Automobiles Spare Parts Trading

A web-application for buying and selling car spare parts based on business to business policy

Product Owners: Taimaz Lakpour, Mehdi Arfaei

Project Members: Taimaz Lakpour, Mehdi Arfaei

Authored by : Taimaz Lakpour

CONTENTS:

Team goals and business objective:

In this web-application, we tried to implement the best policy of buying and selling of automobiles spare parts (B2B policy), to be able to have a positive effect on the trading process of automobile spare parts in markets. In this document, we tried to explain how this web-application works.

Our biggest goal in this project is transaction any automobiles spare parts at the best market price, So that buyers can have different choices between different sellers based on price and quality. On the other hand, sellers have the rights to make many choices based on the number and price of the products they offer.

This policy can be based on the number of purchases or the type of product between major or minor sellers or buyers, and ultimately between consumers of parts. Simply put, a major buyer of spare parts to purchase a specific part that requires a large number of it can buy that part from every seller as a major supplier or a minor spare parts seller. Thus, it could have many choices from different vendors.

Our vision was to create this web-application for the Iranian market and the choice of the subject of the transaction, which is the same as car spare parts, has been selected according to the needs of the Iranian online market. Also, the details of this program have been designed based on the Iranian online market according to its activities. However, due to the coding and design standards, with a few changes in the details of the web-application, you could change the language of the program or the subject of the transaction and present another version of the same web-application with a different transaction topic in another language.

Before explaining the web-applications further, let's take a look at the business-to-business policy:

Business to Business marketing (B2B):

Business to Business marketing (commonly known as B2B marketing) involves selling one company's product or service to another. B2B marketing techniques rely on the same basic principles of consumer marketing, but are implemented in a unique way. Given that consumers choose products not only in terms of price but also in terms of popularity, status, and other emotional stimuli, while for B2B buyers only the price and potential profit of the product is important.

In B2B Business, anything that can make buying and selling faster and easier, such as partners and business-related services such as suppliers, buyers, product features, service support, software applications, etc. can be gathered in one place. As a result, B2B sales methods make it easier for buyers and sellers to buy and sell, auction and develop services and products, and this reduces costs.

General description of how this web-application works using the B2B method:

We have already got a policy that is based on politics, business to business, So that we can provide buyers with a list of sellers based on the needs of buyers. Therefore, the buyer is faced with a list of sellers based on buyers need and could buy the desired product from one of the sellers. All buyers' requests based on product type classification will be provided to various vendors and suppliers of auto spare parts.

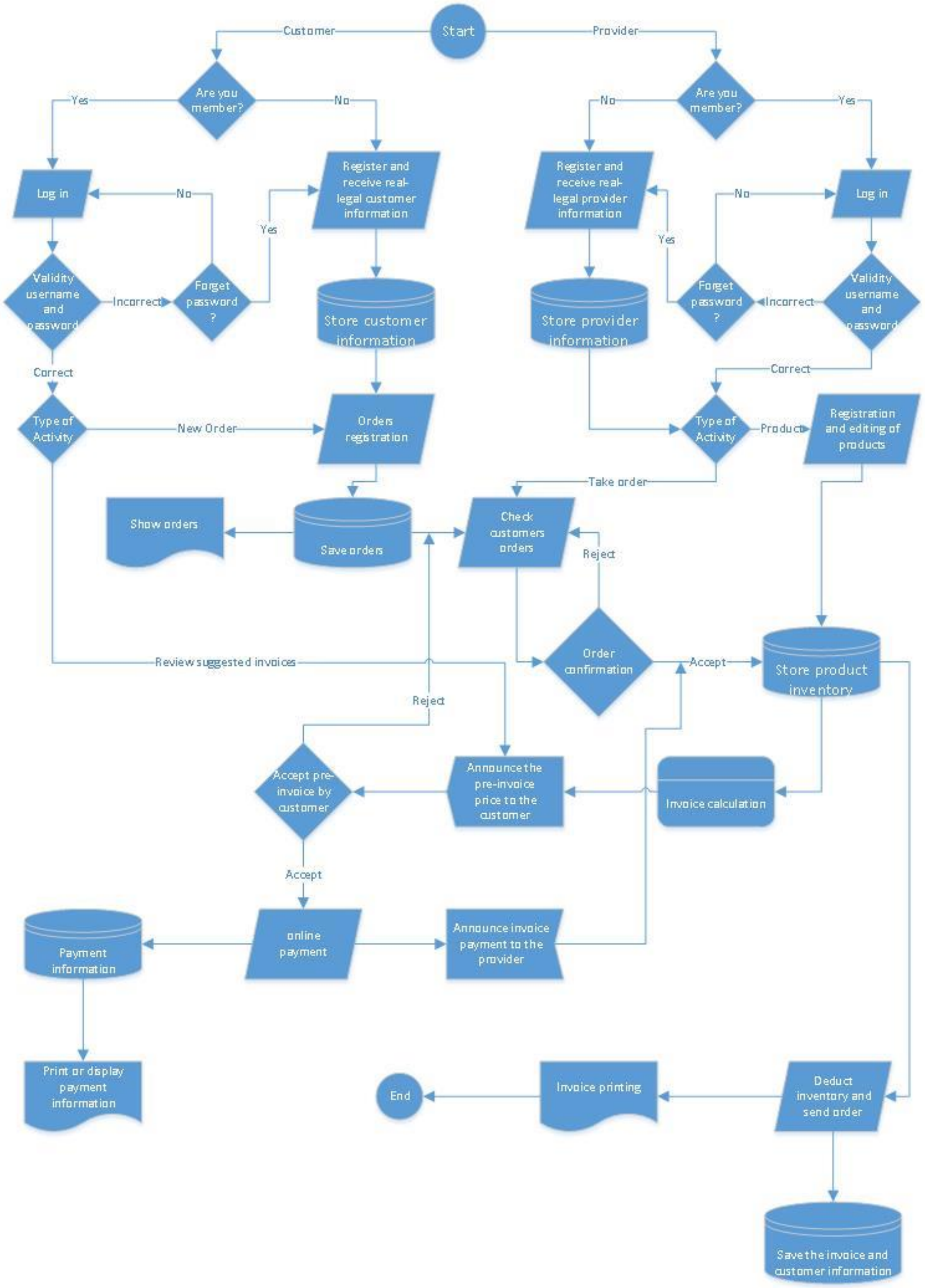
The biggest important feature of this web-application is that the web-application itself does not enter directly into the transaction with the buyer and seller. This means that the buyer buys the product directly from his chosen seller and not from the auto spare parts website, and this web-application only provides a platform for the best deal for buyers and sellers. This feature allows the buyer and seller to enter into a direct transaction and the web-application is only to introduce sellers or suppliers to buyers.

The second important feature of this web-application is that each user can act both in the role of seller and in the role of buyer in this web-application. Nevertheless, the user can buy products as a buyer and on the other hand sell the same purchased products to another customer as a seller. In this web-application, any user with any amount of requests could make their purchase at the best price. Whether suppliers, major manufacturers or users who need spare parts for personal consumption, reach their goal through this web-application.

Vendors and providers are also faced with a list of requests for parts they needed by buyers who can choose the requests that they are able to supply, and according to the type of part, the number of requests for a specific part and the quality of that part, according to the manufacturer company and the country of manufacture of that part, could offer an acceptable price to the buyer. In this policy, it is very important that sellers and providers are able to offer acceptable prices to buyers. In general, this policy could create a competitive market in terms of price and quality of products and on the other hand reduce the brokerage rate between buyers and sellers. So buyers can buy their desired spare parts at the lowest price by their choice.

This sales policy could be based on the number of purchases or the type of product between major suppliers, wholesale or retail sellers, or wholesale and retail buyers, and ultimately between parts consumers. For example, a major buyer of auto spare parts to purchase a specific part that requires a large number of it can buy that part from a major supplier or a minor parts retailer, and can have many different vendors to choose from. In the following, we will take a closer look at this trading process.

Web-application performance by flowchart:


Start
Customer
Provider
Are you member?
Yes
No
Log in
Register and receive real-legal customer information
Register and receive real-legal provider information
Validity username and password
Incorrect
Forget password ?
Store customer information
Store provider information
Correct
Type of Activity
New Order
Orders registration
Product
Registration and editing of products
Take order
Show orders
Save orders
Check customers orders
Reject
Order confirmation
Accept
Store product inventory
Review suggested invoices
Accept pre-invoice by customer
Announce the pre-invoice price to the customer
Invoice calculation
online payment
Payment information
Announce invoice payment to the provider
Print or display payment information
End
Invoice printing
Deduct inventory and send order
Save the invoice and customer information

Generalities in Design and Develop:

This software is a "Web-Application" that we implemented with "C#" programming language at Backend level and with "ASP.Net" programming language at Interface level. And we used the ".Net Framework – IIS" web server to present "HTML" pages and files. In this web-application, the database is designed with "MS. SQL server" and contained 26 designed original tables and 11 sub-tables (gained from normalization process) and 11 tables related to the membership module and profile according to Microsoft security standards. All users' activities with the name of the "log" are recorded and maintained in the "MS. Office Access" database.

In this two-person work group, we agreed to use meaningful naming in the following different types, to make the code more readable and better understanding the performance of different sections.

Principles of naming "Types" and "Parameters":

- Methods
- Properties
- Events
- Fields
- Parameters

Standard coding is one of the most important factors in doing a good team work. In fact, it helps team members to mastering written codes by another member at a lower cost and faster and makes the code have the same pattern and harmony, In such a way that all "Types" and their members, variables, parameters, files, etc. are examined. This process reduces the descriptions written in the code or possible errors, because their names carry enough information to the programmer.

Security and efficient layers:

This web-application is programmed in three levels "Data Layer", "Business Layer" and "Presentation Layer".

Date Access Layer:

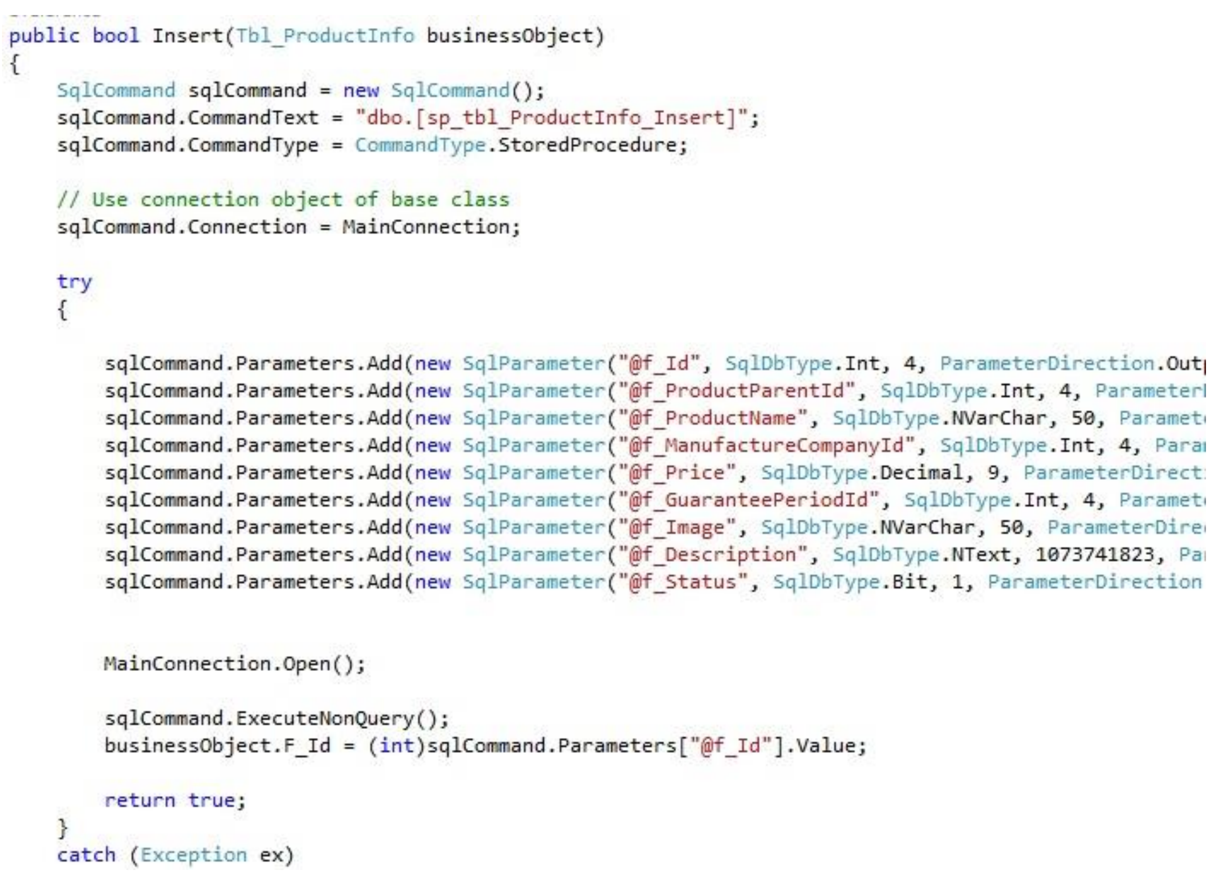
This layer, which is also called the "Database Layer", is responsible for managing the information in the database, based on the requests it receives from its top layer (Business Logic Layer), operations such as: "delete, add, modify and read information" on the database and sends the result to the top layer. It should be noted that communication with the database is done only through this layer. We used the "connection string" to connect between the database and this layer.

Several original and primary instructions are developed for each form individually such as Insert, Update, Select by Primary Key, Select All, Select by Field, Delete by Primary Key and Delete by Field.

In the following, for example, for the ProductInfo form, we show parts of the codes and how to communicate with the database. We implemented these instructions for all web-application's forms according to the function of each form:

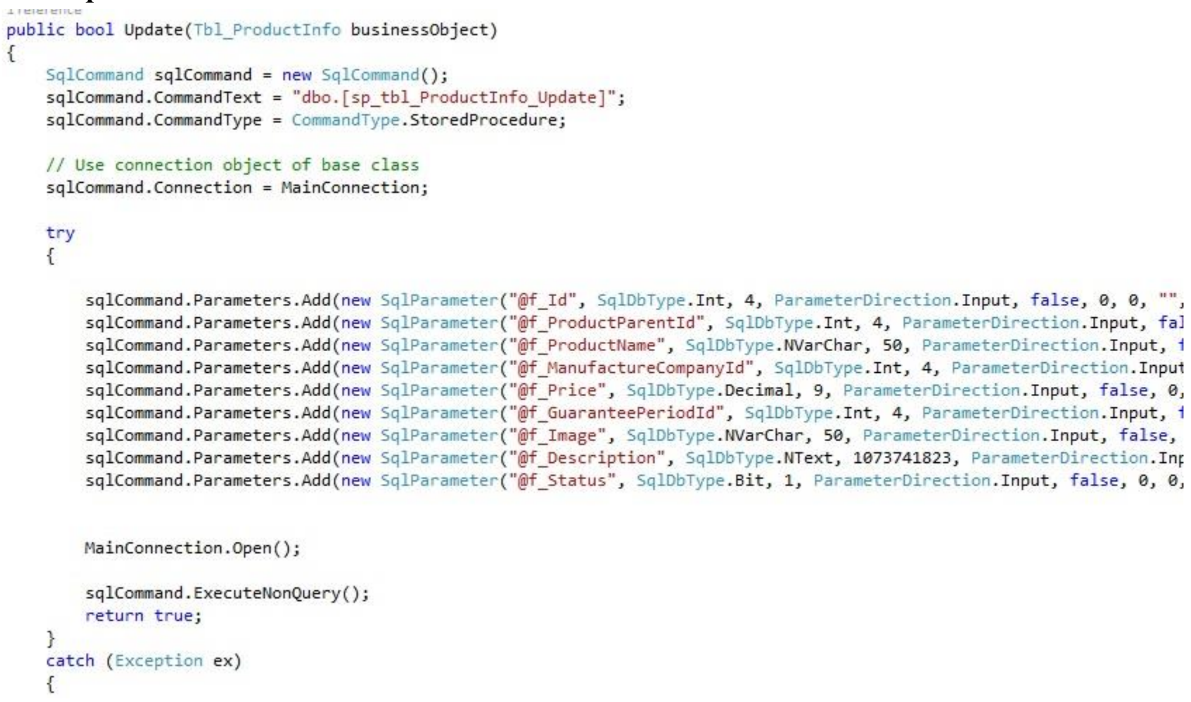
Examples of written code:

- **Insert:**



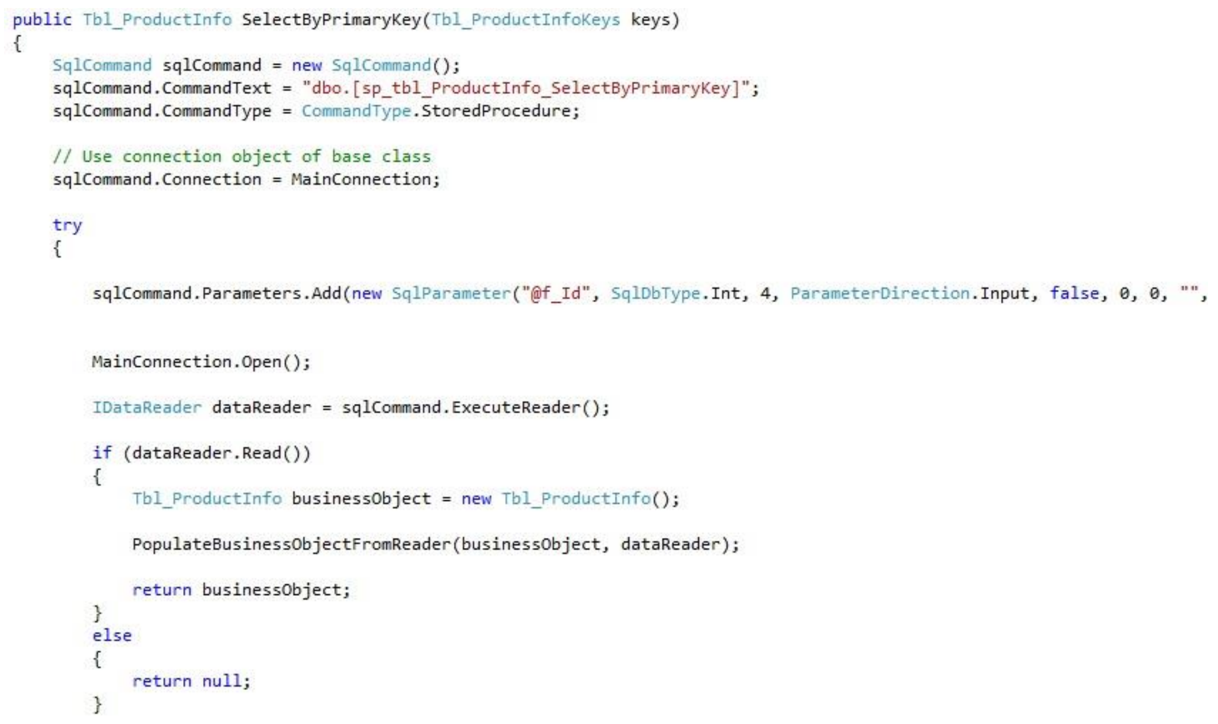
```
public bool Insert(Tbl_ProductInfo businessObject)
{
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_Insert]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        sqlCommand.Parameters.Add(new SqlParameter("@f_Id", SqlDbType.Int, 4, ParameterDirection.Out
        sqlCommand.Parameters.Add(new SqlParameter("@f_ProductParentId", SqlDbType.Int, 4, Parameter
        sqlCommand.Parameters.Add(new SqlParameter("@f_ProductName", SqlDbType.NVarChar, 50, Paramet
        sqlCommand.Parameters.Add(new SqlParameter("@f_ManufactureCompanyId", SqlDbType.Int, 4, Para
        sqlCommand.Parameters.Add(new SqlParameter("@f_Price", SqlDbType.Decimal, 9, ParameterDirect
        sqlCommand.Parameters.Add(new SqlParameter("@f_GuaranteePeriodId", SqlDbType.Int, 4, Paramet
        sqlCommand.Parameters.Add(new SqlParameter("@f_Image", SqlDbType.NVarChar, 50, ParameterDire
        sqlCommand.Parameters.Add(new SqlParameter("@f_Description", SqlDbType.NText, 1073741823, Pa
        sqlCommand.Parameters.Add(new SqlParameter("@f_Status", SqlDbType.Bit, 1, ParameterDirection


        MainConnection.Open();

        sqlCommand.ExecuteNonQuery();
        businessObject.F_Id = (int)sqlCommand.Parameters["@f_Id"].Value;

        return true;
    }
    catch (Exception ex)
```

- **Update:**



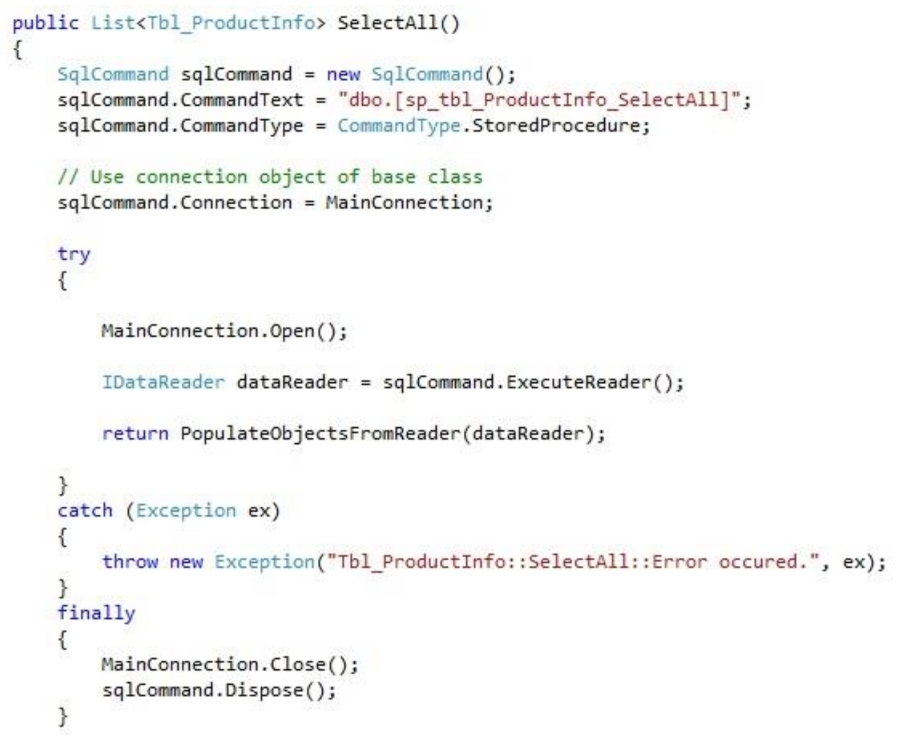
```
1 reference
public bool Update(Tbl_ProductInfo businessObject)
{
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_Update]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        sqlCommand.Parameters.Add(new SqlParameter("@f_Id", SqlDbType.Int, 4, ParameterDirection.Input, false, 0, 0, "",
        sqlCommand.Parameters.Add(new SqlParameter("@f_ProductParentId", SqlDbType.Int, 4, ParameterDirection.Input, fal
        sqlCommand.Parameters.Add(new SqlParameter("@f_ProductName", SqlDbType.NVarChar, 50, ParameterDirection.Input, f
        sqlCommand.Parameters.Add(new SqlParameter("@f_ManufactureCompanyId", SqlDbType.Int, 4, ParameterDirection.Input
        sqlCommand.Parameters.Add(new SqlParameter("@f_Price", SqlDbType.Decimal, 9, ParameterDirection.Input, false, 0,
        sqlCommand.Parameters.Add(new SqlParameter("@f_GuaranteePeriodId", SqlDbType.Int, 4, ParameterDirection.Input, f
        sqlCommand.Parameters.Add(new SqlParameter("@f_Image", SqlDbType.NVarChar, 50, ParameterDirection.Input, false,
        sqlCommand.Parameters.Add(new SqlParameter("@f_Description", SqlDbType.NText, 1073741823, ParameterDirection.Inp
        sqlCommand.Parameters.Add(new SqlParameter("@f_Status", SqlDbType.Bit, 1, ParameterDirection.Input, false, 0, 0,


        MainConnection.Open();

        sqlCommand.ExecuteNonQuery();
        return true;
    }
    catch (Exception ex)
    {
```

- **Select by Primary Key:**



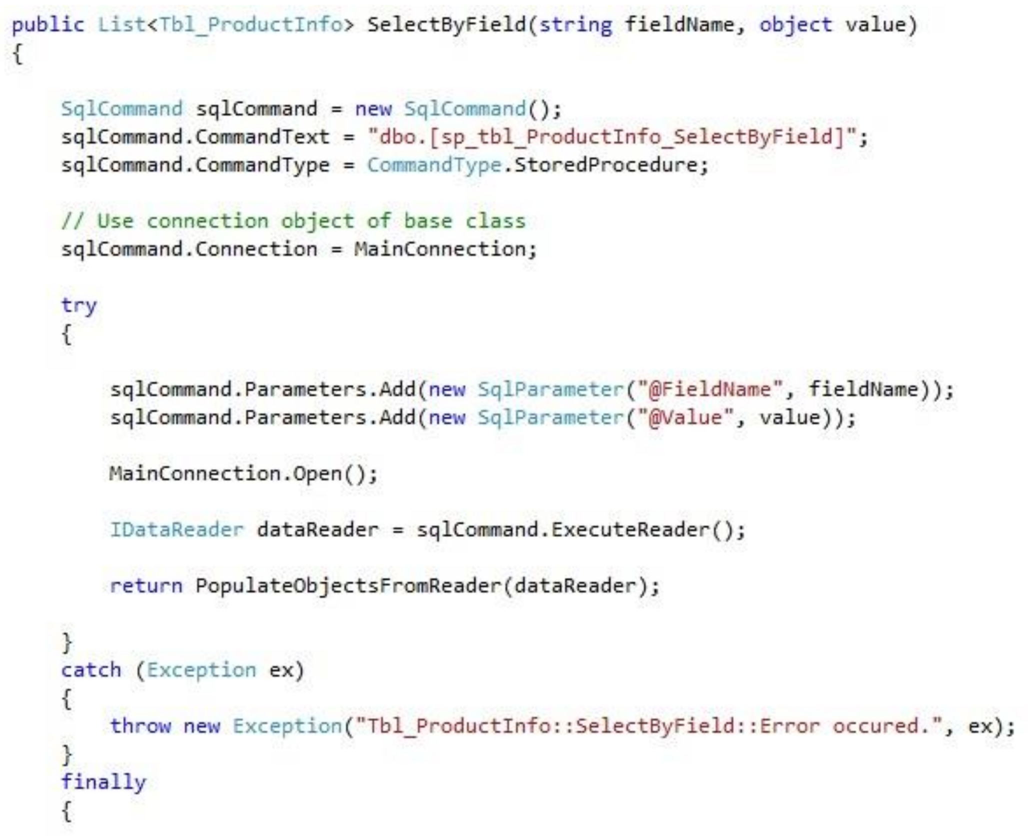
```
public Tbl_ProductInfo SelectByPrimaryKey(Tbl_ProductInfoKeys keys)
{
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_SelectByPrimaryKey]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        sqlCommand.Parameters.Add(new SqlParameter("@f_Id", SqlDbType.Int, 4, ParameterDirection.Input, false, 0, 0, "",


        MainConnection.Open();

        IDataReader dataReader = sqlCommand.ExecuteReader();

        if (dataReader.Read())
        {
            Tbl_ProductInfo businessObject = new Tbl_ProductInfo();

            PopulateBusinessObjectFromReader(businessObject, dataReader);

            return businessObject;
        }
        else
        {
            return null;
        }
```

- **Select All:**



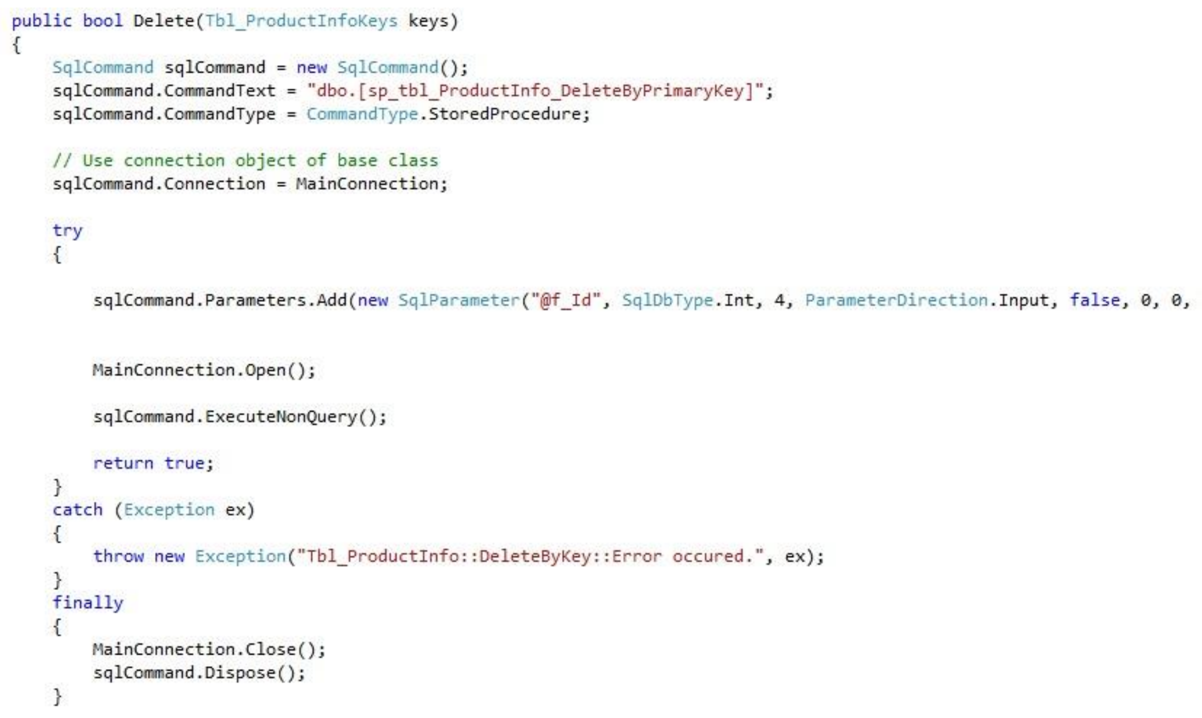
```
public List<Tbl_ProductInfo> SelectAll()
{
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_SelectAll]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        MainConnection.Open();

        IDataReader dataReader = sqlCommand.ExecuteReader();

        return PopulateObjectsFromReader(dataReader);

    }
    catch (Exception ex)
    {
        throw new Exception("Tbl_ProductInfo::SelectAll::Error occured.", ex);
    }
    finally
    {
        MainConnection.Close();
        sqlCommand.Dispose();
    }
```

- **Select by Field:**



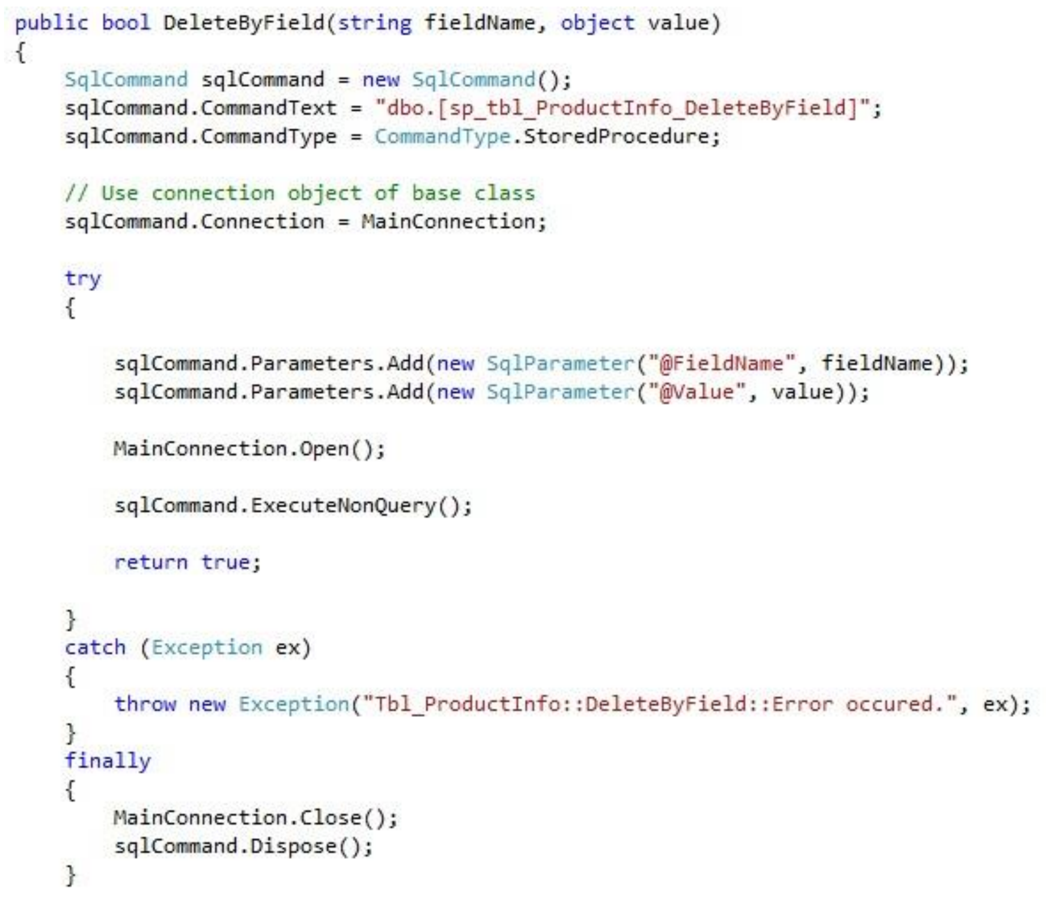
```
public List<Tbl_ProductInfo> SelectByField(string fieldName, object value)
{

    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_SelectByField]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        sqlCommand.Parameters.Add(new SqlParameter("@FieldName", fieldName));
        sqlCommand.Parameters.Add(new SqlParameter("@Value", value));

        MainConnection.Open();

        IDataReader dataReader = sqlCommand.ExecuteReader();

        return PopulateObjectsFromReader(dataReader);

    }
    catch (Exception ex)
    {
        throw new Exception("Tbl_ProductInfo::SelectByField::Error occured.", ex);
    }
    finally
    {
```

- **Delete by Primary Key:**



```
public bool Delete(Tbl_ProductInfoKeys keys)
{
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_DeleteByPrimaryKey]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        sqlCommand.Parameters.Add(new SqlParameter("@f_Id", SqlDbType.Int, 4, ParameterDirection.Input, false, 0, 0,


        MainConnection.Open();

        sqlCommand.ExecuteNonQuery();

        return true;
    }
    catch (Exception ex)
    {
        throw new Exception("Tbl_ProductInfo::DeleteByKey::Error occured.", ex);
    }
    finally
    {
        MainConnection.Close();
        sqlCommand.Dispose();
    }
```

- **Delete by Field:**



```
public bool DeleteByField(string fieldName, object value)
{
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText = "dbo.[sp_tbl_ProductInfo_DeleteByField]";
    sqlCommand.CommandType = CommandType.StoredProcedure;

    // Use connection object of base class
    sqlCommand.Connection = MainConnection;

    try
    {

        sqlCommand.Parameters.Add(new SqlParameter("@FieldName", fieldName));
        sqlCommand.Parameters.Add(new SqlParameter("@Value", value));

        MainConnection.Open();

        sqlCommand.ExecuteNonQuery();

        return true;

    }
    catch (Exception ex)
    {
        throw new Exception("Tbl_ProductInfo::DeleteByField::Error occured.", ex);
    }
    finally
    {
        MainConnection.Close();
        sqlCommand.Dispose();
    }
```

Business Logic Layer:

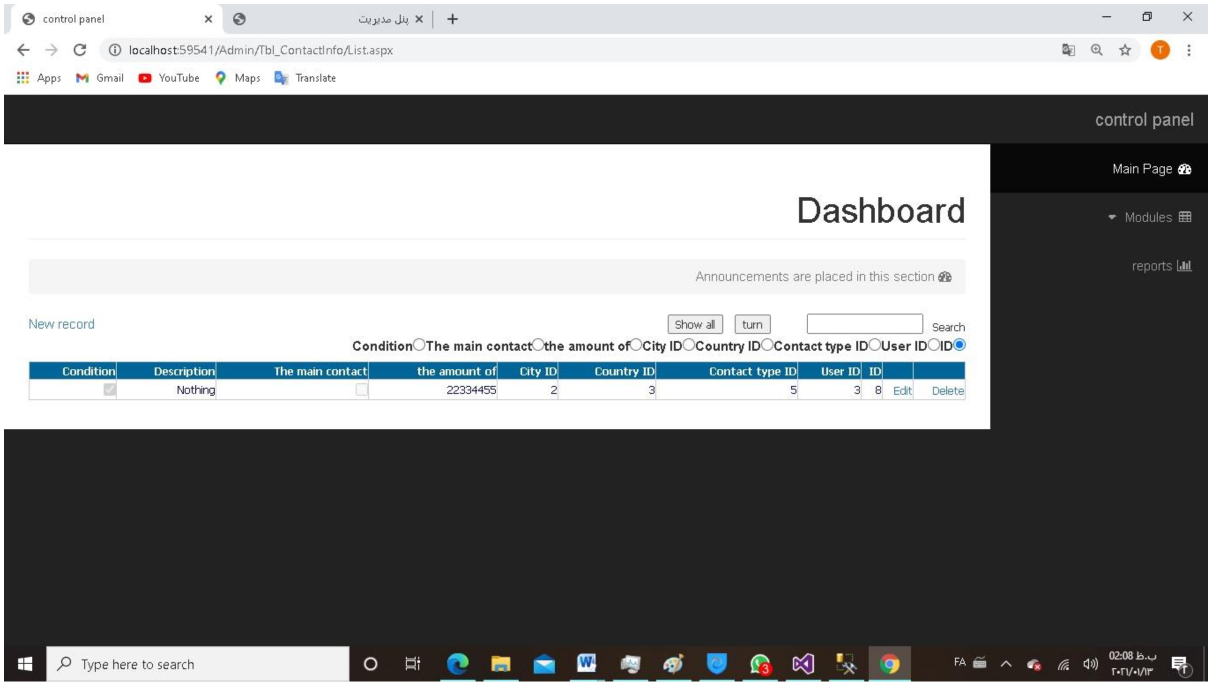
This layer, also known as the "Middle Tier", contained the main logic of the program and is responsible for the connection between the presentation layer and the data layer. In fact, all requests are created as a result of user's interaction with the presentation layer, moved to this layer and all the necessary processing is done based on the main logic of the program in this layer. The result of this processing is transferred back to the presentation layer and displayed to the user. Sometimes the user's request is such that the business logic layer needs to interact with the data layer, the bottom layer. For example, the user may want to search for a company's products by requesting the company's product list through the presentation layer. The presentation layer sends the user's request to the business logic layer and this layer also sends the user's request to the data layer because the user's request needs to communicate with the data layer. The data layer also performs the user's request and receives the list of the company's products from the database. Then send it back to the business logic layer and the business logic layer transfers this list exactly to the presentation layer. Finally, the presentation layer displays this list to the user. Nonetheless, the main task of the business logic layer is applied to the main logic of the program at the requests of users and also to establish a connection between the presentation layer and the data layer.

Presentation Layer (Display):

This layer, also called the "Interface layer", contains all the reliable visual elements related to the graphical user interface. In fact, everything that the user eventually sees from the system, such as forms, controls on forms, images, program menus, etc. are placed in this layer. The user only communicates with this layer and has no connection with other layers, and in fact transmits his request to the lower layers through the presentation layer. The task of the presentation layer is to obtain the necessary information from the user and if necessary, some validations that should be done in this layer, for example: controlling the length of each field, controlling the amount of fields that should have a value, controlling the type of data in each field based on the type of data for which the field is defined etc. It sends this information to the next layer for any other processing required and in fact, no trace of the main logic of the program and the connection to the database can be seen in this layer.

Admin panel:

We have created a panel called the "Admin Panel" for making some emergency changes, easy access and control of all parts of this web-application and database. In this panel, the administrator has control over all parts of this web-application and database tables and could edit or delete in any part of the whole database.



Dashboard

Announcements are placed in this section

New record

Show all | turn | Search

Condition ○The main contact ○the amount of ○City ID ○Country ID ○Contact type ID ○User ID ○[✓ ID]

Condition	Description	The main contact	the amount of	City ID	Country ID	Contact type ID	User ID	ID		
✓	Nothing	☐	22334455	2	3	5	3	8	Edit	Delete

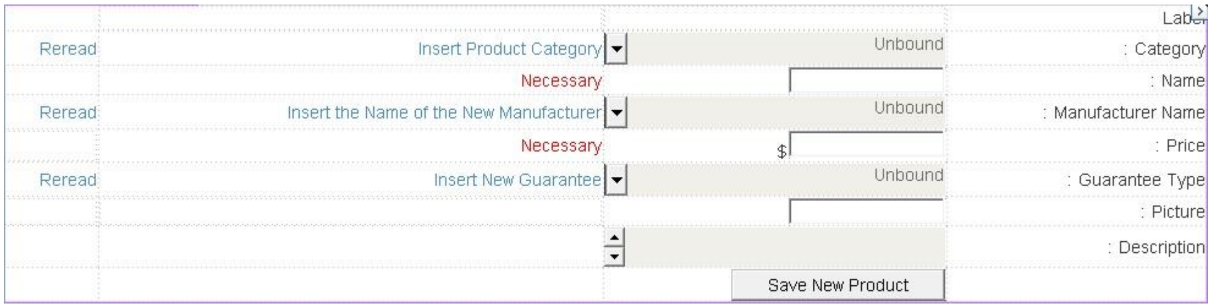
On the right side of this panel there is a menu that consists of three sections:

1- Home page and announcements
2- Modules: It contains a menu of all the tables that exist in the database. (Except for Microsoft membership and profile tables that copied from Microsoft) By selecting any of these pages, the admin connects to one of the main tables of the database and able to make emergency changes and corrections. However, considering the data type of that field and with a small change, other changes may need to be made to other parts of the database.

3- Reports: Contains statistical reports. In addition, the definition of report types and changes to the definition of reports are also done in this menu.

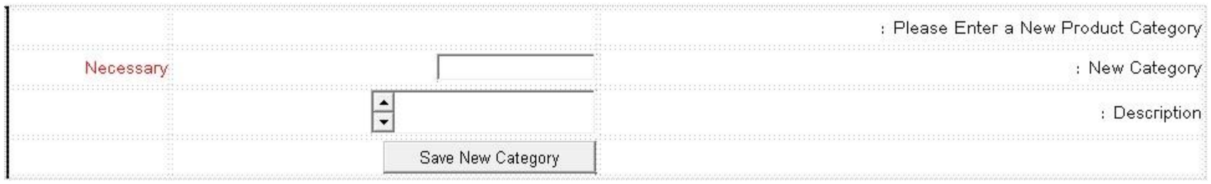
When the administrator opens a table in the module menu, all the records are visible to him. The administrator could search each table based on all the fields of a record of a specific table, change the values and use the "New Record" option, can create a new record on that table and could also delete or edit a record.

Due to the normalization of the database, we created a number of sub-tables in the database. We did this normalization in such a way that on some tables, users have access to create a new record in them based on their need. (In the next title, we will fully define normalization.) For example, the product information form is to define a new product in the system. When the user does not find the product that he wants to order in the product list, he will register that product and the following form will open for him.



			Lab
Reread	Insert Product Category	Unbound	: Category
	Necessary		: Name
Reread	Insert the Name of the New Manufacturer	Unbound	: Manufacturer Name
	Necessary	$	: Price
Reread	Insert New Guarantee	Unbound	: Guarantee Type
			: Picture
			: Description
		Save New Product	

The user enters information about the product they want, but we assume that the category related to that product is not in the list of categories. We put an option in the form called "Insert Product Category" that the user could use this option to add the desired product category to the list. As shown below:



		: Please Enter a New Product Category
Necessary		: New Category
		: Description
	Save New Category	

Database normalization:

Definition of normalization:

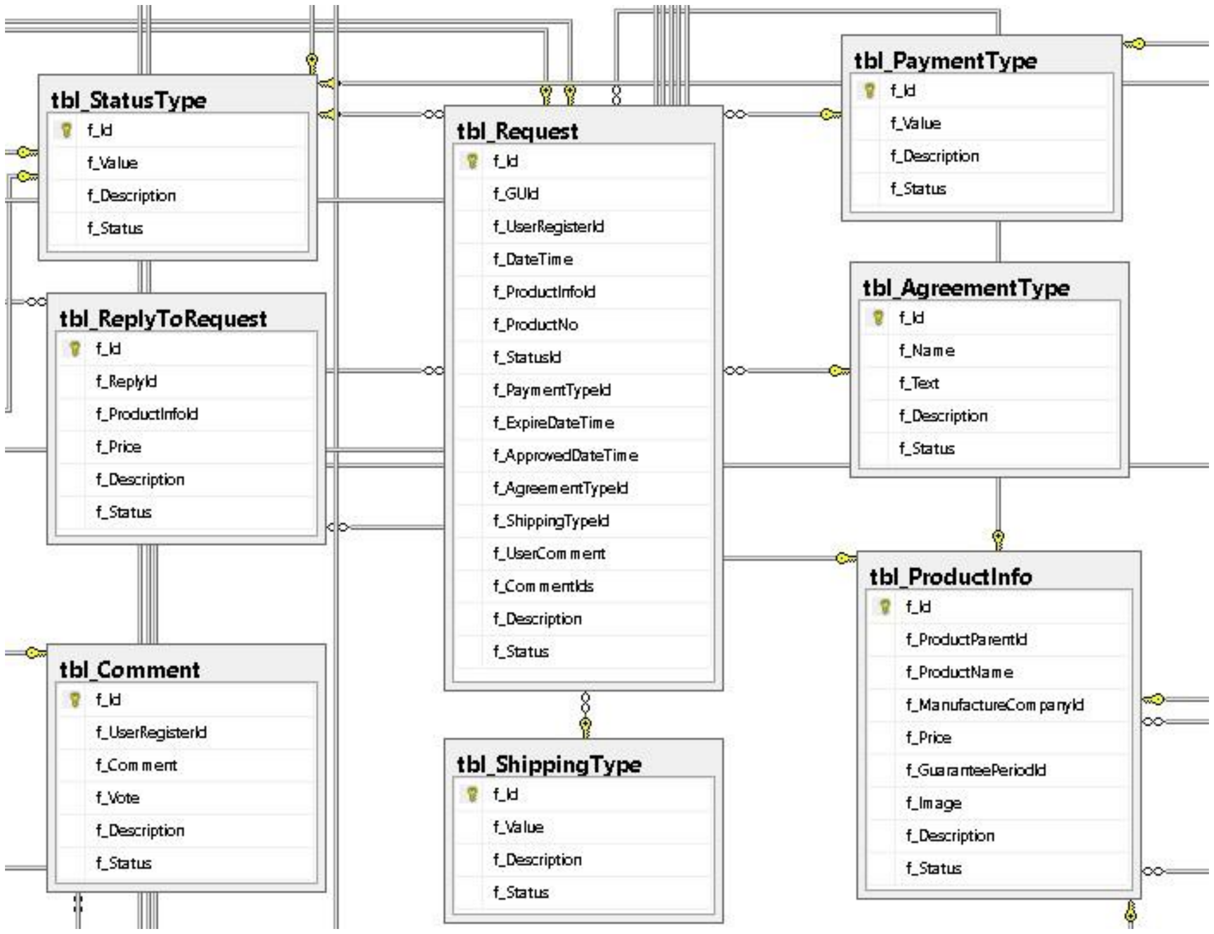
Database normalization is one of the principles of database science to eliminate redundancy. Redundancy means that a particular data is stored in several different locations in the database. This poses a potential risk that the data will conflict with each other at any time and that it will be impossible to extract the truth from it. In other words, it is a process based on which data and information are distributed in logical units called tables in a way that, in addition to maintaining the existence of data, prevents the occurrence of redundancy. For this purpose, several normal forms are defined and used. To normalize a table, it is necessary to first normalize in the first form and then check the normalization in subsequent forms. As tables are normalized, the numbers of tables in the database increase.

How to normalize this database:

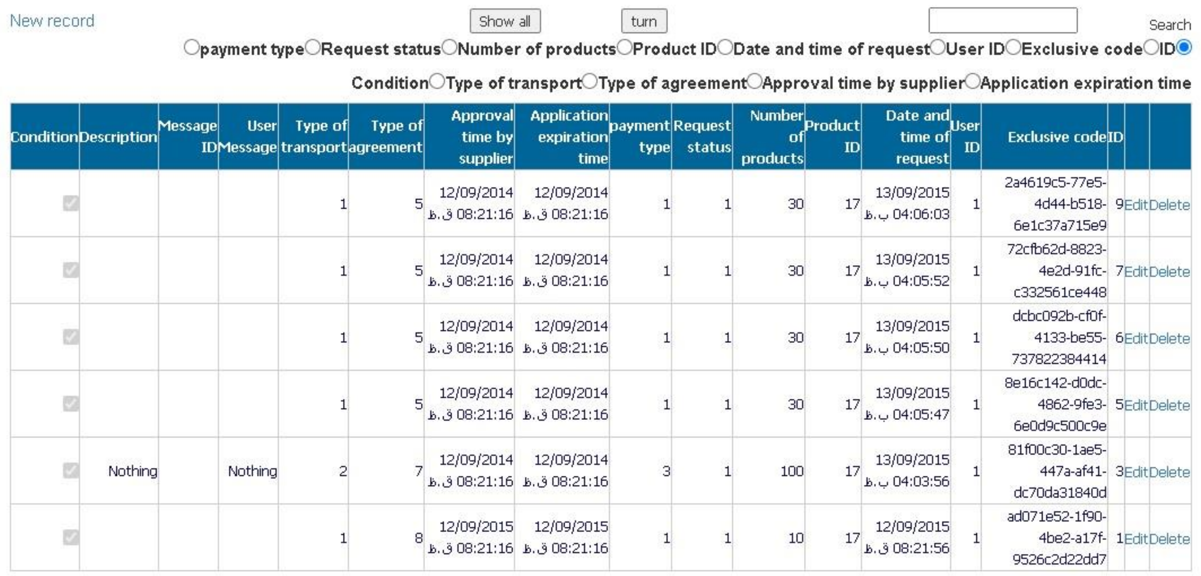
Normalization program has been done in all tables of this software to reduce the volume of the database and increase the loading speed. Another advantage of normalization is the easier and more definition of different filters to create more diverse reports with multiple filters. In this program, we examined all the tables that might suffer from information redundancy in the future and create many duplicate fields in the database and we separated those fields that were prone to redundancy as much as possible by separate tables. Instead of the original value of a field, we put the code that data has stored in the intermediate table.

Example for normalization:

For example, we used seven more tables to normalize the "Request" table. Instead of possible duplicate values, we used "ID" to communicate with other tables that contain that value. So instead of putting the value itself in each field, this value is stored in a field with a unique "ID" in another table and its "ID" is placed in the main table. These tables include information like "UserRegister", "ProductInfo", "Status", "PaymentType", "AgreementType", "ShippingType" and "Comment". Each of these values has an "ID" value (not the value itself) in the "Request" table, which "ID" is used to retrieve their information from the tables. In the image below illustrate some connections of the "Request" table and other sub-tables that have been created due to normalization.



In the image below (request form in the admin panel) we see how to value the fields that have been normalized in other tables.



payment type Request status Number of products Product ID Date and time of request User ID Exclusive code [✓ ID]
Condition Type of transport Type of agreement Approval time by supplier Application expiration time

Condition	Description	Message ID	User Message	Type of transport	Type of agreement	Approval time by supplier	Application expiration time	payment type	Request status	Number of products	Product ID	Date and time of request	User ID	Exclusive code	ID		
✓				1	5	12/09/2014 ق.ظ 08:21:16	12/09/2014 ق.ظ 08:21:16	1	1	30	17	13/09/2015 ب.ظ 04:06:03	1	2a4619c5-77e5-4d44-b518-6e1c37a715e9	9	Edit	Delete
✓				1	5	12/09/2014 ق.ظ 08:21:16	12/09/2014 ق.ظ 08:21:16	1	1	30	17	13/09/2015 ب.ظ 04:05:52	1	72cfb62d-8823-4e2d-91fc-c332561ce448	7	Edit	Delete
✓				1	5	12/09/2014 ق.ظ 08:21:16	12/09/2014 ق.ظ 08:21:16	1	1	30	17	13/09/2015 ب.ظ 04:05:50	1	dcbc092b-cf0f-4133-be55-737822384414	6	Edit	Delete
✓				1	5	12/09/2014 ق.ظ 08:21:16	12/09/2014 ق.ظ 08:21:16	1	1	30	17	13/09/2015 ب.ظ 04:05:47	1	8e16c142-d0dc-4862-9fe3-6e0d9c500c9e	5	Edit	Delete
✓	Nothing		Nothing	2	7	12/09/2014 ق.ظ 08:21:16	12/09/2014 ق.ظ 08:21:16	3	1	100	17	13/09/2015 ب.ظ 04:03:56	1	81f00c30-1ae5-447a-af41-dc70da31840d	3	Edit	Delete
✓				1	8	12/09/2015 ق.ظ 08:21:16	12/09/2015 ق.ظ 08:21:16	1	1	10	17	12/09/2015 ق.ظ 08:21:56	1	ad071e52-1f90-4be2-a17f-9526c2d22dd7	1	Edit	Delete

As you can see, in some fields, instead of real values, there are codes, each of which has a specific value in its sub-table.

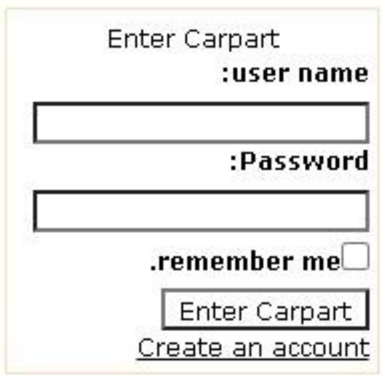
Notice: There is one more example about the normalization in the part of "How this web-application works".

Notice: At the end of this documentation, we explain the names of the tables and the meaning of each in the part of "Introducing Database's Tables".

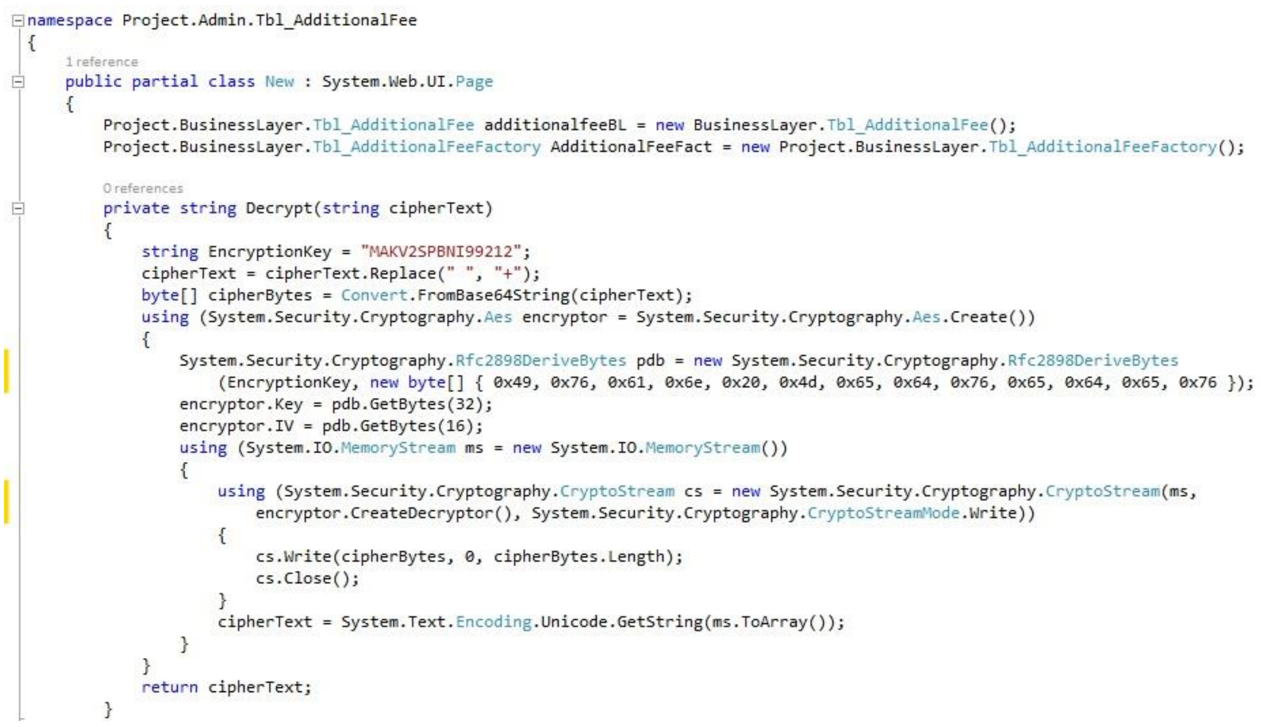
How this web-application works:

Sign in and Sign up:

First of all, by opening this web-application, asked the user about username and password. If the user is already a member of the web-application, will be able to enter to this web-application by entering the username and password. Otherwise, in the first step, the user must become a member of this web-application.

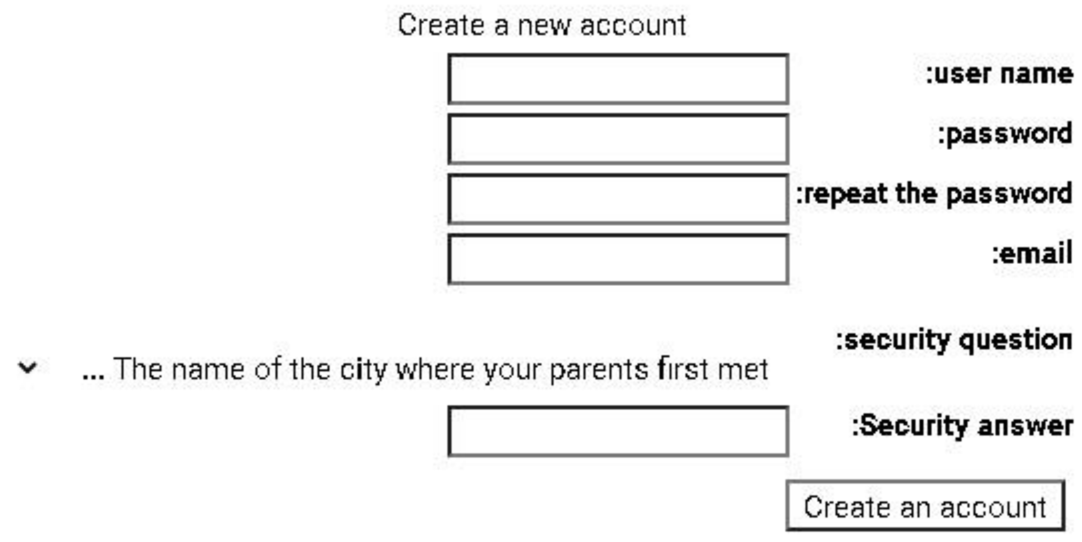


Notice: The user password and the answer to the security question are encrypted based on the key defined in the program code and stored in the “aspnet_Membership” table. Also, in the “aspnet_Membership” table, the “UserID” field is converted to a GUID according to the Microsoft standard. This operation is performed to change the “UserID” and the user password and helps us to be more secure about user data. Below is the part of the code where the encryption key is defined.



```
namespace Project.Admin.Tbl_AdditionalFee
{
    1 reference
    public partial class New : System.Web.UI.Page
    {
        Project.BusinessLayer.Tbl_AdditionalFee additionalfeeBL = new BusinessLayer.Tbl_AdditionalFee();
        Project.BusinessLayer.Tbl_AdditionalFeeFactory AdditionalFeeFact = new Project.BusinessLayer.Tbl_AdditionalFeeFactory();

        0 references
        private string Decrypt(string cipherText)
        {
            string EncryptionKey = "MAKV2SPBNI99212";
            cipherText = cipherText.Replace(" ", "+");
            byte[] cipherBytes = Convert.FromBase64String(cipherText);
            using (System.Security.Cryptography.Aes encryptor = System.Security.Cryptography.Aes.Create())
            {
                System.Security.Cryptography.Rfc2898DeriveBytes pdb = new System.Security.Cryptography.Rfc2898DeriveBytes
                    (EncryptionKey, new byte[] { 0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
                encryptor.Key = pdb.GetBytes(32);
                encryptor.IV = pdb.GetBytes(16);
                using (System.IO.MemoryStream ms = new System.IO.MemoryStream())
                {
                    using (System.Security.Cryptography.CryptoStream cs = new System.Security.Cryptography.CryptoStream(ms,
                        encryptor.CreateDecryptor(), System.Security.Cryptography.CryptoStreamMode.Write))
                    {
                        cs.Write(cipherBytes, 0, cipherBytes.Length);
                        cs.Close();
                    }
                    cipherText = System.Text.Encoding.Unicode.GetString(ms.ToArray());
                }
            }
            return cipherText;
        }
```

To become a new user, in the first step, the user must click on the "Create New Account" option. After that, enter the "Username", "Password", "Repeat Password" and "Email". The user then answers one of the predefined "security questions" so that in the future, if user forgets the password, can easily recover the password.



Create a new account

	:user name
	:password
	:repeat the password
	:email
... The name of the city where your parents first met	:security question
	:Security answer

On the next page, the user must enter his mobile number to receive the security code on the site, and after sending the security code to the user's mobile and entering it on the website, the account creation will complete.

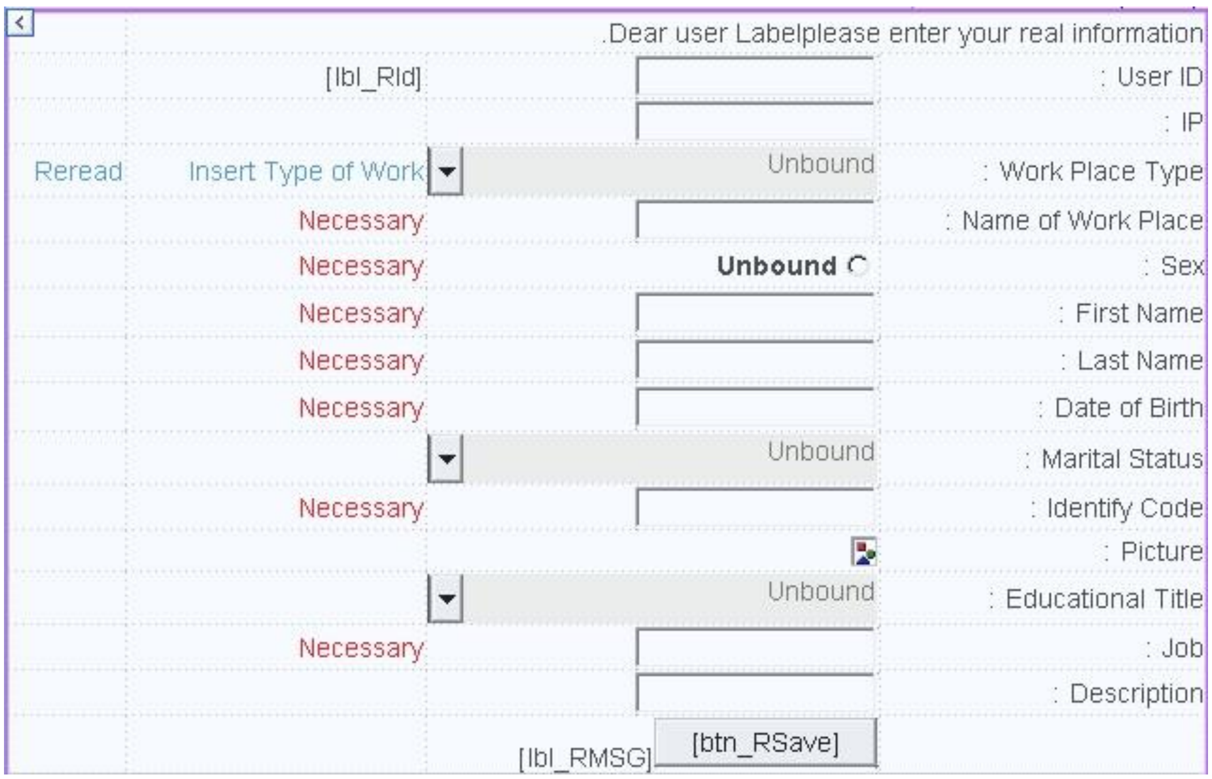
Notice: Sending a security code to the user's mobile phone and entering it by the user on the website will create more security for the website in terms of system hacking by viruses and possible trojans in the future.

Legal User or Real User Information:

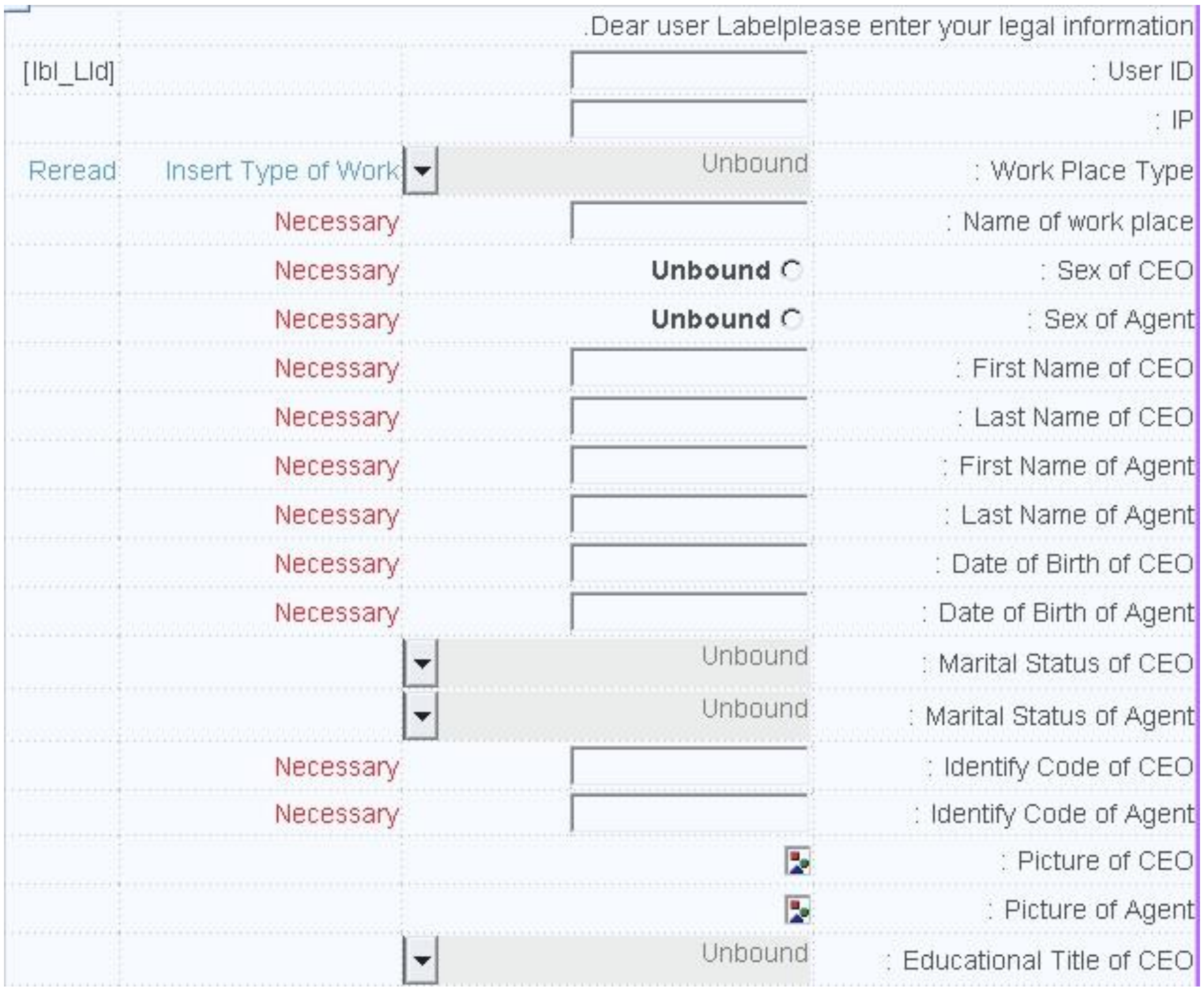
The user then chooses whether to create an account as a natural person (personal account) or as a legal user (corporate account). Then, depending on the type of account, the user enters their personal or company information.

Notice: Users of this software can act both as a buyer of spare parts and as a seller of spare parts in this web-application. We have designed this system so that, there is no difference between the users of the buyer or the seller when registering on the website. This is because in fact, many of the users who are going to work in this software are car spare parts sellers who buy their parts from the main suppliers or major sellers and sell them to minor buyers or end consumers. But for a simpler explanation of this system and for your higher understanding of how this software works, we divide users into two categories, buyers and sellers.

The fields of real or legal user information are as follows:



			.Dear user Labelplease enter your real information
	[lbl_RId]		: User ID
			: IP
Reread	Insert Type of Work	Unbound	: Work Place Type
	Necessary		: Name of Work Place
	Necessary	Unbound	: Sex
	Necessary		: First Name
	Necessary		: Last Name
	Necessary		: Date of Birth
		Unbound	: Marital Status
	Necessary		: Identify Code
			: Picture
		Unbound	: Educational Title
	Necessary		: Job
			: Description
	[lbl_RMSG]	[btn_RSave]	



			.Dear user Labelplease enter your legal information
[lbl_LId]			: User ID
			: IP
Reread	Insert Type of Work	Unbound	: Work Place Type
	Necessary		: Name of work place
	Necessary	Unbound	: Sex of CEO
	Necessary	Unbound	: Sex of Agent
	Necessary		: First Name of CEO
	Necessary		: Last Name of CEO
	Necessary		: First Name of Agent
	Necessary		: Last Name of Agent
	Necessary		: Date of Birth of CEO
	Necessary		: Date of Birth of Agent
		Unbound	: Marital Status of CEO
		Unbound	: Marital Status of Agent
	Necessary		: Identify Code of CEO
	Necessary		: Identify Code of Agent
			: Picture of CEO
			: Picture of Agent
		Unbound	: Educational Title of CEO

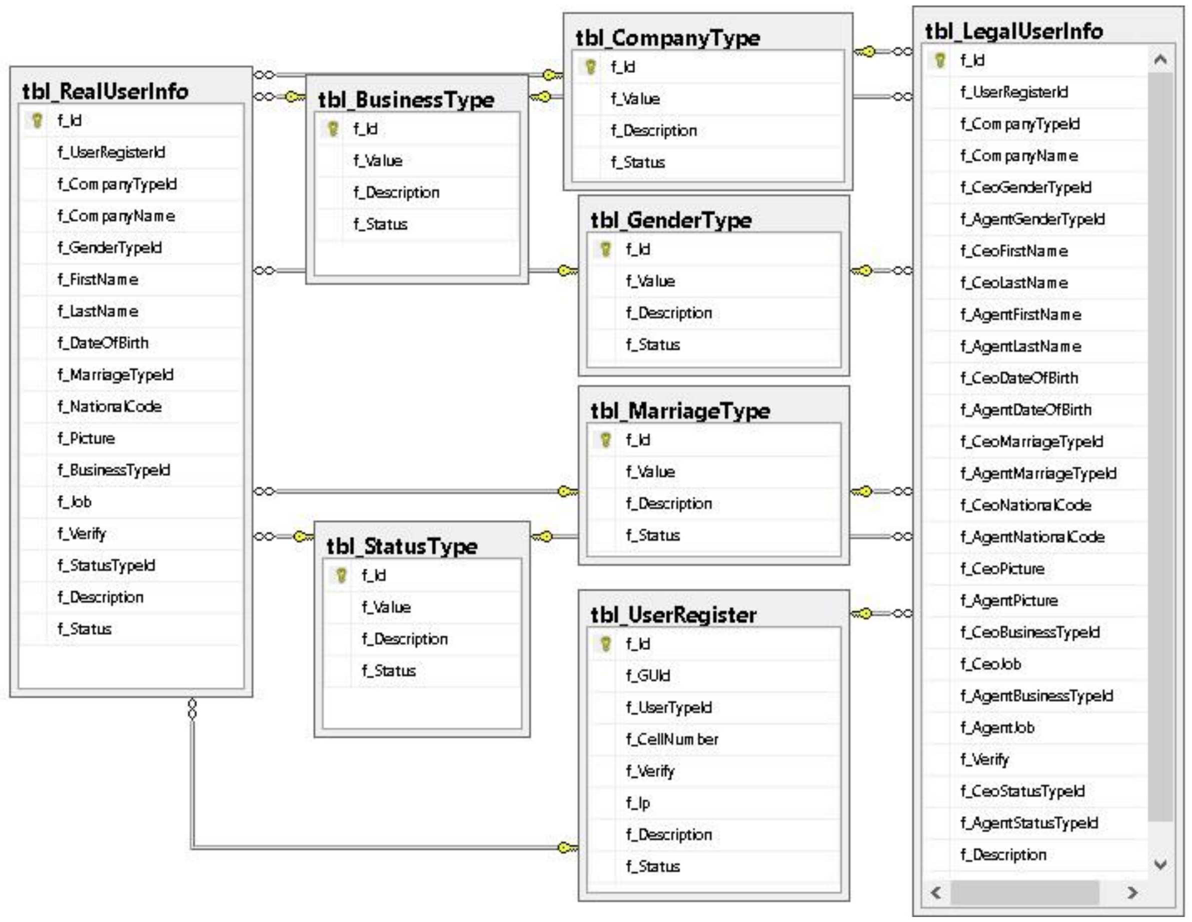
In the form of entering the information of real and legal users, for better access of users to the details, we put the insert option for the Work Place Type option, so that if the types of work were not embedded in the program, the user could have the access to add it to his personal program and profile. After adding that option, it finds it in the menu with the Reread option.

Notice: The "UserID" value is converted to a GUID code and set in the database by the "aspnet_Membership" table which is a Microsoft standard.

Notice: The IP value actually returns the computer's IP of each user automatically and stores it in the "tbl_UserRegister" table.

Notice: To better manage user's personal information and reduce the size of the database, as well as to speed up the loading of pages and applications, in the user's information table, fields of "type of workplace", "gender", "marital status", "educational title" and "user status" each by "Primary Key " respectively connects to "CompanyType", "GenderType", "MarriageType", "BusinessType" and "StatusType" tables , and only their code is displayed in the admin panel. (These values are displayed as text to the user in display mode and are displayed as code only in the admin panel.)

The following image shows the relationship of some of the sub-tables related to the registration section.

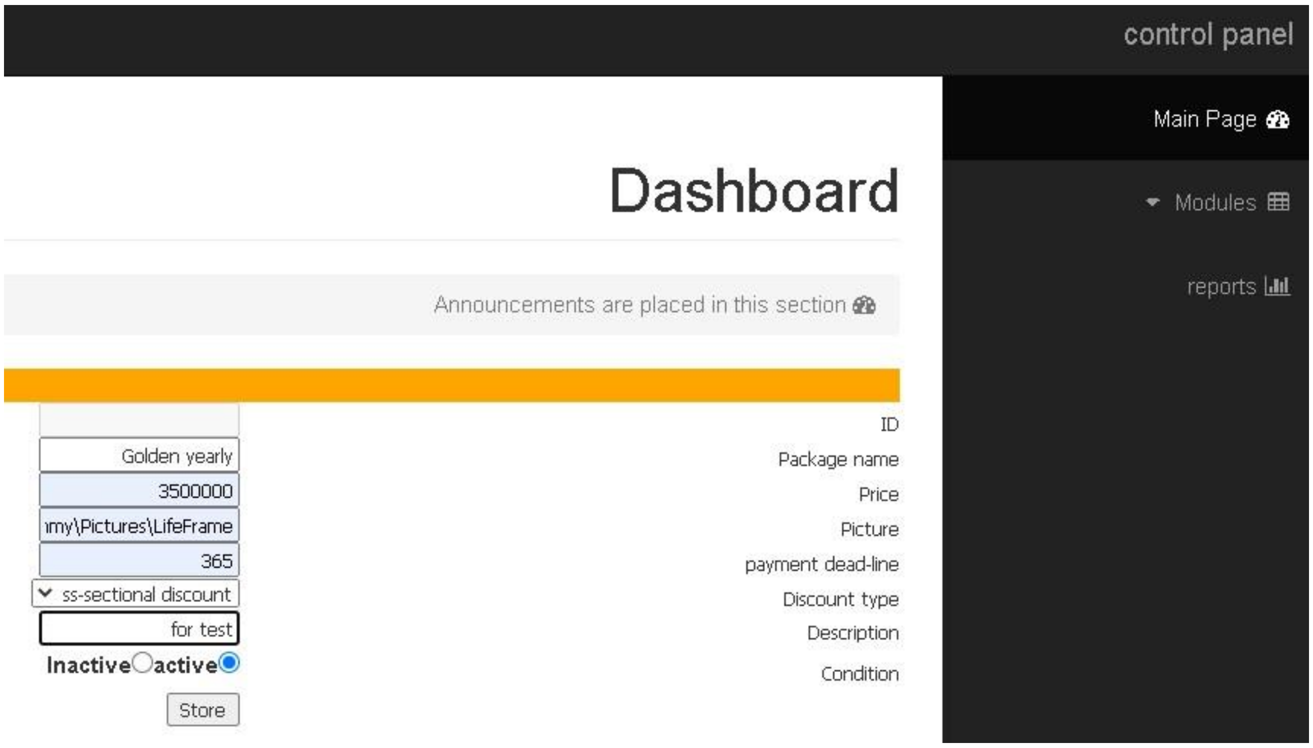


Package Type:

By package in this web-application, we mean the amount of access of buyer users to request a piece in a period of time, and sellers do not need to buy a package through this web-application to sell their products. Simply put, accessibility varies depending on the type of package that each buyer user has purchased.

Packages are provided to the user based on these parameters:

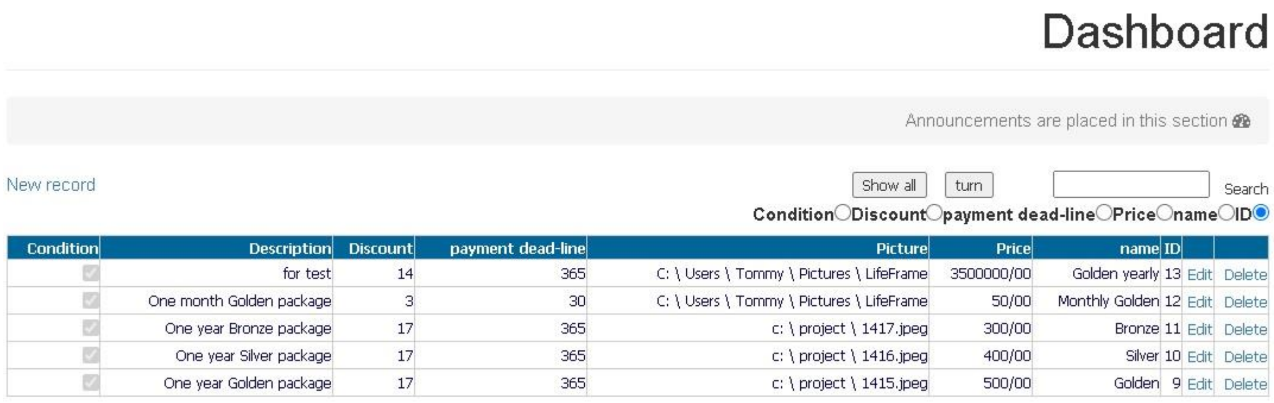
1- Based on the number of requests
2- Based on a period of time
3- Based on the number of items per each request



ID	
Package name	Golden yearly
Price	3500000
Picture	ımy\Pictures\LifeFrame
payment dead-line	365
Discount type	ss-sectional discount
Description	for test
Condition	Inactive [✓ active]

In this page, the types of packages are defined and priced based on the above parameters and will be available to the user on the order page, and the user can purchase the desired package from the website based on his needs. (Website revenue is through the sale of various packages to buyer users.)

In the image below, we see the list of defined packages in the admin panel. (All these defined packages are for example)


Dashboard
Announcements are placed in this section
New record
Show all
turn
Search
Condition Discount payment dead-line Price name [✓ ID]

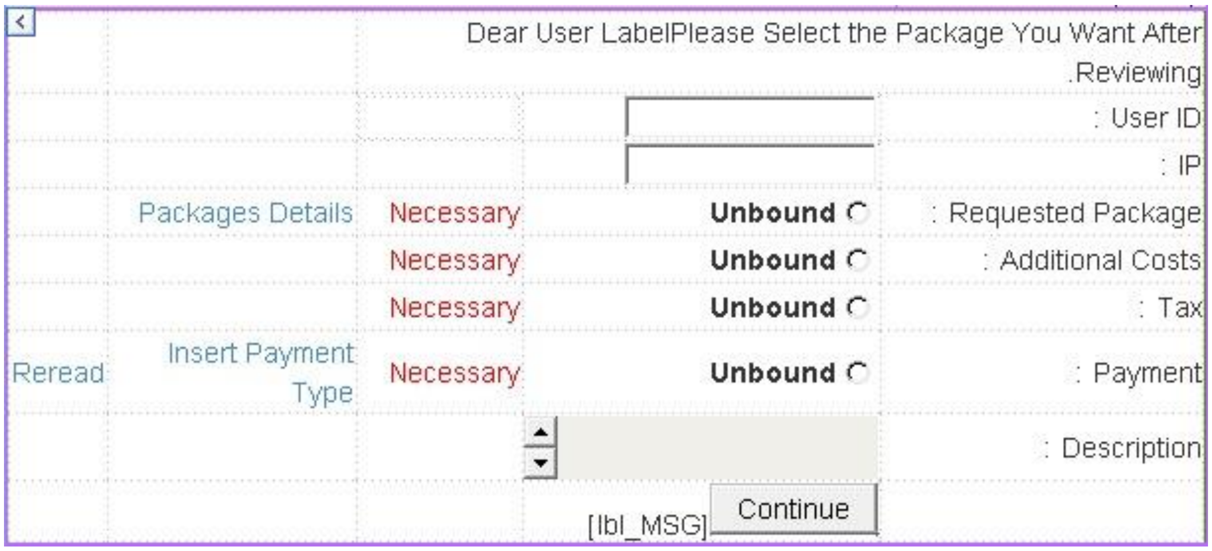
Condition	Description	Discount	payment dead-line	Picture	Price	name	ID		
✓	for test	14	365	C: \ Users \ Tommy \ Pictures \ LifeFrame	3500000/00	Golden yearly	13	Edit	Delete
✓	One month Golden package	3	30	C: \ Users \ Tommy \ Pictures \ LifeFrame	50/00	Monthly Golden	12	Edit	Delete
✓	One year Bronze package	17	365	c: \ project \ 1417.jpeg	300/00	Bronze	11	Edit	Delete
✓	One year Silver package	17	365	c: \ project \ 1416.jpeg	400/00	Silver	10	Edit	Delete
✓	One year Golden package	17	365	c: \ project \ 1415.jpeg	500/00	Golden	9	Edit	Delete

Notice: In the package definition table, the value of the discount field is the ID that is connected to the Discount table and the discounts are defined there and its ID is returned to this table.

Notice: These packages are defined and are available to the user when the software is fully implemented and located on the domain and Introduced to the public.

Order:

In this section, the user selects the type of package he wants based on the amount of parts purchased per month or year, and according to the type of package and its duration, pays a fee for using the services on the website.



					Dear User LabelPlease Select the Package You Want After .Reviewing
					: User ID
					: IP
	Packages Details	Necessary	Unbound	◌	: Requested Package
		Necessary	Unbound	◌	: Additional Costs
		Necessary	Unbound	◌	: Tax
Reread	Insert Payment Type	Necessary	Unbound	◌	: Payment
					: Description
			[lbl_MSG]	Continue	

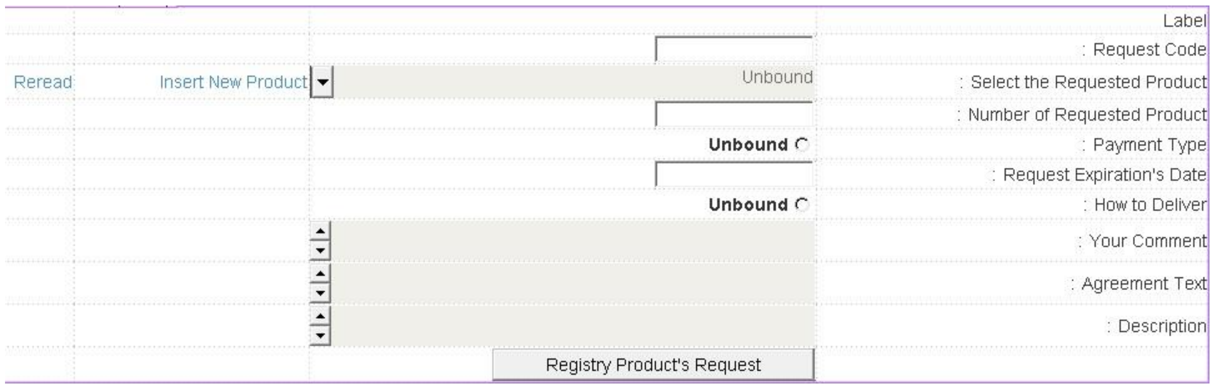
The "User ID" and "IP" of the user's computer are displayed automatically. After selecting the type of package by the user in the section "Requested packages", "Additional Costs" and "Tax" are automatically calculated and displayed. The user then determines their payment type. We have considered online and cash payments for users to pay.

If the user chooses the type of cash payment, an additional amount will be entered in the field of "Additional Costs", which is for the postage that he must go to the user's location and receive the amount from the user.

Notice: If the user selects the type of online payment, the next page displayed to the user is the bank's online payment page, which has not yet been designed. (The design of this page is required to have an electronic payment gateway of the bank, which must be requested from the relevant bank to be created by the bank.)

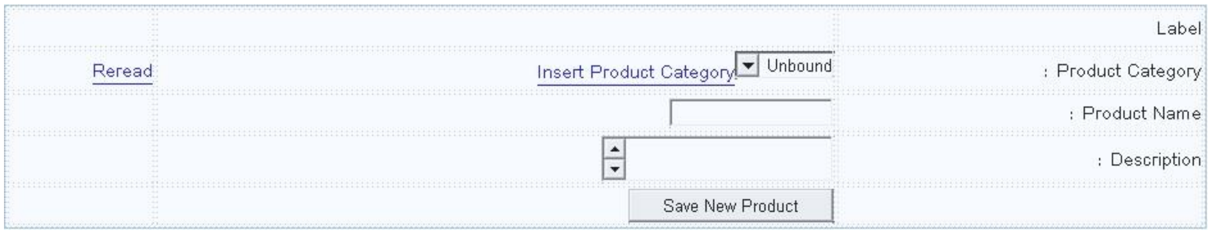
Request:

After registering the user's personal information and selecting the type of package and paying the package amount, the user can register a request in the new request section, regarding his needs to purchase different types of spare parts. In this section, the user enters the desired information in the form in the following order to request the purchase of a specific spare part. Select the category of the spare part, then select the name of the desired spare part and the required number and choose how to pay the purchase amount and how to send it. The "Request Code" field is automatically set and each request has a unique code. In the "Your Comment" field, the buyer user can write a message and the seller user can see it when accepting this request. The "Agreement Text" field is a pre-written text for accepting website policies and online sales policies and must be approved by the user before submitting a request. Finally, the buyer user presses the request registration button and waits for the answers that the seller users will give.



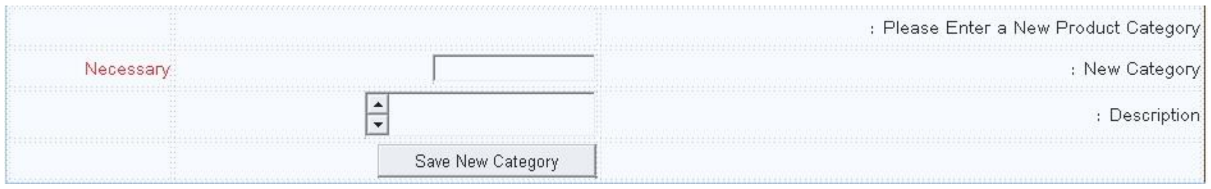
			Label
			: Request Code
Reread	Insert New Product	Unbound	: Select the Requested Product
			: Number of Requested Product
		Unbound	: Payment Type
			: Request Expiration's Date
		Unbound	: How to Deliver
			: Your Comment
			: Agreement Text
			: Description
		Registry Product's Request	

As we know, in the product request field, if the product name does not exist in the product list, there is an option for the user to add the product name. The user clicks on it, then the following form appears to add the product.



			Label
Reread	Insert Product Category	Unbound	: Product Category
			: Product Name
			: Description
		Save New Product	

The user can add it to the product list by selecting the product category and then typing the product name. Then tap the reread option to see the product name in the product list. If in the product category list, the desired category name of that product did not exist, the user can click on add category name, add it to the category list via the following form.

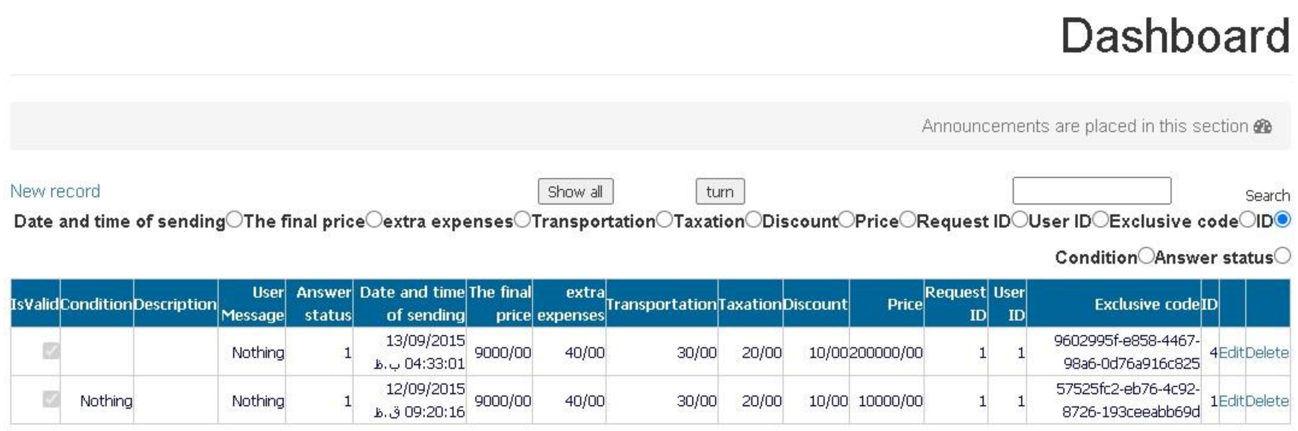


		: Please Enter a New Product Category
Necessary		: New Category
		: Description
	Save New Category	

In this section, the user adds the product category name and then presses the save option, adds it to the category list.

Reply:

After the buyer user submits his purchase request, this request will be sent to all seller users, including suppliers, manufacturers and minor sellers, and seller users will see this request on their home page. After reviewing the available requests by seller users, they could select the requests that they want to cooperate with and then respond to those requests through the "Reply" form.


Dashboard

Announcements are placed in this section

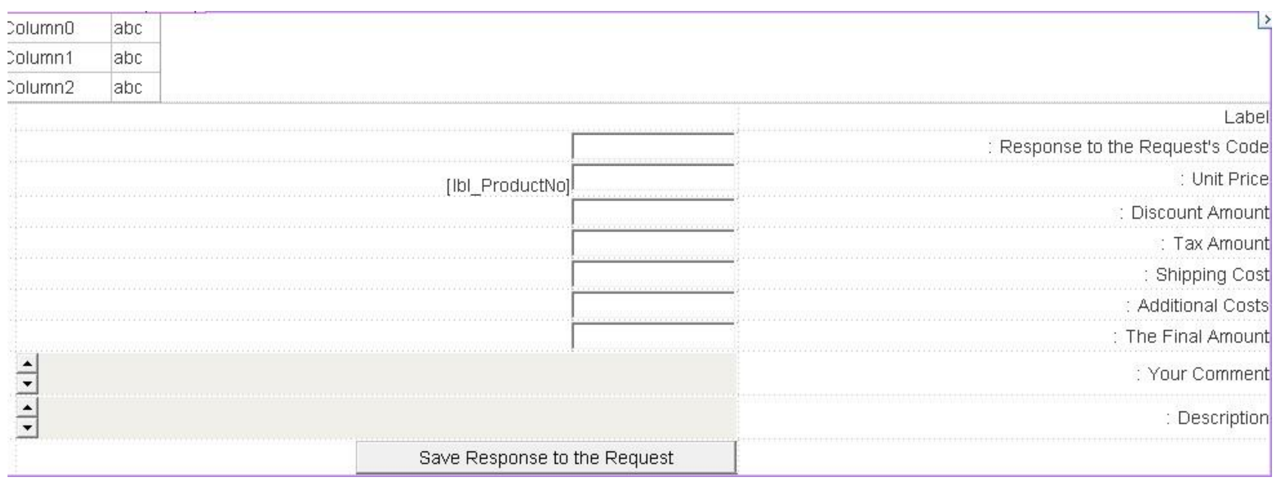
New record | Show all | turn | Search

Date and time of sending ◯ The final price ◯ extra expenses ◯ Transportation ◯ Taxation ◯ Discount ◯ Price ◯ Request ID ◯ User ID ◯ Exclusive code ◯ [✓ ID]

Condition ◯ Answer status ◯

IsValid	Condition	Description	User Message	Answer status	Date and time of sending	The final price	extra expenses	Transportation	Taxation	Discount	Price	Request ID	User ID	Exclusive code	ID		
✓			Nothing	1	13/09/2015 ب.ظ 04:33:01	9000/00	40/00	30/00	20/00	10/00	200000/00	1	1	9602995f-e858-4467-98a6-0d76a916c825	4	Edit	Delete
✓	Nothing		Nothing	1	12/09/2015 ق.ظ 09:20:16	9000/00	40/00	30/00	20/00	10/00	10000/00	1	1	57525fc2-eb76-4c92-8726-193ceeabb69d	1	Edit	Delete

The seller user sees the list of requests of the buyer users at the top of the form and can respond to that request by selecting any of them.



Column0	abc
Column1	abc
Column2	abc

	Label
	: Response to the Request's Code
[lbl_ProductNo]	: Unit Price
	: Discount Amount
	: Tax Amount
	: Shipping Cost
	: Additional Costs
	: The Final Amount
	: Your Comment
	: Description

Save Response to the Request

Notice: In the "Response to the Request's Code" field, a unique code is automatically given to this response. In this web-application, we have designed a separate table called "Reply to Request" table to manage the connections between the "Request" and "Reply" tables.

In the next field, the seller enters the price of product. (The name of the product is displayed in front of the field of product's price.) Then, if the seller user has a discount for the traded product, he enters it in the discount amount field. (In this field, the total amount of the discount is entered. By using the "Exceptions" required in the presentation layer, we prevent the entry of a discount percentage or a discount amount in relation to the price per product.) After that, the seller user sets the tax fields, shipping costs and additional costs as needed.

Notice: The tax, shipping, and additional costs fields could be "null" meaning they can be underestimated and left blank.

The values of the "Unit Price", "Discount Amount", "Tax Amount", "Shipping Costs" and "Additional Costs" fields are then automatically added together and the final value is poured into the "Final Amount" field. Then, if the seller user has a message for the buyer user, type it in the "Your Comment" field and finally click the "Save Response to the Request" button.

Reply to Request:

After the reply to the request has been reviewed by the buyer user, the buyer user confirms one of those reply's to the request based on price per product, additional costs and delivery time of the product by the seller user. This means that a transaction has been made between the buyer user and the seller user. In this type of transaction, out of the large number of different responses given to the buyer user's request from different seller users, only one of these requests is selected by the buyer user, which in the buyer's opinion was the best response to his request.

Introducing Database's Tables:

Tbl_AdditionalFee: In the table of “AdditionalFee”, each item includes a cost other than the cost of purchasing the package, cost of purchasing package's taxes and cost of shipping costs of purchasing package. These costs are saved based on the amount or percentage, such as customs clearance costs, transportation costs, additional freight costs, etc.

Tbl_AgreementType: In the table of “AgreementType”, Types of agreements related to the transactions or privacy policies is stored.

Tbl_Attribute: In the table of “Attribute”, all types of attributes that each spare part may have will be placed. Such as, the color of the spare part, the life of the spare part, the car model that belongs to that spare part, the material of the spare part, the weight of the spare part, etc.

Tbl_BusinessType: In the table of “BusinessType”, all job titles are included. Such as engineer, shopkeeper, wholesaler, supplier, consumer, etc.

Tbl_City: In the table of “City”, the name of each city that may be used to determine the position of buyer and seller is placed. One of the uses of this table could be filtering buyers and sellers by city. Another use is to define the city name to set the phone code of each city.

Tbl_Comment: In the table of “Comment”, all user comments will be placed. We can filter or display comments related to each section. Another use is to determine which users voted for which comments and how many votes each comment received.

Tbl_CommentReply: In the table of "CommentReply", all the answers that are given to each specific comment along with the votes of that answer are placed. This way we can evaluate the feedback of each comment. We can also use valuable comments and responses to improve the system process.

Tbl_CommentVote: In the table of "CommentVote", votes are defined and stored based on five stars (five levels of points).

Tbl_CompanyType: In the table of "CompanyType", the location of jobs and its type are specified. Such as factory, workshop, organization, website, etc.

Tbl_ContactInfo: In the table of "ContactInfo", the type of number and contact number of each user is stored along with the telephone code of that user's city and country (The telephone code of each city and country has already been normalized in the relevant tables).

Tbl_ContactType: In the table of "ContactType", different types of calls such as mobile number, company number, email, etc. are stored.

Tbl_Country: In the table of "Country", the names of all the countries used in the whole program are stored. Like the names of the countries that make a specific spare part.

Tbl_Discount: In the table of "Discount", all discounts used in the program are saved. Including percentage of discounts, start and end dates of discounts and name of discounts.

Tbl_GenderType: The table of "GenderType", stores the gender types and names used when addressing users. Such as female, male, Mr., Miss, etc.

Tbl_Guarantee: In the table of "Guarantee", the types of spare parts guarantee are defined using the guarantee name and its validity period.

Tbl_LegalUserInfo: In the table of "LegalUserInfo", all the personal details of the legal users are stored along with their photos and job titles.

Tbl_ManufactureCompany: In the table of "ManufactureCompany", the names of the spare parts manufacturers and their country of production are listed.

Tbl_MarriageType: In the table of "MarriageType", the types of marriage statuses such as married, single, divorced, etc. are defined and stored.

Tbl_Order: In the table of "Order", it is specified what kind of package each user uses and what date that package expires. Also, the information of all user expenses and payments is stored in this table.

Tbl_Package: In the table of "Package", the information of the defined package types such as package name, package validity period, and price or discount type specified for that package is stored.

Tbl_PaymentType: In the table of "PaymentType", types of payments such as cash payment, online payment, bank's cheque, etc. are stored.

Tbl_PriceHistory: In the table of “PriceHistory”, the price of each product is stored relative to the date. Report of price change of each specific spare part or inflation percentage of each spare part in relation to a historical period is one of the applications of this table.

Tbl_ProductAttribute: In the table of “ProductAttribute”, the ratio of each attribute to each spare part is stored by "ID". In fact, this table is how to create a connection between two tables, "attribute" and "product information", which are connected by unique "IDs". For example, it shows that a spare part with an "ID" has several "IDs" attributes, that each of those "IDs" represents a specific attribute.

Tbl_ProductInfo: In the table of “ProductInfo”, the information of each spare part such as category, product name, manufacturer, price, warranty and photo are stored.

Tbl_ProductParent: In the table of “ProductParent”, different categories are stored. These categories are embedded to make it easier for users to search between spare parts and organize parts. In this way, each spare part is placed in a specific category and all categories are stored in this table.

Tbl_RealUserInfo: In the table of “RealUserInfo”, all the real user's personal details are stored along with their photos and job titles.

Tbl_Reply: In the table of “Reply”, the information of the users who have answered a request is recorded. Such as user ID, request ID, The amount offered to that request, discount amounts, taxes, shipping and additional costs, final price, shipping date and time, response status and message from the seller user to the request of the requesting user.

Tbl_ReplytoRequest: In the table of “ReplytoRequest”, when the buyer user creates a request and the seller users respond to that request, the buyer user is only able to select one of the sellers' answers that he intends to provide. This selection is saved as a transaction resume in the response table to the request, which includes fields such as "response ID", "ID", product name and total price.

Tbl_Request: In the table of “Request”, there are fields such as requesting user information, product information, requested product number, product status, type of payment, date and time of request and its expiration, type of agreement, type of shipping and user comment.

Tbl_SecurityQuestion: In the table of “SecurityQuestion”, stores the security questions that users will be asked when registering or forgetting a password.

Tbl_ShippingType: In the table of “ShippingType”, the available types of transport are stored. Such as air freight, postal shipping, free shipping and etc.

Tbl_StatusType: In the table of “StatusType”, the types of situations that can be created throughout the web-application in different conditions for different values are defined and stored.

Tbl_TaxType: In the table of “TaxType”, the types of taxes that may be levied on users in the country are defined and stored based on the name of the tax and the percentage it adds to the transaction amount and the date that the tax law was enacted.

Tbl_Transaction: In the table of "Transaction", the information of each transaction created to purchase the package is stored. Information such as transaction ID, transaction time and date, transaction result, transaction amount and etc.

Tbl_UserInfoPackage: In the table of "UserInfoPackage", when users purchase a package, their user ID is stored along with their package ID on this table. In fact, this table is for effective communication between the two tables of the user information and the package, in which it shows Each "ID" of the user belongs to which "ID" of the package.

Tbl_UserRegister: : In the table of "UserRegister", information such as "GUID" (Globally Unique Identifier), user type, user mobile number, user status and user's computer IP are stored.

Tbl_UserType: In the table of "UserType", user types such as seller, supplier, buyer, etc. are defined and stored.

Web-application progress level:

So far, we have done most of the programming and designing of backend parts of this web-application, such as implementing web-application logic, web-application communication with the database, database designing, etc. thus, user interface design, definition of exceptions in the presentation layer and definition of statistical reports that will be completed in the future.

Conclusion:

According to these principles and systemic process of this web-application, Buyer users can choose the best price based on product quality from all their requests from all seller users who have responded to the buyer user and with the lowest market price, buy their spare parts in any number from all available sellers. Firstly this policy helps buyers to have many choices to buy. Secondly, buy the product they want at the best market price. It also creates a competitive market between sellers and the sellers who sell their products at the lowest price will have the highest sales and always have a customer to sell their products. These reasons are a document for the disappearance of product brokerage and the possible connections between the largest manufacturers and suppliers between minor buyers.

We have designed this software so that if we need to apply this sales policy to products other than auto spare parts as needed in the future. We have designed this software in such a way that if we want to apply this sales policy for products other than car spare parts according to future needs, by applying minor changes in different parts of this software and redesigning the User Interface, we can implement this software to trade in relation to any other product and turn that product market into a competitive market.