



KOÇ UNIVERSITY
College of Engineering

DEPARTMENT OF COMPUTER ENGINEERING

COMP 391 SUMMER PRACTICE II

Mertcan AKARDERE

Internship Company and Department:

E&M Bilgisayar ve Mühendislik

29.08.2016 / 23.09.2016

Supervisor (Name, Department, Phone, Fax, E-mail, etc.):

Aykut Altundağ

Table of Contents

1. Introduction.....	2
2. Company Description	4
3. Database Programming with SQL and Visual Studio	6
3.1. Problem Statement	6
3.2. Tools and Techniques Used.....	8
3.3. Detailed Explanation	10
3.4. Results	19
4. Conclusions.....	20
5. Appendix.....	22

1. Introduction

I was meant to help in a database project that was taken by E&M Bilgisayar ve Mühendislik so I started my internship by practicing my SQL knowledge and familiarize myself with the interface of Microsoft SQL Server Management Studio as I was going to use it in my project. I had taken a course on data management systems (COMP 306) which gave a foundation for SQL as well as my last internship on SAP systems at MBIS Danışmanlık. But I had to learn new stuff to make the database safer to use for the end-user.

Then I have taken few tasks to do queries that were required by the project and save them as stored procedures to be later implemented with an interface. Some of these queries were challenging as they required the combining of data, like showing all the telephone numbers a customer might have in a single column.

Lastly I had to prepare a form with lots of features using the Microsoft Visual Studio and C# for customers to interact with their database. This form had lots of functionality. It would take the customer data in the database and allowed it to be searched. Also new customers could be added or deleted or already existing customers' data could be updated. All of these processes had safety checks so that the data in the database would remain reliable and minimize the chance of entering a wrong data from the end-user.

I am interested in working with databases as I worked on SAP in my last internship and found it very interesting and useful. I wanted to pursue this goal further but at the same time having a different experience. Working with SAP allowed me into focusing implementing functionalities by request from other consultants so I was not directly focused on the end-user experience and I had SAP's safety procedures to ensure the safety of data from the user errors. These allowed me to work more carefree in my last internship but I wanted to challenge myself by removing those safety nets. I wanted to work here too as I had a chance to work closely with customers' demands and experience end product focused mentality for working. As a consequence of working with a different mentality, I now have more complete knowledge on implementing databases.

2. Company Description

E&M Bilgisayar ve Mühendislik Ltd. provides technical assistance to companies by implementing business solution programs, creating databases and programs for users to interact with them, installing security systems and assisting to hardware and software issues.

The company can provide a customized database to fit to their customers' needs and provide features and interface as desired by the customers. The company can also implement ERP programs like Link and Logo. But the company can also provide databases build from the ground up in SQL. The company mainly works with middle size companies therefore it needs to provide support to users having difficulty using the database systems. So the databases implemented in this company are focused on being as end-user friendly as possible to minimize user errors and maximize efficiency.

My task here was creating such an end-user friendly database and a form to interact with the database.

The company was setup by Mithat Altaç and Erkan Topçuol in 1991 and company name was their initials. But Mr. Topçuol has left the company after 3 years to move on to food sector. The company was always a provider of Link Bilgisayar. But besides that they provided assistance on many different things including UNIX, XENIX, EIX, Dos, Novel and Microsoft.

Company is tiny compared to tasks it's taken. It's formed of few programmers that implement the database and interface for the customers and few people to provide help to deal with hardware and software issues.

3. Database Programming with SQL and Visual Studio

3.1. Problem Statement

E&M Bilgisayar ve Mühendislik had taken a job with a plastics company to program a database and interface to provide them an easy way to organize and allow them to be quicker in their workflow as the software would also have features like standardizing and digitalizing their workflow, easily getting organized by assigning workers to each machine, generating sample agreements that are commonly used with their customer firms. The plastics company was also negotiating with E&M Bilgisayar about safety systems like back-up database and cameras, but that is besides the project I was working on.

I was tasked with helping with the requests of the customers and I had to prepare queries and form interfaces. Saving the information customers of the company was meant to be the part of the program so I did a part of the database where I could save customer information in a database, and allow it to be deleted or updated. I added searching functionality through these customers and listing functionality. I also designed an end-user friendly interface for these functions in Visual Studio and included error messages in case the users gave wrong input for the fields.

The customers did not have software like this previously and they did everything with Microsoft Excel and on paper. This resulted in a difficult task as even the customers did not know exactly what they wanted. So we had to work with them back and forth, trying different solutions until they would be satisfied with the results. We talked with a different person from each step of their workflow to make a program that can be used in every department of the company. Therefore we needed to develop quite a large program with many tabs for every step of the workflow so everyone could easily do their tasks without getting overwhelmed by the other parts of the program.

3.2. Tools and Techniques Used

We used SQL with Microsoft SQL Server Management Studio for the database. We used SQL because it is fast and reliable. Management Studio natively supports SQL and provides ease for almost all features of SQL, so we utilized that. This server was later going to be hosted in the company to be accessed.

We used Microsoft Visual Studio to design the interface for the software the customers were going to use as it provides an easy way to design and allow programming at the same time. This way our program had windows, boxes, buttons similar to Windows style, which is familiar to many end-users. We could also design warnings when users tried to put faulty information or not enough data, so users would be informed about the mistake. The interface of the program was similar to regular Windows style therefore it was easier for employees of the company as most of them were not experts on computers. The familiar look, simple design and warning messages were there to guide the employees into operating correctly and minimizing the errors. We used C# to program in Visual Studio which is a quite an efficient yet a simple language. C# was similar to Java and C++, so I was able to quickly adapt to the language thanks to the information I had gathered from the advanced programming classes I had taken.

We were making relational databases with SQL and I made use of the knowledge I gathered in the COMP 306 course. I used composite keys, ID method, and set whether a column can be null or have to unique. I used normalization techniques to create an efficient database in the 3NF format.

3.3. Detailed Explanation

First we had to develop a database to store the required data for the processes demanded by the customer's request. We created tables to store information contacts of the company, plastics orders given to company with the details of where the order would be produced, shaped, shipped, and other stages of plastic object production, agreements and plastic ratios set for each order, types of plastics and objects produced, and factory information like where and when the production would take place and assigned workers. These tables were filled a little from the information given by the company to set an example and prepare a foundation for the database when it goes in effect at the company.

Also some data had to be put in to save time, like the templates for common agreements that the company gives and receives. This also provides efficiency for the sales part as the customers of the company can give orders from the company's templates.

My task was saving information about customers, so I created tables for customers and cities of Turkey (Figure 1). City tables were there to show city names from their official plate codes. Customer tables saved names, location, and all available telephone numbers (Figure 2).

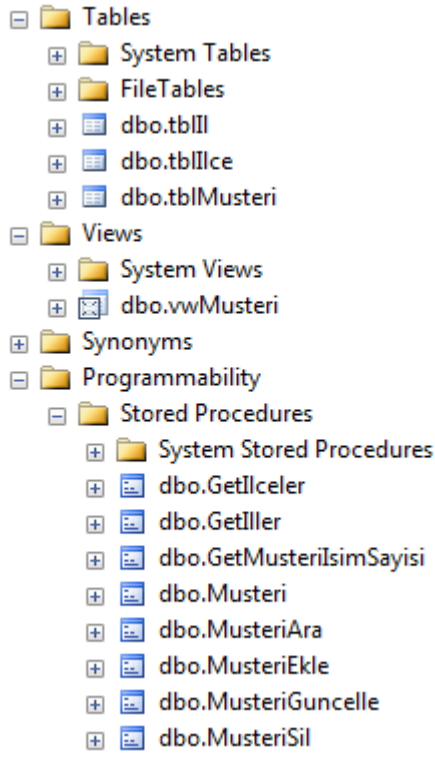


Figure 1) Tables and Procedures done by me for the project. Tables store the data for customers and cities in Turkey. Procedures perform standard queries that will later be done with a interface in Visual Studio

	Column Name	Data Type	Allow Nulls
▶	id	int	<input type="checkbox"/>
	ad	nvarchar(30)	<input type="checkbox"/>
	aciklama	nvarchar(MAX)	<input checked="" type="checkbox"/>
	il_id	int	<input checked="" type="checkbox"/>
	ilce_id	int	<input checked="" type="checkbox"/>
	telefon	nvarchar(20)	<input checked="" type="checkbox"/>
	yetki	nvarchar(30)	<input checked="" type="checkbox"/>

Figure 2) Table for customer. This shows data types for columns. I used nvarchar as it saves space for the empty fields.

We also created table to store transaction and action history for safety precautions for the company. This will help if someone mistakenly deletes something from the database or tried maliciously damaging it.

Secondly, we had to write queries and stored procedures that would satisfy the needs and demands of the customers in SQL (Figure 3-4). Besides saving the data in a certain way, the company wanted to be able to see the data in different structures, therefore we had to write queries that combined many different data from various tables to create summaries that the employees could see and investigate. We also had to implement functions to customize the stored data as mistakes could happen and various functions to do undo changes and functions to save history of action as precautions. These would later be implemented into visually appealing programs in the Visual Studio for the ease of end-user.

```
ALTER PROCEDURE [dbo].[MusteriEkle]
(
    @ad as nvarchar(30),
    @aciklama as nvarchar(max),
    @il_id as int,
    @ilce_id as int,
    @telefon as nvarchar(20),
    @yetki as nvarchar(30)
)
AS
BEGIN
INSERT INTO [dbo].[tblMusteri]
(
    [ad]
    , [aciklama]
    , [il_id]
    , [ilce_id]
    , [telefon]
    , [yetki]
)
VALUES
(
    @ad
    , @aciklama
    , @il_id
    , @ilce_id
    , @telefon
    , @yetki
)
END
```

Figure 3) A stored procedure for adding a customer.

```
USE [CRM]
GO
/***** Object: StoredProcedure [dbo].[MusteriAra] *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[MusteriAra]
(@ad as nvarchar(30))
AS
BEGIN
SELECT *
FROM vwMusteri
WHERE isim LIKE '%' + @ad + '%'
END
```

Figure 4) A stored procedure for searching a customer. It takes a name as an input to search. %'s allows it to show all customers that include the input anywhere in their saved names so the user doesn't have to write exactly correct name to find a customer.

This program was designed with a theme similar to Windows. Textboxes, buttons and lists that were used looked like the common Windows style therefore the employees would feel familiar to the style of the program.

I also implemented different error messages for every different scenario, thus the mistaking would not harm the accuracy of the database and the employees would know what they have done wrong and fix the mistake instead of getting confused and frustrated (Figure 5).

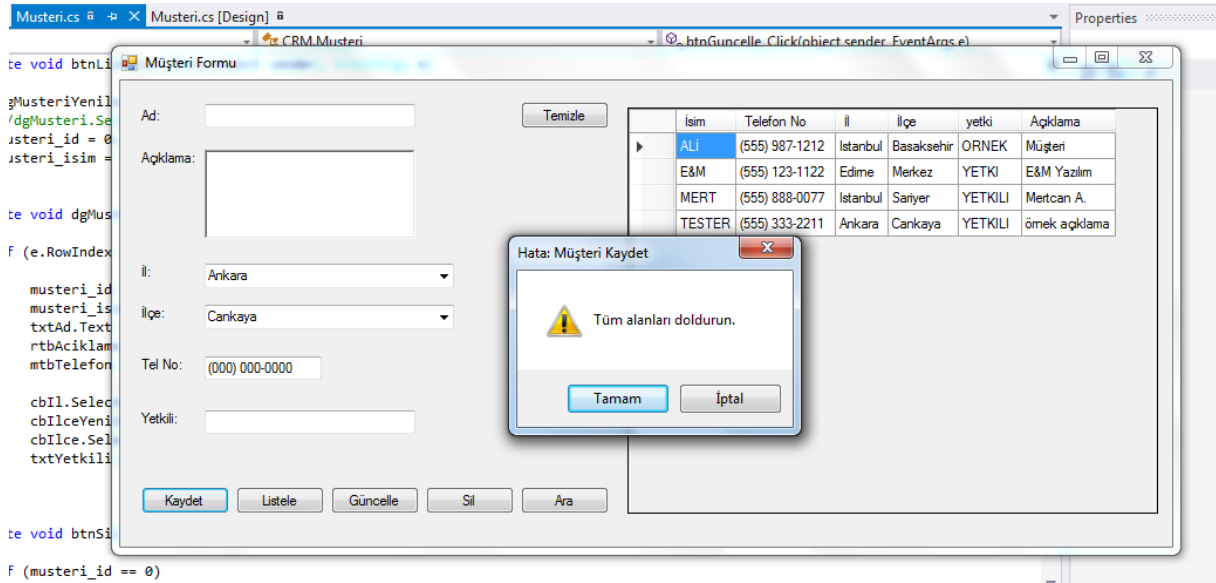


Figure 5) An example of an error. All the information is required to register a customer, so leaving them empty results in an error message explaining this condition to the user.

I also had to think out few corner cases that would harm the accuracy of the database. Firstly unique fields had to remain unique. So I had to check whether customer names were present in the database before adding them or editing them in since SQL would give off an error message if that was the case. The program informed the user in such cases (Figure 6).

I also minimized this sort of mistake by standardizing the telephone number. That textbox only accepts numeric inputs and shows them in a format that is easily readable by users. But the program converts them into numbers before putting them into the database as I used integer for saving telephone numbers to save space. Then it translates them back to the readable form when user lists the customers in the database.

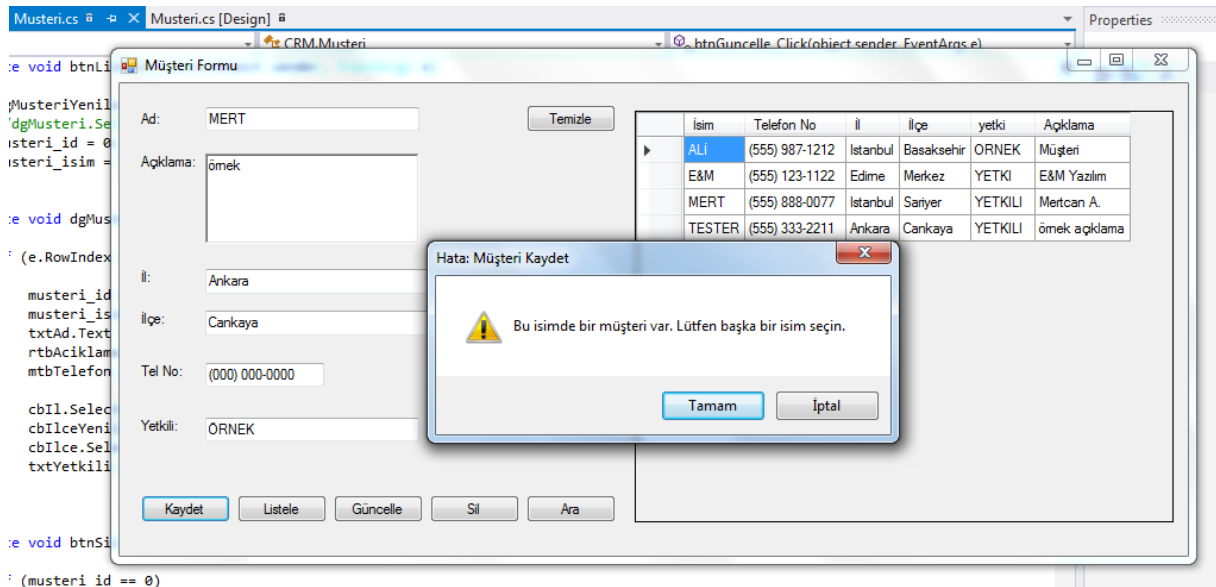


Figure 6) An error message pointing that not another customer can be added with this name as there is already one existing with that name. Both buttons do nothing as name is a unique field in the customer table.

I also wanted to implement confirmation messages on undoable actions, like deleting data from the database (Figure 6). This was an important precaution as human error occurs too often to be ignored, and crucial information could be lost if it was so easy to delete data permanently. Both the No and Cancel do nothing, but I wanted to add a cancel button, since many computers like to use cancel when they accidentally click on something and a pop-up appears.

We'd also have a logging system where actions like deleting would be logged as a safety precaution against malicious attempts to harm to database or any mistakes that happen addressing. But the implementation of this was not my part.

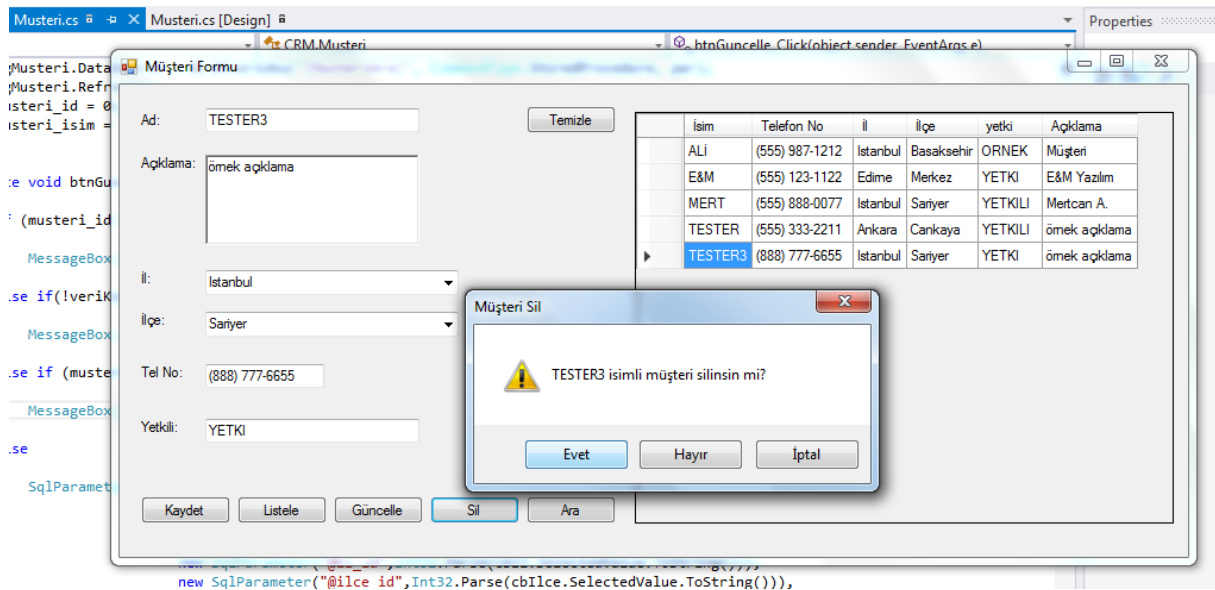


Figure 6) The confirmation box for deleting a customer. Yes runs a procedure for deleting a customer from the database whereas no and cancel do nothing.

Besides warning the user when they have done wrong things, I also wanted to inform the users when tasks they wanted to do were done. This was to reassure the user everything went right as not many employees of the company are experts on the computer and to lessen the chance of users trying to redo the things like adding a customer just because they are not sure if it has been done or not. So I added little confirmation box after important adjustments like adding or editing have been done on the database (Figure 7).

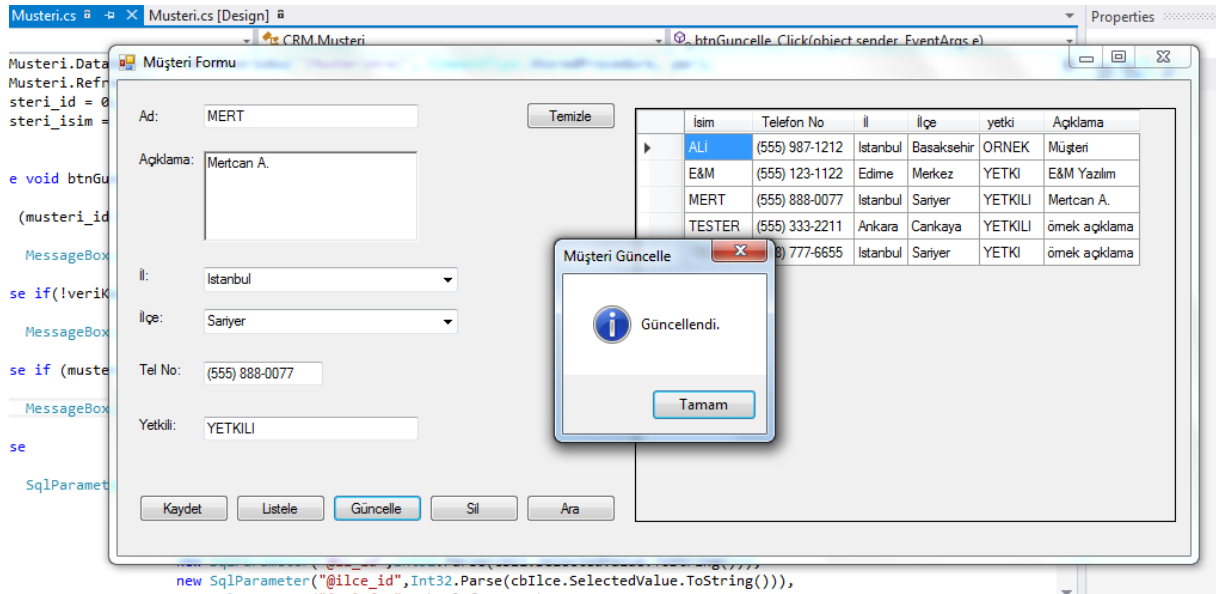
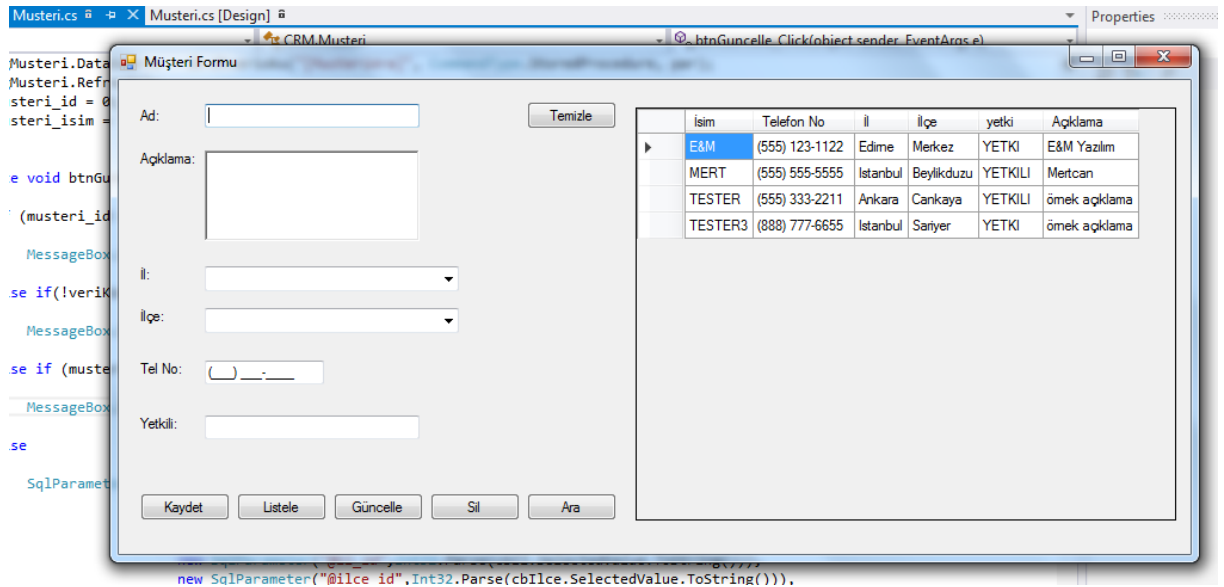


Figure 7) A confirmation box showing that user had edited the information on a customer. Okay button does nothing as the process has already been completed successfully.

After those steps, I had a pretty complete form for interacting with customer table in the database (Figure 8). Users could add, remove, update customer information and they were informed when something went wrong or their task was completed successfully. They were informed of the mistake if they had done any to solve it. The program neatly showed the customer information on a data grid, and this could be used to easily sort and select customers.



İsim	Telefon No	İl	İlçe	Yetki	Açıklama
E&M	(555) 123-1122	Edirne	Merkez	YETKİ	E&M Yazılım
MERT	(555) 555-5555	İstanbul	Beylikduzu	YETKİLİ	Mertcan
TESTER	(555) 333-2211	Ankara	Cankaya	YETKİLİ	örnek açıklama
TESTER3	(888) 777-6655	İstanbul	Sarıyer	YETKİ	örnek açıklama

Figure 8) A form used to save, update, delete a customer from the database. It can also search through it and sort it however the use likes it.

Then we had to design forms that interact with the database and they would be used in all stages of production. These forms had to be customized greatly for quicker business processes. Each step of workflow had a different segment in the program so different departments could easily access their commands and not interfere others by accident.

Later we had to present the application to our customer, get feedback and customize the program again to satisfy their demands until all of them were met. Again we had to meet with everyone from different stages of the production to communicate what exactly was required for them to operate accurately and efficiently. This was a difficult process as most of the employees had not used such a program or a database except Excel, so we had to ask many questions to minimize misunderstandings. The program we had developed was very modular so it could be easily changed and adapt to demands by the company. This was beneficial at the development stage as it undergo various changes and it will be useful years after being used and company wants to adapt new functionalities to the program.

Lastly SQL Server should be setup in the customers' office and filled with required data like the products they provide. The program should be installed in every computer and explain the usage to the employees of the company.

3.4. Results

The project the company had taken was in the later stages when I started my internship and it was not completed when I was done working. So my task was only a small part of the project. I did my part successfully as the form I setup was working accurately. I had a working program for adding, editing, removing, searching customers from the database that was usable by the end-user.

Final product of my project has screenshots in the appendix. Text boxes, lists, and buttons have appropriate names for users to understand their functionality, error messages have explanatory descriptions and, button and error messages have the Windows interface for users to feel familiar. So it is not the best at being visually appealing but it is highly intelligible.

4. Conclusions

I have already taken the COMP 306 where I learnt SQL and concepts of databases so I was able to learn new functions of SQL easily and familiarize myself with Management Studio with ease. I was also able make the database efficiently since I knew concepts like normalization.

I also had taken advanced programming course COMP 132, where I learnt Java and C. Therefore it was beneficial for me as C# is like the combination of those both languages so I could easily program.

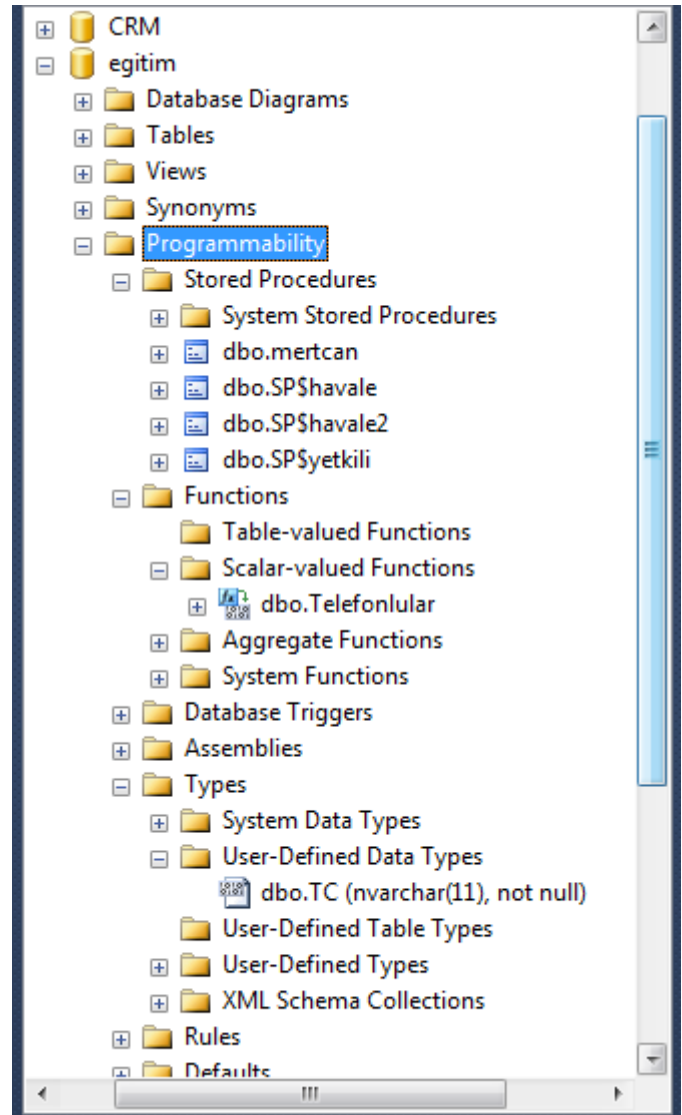
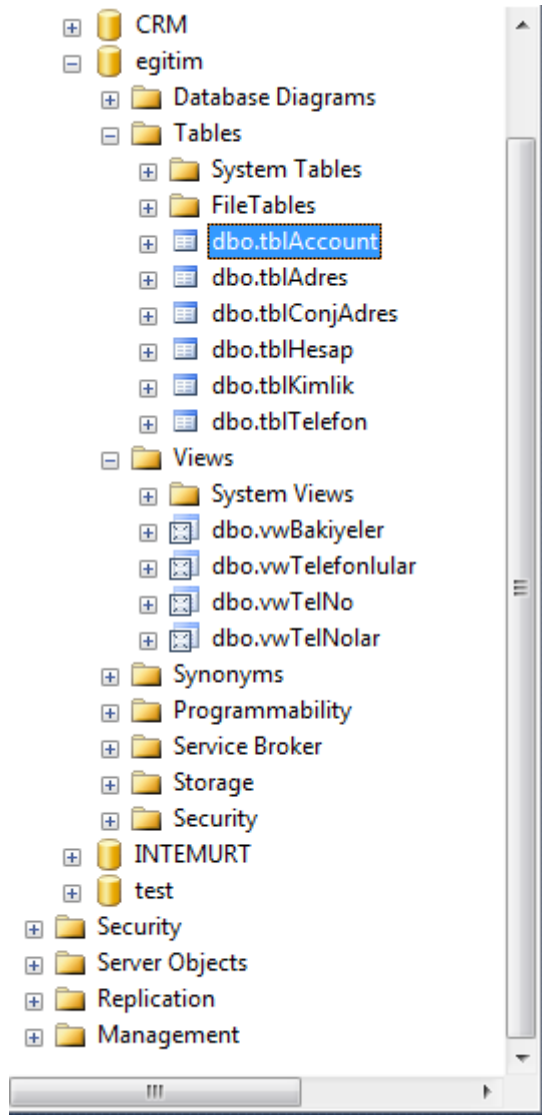
In class of COMP 306, we were using MyWebSQL for the database. In my workplace we used Management Studio with SQL. There are small syntax differences with MyWebSQL and SQL but real difference was the utilities that were provided with Management Studio. I learnt how to implement many different features of SQL with the interface of Management Studio which speed up the process of setting the database up. Setting composite keys, unique fields, incrementing fields are an example features that can easily implement on Management Studio. I used these features as often as possible to learn them.

Also my work was more focused onto producing user friendly end products. For example showing tables from the database in Excel or saving transaction history was important for work but they were not part of the classwork.

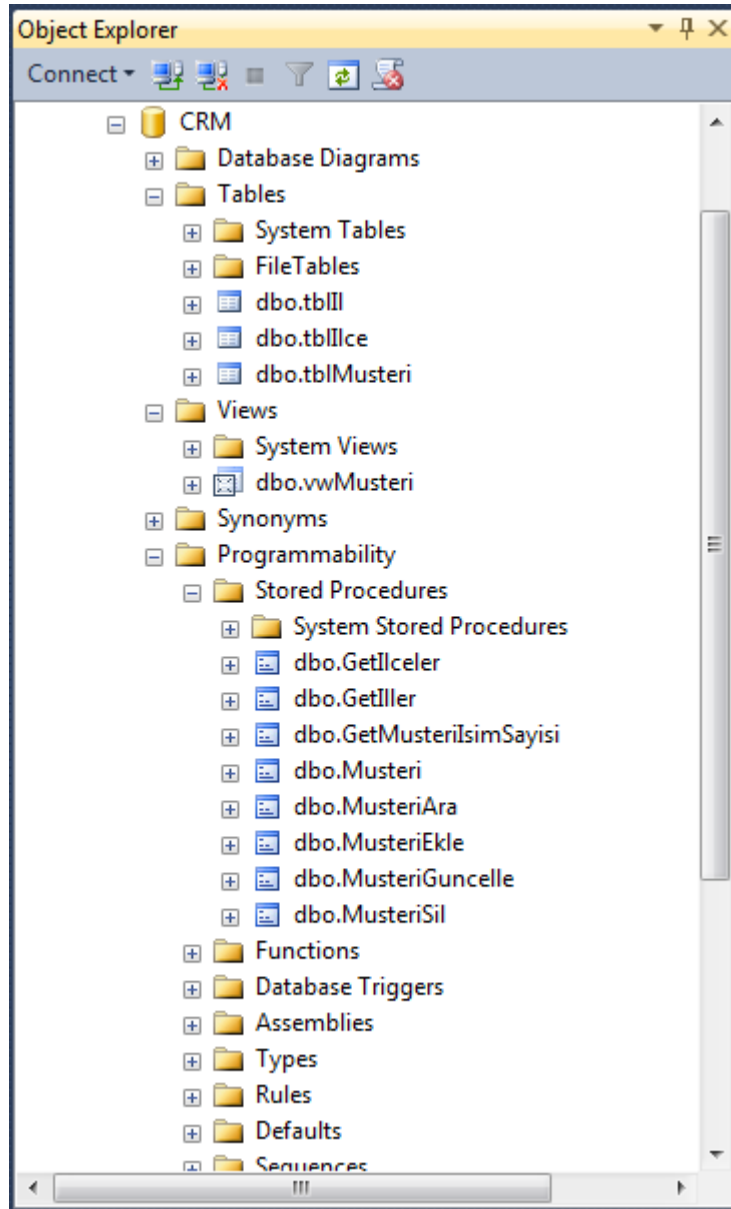
In my internship I had a chance to learn pulling data from a SQL server into Excel files and even run stored procedures that took an input from a excel cell and show a query based on that in real-time. This method could be used to display data from a database without designing a program and not giving a chance for a user to edit or delete something from the database as it only pulls data from the database and editing done in the Excel does not change the database.

In my internship I learnt about setting up visual interface with Microsoft Visual Studio and connecting the database to excel and Visual Studio so I feel more confident in my database knowledge as now I know how to implement a database in a real work environment.

5. Appendix



1) "egitim" is the database I made to practice SQL with all the tables, views, stored procedures and user defined date type.



2) Database I setup for my form project with tables and stored procedures.

	id	ad	aciklama	il_id	ilce_id	telefon	yetki
▶	1	MERT	Mertcan A.	34	3	(555) 888-0077	YETKILI
	2	E&M	E&M Yazılım	22	1	(555) 123-1122	YETKI
	5	TESTER	örnek açıklama	6	1	(555) 333-2211	YETKILI
	27	ALİ	Müşteri	34	2	(555) 987-1212	ORNEK
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3) Table "tblMusteri"

Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
ad	nvarchar(30)	<input type="checkbox"/>
aciklama	nvarchar(MAX)	<input checked="" type="checkbox"/>
il_id	int	<input checked="" type="checkbox"/>
ilce_id	int	<input checked="" type="checkbox"/>
telefon	nvarchar(20)	<input checked="" type="checkbox"/>
yetki	nvarchar(30)	<input checked="" type="checkbox"/>

Column Properties	
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
Indexable	Yes
Is Columnset	No
Is Sparse	No
Merge-published	No
Not For Replication	No
Replicated	No
RowGuid	No

4) "tblMusteri" data types of the fields

```

USE [CRM]
GO
/***** Object: StoredProcedure [dbo].[GetIlceler]    Script Date: 23.09.2016 16:53:24 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[GetIlceler]
    @il_id int
AS
BEGIN
    SELECT * FROM tblIlce WHERE il_id=@il_id
END

```

5) Procedure that gets the Provinces in the database

```

ALTER PROCEDURE [dbo].[GetMusteriIsimSayisi]
    (@ad as nvarchar(30))
AS
BEGIN
    SELECT COUNT(*)
    FROM tblMusteri
    WHERE (SELECT REPLACE(ad, ' ', '')) = (SELECT REPLACE (@ad, ' ', ''))
end

```

6) Stored Procedure which counts customers with the same name in the table. It's used for checking whether name is available since it needs to be unique.



```
ALTER PROCEDURE [dbo].[MusteriEkle]
(
    @ad as nvarchar(30),
    @aciklama as nvarchar(max),
    @il_id as int,
    @ilce_id as int,
    @telefon as nvarchar(20),
    @yetki as nvarchar(30)
)
AS
BEGIN
    INSERT INTO [dbo].[tblMusteri]
        ([ad]
        ,[aciklama]
        ,[il_id]
        ,[ilce_id]
        ,[telefon]
        ,[yetki])
    VALUES
        (@ad
        ,@aciklama
        ,@il_id
        ,@ilce_id
        ,@telefon
        ,@yetki)
END
```

7) Stored Function for adding customers to the database



```
USE [CRM]
GO
/***** Object: StoredProcedure [dbo].[MusteriGu
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
|
ALTER PROCEDURE [dbo].[MusteriGuncelle]
(
    @id as int,
    @ad as nvarchar(30),
    @aciklama as nvarchar(max),
    @il_id as int,
    @ilce_id as int,
    @telefon as nvarchar(20),
    @yetki as nvarchar(30)
)
AS
BEGIN
UPDATE [dbo].[tblMusteri]
    SET [ad] = @ad
      ,[aciklama] = @aciklama
      ,[il_id] = @il_id
      ,[ilce_id] = @ilce_id
      ,[telefon] = @telefon
      ,[yetki] = @yetki
    WHERE id=@id
END
```

8) Stored Function for updating the customer details



```

USE [CRM]
GO
/***** Object: StoredProcedure [dbo].[MusteriAra]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[MusteriAra]
    (@ad as nvarchar(30))
AS
BEGIN
    SELECT *
    FROM vwMusteri
    WHERE isim LIKE '%' + @ad + '%'
END

```

9) Stored Function for Search

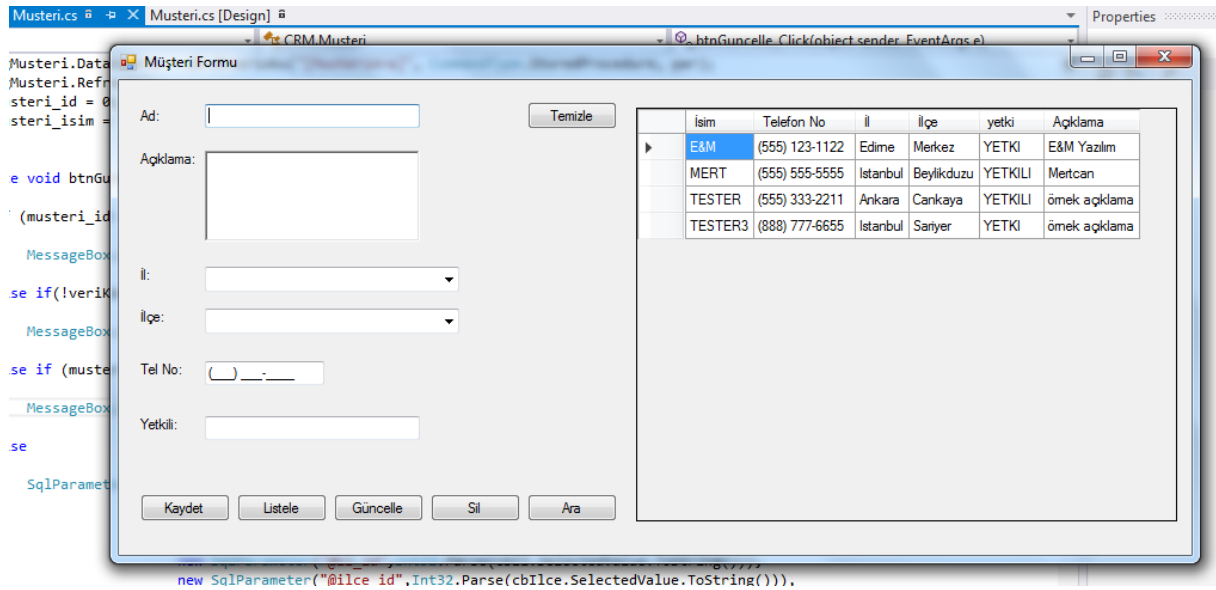
```

USE [CRM]
GO
/***** Object: StoredProcedure [dbo].
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[MusteriSil]
    (@id as int)
AS
    DELETE FROM [dbo].[tblMusteri]
    WHERE id=@id

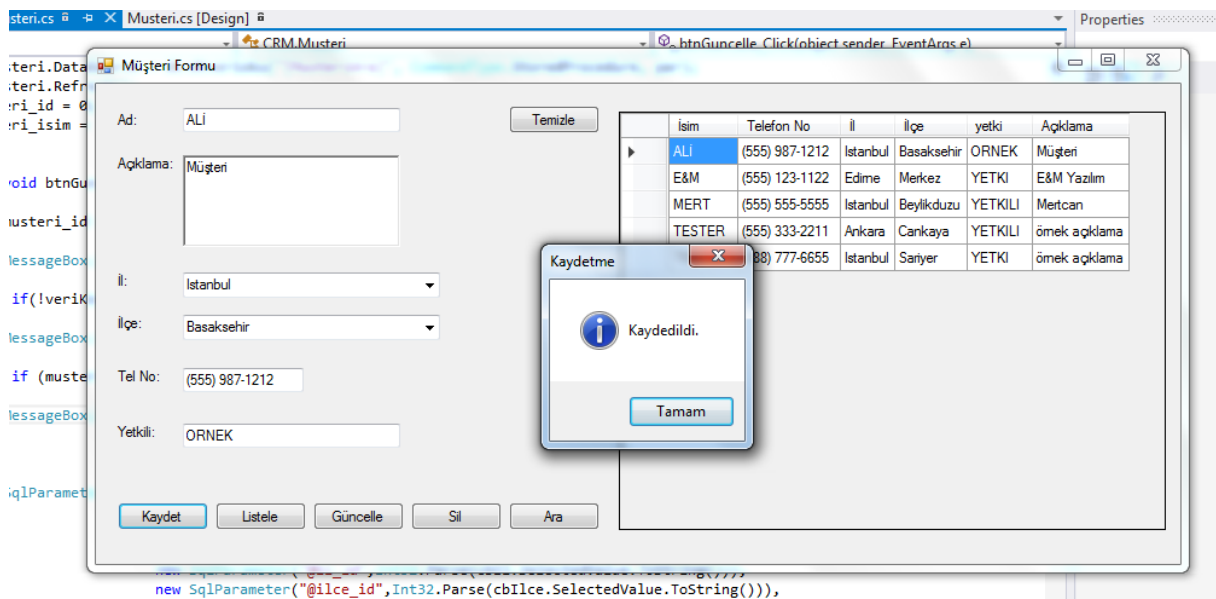
```

10) Stored Function for Delete



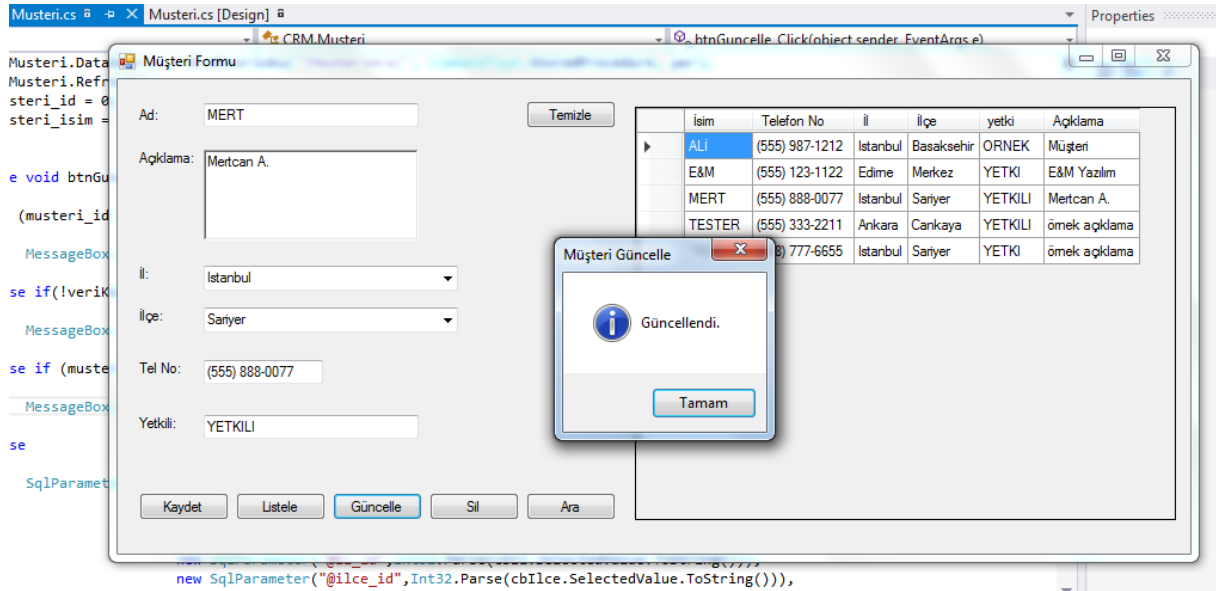
İsim	Telefon No	İl	İlçe	yetki	Açıklama
E&M	(555) 123-1122	Edirne	Merkez	YETKİ	E&M Yazılım
MERT	(555) 555-5555	İstanbul	Beylikduzu	YETKİLİ	Mertcan
TESTER	(555) 333-2211	Ankara	Cankaya	YETKİLİ	örnek açıklama
TESTER3	(888) 777-6655	İstanbul	Sarıyer	YETKİ	örnek açıklama

11) The interface of the form

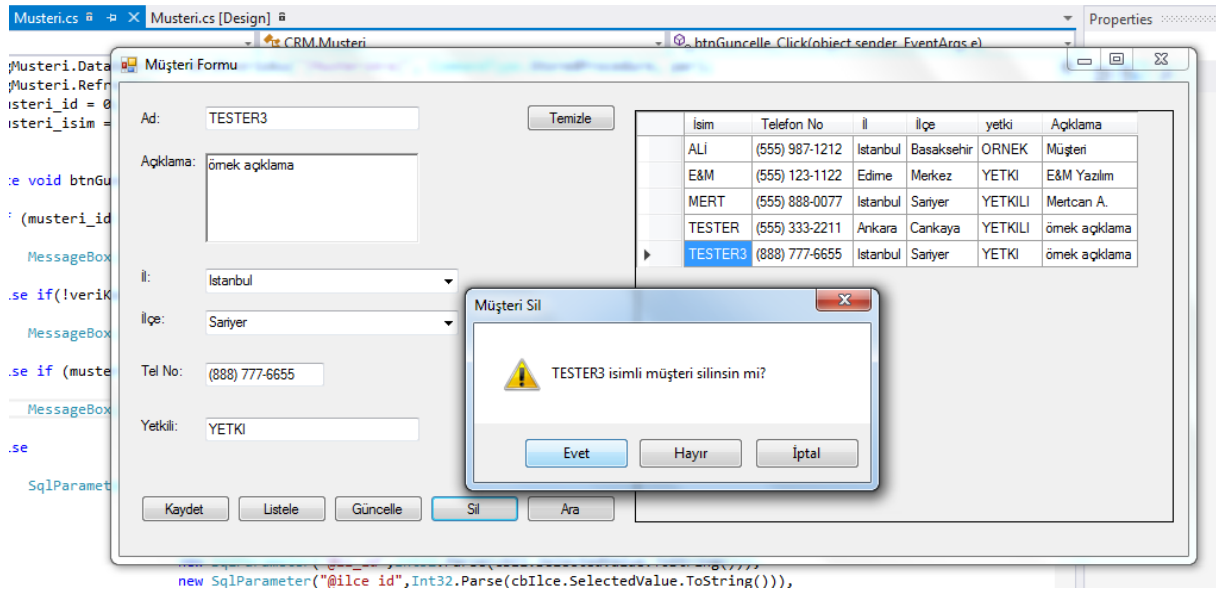


İsim	Telefon No	İl	İlçe	yetki	Açıklama
ALI	(555) 987-1212	İstanbul	Basaksehir	ORNEK	Müşteri
E&M	(555) 123-1122	Edirne	Merkez	YETKİ	E&M Yazılım
MERT	(555) 555-5555	İstanbul	Beylikduzu	YETKİLİ	Mertcan
TESTER	(555) 333-2211	Ankara	Cankaya	YETKİLİ	örnek açıklama
TESTER3	(888) 777-6655	İstanbul	Sarıyer	YETKİ	örnek açıklama

12) Adding a customer



13) Updating a customer



14) Confirmation on delete

İsim	Telefon No	İl	İlçe	yetki	Açıklama
ALI	(555) 987-1212	İstanbul	Basaksehir	ORNEK	Müşteri
E&M	(555) 123-1122	Edirne	Merkez	YETKİ	E&M Yazılım
MERT	(555) 888-0077	İstanbul	Sarıyer	YETKİLİ	Mertcan A.
TESTER	(555) 333-2211	Ankara	Cankaya	YETKİLİ	örnek açıklama

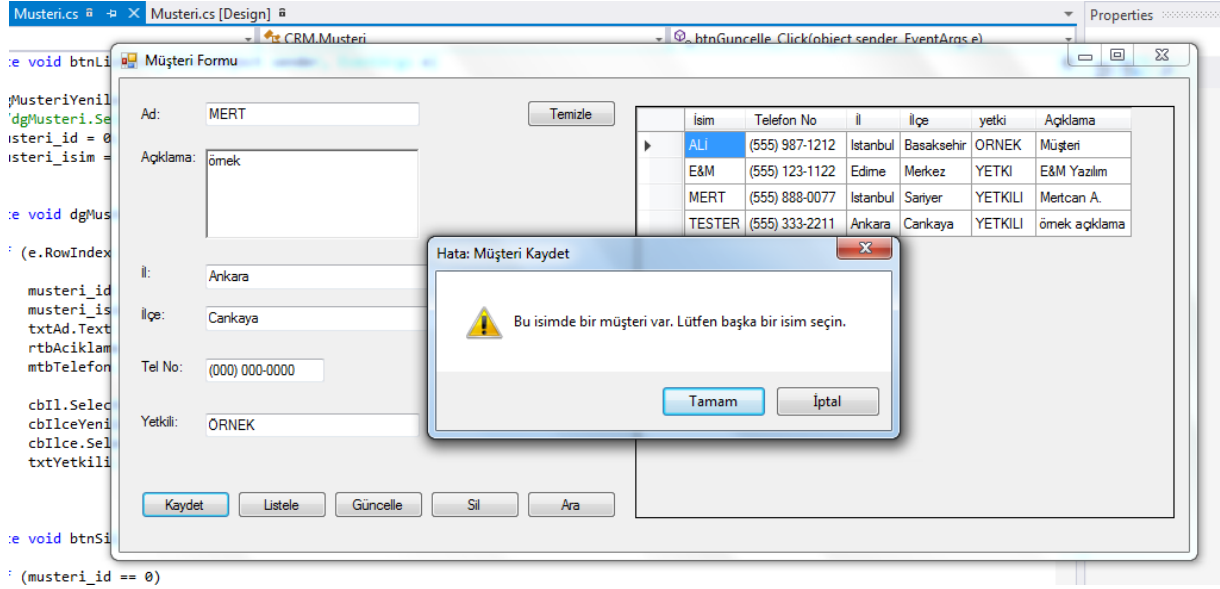
The 'Sil' button is highlighted in blue. The background shows the Visual Studio IDE with the 'Musteri.cs' file open and some code visible." data-bbox="114 166 881 417"/>

15) Table after deleting "TESTER3"

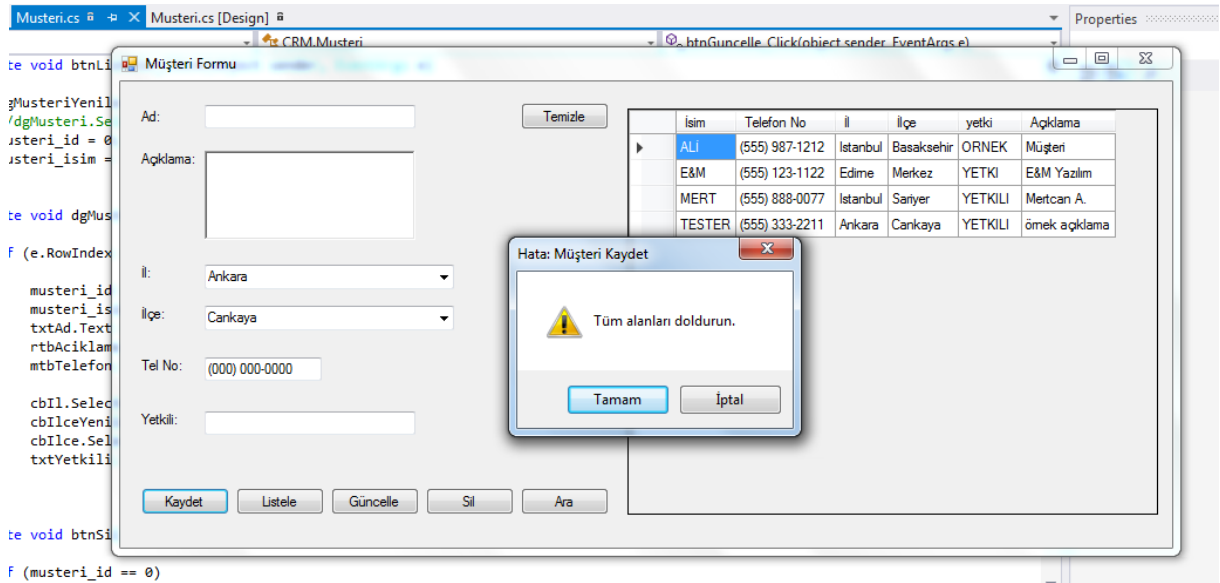
İsim	Telefon No	İl	İlçe	yetki	Açıklama
MERT	(555) 888-0077	İstanbul	Sarıyer	YETKİLİ	Mertcan A.
TESTER	(555) 333-2211	Ankara	Cankaya	YETKİLİ	örnek açıklama

The 'Ara' button is highlighted in blue. The background shows the Visual Studio IDE with the 'Musteri.cs' file open and some code visible." data-bbox="114 484 881 741"/>

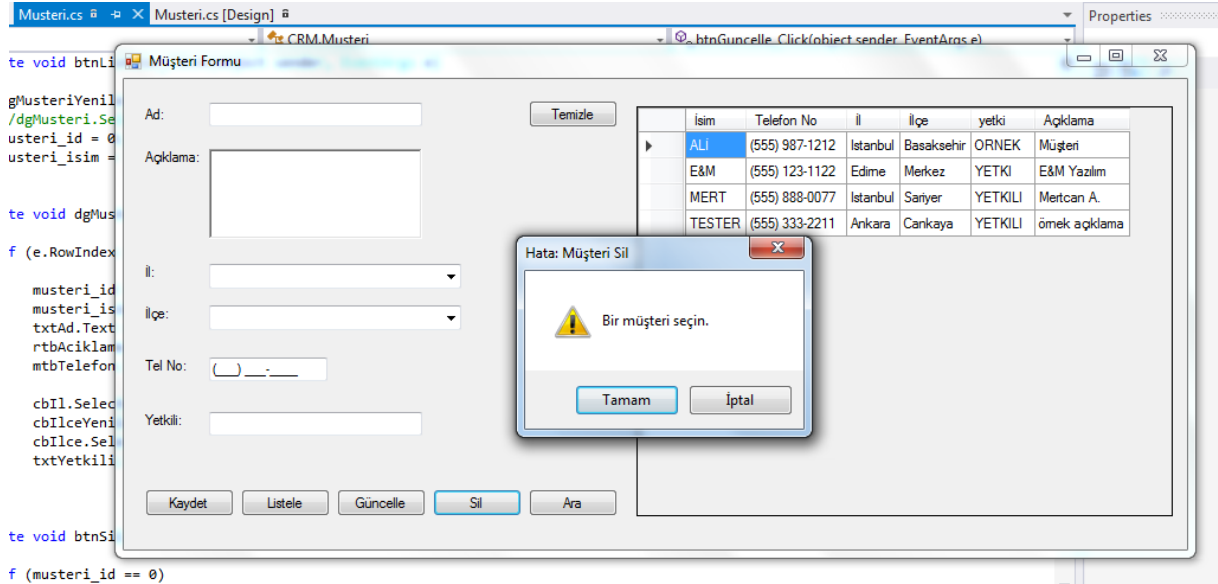
16) Searching for customers with "ER" in their name



17) Error message for trying to add a customer with an existing name



18) Error message for not filling all the fields



19) Error message for trying delete without selecting a customer