CODE STRUCTURE

A SIMPLE DIVE IN

- ✓ To create The Agro- Industrial Enterprise Management System Application we have used Windows Forms .Net Framework Technology.
- ✓ We have used SQL Server database to perform Data based Operations.
- ✓ This application Consists with 10 Forms.
    1. The Login Page.
    2. Home Page.
    3. Inventory Management
    4. Sales And Marketing
    5. Crop Management
    6. Supply chain Management.
    7. Financial Analysis
    8. Employee Management
    9. Quality Control
    10. Equipment Maintenance
- ✓ Our vision was here to create an responsible application to manage an entire Industrial Enterprise Empire through single application.
- ✓ So Above Each Form focuses on managing each part that is department of the company.
- ✓ And also in Database we had to create tables for each of those application in order to manage and manipulate data clearly and easily.

CODING STRUCTURE

- ✓ Basic method and event handling structure for the code of each form is as follows-

1. Name space for Sql database classes.
2. Class for the Form
3. Constructor to Initialize Objects with Class components.
4. Sql Connection Initialization.
5. Method for Data Grid View and Database connection Initialization.
6. Method for Resetting Components.
7. Event Handler for Save button click event.
8. Event Handler for Edit button click event.
9. Event Handler for Delete button click event.
10. Event Handler for Data Grid View Cell Content click event.
11. Event Handlers for Label Click for transition through the Forms.
12. Logout Sequence.

1.NAME SPACE FOR SQL DATABASE CLASSES.

- ✓ Using System.Data.sqlClient ;

2.CLASS AND CONSTRUCTOR

- ✓ Each form has a class and a constructor named after itself.
- ✓ This is for the creation of objects of that class.

3.SQL CONNECTION

- ✓

```
 SqlConnection Con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=""F:\favo\New folder44\New
folder\Agr-Industrial Enterprise Management System.mdf"";Integrated
Security=True;Connect Timeout=30");
```

- ✓
- ✓ This Initialize the connection between the backend and the database. We have used the connection string inside of the Sqlconnection class and have named it as “Con” so we can Manipulate connections through each private fields easily.

4.DGV DATA SHOWING METHOD

- ✓ Through this method we intend to show data base data in the user interface. So user in our case the company can directly see and analyze data.

```
 private void showinventory()
{
    Con.Open();
    String Query = "Select * from [Items Table]";
    SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
    SqlCommandBuilder builder= new SqlCommandBuilder(sda);
    var ds= new DataSet();
    sda.Fill(ds);
    InventoryDGV.DataSource = ds.Tables[0];
    Con.Close();
     }
```

- ✓ We Have used Sql query to select all the Data rows of the data table here.
- ✓ Then we have create an Sql Data Adapter in order to Extract data from data base towards the backend.
- ✓ Then We have created a Data set to fill data that extracted with the SDA.
- ✓ Then we set the Datasource of the Data Grid view to That Dataset.

5.RESET METHOD

```
private void Reset()
 {
     ItemIDTextBox.Text = "";
     ItemNameTextBox.Text = "";
     DescriptionTextBox.Text = "";
```

```
    SupplierInformationTextBox.Text = "";
    PurchasePriceTextBox.Text = "";
    QuantityOnHandTextBox.Text = "";
    UnitOfMeasureTextBox.Text = "";
    StorageLocationTextBox.Text = "";
    DateOfPurchaseDateTimePicker.Value = DateTimePicker.MinimumDateTime;
  ✓   }
```

- ✓ The reset method will be later called on the other parts of the code.
- ✓ In this we have Set User interface Data input fields to their empty fields.

6.EVENT HANDLER FOR SAVE,EDIT,DELETE BUTTON CLICK

```
private void SaveBtn_Click(object sender, EventArgs e)
{
    if(ItemIDTextBox.Text==""|| ItemNameTextBox.Text==""||
DescriptionTextBox.Text==""|| SupplierInformationTextBox.Text==""||
PurchasePriceTextBox.Text==""|| QuantityOnHandTextBox.Text==""||
UnitOfMeasureTextBox.Text==""|| StorageLocationTextBox.Text == "" ||
DateOfPurchaseDateTimePicker.Value == DateTimePicker.MinimumDateTime)
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
        Con.Open();
            SqlCommand cmd = new SqlCommand("Insert into [Items
Table](ItemID,ItemName, Description, QuantityOnHand, UnitOfMeasure, ISupplierID,
PurchasePrice, DateOfPurchase, StorageLocation) values(@ItemID,@ItemName,
@Description, @QuantityOnHand, @UnitOfMeasure, @ISupplierID, @PurchasePrice,
@DateOfPurchase, @StorageLocation)", Con);
            cmd.Parameters.AddWithValue("@ItemID", ItemIDTextBox.Text);
            cmd.Parameters.AddWithValue("@ItemName", ItemNameTextBox.Text);
            cmd.Parameters.AddWithValue("@Description", DescriptionTextBox.Text);
            cmd.Parameters.AddWithValue("@QuantityOnHand",
QuantityOnHandTextBox.Text);
            cmd.Parameters.AddWithValue("@UnitOfMeasure",
UnitOfMeasureTextBox.Text);
            cmd.Parameters.AddWithValue("@PurchasePrice",
PurchasePriceTextBox.Text);
            cmd.Parameters.AddWithValue("@DateOfPurchase",
DateOfPurchaseDateTimePicker.Value);
            cmd.Parameters.AddWithValue("@StorageLocation",
StorageLocationTextBox.Text);
            cmd.Parameters.AddWithValue("@ISupplierID",
SupplierInformationTextBox.Text);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Data Added");
            Con.Close();
            showinventory();
            Reset();
        }
```

```
        catch(Exception ex)
        {
        MessageBox.Show(ex.Message);
        }
    }
✓   }
```

- ✓ Through this we have used an if statement to check if all the fields of the form interface are filled. We have connected each input fields with OR statements in order to check that even single component is not empty .
- ✓ If not we tend to show a message as missing information.
- ✓ Inside of the else statement we have used an exception handling.
- ✓ Inside of the try block we have given the code that may rises an exception.
- ✓ Inside of the try block we have given sql command to insert, update or delete which depends on the event.
- ✓ In the query we have given the purpose then which data table then Data Table Fields after that all the placeholders in order to insert user data input fields.
- ✓ Then we have initialize each parameter or placeholder with the respective data input fields one by one.
- ✓ As we create the command now we execute it.
- ✓ Then we show a message box about the activity that have done.
- ✓ Then we call the datagrid view show method to update the DGV with respective updated values.
- ✓ Then we call the Reset method to set data input fields back into their empty states in order to enter new values.
- ✓ After all that Inside catch block set error exception message to show up on a message box if there is any error in the code.

7.DGV CELL CONTENT CLICK EVENT HANDLER

```
private void InventoryDGV_CellContentClick_1(object sender,
DataGridViewCellEventArgs e)
 {
     if (e.RowIndex >= 0 && e.ColumnIndex >= 0)
     {
         // Assuming SupplierID is stored in the first column (index 0) of the
DataGridView
         string IItemID = InventoryDGV.Rows[e.RowIndex].Cells[0].Value.ToString();
         string ItemName = InventoryDGV.Rows[e.RowIndex].Cells[1].Value.ToString();
         string Description =
InventoryDGV.Rows[e.RowIndex].Cells[2].Value.ToString();
         string QuantityOnHand =
InventoryDGV.Rows[e.RowIndex].Cells[3].Value.ToString();
         string UnitOfMeasure =
InventoryDGV.Rows[e.RowIndex].Cells[4].Value.ToString();
         string ISupplierID =
InventoryDGV.Rows[e.RowIndex].Cells[5].Value.ToString();
         string PurchasePrice =
InventoryDGV.Rows[e.RowIndex].Cells[6].Value.ToString();
```

```
        string DateOfPurchase =
InventoryDGV.Rows[e.RowIndex].Cells[7].Value.ToString();
        string StorageLocation =
InventoryDGV.Rows[e.RowIndex].Cells[8].Value.ToString();
        string ItemID = InventoryDGV.Rows[e.RowIndex].Cells[9].Value.ToString();

        ItemIDTextBox.Text = ItemID;
        ItemNameTextBox.Text = ItemName;
        DescriptionTextBox.Text = Description;
        QuantityOnHandTextBox.Text = QuantityOnHand;
        UnitOfMeasureTextBox.Text = UnitOfMeasure;
        SupplierInformationTextBox.Text = ISupplierID;
        PurchasePriceTextBox.Text = PurchasePrice;
        DateOfPurchaseDateTimePicker.Value = DateTime.Parse(DateOfPurchase);
        StorageLocationTextBox.Text = StorageLocation;
        primaryKeyTextBox.Text = IItemID;




    }
  ✓  }
```

- ✓ We have used event arguments to find where and which part of the interface specifically that raised the event.
- ✓ Then we have used an if statement to check if the cell that clicked by the user is not a header cell.
- ✓ We have created variables to assign each cell data inside the Data grid view back to their respective string values so we can set those data inputs back into their respective input fields so editing and deleting data process can be done.
- ✓ Also we have used a read only field to be updated the primary key. Which shows the count of the data entries.
- ✓ With that primary key we can access the entire data grid view row to be reloaded into the fields.

8.NEVIGATION PANEL FORMS TRANSITION

```
private void label8_Click(object sender, EventArgs e)
 {
     EmployeeManagement obj = new EmployeeManagement();
     obj.Show();
     this.Hide();
  ✓   }
```

- ✓ In this event we have assign an entire form to an object named as obj.
- ✓ So we can simply call Show method through that object to show that form and hide method to hide the current form.

9.LOGOUT SEQUENCES

```
private void label9_Click(object sender, EventArgs e)
{
    Login obj=new Login();
    obj.Show();
    this.Hide();
   ✓ }
```

- ✓ Same method as above. But as it is Logout event we have shown the path back to the login form hiding any form performing a logging out sequence.

HOME PAGE

- ✓ In the Home page we have used count and Sum queries to find the sum of salaries and count of employees and crops such events to show any user a brief view through the application. To this also we have used SQL DATA ADAPTER to retrieve data and DATA TABLE to store that data.
- ✓ We have set labels to be updated in each event here.
- ✓ Then we have set the label text fields to that retrieved data's string values.