What is Dictionary System?**

Is provide translation from English to Turkish and from Turkish to English  
This system is designed and developed to support easy and powerful search  
About the desired word or translation.

### **1.4 Dictionary System Specification**

At the beginning my system consist of main three process which are search from English to Turkish, From Turkish to English, and add new words to database

My project is consist too many things at beginning my project is involve to important thing is database which database consist on the information that requirement through using the system which in database have 2-table each table has information one is about the words under English and Turkish anther one is about the names under administrator name and user name. Also my project involved to many forms which each form has consist Button component and Edit component too and also Data Source component and Label component and Chick Box component.

All of this option will be help us through using the system.

See next part to know how each form work and each component work.

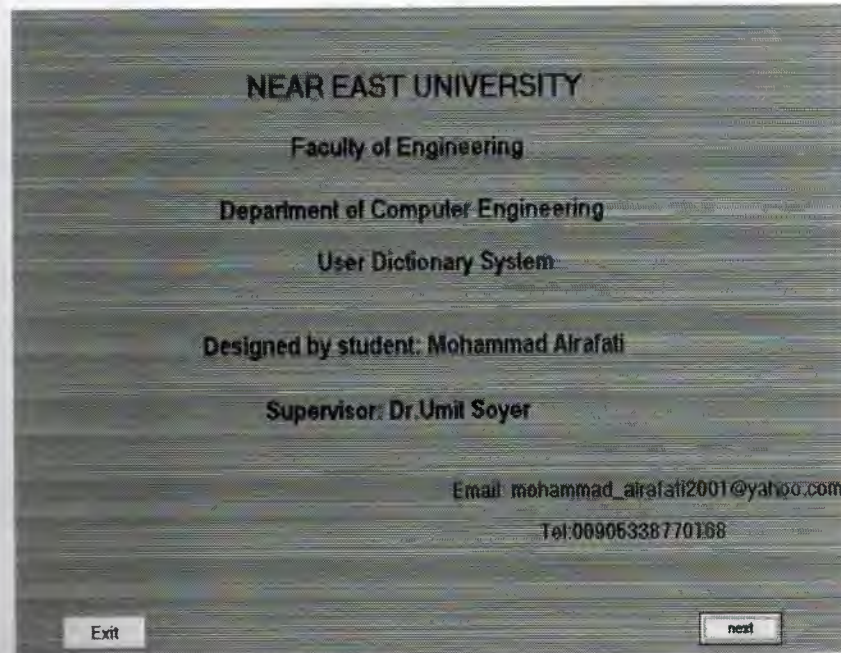
### **1.5 System Test Description**

In this part I will be explain how we can use Dictionary System and explaining about the forms and what is each form do?



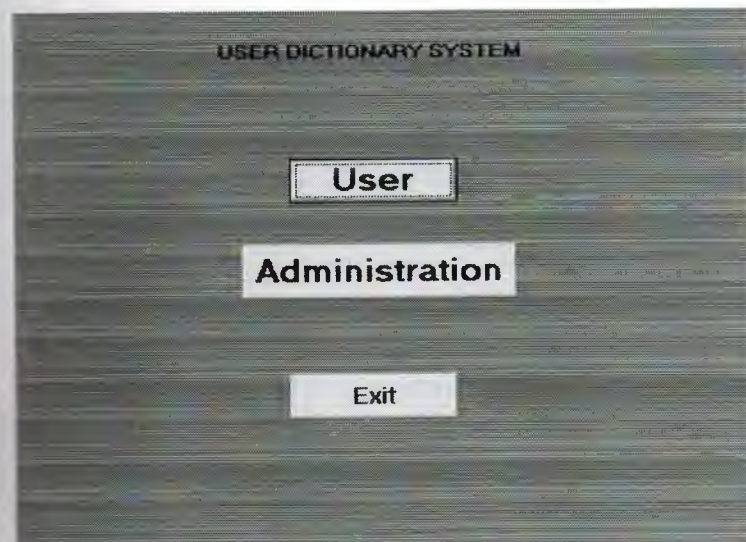
## User Dictionary System

At the beginning when the dictionary system opened we will see the first page is call Data Personal page. This page is talking about what is the project name who designed and who was supervisor on this project. This page also has 2-button one is exit button anther one is next button if we pressed on exit button we will close the system but if we pressed on next button will be go to the next page see figure (1).



Personal data Figure (1)

After when we pressed on next button from Data Personal page will be see next page this page is name User Menu is talking about which you are exactly, user or administration which is Consists 3-bytton first button is about user second one is about administrator third one is exit button if we presses exit we will back to last page (Data Personal page) but if we press on User button we will enter to the pages which are talking about what is user can do? If we pressed on administration button we will enter to the pages which are talking about administrator job. See figure (2).

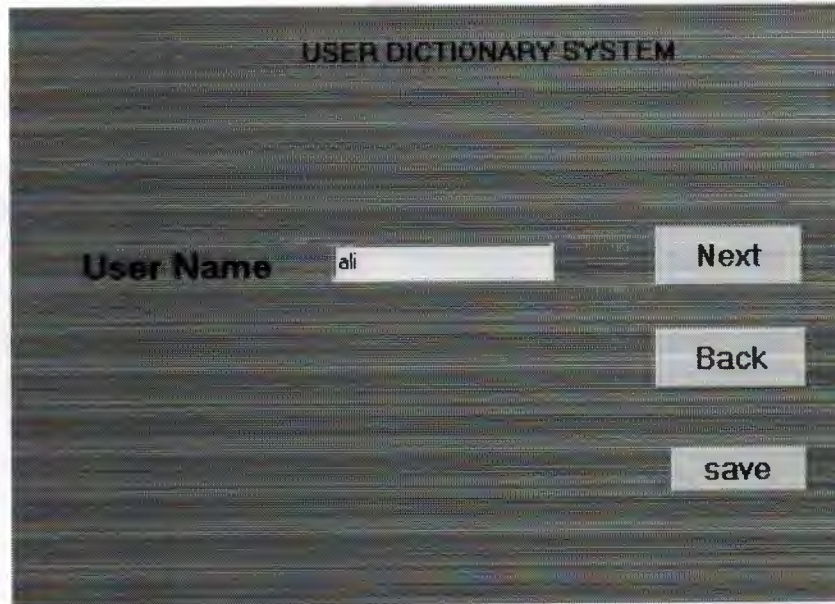


User Menu Figure (2)



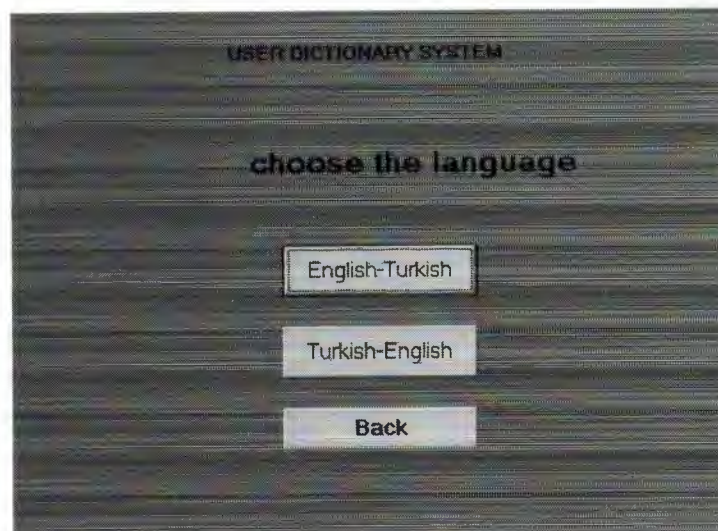
## User Dictionary System

Let's go to press on the User button then will show us page this page is name User Register page. This page is special for user see the figure (3)



User Register Figure (3)

Which the user must register his name before to use the system this step it allow the user to use this program again. It have Edit component which from through it the user can write his name inside it and 3-button component first one is next button second one is back button third one is save button. If we pressed back button we will back to the last page (User Menu) if we pressed on save button directly the name of the user will be save into database under the table who responsible about the user name and show message to tell the user your name is saved also he does not need to save his name again because already saved after the save step the user can press Next button to continue. The next button is let us to enter to the next page. Next page is name Main Menu which is consist which language we want to translation from English to Turkish or Turkish to English see the figure (4).



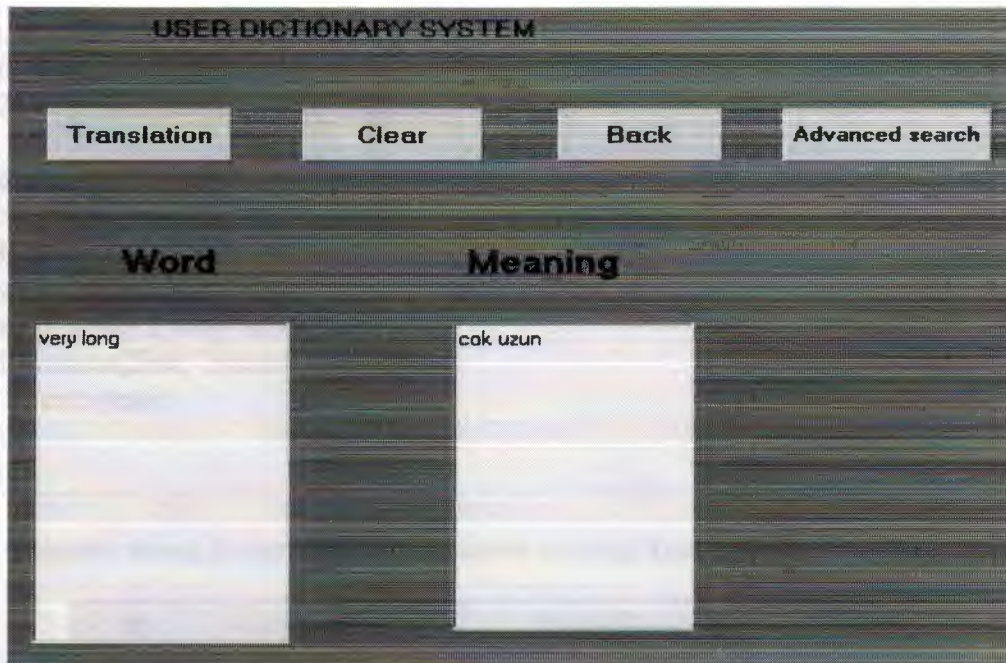
Main Menu Figure (4)



## User Dictionary System

This page is consist on 3-button first one is English to Turkish button second one is Turkish to English third one is Back button. Back button is let us to back to the last page (User Register page). English to Turkish button is allowing us to enter to the translation page which translates from English to Turkish. Also Turkish to English button is allowing us to enter same page but translate from Turkish to English.

If we pressed on English to Turkish button we will to enter to the next page is name translation page which is allow us to translate from English to Turkish see figure (5).



Translation Page Figure (5)

This page is consists 2-Edit box first one about the word which we write the word inside it, second one about the meaning and have 4-button first one is Back button it let us back to the last page (Maine Menu) second one is clear button its work on clean inside Edit boxes, third one is translation button. If we press on translation button after writing the word inside Edit box under the word it will translate the word and show the meaning inside Edit box under the meaning. The special thing in this system it can be translated more than word look to the figure (5) which the words are separated in database table about the 3<sup>rd</sup> button is about advanced search I will be explain after the translate section. Here about how this system can translate tow word together.

```
Procedure TForm4.Button1Click (Sender: TObject);
Begin
  str1:=edit1.Text;
  stt:=length(str1);
  stt1:=pos (' ', str1);
  stt2:=IntToStr(stt1);
  stcop:=StrToInt(stt2);
  if stt1=0 then
  Begin
    table1.locate ('English', str1,[lopartialkey]);
```



```

If str1 <> table1 ['English'] then

showmessage('not found')
  begin
else
edit2.text:=table1.FieldValues['Turkish'];
End
Else
  cop1:=copy(str1,0,stcop-1);
  cop2:=copy(str1,stcop+1,stt);

table1.locate('English',cop1,[lopartialkey]);
first:=table1.FieldValues['Turkish'];
total:=first;
table1.locate('English',cop2,[lopartialkey]);
second:=table1.FieldValues['Turkish'];
total:=concat(first,' ',second);
edit2.Text:=total;
End;
End;

```

But if there's wrong in the word will be show message (not found). See figure (5.1).

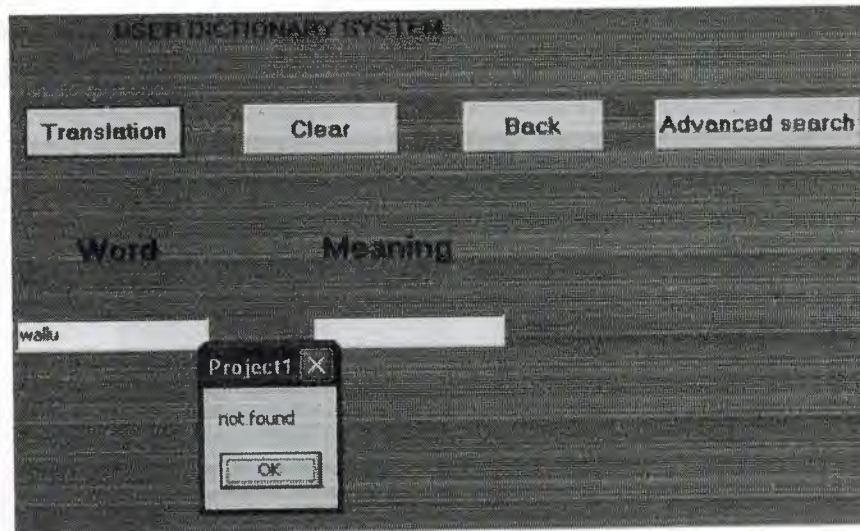


Figure (5.1)

Let's enter into advanced search this part about if the user didn't know how the word write just he write the first 2 character and make search the system will show all the words which start on this character and the user choose the word which he want it and press search on the word he want. See the figure (5.2)

## User Dictionary System

The screenshot shows a web interface titled "User Dictionary System". At the top, there are three buttons: "Translation", "Clear", and "Back". Below these, on the right, is a "search" button. On the left, there are two labels: "Word" and "Meaning". Next to "Word" is a text input field containing "ap". Next to "Meaning" is a text input field containing "elma". To the right of the "Meaning" input field is a dropdown menu with "apple" selected.

Advanced Search figure (5.2)

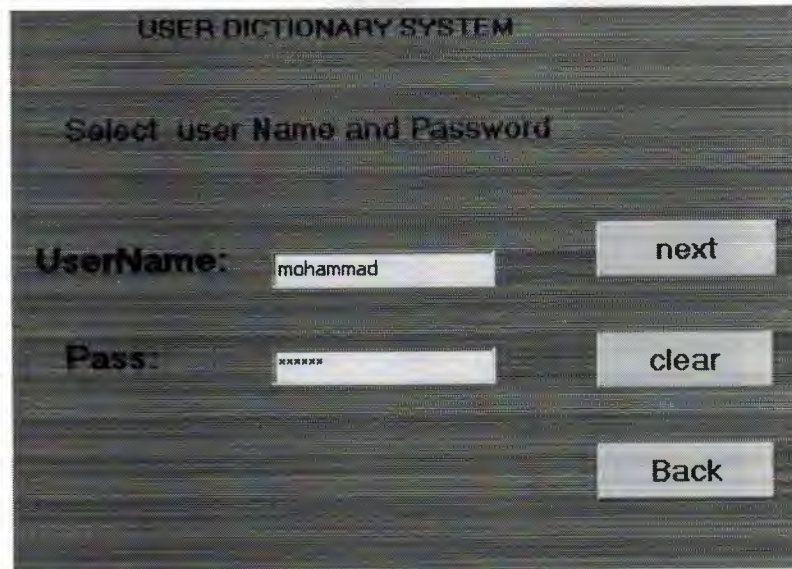
Let's back to the Main menu figure (4) and choose Turkish to English button. Are the same steps in English to Turkish button but the difference between it here the translation from Turkish to English. Until now I explained about the work of the User. Let's back to figure (2) and choose the administration will be showing us this page. See figure (7). This page is name Administration menu it's consist administrator button and manager button, which administrator button just the administrator can enter here and can do it anything, Translate, Add new word, Update, Delete and register the manager name .

The screenshot shows a web interface titled "User Dictionary System" with a subtitle "Administration Menu". Below the subtitle, there are three buttons arranged vertically: "Administrator", "manager", and "Back".

Administration Menu Figure (7)

.for the manager button here the administrator and the manager can enter but the beginning let's choose administrator button, if we pressed on administrator will show us this page (8) is name Administrator Register.





Administrator Register Figure (8)

This page is ask about the administrator name and password to let him to continue it have 2-Edit box which firs one is about the name and second one about the password. And have 3-button one is back button which let us back to the last page (Administration Menu) second one is clear button its work on clean inside Edit boxes third one is next button, when we write the user name and the password then we press on the next to continue and see what the Administrator can do but if the user name And password is wrong will show us message to tell us you can not enter see figure (8.1).

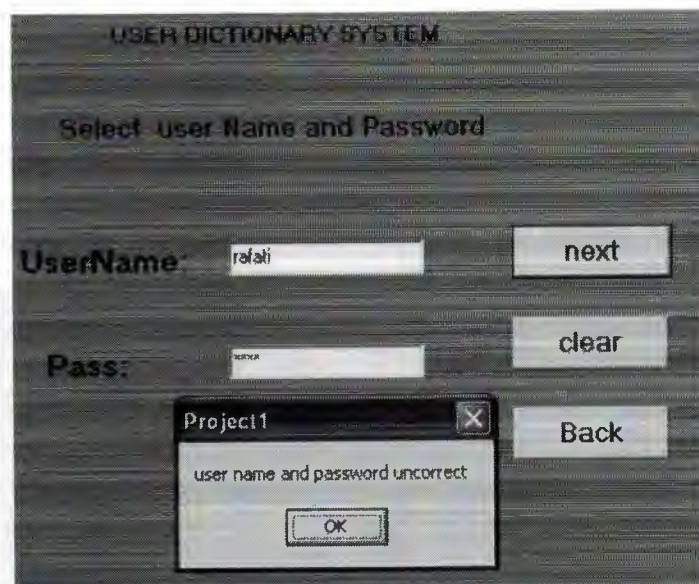
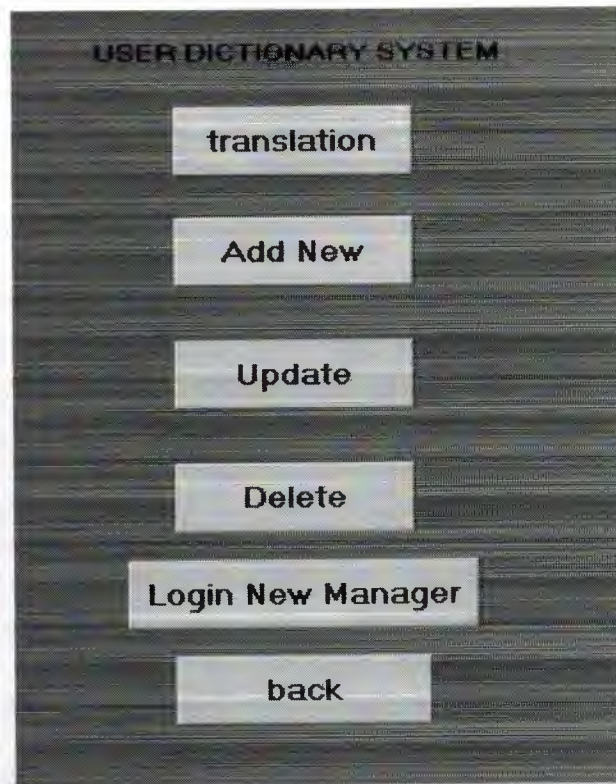


Figure (8.1)

So must to write the correct user name and the password. If we press next will show us next page is name Administrator Menu see figure (9).



## User Dictionary System



Administrator Menu figure (9).

This page is talking about what the Administrator can do. At beginning this page is consists on 6-button, first button is translate button second one is Add New button third one is Update button fourth one is Delete button fifth button is Login New manager sixth button is Back button to. If we press on Back button we will back to User Menu page figure (2), if we press on translate button we will enter to translation page, if we press on Add New button we will enter to Add New page, if we press on Update button we will enter to update page, if we press on Delete button we will enter to delete page, if we press on login new manager button we will enter to login new manager page. Let's go to enter translation page see the figure (5). Let's go to enter to the Add New Administration see the figure (10)

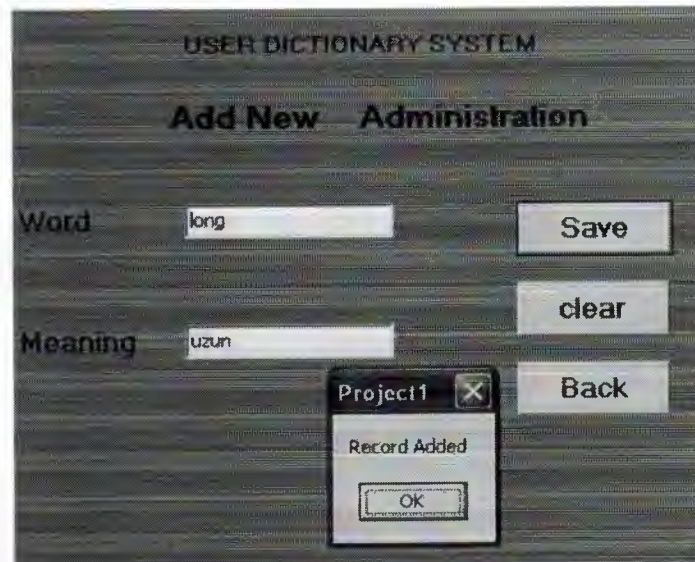
A screenshot of a web application titled "USER DICTIONARY SYSTEM" with the subtitle "Add New Administration". It contains two input fields: "Word" with the value "beautiful" and "Meaning" with the value "guzal". To the right of these fields are three buttons: "Save", "clear", and "Back". The interface has a dark, textured background.

Add New Figure (10)



## User Dictionary System

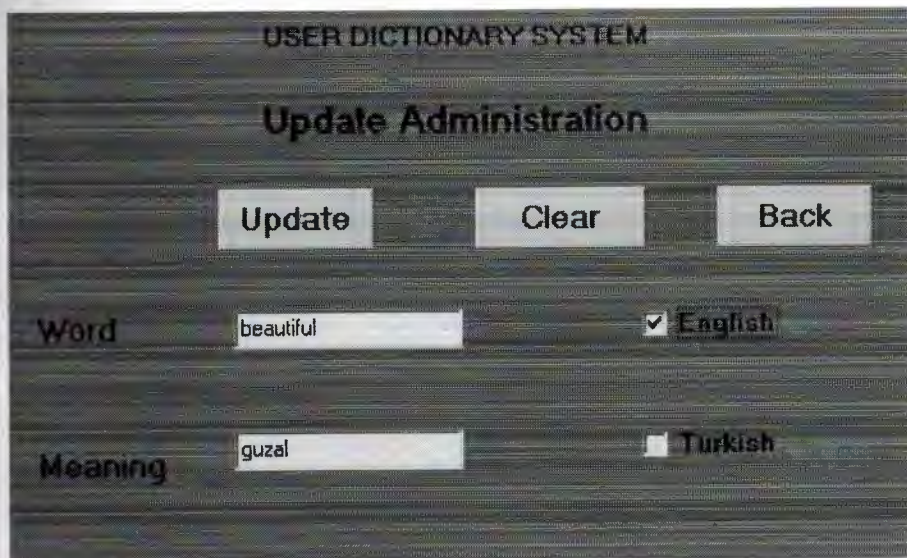
This page is name Add New is consists on 2-Edit box which first box about the word we write the word inside this box second box about the meaning we write the meaning inside this box. Also have 3-button first button is Save button if we press directly the word and the meaning will save inside the database and show message to tell us record Added see figure (10.1) but if the word already saved before it will show message to tell us already found.



Add New figure (10.1)

If we press on clear button will be clean inside Edit boxes and if we press on back button we will back to the Administrator menu figure (9).

If we press on the Update button from the Administrator menu page we will see a page this page is name Update Administration see figure (11).



Update Administration Figure (11)

This page is name Update Administration which the administrator can be correct the words if there's any wrong in the words. This page is consists on 2-Edit box one is



## User Dictionary System

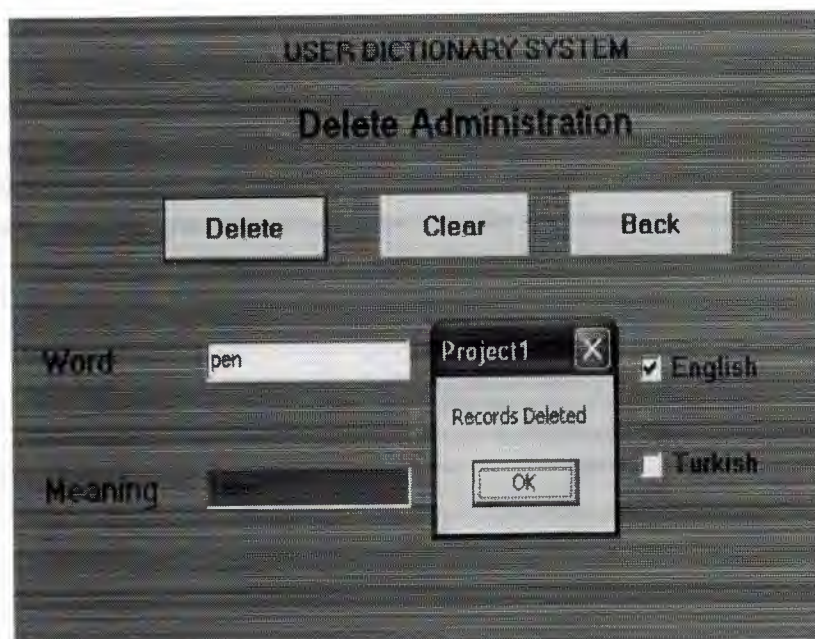
About the word we write the word inside it another box about the meaning. And have 2-Click box one is about English another one about the Turkish, here if the word was in English we press on English to find the meaning in Turkish and if the word was in Turkish we press on Turkish to find the meaning in English as we see in the figure and have 3-button first one is Update after to fixing the wrong word or the meaning we press the Update to save the fixing into database and show message to tell us recorded Update see figure (11.1).



Figure (11.1)

Second button is clear it work to clean inside Edit boxes. Third button is back. If press back we will back to the Administrator Menu see figure (9).

If we press on Delete Administration from Administrator Menu we will enter to the Delete Administration page see figure (12)



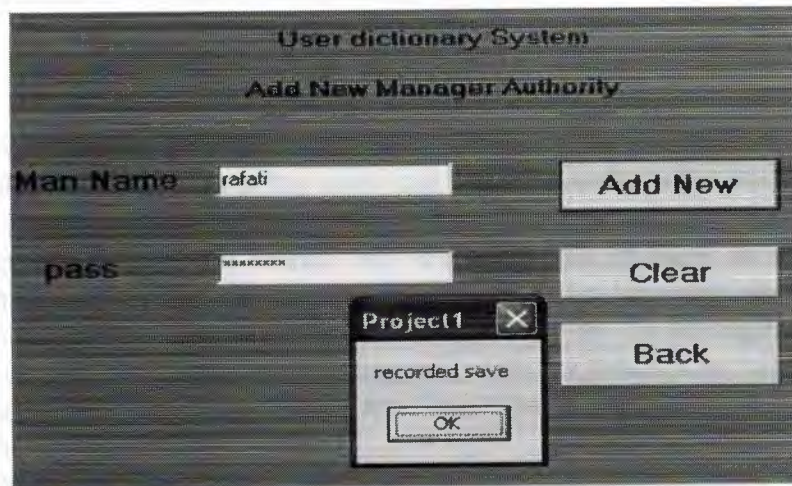
Delete Administration Figure (12)



This page is talking about the delete the word from database. Which the administrator can be deleting the words this page is consists on 2-Edit box one is about the word we write the word inside it another box about the meaning. And have 2-Chick box one is about English another one about the Turkish, here if the word was in English we press on English to find the meaning in Turkish and if the word was in Turkish we press on Turkish to find the meaning in English as we see in the figure and have 3-button first one is delete if we press on delete will be delete the word and the meaning together from database and show message to tell us recorded deleted see figure (12).

Second button is clear it work to clean inside Edit boxes. Third button is back if we press back we will back to the Administrator Menu see figure (9).

If we press on login new manager from Administrator Menu we will enter to the login new manager page see figure (13).



Login new manager Figure (13)

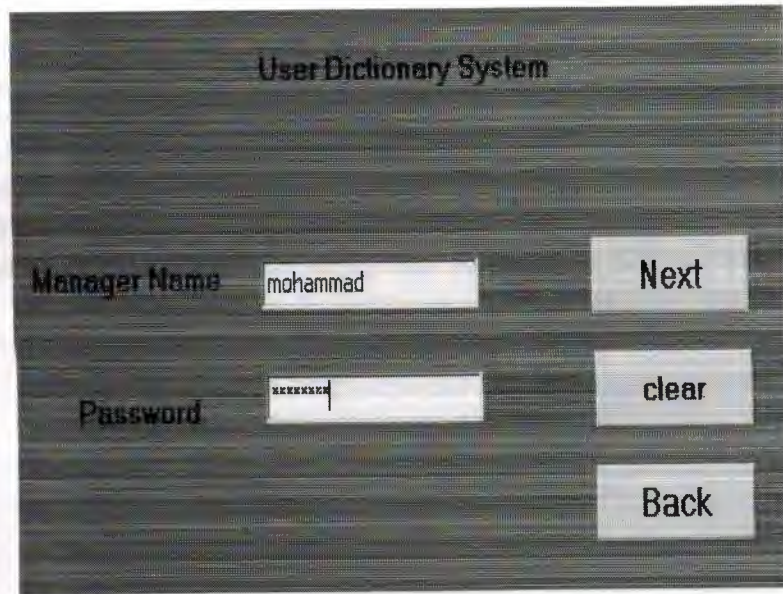
This page is talking about manager register. It consists of 2-Edit box which first Edit box is about the manager name and second Edit box is about the password so we write the name of manager inside the first box and the password in the second box, and it has 3-buttons first button is Add New button, if we press Add New button directly the name of the manager and the password will be saved into the database and a message will be shown to tell us recorded save see the figure (13). When we press on clear button it will clean inside the Edit box and when we press on back button we will go back to the Administrator Menu figure (9). Now we understand what is the Administrator work.

Let's go back to the figure (7) to Administration Menu and choose Manager button.

When we press on manager button it will show us Manager Register page look to the figure (14).



## User Dictionary System




The screenshot shows a dark-themed interface titled "User Dictionary System". Below the title, there are two input fields. The first is labeled "Manager Name" and contains the text "mohammad". The second is labeled "Password" and contains a series of asterisks "xxxxxxx". To the right of the "Manager Name" field is a button labeled "Next". To the right of the "Password" field are two buttons: "clear" and "Back".

Manager Register Figure (14)

This page is ask about the Manager name and password to let him to continue it have 2-Edit box which first one is about the name and second one about the password. And have 3-button one is back button which let us back to the last page Administration Menu figure (7) second one is clear button its work on clean inside Edit boxes third one is next button, when we write the Manager name and the password then we press on the next to continue and see what the Manager can do but if the Manager name and password is wrong will show us message to tell us you can not enter so must write the correct name and the password here not just the manager can use also the Administrator.

When we press on the next button it will show us next page and this is name Manager Menu and is talking about the manager work. See figure (15).



The screenshot shows a dark-themed interface titled "User Dictionary System". Below the title, there is a subtitle "Manager Edit Menu". Underneath, there are three buttons stacked vertically: "update", "Add New", and "Back".

Manager Edit menu Figure (15)

Through this page we can know what is the manager job, which is consists on 3-button first button is Update which the manager can be do update like Administrator so when we press on Update button we will enter to Update page see figure (11). Second button is Add New button which the manager can be do Add New like Administrator so when we press on Add New button we will enter to the Add New page see figure (10) so the manager have two work just he can do Add New and Update. About Back button it let us to back to Manager Register page back to figure (14).



## **Chapter 2. DATABASE. DELPHI LANGUAGE**

### **2.1 Introduction**

The database is an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides you with the software tools you need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.

### **2.2 What is a database?**

The database one or more, large structured sets of persistent data. Usually associated with software to update and query the data. A simple database might be a single file containing many records, each of which contains the same set of fields, where each field is a certain fixed width. A database is one component of a database management system.

#### **2.2.1 Database Management System (DBMS)**

The DBMS is in charge of Delphi, security, storage and a host of other functions for the database system. It does this through a selection of computer programs. This allows it to manage the large, structured sets of persistent data, which make up the database, and provide access to the data for multiple, concurrent users whilst maintaining the integrity of the data.

The DBMS provides security facilities in a variety of forms, both to prevent unauthorized access and to prevent authorized users from accessing data at the same time as each other or overwriting information, which they should not. As a first line of security to prevent unauthorized users from accessing the system, it uses user names and passwords to identify operators, programs and individual machines and sets of privileges assigned to them. These privileges can include the ability to read, write and update data in the database.

The DBMS controls who is able to read and write data through the use of locks. Locks are used for read and write to specific rows of data in relational systems, Or objects in object oriented ones, which are currently being used. By doing this only one user at a time can alter data.

DBMS's typically run on special hardware, for example servers which have been specially designed to only run databases and often only with specific database systems in mind. This allows the hardware designers to increase the number and speed of its network connections, use multiple processors and discs to speed up searched for information, increase the amount of memory and cache, as well as a host of other features not found on standard hardware. This also allows the programmers to take advantage of special features in the hardware to get the best possible performance from the system.

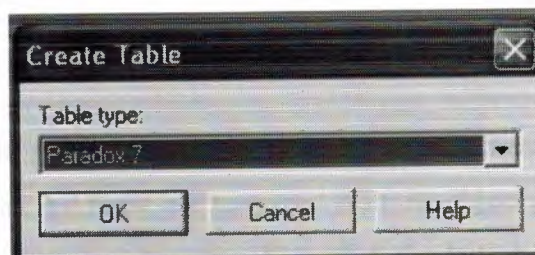
### 2.2.2 Database Design

Database Design is the process of taking the requirements for the database, i.e. what information is to be stored, who can enter it, how many people may be accessing it simultaneously etc. and designing a system which will achieve this.

This initial stage is carried out by either a database design specialist, or the Database Administrators and Software Analysts. At the end of it, a schema is produced which defines what data is stored, how it relates to other data, and who can enter, add and modify data. This schema is broken down into several sub-schemas which define individual tables and relationships between the tables and the data contained within.

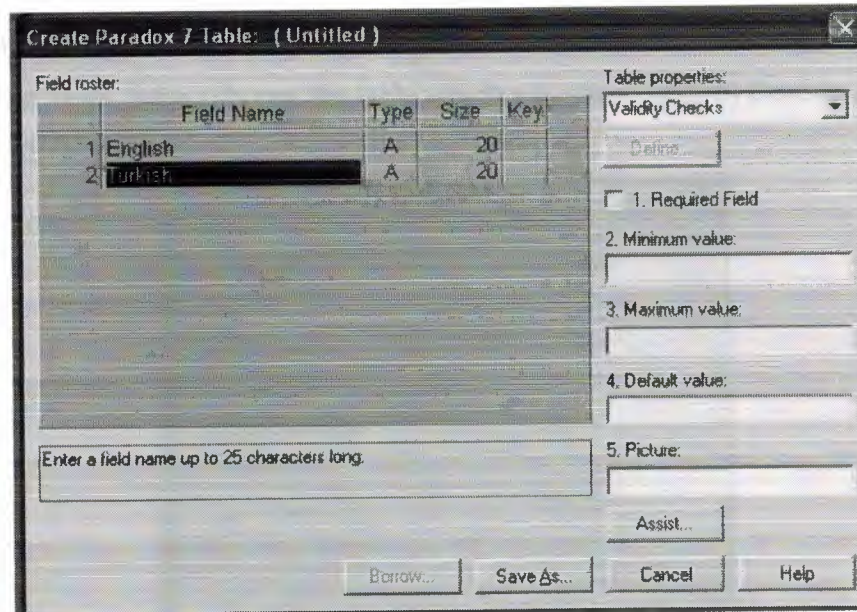
### 2.2.3 Creating the Database

To create a database your first need to open Delphi program and choose 'tools And choose Database Desktop' from the starting menu. And create new table, file, new, table new you will be see box is consist on the type table so choose Paradox 7 as shown figure above





Prompted for a name for the database and where you want it saved. Call the database 'translation. Db' and save it in the same directory as the dictionary system connecting to the database is going to be. Now, we can begin setting up the field in our database. Add the fields and set their data types so that it looks like this



Enter a field name up to 25 characters long as you see in the figure and have the data type of text. Use (A) for char. And change the default field size of 20 characters. Now the table has been created you need to enter some test data into the table. You can do this by open the table from where you saved. From here, you can enter some test data.

### 2.2.4 Database objects

There are six different types of database objects (tables, queries, forms, reports, macros and modules). However, I want to explain the Table and Form only, because it is used in my project.

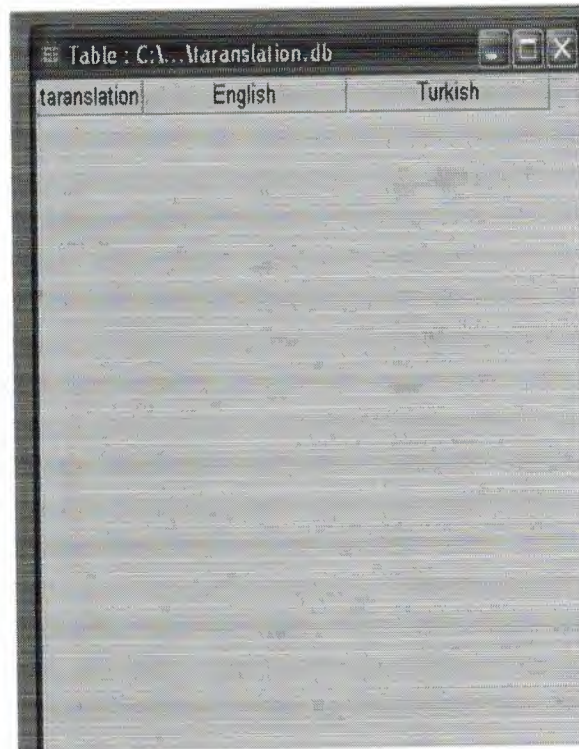
## 2.3 Tables

Database in Delphi program is a file made of various internal objects: tables, queries, forms, reports, etc. All these are managed from an object called the Database Window. The objects are kept in categories. You click the button that corresponds to its category.

A table is the central point of a database, because all data is stored in tables. For better organization, you will have various tables in your database, each for a different purpose.

### 2.3.1 Opening a Table

1. On the database window, in the Object Bar (Delphi program) click the tools, choose database desktop.
2. File, open, table (choose the table which you saved before) Double – click



### 2.3.2 Connecting To Database

When we come to write the code for our application, we will need to refer to our database. Must be performance Data Source component from Data Assess and Table component from BDE to make connection between the database and the forms.

### 2.4 Summary

If you are reading this page then I shall assume that you already know a little bit about database and how you can make table from database by using Delphi program.



The first thing we are establish a connection with the DBMS, and we use in this case is Delphi program. Before we can connect to a database, we need a database to connect too.

## Chapter 3. DELPHI LANGUAGE

### 3.1 History and Introduction

Delphi was originally envisioned as a next-generation visual development environment for Windows based on Borland's Object Pascal programming language and building upon the success of the popular Turbo Pascal IDE. At the intersection of RAD and client-server development trends, the original Delphi combined a leading-edge visual environment and visual component library (VCL) with superior database access features. For the last ten years, Delphi has continued to help developers take advantage of new technologies, from multi-tier Internet computing to Web services and Model-Driven Architecture (MDA), in a consistent visual environment and with the same tradition of superior connectivity as the original Delphi. As underlying platforms evolved, Borland has remained committed to helping developers maintain existing applications while adopting new technologies at their own pace.

Borland Developer Studio 2006 is the latest Integrated Development Environment product release from Borland Software Corporation. Borland Developer Studio includes Delphi 2006, C++Builder 2006, and C#Builder 2006.

What has always set Borland apart from other vendors has been its pragmatic approach to providing developer tools that are right for the challenges that commercial developers face today, while leading them into the emerging technologies of tomorrow with the confidence that their development investments remain relevant, adaptable and extendable in the future.

This paper provides an introductory look at what makes this release compelling from a Delphi perspective. While this document covers features that span Delphi, C++, and C#, it focuses on the Delphi capabilities and is not intended to cover specifically what's new for the C/C++ language specific capabilities.

Delphi 2006 is the tenth version of Delphi, Borland's flag ship Rapid Application Development (RAD) environment and continues the Delphi's RAD traditions and has extended capabilities in significant areas with considerable implications for software developers at every level.

While Microsoft Visual Basic developers struggled with the complexity of transitioning from the underlying VBRUN Win32 framework to the .NET framework, Delphi developers were able to seamlessly move back and forth between VCL and VCL.NET.

This focus on seamless integration of new technologies has applied equally well to new technologies introduced by Borland, as recent versions of Delphi have incorporated more sophisticated enterprise development capabilities, such as model-driven development and application lifecycle management, with the same focus on developer usability and ease of adoption.



### 3.2 What is Delphi?

Delphi is the premier development tool within Windows computing, and communications is one of the fastest growing segments of the Windows development platform. The Tomes of Delphi: Basic 32-Bit Communications Programming focuses on two essential communications technologies-Windows Sockets and the Telephony Application Programming Interface-and provides a guide to using them in Windows applications.

### 3.3 Starting a new application

Before beginning a new application, create a directory to hold the source files:

3.3.1 Create a directory called TextEditor in your C:\Program Files\Borland\Delphi6\Projects directory.

3.3.2 Open a new project.

Each application is represented by a project. When you start Delphi, it creates a blank project by default. If another project is already open, choose File|New| Application to create a new project.

When you open a new project, Delphi automatically creates the following files:

Project1.dpr: a source-code file associated with the project. This is called a project file.

Unit1.pas: a source-code file associated with the main project form. This is called a unit file.

Unit1.dfm: a resource files that stores information about the main project form. This is called a form file. Each form has its own unit (Unit1.pas) and form (Unit1.dfm) files. If you create a second form, a second unit (Unit2.pas) and form (Unit2.dfm) file are automatically created.

3.3.3 Choose File / Save all to save your files to disk. When the Save dialog box appears:

Navigate to your TextEditor folder.

Save Unit1 using the default name Unit1.pas.

Save the project using the name TextEditor.dpr. (The executable will be named the same as the project name with an .exe extension.)

Later, you can resave your work by choosing File|Save All.

When you save your project, Delphi creates additional files in your project directory. These files include TextEditor.dof, which is the Delphi Options file, TextEditor.cfg, which is the configuration file, and TextEditor.res, which is the Windows resource file. You don't need to worry about these files but don't delete them.

### 3.4 Setting property values

When you open a new project, Delphi displays the project's main form, named Form1 by default. You'll create the user interface and other parts of your application by placing components on this form.

Next to the form, you'll see the Object Inspector, which you can use to set property values for the form and the components you place on it. When you set properties, Delphi maintains your source code for you. The values you set in the Object Inspector are called design-time settings.

### 3.4.1 Form files

Forms are a very visible part of most Delphi projects. Normally, you design forms using Delphi's visual tools, and Delphi stores a description of the designed forms in form files. Form files (extension .dfm or .xfm) describe each component in Your form, including the values of all persistent properties. You do not specify the form file programmatically; you simply create the form by selecting components from the Component palette and customizing them to suit your needs by setting properties and events with the Object Inspector.

Each form in a Delphi project also has an associated unit. The unit contains the source code for any event handlers attached to the events of the form or the components it contains. A unit associated with a form is sometimes called a form unit. When you save a form unit or a project containing unsaved forms, Delphi prompts you to enter a name for each unit, which it uses as the name of the unit file, appending the extension .pas. The form file gets the same name, but with the extension .dfm for Windows projects and xfm for CLX projects. You can use any extension you want on your unit files, but Delphi expects the .dfm or xfm extension on the corresponding form file.

Warning: You can't define more than one form in a single unit. This is because each dfm or xfm file can only describe a single form (or data module).

Form files can be saved in either binary or text format. The Environment Options dialog lets you indicate which format you want to use for newly created forms.

To view the text version of .dfm or xfm files in the Code editor:

1. Select the form.
2. Right-click and choose View as Text.

To return to viewing the form graphically, follow the above steps and choose View as Form.

To change the format (text or binary) in which the form file is saved:

1. Select the form.
2. Right-click and check or uncheck Text DFM (VCL applications) and Text XFM (CLX applications).

### 3.4.2 Unit files

Delphi's Object Pascal language supports separately compiled modules of code called units. Using units promotes structured, reusable code across projects. The most common units in Delphi projects are form units, which contain the event handlers and other code for the forms used in Delphi projects. But units don't have to have forms associated with them. You can create and save a unit as a standalone file that any project can use. For example, you can write your own procedures, functions, DLLs, and components, and put their source code in a separate unit file that has no associated form. If you open and save a default new project, the project directory initially contains one unit source-code file (unit1.pas) and its associated form file (unit1.dfm or unit1.xfm).

Caution: Do not add more than one form into a single unit file. The associated form file (.dfm or .xfm) can only describe a single form.

When you compile or run the project or perform a syntax check on the project, Delphi's compiler produces an intermediate output file on disk from each unit's source code. By default the compiled version of each unit is stored in a separate binary-format file with the same name as the unit file, but with the extension .dcu (Delphi compiled unit). You should never need to open these binary files, and you do not need



To distribute them with the completed project. The compiled-unit format is specific to the Delphi compiler, and enables rapid compiling and linking.

Note: As an option, you can choose to have the compiler generate standard Intel object files (with the extension .obj) for greater compatibility with other compilers, but this greatly reduces the speed of compiling and linking your project. It should have no effect on the quality of the final generated code, however.

Unit files for forms most unit files you'll work with will probably be associated with forms. Whenever you create a new form, Delphi creates the corresponding unit file with the following code. The default unit identifier is incremented (Unit2, Unit3, and so on) for each new form.

```
unit Unit1;    { unit identifier }
interface
uses    { uses clause }
  SysUtils, Windows, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;
type
  TForm1 = class(TForm)    { class declaration }
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;    { instance declaration }
implementation
{$R *.DFM} { compiler directive to link form file }
end.
```

The type declaration (or class declaration) part introduces the form as a class. A class is simply an object, which you will recognize if you are familiar with previous versions of Borland Pascal products, or another object-oriented programming language. The default type declaration makes the new form a descendant of the generic form class, TForm. This means it contains all the behaviors and characteristics of a TForm object.

The variable declaration declares your form as an instance of the class TForm1.

The \$R compiler directive links the TForm's binary form file. This adds the form file(s) in your project to the compiled executable.

Caution: Do not remove the {\$R \*.dfm} or {\$R \*.xfm} directive from a form unit file. Doing so will result in code that will never work correctly.

Unit files for procedures and functions

You can write custom procedures or functions within a unit that's associated with a form. However, if you want to reuse the routines that you write, it's better to create a separate unit to contain those routines. By creating standalone units that have no associated forms, you can easily make your procedures and functions available to other projects.

To create a unit file not associated with a form:

Choose File|New|Unit.

It is not necessary to have a project open unless you want the new unit to be part of a project.

### 3.4.3 Naming unit and project source code files

As you open new units in a project, the product gives them default names: UNIT1.pas, UNIT2.pas, UNIT3.pas, and so on. You can change a unit's default name to a meaningful (and unique) name when you save the project.

To name a unit file:

1. Select the unit files.
2. Edit the Name property for the unit in the Object Inspector.

The product also supplies a default name for the project file (project1.dpr) which you can rename when you save the project.

All unit and project file names must be legal Object Pascal identifiers. When the compiler looks for a unit or project file, it first searches for a file with the full name of the unit or project identifier. If it does not find that file, it then searches for a version of the identifier name, truncated to eight characters. This is for backward compatibility and for compatibility with file servers that store only short file names. You should not manually truncate your file names.

## 3.5 Working with projects

When you're working with the product, you're working on a project. These topics describe the files that make up a project and then provide information on working with projects. The topics covered here include:

- 3.5.1) what is a project?
- 3.5.2) Viewing a project's contents
- 3.5.3) Saving projects and individual project files
- 3.5.4) Managing projects
- 3.5.5) Sharing objects
- 3.5.6) Creating a project group
- 3.5.7) Specifying a default project, new form, and main form
- 3.5.8) Running project

### 3.5.1 What is a project?

A project is a collection of files that make up an application or dynamic-link library. Some of these files are created at design time. Others are generated when you compile the project source code.

You can combine projects into a project group. Project groups let you organize and work on related projects, such as applications and DLLs that function together or parts of a multi-tiered application.

You can view the files that make up a project in the Project Manager (see Viewing a project's contents). Although you can edit many of these files directly, it is often easier and more reliable to use the visual tools in this product. You should, however, understand the files and file types that make up a project.

Single project files, which describe individual projects, have a .dpr extension. Project files contain directions for building an application or library. When you add and remove files using the Project Manager, the product updates the project file.

The product reads the uses clause of the project (.dpr) file to determine which units is part of a project. Only units that appear in the uses clause followed by the keyword in



And a file name is considered part of the current project. For example, here is the default project file for new applications:

```
Program Project1;  
Uses  
  Forms,  
  Unit1 in 'Unit1.pas' {Form1};  
{ $R *.res }  
Begin  
  Application.Initialize;  
  Application.CreateForm(TForm1, Form1);  
  Application.Run;  
End.
```

The project defined above uses two units: Forms and Unit1. Only Unit1, however, is actually part of the project.

The project group file contains make commands to build the projects in the project group, has a .bpg extension. Any time you add a project to the project group, a reference to that project is added to the .bpg file.

You can also add additional types of files to your project (using drag and drop or Project|Add to Project) and view them in the editor as text files. You can also add resource files, and they are compiled into .res files and linked when you compile the project.

### 3.5.2 Viewing a project's contents

The Project Manager displays the contents of your current project group and any project it contains. It allows you to navigate among various projects and the projects' constituent files. You can perform project management tasks using its toolbar and context (right-click) menus.

To display the Project Manager:

1. Open a project.
2. Choose View|Project Manager.

To view a unit, form or other file, double-click it. It is displayed in the Code editor.

The Project Manager gives you a high-level view of the projects contained in a project group, and of the form, unit files, and resource, object and library files contained in the project file.

Many commands are available by selecting an item in the Project Manager and right-clicking to display a Project Manager Context menu. You can use the Project Manager to open, add, save, and remove project files. You can also use the Project Manager to access the Project Options dialog box, which lets you configure your default project settings.

You can also add additional types of files to your project (using drag and drop or Project|Add to Project) and view them in the editor as text files. You can also add resource files, and they are compiled into resource files and linked when you compile the project.

The Project Manager is an invaluable tool if you share files among different projects because it lets you quickly find each file in the project

(See adding existing forms and Units to a project and Sharing objects). It is also useful when backing up all the files in your project.

Certain operations, such as commands available on context menus, operate on the active project. The active project is the one that is highlighted in bold in the Project Manager and is the project you are currently working on. The active project is also shown in the project selector which is the top left of the Project Manager. See selecting a project to work on.

To make a project the active project:

1. Display the Project Manager.
2. Select the project you want to make active and click Activate.

When you view a project item, such as a form or source code, the Project Manager automatically makes the project it belongs to the active project. Projects you select using the project selector automatically become the active project.

### 3.5.3 Saving projects and individual project files

At any time during project development, you can save an open project in its current state to the project directory. You can optionally save a copy of the project in a different directory under the same or a different name.

You can also add your project to the Object Repository so that you or others can use it as a template. For more information, see Adding items to the Object Repository.

You are not limited to saving a project as a whole; you can save individual constituent files of a project, including saving a copy of a file to a different directory or with a different file name.

### 3.5.4 Managing projects

You can use the Project Manager to manage multiple projects within a project group. The Project Manager displays information about the status and file content of the current project and provides a convenient way to open, add, save, and remove project files (you can do some of these tasks from the File menu as well). You can also create new projects to add to the current project group.

You can perform the following tasks from the Project Manager:

#### 3.5.4.1 Selecting a project to work on

The project selector is the list box at the top left of the Project Manager. You can select a project from the project group using the project selector. A drop-down list shows all projects in the current project group. The project you select becomes the active project. The active project is the one that is highlighted in bold in the Project Manager and is the project you are currently working on.

#### 3.5.4.2 Searching for files

You can locate files in large projects using an incremental search within the Project Manager. With the Project Manager displayed, start typing the name of a file and the Project Manager moves to the nearest match. Press the Spacebar to repeat the last search. If you share files among different projects, using the Project Manager is highly recommended because you can quickly and easily tell the location of each file in the project. This is especially helpful to know when creating backups that include all files the project uses. (See Backing up a project for more information.)

#### 3.5.4.3 Removing items in the Project Manager

You can remove selected items in the Project Manager in the following ways: Click the Remove button in the Project Manager.



Press the Delete key on the keyboard. Choose Project|Remove from Project from the main menu. The Project|Remove from Project dialog allows you to multiselect and remove multiple items from the project. If a project is selected, the whole project including all it contains is deleted from the current project group. If one file is selected, only that file is deleted. If you remove, for example, a library file that you added to a project, references to it are removed from your application. Realize that the files are not deleted from disk; they are only disassociated from the current project or group. The Project Manager verifies that you want to remove the project or item before doing so.

Note: If you copy a file from one project to another then remove the first project without saving the second one; the copied item is removed from both projects. The Project Manager prompts you save the project before removing the item. If you save the project when prompted to do so, the copied item is retained.

The section Removing forms and units from a project provides more details about removing forms from projects.

#### **3.5.4.4 Copying in the Project Manager**

You can add items into projects in the Project Manager from other Windows folders using drag and drop. You can also copy and paste items from one project to another within a project group. For information on copying an entire project, see Copying a project.

Note: When you copy items in the Project Manager, you are not making a physical copy on the disk. You are including the file or item as part of the active project. Drag and drop you can drag one or more selected items from any Windows folder and drop them into a project in the Project Manager:

1. Activate the project where you want the item to be added. The active project appears in bold in the Project Manager.
2. in any Windows folder (such as the Windows Explorer or My Computer), select one or more items to copy.
3. Drag the item or items onto the name of a project in the Project Manager.
4. Drop the items.

You are asked to verify that you want to add the item or items, and if you click yes, the items are added to the active project.

5. Save the project where you added the items.

#### **Copying files between projects**

You can copy any project item from one project to another:

1. Select the project item you want to copy.
2. Choose Edit|Copy (or type Ctrl+C).
3. Select the project where you want to place the copy (or move the cursor where you want to place the copy).
4. Paste the copy using Edit|Paste (or Ctrl+V).
5. Save the project where you placed the copy.

You can also use drag and drop to copy items from one project to another.

Note: If you copy a file from one project to another then remove the first project without saving the second one, the copied item is removed from both projects.

Project Manager prompts you save the project before removing the item. If you save the project when prompted to do so, the copied item is retained.

#### 3.5.4.5 Getting project path and unit information

The status bar at the bottom of the Project Manager window displays the full path name of the project file. You can display and hide the status bar by right-clicking and choosing Status Bar.

The project file path name can be a useful reference if you are bringing many forms and units that reside in locations other than the main project directory into the current project.

#### 3.5.4.6 Adding projects to the project group

To add a new project to this project group, select the project group, right-click, and choose New. You can also use the New button on the Project Manager toolbar.

Types of projects you can add are shown in the Object Repository. You can choose Project/ Add Existing Project to add a project that was already created to the current project group.

- Adding existing forms and units to a project

- Removing forms and units from a project

- Copying a project

#### 3.5.5 Sharing objects

The Object Repository (Tools|Repository) is a versatile tool that makes it possible to easily share (or copy) forms, dialog boxes, and data modules across projects and within a single project. It also provides project templates as starting points for new projects. By adding forms, dialog boxes, and data modules to the Object Repository, you make them available to other projects. In fact, you can add an entire project to the Object Repository as a template for future projects.

You'll also see wizards in the Object Repository. Wizards are small applications that lead you through a series of dialog boxes to create a form or project. The product provides a number of wizards, and you can also create and add your own customized wizards to simplify and standardize your work.

The repository stores settings in a text file named delphi32.dro (Delphi repository objects) in the \BIN directory that contains references to the items that appear in the Object Repository dialog box and the New Items dialog box. You can open this file in any text editor.

#### 3.5.6 Creating a project group

Create project groups to handle related projects at once. For example, you can create a project group that contains multiple executable files such as a .DLL and an .EXE. By organizing them into a group, you can compile them at the same time. To create a project group:

1. Choose View|Project Manager to display the Project Manager, if necessary. If no project is currently loaded, the Project Manager lists <No Project Group>. If a project is currently loaded, the Project Manager lists <ProjectGroup1>.
2. Select the project group, right-click, and choose either:
  - Add New Project to open the New Items dialog box to add a new project.
  - Add Existing Project to add an existing project to this project group.
3. When you have completed adding projects, select ProjectGroup1, right-click, and choose Save to rename the project group to a meaningful name.



To manage a project group:

Select the project group, right-click, and a context menu appears.

Whenever you open a project that is not currently part of a project group, the product displays the project as ProjectGroup1. You may choose to save the Project Group, thereby creating a project group for this project. However, it is not necessary.

### 3.5.7 Specifying a default project, new form, and main form

You can specify defaults for a new project, a new form, and a main form. You always have the option to override the defaults by choosing File|New|Other and selecting from the New Items dialog box. When you are developing a complex programming project in a team setting, or managing several development projects, you might soon develop the need for a version control system. A version control system can archive files, control access to project files, and track multiple versions of your projects.

Some versions of the product ship with Team Source, a tool specifically designed to manage the complexities of developing in a team environment. Not only does Team Source use a version control system to archive files, it provides a mechanism for reconciling changes made by individual developers with the changes to the overall project. All projects have as a target a single distributable executable file, either an .EXE or a .DLL file. You can view or test your application at various stages of development by compiling, building, or running it. You can also test the validity of your source code without attempting to compile the project.

If you have grouped several projects together, you can compile or build all projects in a single project group at once. Choose Project|Compile All Projects or Project|Build All Projects with the project group selected in the Project Manager.

### 3.5.8 Running a project

You can test run a project from within the product or you can run the compiled executable file from the Windows operating environment without having to run the product.

To compile and then run your application from within the product, either: Choose Run|Run. Or Choose the Run button on the toolbar.

These actions are identical to choosing the Project|Compile command, except that the product runs your application immediately if the compile operation succeeds.

Executing a project on Windows because the compiler always creates a fully compiled standalone executable file (.EXE), you can run your application from the Windows operating environment using the same techniques as you would for any other Windows application. If you have specified an icon for your project, it will appear beside the file name in the Windows Explorer, in a shortcut on the desktop, on the Windows Start menu, and on the taskbar when you minimize the application while it is running.

## 3.6 Using Object Pascal

Object Pascal is a high-level, compiled, strongly typed language that supports structured and object-oriented design. Its benefits include easy-to-read code, quick compilation, and the use of multiple unit files for modular programming.

Object Pascal has special features that support Borland's component framework and RAD environment. For the most part, descriptions and examples in this language reference assume that you are using Object Pascal to develop applications using Borland development tools such as Delphi or Kylix.

Most developers using Borland software development tools write and compile their code in the integrated development environment (IDE). Borland development tools handle many details of setting up projects and source files, such as maintenance of dependency information among units. The product also places constraints on program organization that are not, strictly speaking, part of the Object Pascal language specification. For example, Borland development tools enforce certain file- and program-naming conventions that you can avoid if you write your programs outside of the IDE and compile them from the command prompt.

These Help topics generally assume that you are working in the IDE and that you are building applications that use the Visual Component Library (VCL) and/or Borland Component Library for Cross Platform (CLX). Occasionally, however, Borland-specific rules are distinguished from rules that apply to all Object Pascal programming.

### 3.7 Pascal source files

The compiler expects to find Pascal source code in files of three kinds:

- 1) Unit source files (which end with the .pas extension)
- 2) Project files (which end with the .dpr extension)
- 3) Package source files (which end with the .dpk extension)

Unit source files contain most of the code in an application. Each application has a single project file and several unit files; the project file—which corresponds to the “main” program file in traditional Pascal—organizes the unit files into an application. Borland development tools automatically maintain a project file for each application. If you are compiling a program from the command line, you can put all your source code into unit (.pas) files. But if you use the IDE to build your application, you must have a project (.dpr) file.



## Chapter 4. Appendix

### 4.1 Personal Data Page

This page is talking about what is the project name who designed and who was supervisor on this project.

```

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label3: TLabel;
    Label4: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label1: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label2: TLabel;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
  uses Unit2, Unit6;
  ($R *.dfm)
  procedure TForm1.Button1Click(Sender: TObject);
  begin
    form6.show;
  end;
  procedure TForm1.Button2Click(Sender: TObject);
  begin
    close;
  end;

```

### Personal Data coding

#### 4.2 User Menu page

This page to let us to choose which you is user or administration.

```
type
  TForm6 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form6: TForm6;
implementation
uses Unit4, Unit2, Unit7, Unit15;
{$R *.dfm}
procedure TForm6.Button1Click(Sender: TObject);
begin
  form7.show;
  close;
end;
procedure TForm6.Button2Click(Sender: TObject);
begin
  form15.Show;
  close;
end;
procedure TForm6.Button3Click(Sender: TObject);
begin
  close;
end;

end.
```

#### User Menu coding



### 4.3 User Register

This page is special for user which the user must register his name before to use the system this step it allow the user to use this program again.

```

type
  TForm7 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    DataSource1: TDataSource;
    Button3: TButton;
    Label2: TLabel;
    Button4: TButton;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    str1:string;
  public
    { Public declarations }
  end;
var
  Form7: TForm7;
implementation
uses Unit3, Unit8, Unit6;
{$R *.dfm}
procedure TForm7.Button1Click(Sender: TObject);
begin
  str1:=edit1.text;
  table1.Locate('UserName',str1,[lopartialkey]);
  if str1 = table1.FieldValues['UserName'] then
    form3.show
  else
    Showmessage('user name invalid');
    edit1.Clear;
    if str1 = table1.FieldValues['UserName'] then
      Close;
    end;
  procedure TForm7.Button3Click(Sender: TObject);
  begin
    form6.show;
    close;
    edit1.Clear;
  end;
  procedure TForm7.Button4Click(Sender: TObject);
  var
    st1:string;

```

```

begin

str1:=edit1.text;
table1.Locate('UserName',str1,[lopartialkey]);
st1:=table1.FieldValues['UserName'];
if st1=str1 then
showmessage('already Found');
if st1<>str1 then
begin
table1.Insert;
table1['UserName']:=str1;
table1.refresh;
showmessage('recorded save');
end
end;
end.

```

#### User Register coding

#### 4.4 Main Menu Page

Which is consist which language we want to translation from English to Turkish or Turkish to English

```

type
TForm3 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Label1: TLabel;
  Button3: TButton;
  Label2: TLabel;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form3: TForm3;
implementation
uses Unit4, Unit5, Unit6;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
form4.show;
close;
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
form5.show;

```



```
close;
end;
procedure TForm3.Button3Click(Sender: TObject);
begin
form6.show;
close;
end;
```

#### Main Menu coding

### 4.5 Translation Page

#### 4.5.1 Translate from English to Turkish

This page is talking about translate from English to Turkish

```
type
TForm4 = class(TForm)
  DataSource1: TDataSource;
  Edit1: TEdit;
  Edit2: TEdit;
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Button4: TButton;
  Table1: TTable;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
private
  str1:shortstring;
  st2:string;
  stt:integer;
  stt1:integer;
  stt2:string;
  cop1:string;
  cop2:string;
  stcop:integer;
  first:string;
  second:string;
  total:string;
public
  { Public declarations }
end;
var
  Form4: TForm4;
implementation
uses Unit3, Unit14;{$R *.dfm}
```

```

procedure TForm4.Button1Click(Sender: TObject);
begin
  str1:=edit1.Text;
  stt:=length(str1);

  stt1:=pos(' ',str1);
  stt2:=IntToStr(stt1);
  stcop:=StrToInt(stt2);
  if stt1=0 then
  begin
    table1.locate('English',str1,[lopartialkey]);
    if str1 <> table1['English'] then
      showmessage('not found')
    else
      edit2.text:=table1.FieldValues['Turkish'];
    end
  else
  begin
    cop1:=copy(str1,0,stcop-1);
    cop2:=copy(str1,stcop+1,stt);
    table1.locate('English',cop1,[lopartialkey]);
    first:=table1.FieldValues['Turkish'];
    total:=first;
    table1.locate('English',cop2,[lopartialkey]);
    second:=table1.FieldValues['Turkish'];
    total:=concat(first,' ',second);
    edit2.Text:=total;
  end;
end;

procedure TForm4.Button2Click(Sender: TObject);
begin
  edit1.Clear;
  edit2.Clear;
end;

procedure TForm4.Button3Click(Sender: TObject);
begin
  form3.show;
  edit1.Clear;
  edit2.Clear;
  close;
end;

procedure TForm4.Button4Click(Sender: TObject);
begin
  form14.show;
end;
End

```

**Translate from English to Turkish coding**



#### 4.5.2 Translate from Turkish to English

This page is talking about translate from Turkish to English

```

type
  TForm5 = class(TForm)
    DataSource1: TDataSource;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
  public
    { Public declarations }
  end;
var
  Form5: TForm5;
implementation
uses Unit3;
{$R *.dfm}
procedure TForm5.Button1Click(Sender: TObject);
var
  str1:shortstring;
  st2:string;
  stt:integer;
  stt1:variant;
  stt2:string;
  cop1:string;
  cop2:string;
  stcop:integer;
  first:string;
  second:string;
  total:string;
begin
  str1:=edit1.Text;
  stt:=length(str1);
  stt1:=pos(' ',str1);
  stt2:=IntToStr(stt1);
  stcop:=StrToInt(stt2);
  if stt1=0 then
  begin

```

```

if table1.locate('Turkish',str1,[lopartialkey]) then
edit2.text:=table1.FieldValues['English'];
if str1<>table1['Turkish'] then
showmessage('not found')
end
else
begin
cop1:=copy(str1,0,stcop-1);
cop2:=copy(str1,stcop+1,stm);
table1.locate('Turkish',cop1,[lopartialkey]);
first:=table1.FieldValues['English'];
total:=first;
table1.locate('Turkish',cop2,[lopartialkey]);
second:=table1.FieldValues['English'];
total:=concat(first,' ',second);
edit2.Text:=total;
end;
end;
procedure TForm5.Button2Click(Sender: TObject);
begin
edit1.Clear;
edit2.Clear;
end;
procedure TForm5.Button3Click(Sender: TObject);
begin
form3.show;
edit1.Clear;
edit2.Clear;
close;
end;
end.

```

#### Translate from Turkish to English coding

#### 4.6 Advanced search

```

type
TForm14 = class(TForm)
Edit1: TEdit;
Edit2: TEdit;
Button1: TButton;
DataSource1: TDataSource;
Table1: TTable;
Button2: TButton;
ComboBox1: TComboBox;
Button3: TButton;
Label1: TLabel;
Button4: TButton;
Label2: TLabel;

```



```

Label3: TLabel;
procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
private
  str1:string;
  cop1:string;
  st1:string;
public
end;
var
  Form14: TForm14;
implementation
uses Unit4;
{$R *.dfm}
procedure TForm14.Button1Click(Sender: TObject);
begin
  str1:=Edit1.text;
  table1.Locate('English',str1,[lopartialkey]);
  st1:=table1.fieldvalues['English'];
  if str1 <> st1 then
    cop1:=copy(str1,1,2);
    table1.Locate('English',cop1,[lopartialkey]);
    combobox1.Text:=table1.FieldValues['English'];
  end;
procedure TForm14.Button2Click(Sender: TObject);
begin
  edit1.Clear;
  edit2.Clear;
  combobox1.Clear;
end;
procedure TForm14.Button3Click(Sender: TObject);
begin
  str1:=Edit1.text;
  table1.Locate('English',str1,[lopartialkey]);
  edit2.Text:=table1.fieldvalues['Turkish'];
end;
procedure TForm14.Button4Click(Sender: TObject);
begin
  form4.show;
  edit1.Clear;
  edit2.Clear;
  combobox1.Clear;

```

```
close;
end;
procedure TForm14.FormCreate(Sender: TObject);
begin
end;
end.
```

#### 4.7 Administration Menu Page

This page just for the Administrator and the manager

```
type
  TForm15 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form15: TForm15;
implementation
  uses Unit2, Unit16, Unit6;
  {$R *.dfm}
  procedure TForm15.Button1Click(Sender: TObject);
  begin
    form2.show;
    close;
  end;
  procedure TForm15.Button2Click(Sender: TObject);
  begin
    form16.show;
    close;
  end;
  procedure TForm15.Button3Click(Sender: TObject);
  begin
    form6.show;
    close;
  end;
end.
```

#### Administration Menu coding



#### 4.8 Administrator Register page

This page is ask about the administrator name and password to let him to continue it

```

type
  TForm2 = class(TForm)
    DataSource1: TDataSource;
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Label3: TLabel;
    Button3: TButton;
    Button2: TButton;
    Label4: TLabel;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);

  private
    str1:string;
    str2:string;
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
implementation
uses Unit3, Unit9, Unit13, Unit6;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
  str1:=edit1.text;
  str2:=edit2.text;
  table1.Locate('UserName',str1,[lopartialkey]);
  table1.Locate('Pass',str2,[lopartialkey]);
  if ((str1=table1.FieldValues['UserName'])and (str2=table1.FieldValues['Pass'])) then
    form9.show
  else
    Showmessage('user name and password uncorrect') ;
  close;
  edit1.Clear;
  edit2.Clear;
end;
procedure TForm2.Button3Click(Sender: TObject);
begin

```

```
edit1.Clear;
edit2.Clear;
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
form6.show;
close;
edit1.Clear;
edit2.Clear;
end;
end.
```

#### Administrator Register coding

#### 4.9 Administrator Menu page

This page is talking about what the administrator can do.

```
type
TForm9 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  Button4: TButton;
  Button5: TButton;
  Label1: TLabel;
  Button6: TButton;
  procedure Button4Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button5Click(Sender: TObject);
  procedure Button6Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form9: TForm9;
implementation
uses Unit3, Unit10, Unit11, Unit12, Unit6, Unit13, Unit18;
{$R *.dfm}
procedure TForm9.Button4Click(Sender: TObject);
begin
form13.show;
close;
end;
procedure TForm9.Button1Click(Sender: TObject);
begin
form10.show;
close;
```



```

end;
procedure TForm9.Button3Click(Sender: TObject);
begin
form11.show;
close;
end;
procedure TForm9.Button2Click(Sender: TObject);
begin
form12.show;
end;
procedure TForm9.Button5Click(Sender: TObject);
begin
form6.show;
close;
end;
procedure TForm9.Button6Click(Sender: TObject);
begin
form18.show;
end

```

#### Administrator Menu coding

#### 4.9.1 Translation Page

This page is talking about translate from English to Turkish and Turkish to English. And this page just the Administrator can used

```

type
TForm13 = class(TForm)
  Button2: TButton;
  Button3: TButton;
  Edit1: TEdit;
  Edit2: TEdit;
  Label1: TLabel;
  Label2: TLabel;
  CheckBox1: TCheckBox;
  CheckBox2: TCheckBox;
  DataSource1: TDataSource;
  Label3: TLabel;
  Label4: TLabel;
  Table1: TTable;
  procedure CheckBox2Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure CheckBox1Click(Sender: TObject);
private
  str1:string;
  str2:string;
  st1:string;
public
  { Public declarations }

```

```

end;
var
  Form13: TForm13;
implementation
uses Unit2, Unit9;
{$R *.dfm}
procedure TForm13.CheckBox2Click(Sender: TObject);
begin
  str2:=Edit1.Text;
  if CheckBox2.Checked then
    table1.Locate('Turkish',str2,[lopartialkey]);
    st1:=table1.FieldValues ['Turkish'];
    if st1 <> str2 then
      showmessage('Not Found')
    else
      edit2.Text:=table1.FieldValues['English'];
  end;
procedure TForm13.Button2Click(Sender: TObject);
begin
  edit1.Clear;
  edit2.clear;
end;
procedure TForm13.Button3Click(Sender: TObject);
begin
  form9.show;
  edit1.Clear;
  edit2.Clear;
  close;
end;
procedure TForm13.CheckBox1Click(Sender: TObject);
begin
  str1:=Edit1.text;
  str2:=Edit1.text;
  if CheckBox1.Checked then
    table1.Locate('English',str1,[lopartialkey]);
    st1:=table1.FieldValues['English'];
    if st1 <> str1 then
      showmessage('Not Found')
    else
      edit2.Text:=table1.FieldValues['Turkish'];
  end;
end.

```

#### Translation coding



## 4.9.2 Add New Page

This page is talking about the add new words inside the database table

```

type
  TForm10 = class(TForm)
    Label1: TLabel;

    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Label2: TLabel;
    DataSource1: TDataSource;
    Button2: TButton;
    Label3: TLabel;
    Button3: TButton;
    Label4: TLabel;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    str1:string;

    str2:string;
    st1:string;
    st2:string;
  public
    { Public declarations }
  end;
var
  Form10: TForm10;
implementation
uses Unit9;
{$R *.dfm}
procedure TForm10.Button1Click(Sender: TObject);
begin
  str1:=edit1.text;
  str2:=edit2.text;
  table1.Locate('English',str1,[lopartialkey]);
  st1:=table1.FieldValues['English'];
  st2:=table1.FieldValues['Turkish'];
  if ((st1=str1) and (st2=str2)) then
    showmessage('already Found');
  if ((st1<>str1) and (st2<>str2)) then
    begin
      table1.Insert;
      table1['English']:=str1;
    end;
  end;

```

```

table1['Turkish']:=str2;
table1.Refresh;
showmessage('Record Added');
end
end;

procedure TForm10.Button2Click(Sender: TObject);
begin
form9.show;
edit1.Clear;
edit2.clear;
close;
end;
procedure TForm10.Button3Click(Sender: TObject);
begin
edit1.Clear;
edit2.clear;
end;
end.

```

#### Add New coding

#### 4.9.3 Update Page

This page is name Update Administration which the administrator can be correct the words if there's any wrong in the words.

```

type
TForm12 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  Edit1: TEdit;
  Edit2: TEdit;
  Label1: TLabel;
  Label2: TLabel;
  DataSource1: TDataSource;
  CheckBox1: TCheckBox;
  CheckBox2: TCheckBox;
  Label3: TLabel;
  Label4: TLabel;
  Table1: TTable;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure CheckBox1Click(Sender: TObject);
  procedure CheckBox2Click(Sender: TObject);
private
  str1:string;
  str2:string;
  st1:string;

```



```

public
  { Public declarations }
end;
var
  Form12: TForm12;
implementation
uses Unit9;
{$R *.dfm}
procedure TForm12.Button1Click(Sender: TObject);
begin
  str1:=edit1.Text;
  str2:=edit2.Text;
  table1.Edit;
  table1.FieldValues['English']:=str1;
  table1.FieldValues['Turkish']:=str2;
  table1.Refresh;
  showmessage('Records Updated');
end;
procedure TForm12.Button2Click(Sender: TObject);
begin
  edit1.Clear;
  edit2.Clear;
end;
procedure TForm12.Button3Click(Sender: TObject);
begin
  form9.show;
  edit1.Clear;
  edit2.Clear;
  close;
end;
procedure TForm12.CheckBox1Click(Sender: TObject);
begin
  str1:=Edit1.text;
  str2:=Edit1.text;
  if CheckBox1.Checked then
    table1.Locate('English',str1,[lopartialkey]);
  st1:=table1.FieldValues['English'];
  if st1 <> str1 then
    showmessage('Not Found')
  else
    edit2.Text:=table1.FieldValues['Turkish'];
  end;
end;
procedure TForm12.CheckBox2Click(Sender: TObject);
begin
  str2:=Edit1.Text;
  if CheckBox2.Checked then
    table1.Locate('Turkish',str2,[lopartialkey]);
  st1:=table1.FieldValues ['Turkish'];
  if st1 <> str2 then

```

```

showmessage('Not Found')
else
edit2.Text:=table1.FieldValues['English'];
end;
end.

```

### Update coding

#### 4.9.4 Delete Page

This page is talking about the delete the word from database. Which the administrator can be deleting the words

```

type
  TForm11 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    DataSource1: TDataSource;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    Button2: TButton;
    Button3: TButton;
    Label4: TLabel;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    str1:string;
    st1:string;
    str2:string;
  public
  end;
var
  Form11: TForm11;
implementation
uses Unit10, Unit9;
{$R *.dfm}
procedure TForm11.Button1Click(Sender: TObject);
begin
  str1:=Edit1.text;
  table1.Locate('English',str1,[lopartialkey]);
  table1.FieldValues['English'];
  table1.FieldValues['Turkish'];
  table1.Delete;
  table1.Refresh;

```



```

showmessage('Records Deleted');
if CheckBox2.Checked then
table1.Locate('Turkish',str2,[lopartialkey]);
edit2.Text:=table1.FieldValues['English'];
end;
procedure TForm11.CheckBox1Click(Sender: TObject);
begin
str1:=Edit1.text;
str2:=Edit1.text;
if CheckBox1.Checked then
table1.Locate('English',str1,[lopartialkey]);
st1:=table1.FieldValues['English'];
if st1 <> str1 then
showmessage('Not Found')
else
edit2.Text:=table1.FieldValues['Turkish'];
end;

procedure TForm11.CheckBox2Click(Sender: TObject);
begin
str2:=Edit1.Text;
if CheckBox2.Checked then
table1.Locate('Turkish',str2,[lopartialkey]);
st1:=table1.FieldValues ['Turkish'];
if st1 <> str2 then
showmessage('Not Found')
else
edit2.Text:=table1.FieldValues['English'];
end;
procedure TForm11.Button2Click(Sender: TObject);
begin
form9.show;
edit1.Clear;
edit2.Clear;
close;
end;
procedure TForm11.Button3Click(Sender: TObject);
begin
edit1.Clear;
edit2.Clear;
end;
end.

```

**Delete coding**

## 4.9.5 Login New Manager

This page is talking about manager register.

```

type
  TForm18 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Label1: TLabel;
    DataSource1: TDataSource;
    Button2: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Button3: TButton;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    str1:string;
    str2:string;
  public
    { Public declarations }
  end;
var
  Form18: TForm18;
implementation
uses Unit9;
{$R *.dfm}
procedure TForm18.Button1Click(Sender: TObject);
begin
  str1:=edit1.text;
  str2:=edit2.text;
  table1.Insert;
  table1['ManName']:=str1;
  table1['Pass']:=str2;
  table1.refresh;
  showmessage('recorded save');
end;
procedure TForm18.Button2Click(Sender: TObject);
begin
  form9.show;
  close;
  edit1.Clear;
  edit2.Clear;
end;

procedure TForm18.Button3Click(Sender: TObject);
begin

```



```
edit1.Clear;
edit2.Clear;
end;
end.
```

### Login New Manager coding

#### 4.10 Manager Register Page

This page is ask about the Manager Name and password to let him to continue it

```
type
  TForm16 = class(TForm)
    Button1: TButton;
    Label2: TLabel;
    Label3: TLabel;
    DataSource1: TDataSource;
    Label4: TLabel;
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    str1:string;
    str2:string;
  public
    { Public declarations }
  end;
var
  Form16: TForm16;
implementation
  uses Unit17, Unit6;
  {$R *.dfm}
  procedure TForm16.Button1Click(Sender: TObject);
  begin
    str1:=edit1.Text;
    str2:=edit2.Text;
    table1.Locate('ManName',str1,[lopartialkey]);
    table1.Locate('Pass',str2,[lopartialkey]);
    if ((str1=table1.FieldValues['ManName'])and (str2=table1.FieldValues['Pass'])) then
      form17.show
    else
      Showmessage('user name and password uncorrect') ;
  end;
  close;
  edit1.Clear;
  edit2.Clear;
  end;
  procedure TForm16.Button2Click(Sender: TObject);
  begin
```



```

form6.show;
close;
edit1.clear;
edit2.Clear;
end;
end.

```

### Manager Register coding

#### 4.11 Manager Edit Menu Page

This page is talking about the manager work through this page we can know what is the manager job.

```

type
  TForm17 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Button4: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form17: TForm17;
implementation
  uses Unit19, Unit20, Unit15, Unit21;
  {$R *.dfm}
  procedure TForm17.Button1Click(Sender: TObject);
  begin
    form19.show;
  end;
  procedure TForm17.Button2Click(Sender: TObject);
  begin
    form20.show;
  end;
  procedure TForm17.Button4Click(Sender: TObject);
  begin
    form15.show;
  end;
  close;
  end;
end.

```

### Manager Edit Menu coding



## 4.11.1 Update Page

This page is name Update Administration which the Manager can be correct the words if there's any wrong in the words.

```

type
  TForm19 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    Label2: TLabel;
    Label3: TLabel;
    DataSource1: TDataSource;
    Label4: TLabel;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    str1:string;
    str2:string;
    st1:string;
  public
    { Public declarations }
  end;
var
  Form19: TForm19;
implementation
uses
  Unit17;
{$R *.dfm}
procedure TForm19.Button1Click(Sender: TObject);
begin
  str1:=edit1.Text;
  str2:=edit2.Text;
  table1.Edit;
  table1.FieldValues['English']:=str1;
  table1.FieldValues['Turkish']:=str2;
  table1.Refresh;
  showmessage('Records Updated');
end;
procedure TForm19.CheckBox1Click(Sender: TObject);
begin
  str1:=Edit1.text;
  str2:=Edit1.text;
  if CheckBox1.Checked then

```

```

table1.Locate('English',str1,[lopartialkey]);
st1:=table1.FieldValues['English'];
if st1 <> str1 then
showmessage('Not Found')
else
edit2.Text:=table1.FieldValues['Turkish'];
end;
procedure TForm19.CheckBox2Click(Sender: TObject);
begin
    str2:=Edit1.Text;
    if CheckBox2.Checked then
        table1.Locate('Turkish',str2,[lopartialkey]);
        st1:=table1.FieldValues ['Turkish'];
        if st1 <> str2 then
            showmessage('Not Found')
        else
            edit2.Text:=table1.FieldValues['English'];
        end;
    procedure TForm19.Button2Click(Sender: TObject);
    begin
        edit1.Clear;
        edit2.Clear;
    end;
    procedure TForm19.Button3Click(Sender: TObject);
    begin
        form17.show;
        close;
        edit1.Clear;
        edit2.Clear;
    end;
end.

```

#### Update coding

#### 4.11.2 Add New Page

This page is talking about the add new words inside the database table.

```

type
    TForm20 = class(TForm)
        Edit1: TEdit;
        Edit2: TEdit;
        Button1: TButton;
        Button2: TButton;
        Button3: TButton;
        Label1: TLabel;
        Label2: TLabel;
        DataSource1: TDataSource;
        Table1: TTable;
    end;

```



```

procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
private
  str1:string;
  str2:string;
  st1:string;
  st2:string;
public
  { Public declarations }
end;
var
  Form20: TForm20;
implementation
uses Unit17;
{$R *.dfm}
procedure TForm20.Button1Click(Sender: TObject);
begin
  str1:=edit1.text;
  str2:=edit2.text;
  table1.Locate('English',str1,[lopartialkey]);
  st1:=table1.FieldValues['English'];
  st2:=table1.FieldValues['Turkish'];
  if ((st1=str1) and (st2=str2)) then
    showmessage('already Found');
  if ((st1<>str1) and (st2<>str2)) then
    begin
      table1.Insert;
      table1['English']:=str1;
      table1['Turkish']:=str2;
      table1.Refresh;
      showmessage('Record Added');
    end
  end;
procedure TForm20.Button2Click(Sender: TObject);
begin
  edit1.clear;
  edit2.clear;
end;
procedure TForm20.Button3Click(Sender: TObject);
begin
  form17.show;
  close;
  edit1.clear;
  edit2.clear;
end;
end.

```

**Add New coding**

## CONCLUSION

I came across to the point at the end of my project that using clever programming techniques and simple design. There are some techniques for the dictionary designing. From the Database, I used database from Delphi language the database connection is used to create an open connection to a data source. Through this connection, we can make search inside the database and add new inside it.

We see also how we can connect each form together and with database.



## References

**<http://www.borland.com/delphi>**

**<http://community.borland.com/Delphi>**

**<http://bdn.borland.com/eco>**

**Delphi Star team Integration White Paper:**

**[http://www.borland.com/resources/en/pdf/white\\_papers/del\\_collaborative\\_teams\\_wp.pdf](http://www.borland.com/resources/en/pdf/white_papers/del_collaborative_teams_wp.pdf)**

**Together BDNtv demo:**

**<http://bdn.borland.com/article/0,1410,33373,00.html>**

**ECO BDNtv demo:**

**<http://bdn.borland.com/article/0,1410,33375,00.html>**