### INTRODUCTION

### ABSTRACT

This project is Entitled as “**HERO Co”** developed using ASP.NET as a Front End and Microsoft SQL server 2012 as a Back End. This project is ideal for dealers or resellers of any size. The Bike showroom control panel can be access anywhere in any time.

Bike showroom management system describes the complete process of selling a Bike or Scooter to a customer from the dealer’s showroom. Before selling, the Bike or Scooter belongs to the fixed asset of the dealer’s showroom. So the main point of this scenario is posting the Bike or Scooter from a fixed asset to a current asset. While executing the process, the dealer can manually maintain the Bike business transaction type, which means, the Bike or Scooter can be set as a new or used Bike or Scooter to sell to the customer. This scenario shows the process of new Bike or Scooter sales.

* The dealer purchases the Bike or Scooter from Company
* The customer wants to buy Bike or Scooter from dealer’s showroom.
* The dealer creates a sales order for the end customer
* The customer confirms the order
* The dealer creates a service order to prepare the Bike or Scooter.

### PROBLEM DEFINITION

The objective of this project is to create a Bike Showroom Management System which helps to manage the details about the Bikes available and also the employees working in that showroom. Most of the showrooms today are running manually storing data in books and files. As the storage medium are books there is chance of inconsistency, accessing a particular item is very time consuming and boring task and the probability of errors during calculations is very high. Due to these drawbacks of the existing manual system, the need of new computerized system is inevitable.

Using the features of ASP.net, the Showroom Management System has got highly user friendly interface which makes the dealing with the system simpler and easier. With the MSSQL Server 2005, we can store and retrieve required data in efficient manner, so that the precious time of users can be Saved, calculations can be made accurately and thereby make their procedures simple.

### 

### SYSTEM ANALYSIS

* 1. **Existing System**

A Bike or Scooter showroom has to manage a number of forms regarding delivery, exchange, registration, insurance etc. At present all the procedures are doing manually. There are many disadvantages for this system. Data security cannot be assured, retrieving data is difficult. There is chance of losing the stored details. Also errors occur. It is a time consuming process.

### Proposed System

To take advantage of the latest technology and to manage the details stored in the showroom a new system needs to be developed. The new system should accomplish the following functions The system should allow the representative to handle the model, price, The booking of Bike, other details all have to be handled. It should store the customer details, update the price.

### SYSTEM STUDY

* 1. **FEASIBILITY STUDY**

The development and implementation of a new system is definitely expensive. It requires system resources, manpower, time and money. So it increases the necessity of the feasibility study based on the proposed system requirements. During system analysis, the feasibility study of the proposed system is to be Bikeried out.

The study is done in three phases:

1. Technical feasibility
2. Economical feasibility
3. Operational Feasibility

### Technical Feasibility

The assessment of technical feasibility must be based on an outline design of system requirements in terms of input, output, files, programs, and procedures. This can be qualified in terms of volume of data, trends, frequency of updating, cycles of activity etc. in order to give an introduction of technical system. “Bike showroom management system” satisfies technical feasibility because it need not require any additional hardware or system configuration for implementation and execution.

### Operational Feasibility

The windows application is a highly programmable environment that allows mass customization through the immediate deployment of a large and diverse range of applications, to millions of global users.

### 

### 

### MODULES

This project has Several modules:

These modules are used to maintain and enter the essential details Regarding the available Bikes in the showroom their price in showroom and the price in on road, extra (tax) etc. And also these modules maintain the user who manage the system in showroom can enter The details of customer purchased Bike, its details such as price, make, model, color, Date of purchase,etc .

### Bike or Scooter details

It allows the administrator to enter the Bikes available and details such as Color, price, Make, Model etc.

# SYSTEM SPECIFICATION

Analysis will be helpful to produce a software requirement specification (SRS). An SRS contains functional requirements, performance requirements, forms of input and output, design constraints etc.Functional requirements include functionality details of every module. The first module, i.e. Employee module consists of the details of employee including the salary details. In customer module all the details about customer are stored, also about the details of Bike purchasing and delivery details.

SOFTWARE SPECIFICATION

Platform - Windows 10

Front end tool - MS Visual Studio 2017 (ASP.NET 3.5)

Language - VB.NET

Back end tool - MS SQL Server Express 2012

* 1. HARDWARE SPECIFICATION

System - Pentium Dual Core

Memory - 1GB

Hard disk - 1024 GB HDD

Monitor - LCD or LED Monitor

Mouse - Optical Mouse

Keyboard - 104 Keys

# SOFTWARE FEATURES

ASP.NET is a [server-side](http://en.wikipedia.org/wiki/Server-side_scripting) [Web application framework](http://en.wikipedia.org/wiki/Web_application_framework) designed for [Web development](http://en.wikipedia.org/wiki/Web_development) to produce [dynamic Web pages.](http://en.wikipedia.org/wiki/Dynamic_Web_pages) It was developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) to allow [programmers](http://en.wikipedia.org/wiki/Programmer) to build dynamic [web sites,](http://en.wikipedia.org/wiki/Web_site) [web applications](http://en.wikipedia.org/wiki/Web_application) and [web services](http://en.wikipedia.org/wiki/Web_service). It was first released in January 2002 with version 1.0 of the [.NET Framework,](http://en.wikipedia.org/wiki/.NET_Framework) and is the successor to Microsoft's [Active Server Pages](http://en.wikipedia.org/wiki/Active_Server_Pages) (ASP) technology. ASP.NET is built on the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), allowing programmers to write ASP.NET code using any supported [.NET language.](http://en.wikipedia.org/wiki/List_of_CLI_languages) The ASP.NET [SOAP](http://en.wikipedia.org/wiki/SOAP) extension framework allows ASP.NET components to process SOAP messages.

## ASP.NET compared with classic ASP

ASP.NET simplifies developers' transition from [Windows](http://en.wikipedia.org/wiki/Microsoft_Windows) application development to Web development by offering the ability to build pages composed of [*controls*](http://en.wikipedia.org/wiki/Widget_%28computing%29)similar to a Windows [user interface.](http://en.wikipedia.org/wiki/User_interface) A Web control, such as a *button* or *label*, functions in very much the same way as its Windows counterparts: code can assign its properties and respond to its events. Controls know how to render themselves: whereas Windows controls draw themselves to the screen, Web controls produce segments of [HTML](http://en.wikipedia.org/wiki/HTML) and [JavaScript](http://en.wikipedia.org/wiki/JavaScript) which form parts of the resulting page sent to the end-user's browser.

ASP.NET encourages the programmer to develop applications using an [event-driven](http://en.wikipedia.org/wiki/Event-driven_programming) [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) model, rather than in conventional Web-[scripting](http://en.wikipedia.org/wiki/Scripting_programming_language) environments like ASP and [PHP](http://en.wikipedia.org/wiki/PHP). The framework combines existing technologies such as JavaScript with internal components like "[ViewState](http://en.wikipedia.org/wiki/ViewState)" to bring persistent (inter-request) state to the inherently [stateless](http://en.wikipedia.org/wiki/Stateless_server) Web environment. Other differences compared to [classic ASP](http://en.wikipedia.org/wiki/Active_Server_Pages) are:

* + - Compiled code means applications run faster with more design-time errors trapped at the development stage.
    - Significantly improved run-time error handling, making use of [exception handling](http://en.wikipedia.org/wiki/Exception_handling) using try-catch blocks.
    - Similar metaphors to Microsoft Windows applications such as controls and events.
    - An extensive set of controls and class libraries, as well as user-defined controls, allow the rapid building of applications. Layout of these controls on a page is easier because most of it can be done visually in most editors.
    - ASP.NET uses the multi-language abilities of the .NET [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime), allowing Web pages to be coded in VB.NET, C#, J#, Delphi.NET, Chrome, etc.
    - Ability to cache the whole page or just parts of it to improve performance.
    - Ability to use the [code-behind](http://en.wikipedia.org/wiki/ASP.NET#Code-behind_model) development model to separate business logic from presentation.
    - Ability to use t[rue object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) design for programming pages and controls
    - If an ASP.NET application [leaks memory](http://en.wikipedia.org/wiki/Memory_leak), the ASP.NET runtime unloads the AppDomain hosting the erring application and reloads the application in a new AppDomain.
    - [Session state](http://en.wikipedia.org/wiki/ASP.NET#Session_state) in ASP.NET can be saved in a [Microsoft SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server) database or in a separate process running on the same machine as the Web server or on a different machine. That way session values are not lost when the Web server is reset or the ASP.NET worker process is recycled.
    - Versions of ASP.NET prior to 2.0 were criticized for their lack of standards compliance. The generated HTML and JavaScript sent to the client browser would not always validate against [W3C](http://en.wikipedia.org/wiki/World_Wide_Web_Consortium)/[ECMA](http://en.wikipedia.org/wiki/Ecma_International) standards. In addition, the framework's browser detection feature sometimes incorrectly identified Web browsers other than Microsoft's own [Internet Explorer](http://en.wikipedia.org/wiki/Internet_Explorer) as "downlevel" and returned HTML/JavaScript to these clients with some of the features removed, or sometimes crippled or broken. In version 2.0 however, all controls generate valid HTML 4.0, XHTML 1.0 (the default) or XHTML 1.1 output, depending on the site configuration. Detection of standards-compliant Web browsers is more robust and support for [Cascading Style Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets) is more extensive.
    - Web Server Controls: these are controls introduced by ASP.NET for providing the UI for the Web form. These controls are state managed controls and are [WYSIWYG](http://en.wikipedia.org/wiki/WYSIWYG) controls.

## Custom controls

Programmers can also build custom controls for ASP.NET applications. Unlike user controls, these controls do not have an ASCX markup file, having all their code compiled into a [dynamic](http://en.wikipedia.org/wiki/Dynamic-link_library) [link library (DLL)](http://en.wikipedia.org/wiki/Dynamic-link_library) file. Such custom controls can be used across multiple Web applications and [Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) projects.

User controls

User controls are encapsulations of sections of pages which are registered and used as controls in ASP.NET, etc.

Rendering technique

ASP.NET uses a "visited composites" rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the Literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree.

Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the methods of the visitor. The resulting HTML output is sent to the client.

After the request has been processed, the instance of the page class is disBikeded and with it the entire control tree. This is a source of confusion among novice ASP.NET programmers who rely on the class instance members that are lost with every page request/response cycle.

State management

ASP.NET applications are hosted by a [Web server](http://en.wikipedia.org/wiki/Web_server) and are accessed using the [stateless](http://en.wikipedia.org/wiki/Stateless_server) [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) protocol. As such, if an application uses stateful interaction, it has to implement [state](http://en.wikipedia.org/wiki/State_management) [management](http://en.wikipedia.org/wiki/State_management) on its own. ASP.NET provides various functions for state management. Conceptually, Microsoft treats "state" as [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) state. Problems may arise if an application needs to keep track of "data state"; for example, a [finite-state machine](http://en.wikipedia.org/wiki/Finite-state_machine) which may be in a transient state between requests ([lazy evaluation](http://en.wikipedia.org/wiki/Lazy_evaluation)) or which takes a long time to initialize. State management in ASP.NET pages with authentication can make [Web scraping](http://en.wikipedia.org/wiki/Web_scraping) difficult or impossible.

Application state is held by a collection of shared user-defined variables. These are set and initialized when the Application\_OnStart event fires on the loading of the first instance of the application and are available until the last instance exits. Application state variables are accessed using the Applications collection, which provides a wrapper for the application state. Application state variables are identified by name.

Session state

Server-side session state is held by a collection of user-defined session variables that are persistent during a user session. These variables, accessed using the Session collection, are unique to each session instance. The variables can be set to be automatically destroyed after a defined time of inactivity even if the session does not end. Client-side user session is maintained by either a [cookie](http://en.wikipedia.org/wiki/HTTP_cookie) or by encoding the session ID in the URL itself.

ASP.NET supports three modes of persistence for server-side session variables:

### In-process mode

The session variables are maintained within the ASP.NET [process](http://en.wikipedia.org/wiki/Process_%28computing%29). This is the fastest way; however, in this mode the variables are destroyed when the ASP.NET process is recycled or shut down.

State server mode

ASP.NET runs a separate [Windows service](http://en.wikipedia.org/wiki/Windows_service) that maintains the state variables. Because state management happens outside the ASP.NET process, and because the ASP.NET engine accesses data using .NET Remoting, ASPState is slower than In-Process. This mode allows an ASP.NET application to be load-balanced and scaled across multiple servers. Because the state management service runs independently of ASP.NET, the session variables can persist across ASP.NET process shutdowns. However, since session state server runs as one instance, it is still one point of failure for session state. The session-state service cannot be load-balanced, and there are restrictions on types that can be stored in a session variable.

SQL Server mode

State variables are stored in a [database,](http://en.wikipedia.org/wiki/Database) allowing session variables to be persisted across ASP.NET process shutdowns. The main advantage of this mode is that it allows the application to balance load on a server cluster, sharing sessions between servers. This is the slowest method of session state management in ASP.NET.

**Microsoft SQL Server Express**

Microsoft SQL Server Express is a version of [Microsoft'](http://en.wikipedia.org/wiki/Microsoft)s [SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server) [relational database](http://en.wikipedia.org/wiki/Relational_database_management_system) [management system](http://en.wikipedia.org/wiki/Relational_database_management_system) that is free to download or distribute. It comprises a database specifically targeted for [embedded](http://en.wikipedia.org/wiki/Embedded_system) and smaller-scale applications. The product traces its roots to the [Microsoft Database Engine (MSDE)](http://en.wikipedia.org/wiki/MSDE) product, which was shipped with SQL Server 2000. The 'Express' branding has been used since the release of SQL Server 2005.

Microsoft SQL Server Express Edition (SQL Server Express) is a powerful and reliable data management product that delivers rich features, data protection, and performance for embedded application clients, light Web applications, and local data stores. Designed for easy deployment and rapid prototyping, SQL Server Express is available at no cost, and you are free to redistribute it with applications. Because it is part of the Microsoft Windows Server System, SQL Server Express is designed to integrate seamlessly with your other server infrastructure investments. With a database size limit of 10 gigabytes, support for 1 CPU and 1GB of RAM, SQL Server Express Edition is suitable for application embedding or lightweight application development.

Capabilities

SQL Server Express provides many of the features of the paid, full versions of Microsoft SQL Server database management system. However it has technical restrictions that make it unsuitable for some large-scale deployments. Differences in the Express product include: Maximum database size of 10 GB per database in SQL Server 2012.

No SQL Server Agent service[[2]](http://en.wikipedia.org/wiki/SQL_Server_Express" \l "cite_note-2)[[3]](http://en.wikipedia.org/wiki/SQL_Server_Express#cite_note-2) Hardware-utilization limits:

Single physical CPU, but multiple cores allowable[[4]](http://en.wikipedia.org/wiki/SQL_Server_Express" \l "cite_note-4)

1 GB of RAM (runs on a system with any RAM amount, but uses only at most 1 GB)

Unlike the predecessor product, [MSDE,](http://en.wikipedia.org/wiki/MSDE) the Express product does not include a concurrent workload-governor to "limit performance if the database engine receives more work than is typical of a small number of users."[[5]](http://en.wikipedia.org/wiki/SQL_Server_Express#cite_note-5)

SQL Server Express includes several GUI tools for database management. These include: [SQL Server Management Studio Express](http://en.wikipedia.org/wiki/SQL_Server_Management_Studio_Express)

SQL Server Configuration Manager

SQL Server Surface Area Configuration tool

SQL Server [Business Intelligence Development Studio](http://en.wikipedia.org/wiki/Business_Intelligence_Development_Studio)

The predecessor product [MSDE](http://en.wikipedia.org/wiki/MSDE) generally lacked basic GUI management tools,[[6]](http://en.wikipedia.org/wiki/SQL_Server_Express" \l "cite_note-6)

* Features available in SQL Server "Standard" and better editions but absent from SQL Server Express include:
* [Analysis Services](http://en.wikipedia.org/wiki/Microsoft_Analysis_Services)
* [Integration Services](http://en.wikipedia.org/wiki/SQL_Server_Integration_Services)
* [Notification Services](http://en.wikipedia.org/wiki/SQL_Server_Notification_Services)
* Variants

Acknowledgement

We would like to express our special thanks of gratitude to our teacher “**Mamta Pandey**” who gave us the golden opportunity to do this wonderful project on the topic “**HERO Customer Registration Form**”, which also helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to her.

Secondly we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

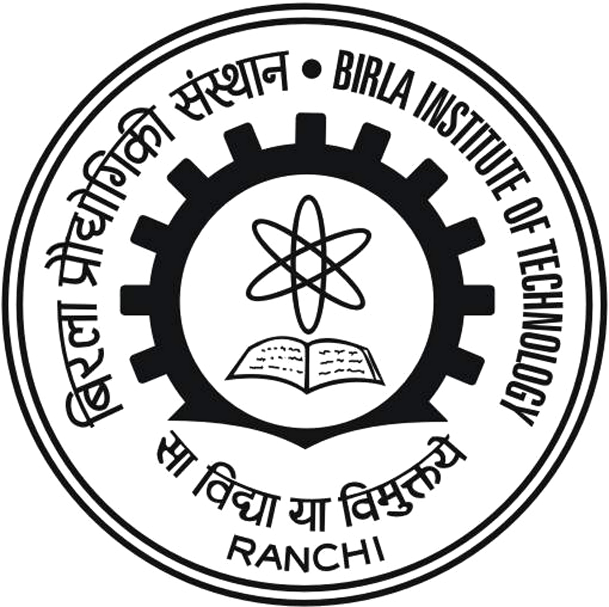
**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **TITLE** | **Page NO.** | **REMARKS** |
| **1.** | Introduction |  |  |
| **2.** | Abstract |  |  |
| **3.** | Problem Definition |  |  |
| **4.** | System Analysis |  |  |
| **5.** | System Study |  |  |
| **6.** | Modules |  |  |
| **7.** | System Specification |  |  |
| **8.** | Customer\_Registration \_Form.VB |  |  |
| **9.** | Customer\_Registration \_Form.VB[Design] |  |  |
| **10.** | View\_Form.VB |  |  |
| **11.** | View\_Form.VB[Design] |  |  |
| **12.** | Outputs |  |  |
|  |  |  |  |
|  |  |  |  |

*Birla Institute Of Technology*

Extenstion Center, Lalpur

***RANCHI***



**<ASSIGNMENT>**

**NAME:- KUMAR SOURAV (BCA/40597/16)**

**SHUBHAM SHARMA (BCA/40592/16)**

**SUBJECT:- VB.Net**

**TOPIC:- Honda Bike & Scooter   
 Customer Registration Form**

**SEMESTER:- 5th**

**CLASS:- BCA 5th B (Boys)**

***Sign :-***

**CUSTOMER\_REGISTRATION\_FORM.VB**

Imports System.Data.SqlClient

Imports System.IO

Public Class Customer\_Registration

Dim connection As New SqlConnection("Server= SRV; Database = HERO; Integrated Security = true")

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Dim command As New SqlCommand("insert into Hero (Owner\_Name, Owner\_Mobile, Owner\_Address, Email\_ID, Price, V\_Type, Reg\_No,Owner\_Image,Date\_Of\_Purchase,Date\_Of\_Birth,Gender,V\_Model)VALUES(@Owner\_Name,@Owner\_Mobile,@Owner\_Address,@Email\_ID,@Price,@V\_Type,@Reg\_No,@Owner\_Image,@Date\_Of\_Purchase,@Date\_Of\_Birth,@Gender,@V\_Model)", connection)

Dim ms As New MemoryStream

PictureBox1.Image.Save(ms, PictureBox1.Image.RawFormat)

command.Parameters.Add("@Owner\_Name", SqlDbType.VarChar).Value = TextBox1.Text

command.Parameters.Add("Owner\_Mobile", SqlDbType.VarChar).Value = TextBox2.Text

command.Parameters.Add("@Owner\_Address", SqlDbType.VarChar).Value = TextBox3.Text

command.Parameters.Add("@Email\_ID", SqlDbType.VarChar).Value = TextBox7.Text

command.Parameters.Add("@Price", SqlDbType.VarChar).Value = TextBox4.Text

command.Parameters.Add("@V\_Type", SqlDbType.VarChar).Value = ComboBox2.Text

command.Parameters.Add("@Reg\_No", SqlDbType.VarChar).Value = TextBox6.Text

command.Parameters.Add("@Owner\_Image", SqlDbType.Image).Value = ms.ToArray()

command.Parameters.Add("@Date\_Of\_Purchase", SqlDbType.Date).Value = DateTimePicker1.Value

command.Parameters.Add("@Date\_Of\_Birth", SqlDbType.Date).Value = DateTimePicker2.Value

command.Parameters.Add("@V\_Model", SqlDbType.VarChar).Value = ComboBox1.Text

Dim rdb As String = ""

If RadioButton1.Checked = True Then

rdb = "MALE"

Else

rdb = "FEMALE"

End If

command.Parameters.Add("@Gender", SqlDbType.VarChar).Value = rdb

connection.Open()

If command.ExecuteNonQuery() = 1 Then

MessageBox.Show("Submitted Successfully")

Else

MessageBox.Show("Error While Submitting")

End If

connection.Close()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

Dim opf As New OpenFileDialog

opf.Filter = "Choose Image(\*JPG;\*.PNG;\*.GIF)|\*.jpg;\*.png;\*.gif"

If opf.ShowDialog = Windows.Forms.DialogResult.OK Then

PictureBox1.Image = Image.FromFile(opf.FileName)

End If

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

View\_Form.Show()

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

For Each control As Control In Me.Controls

If TypeOf control Is TextBox Then

control.Text = String.Empty

End If

If TypeOf control Is ComboBox Then

control.Text = String.Empty

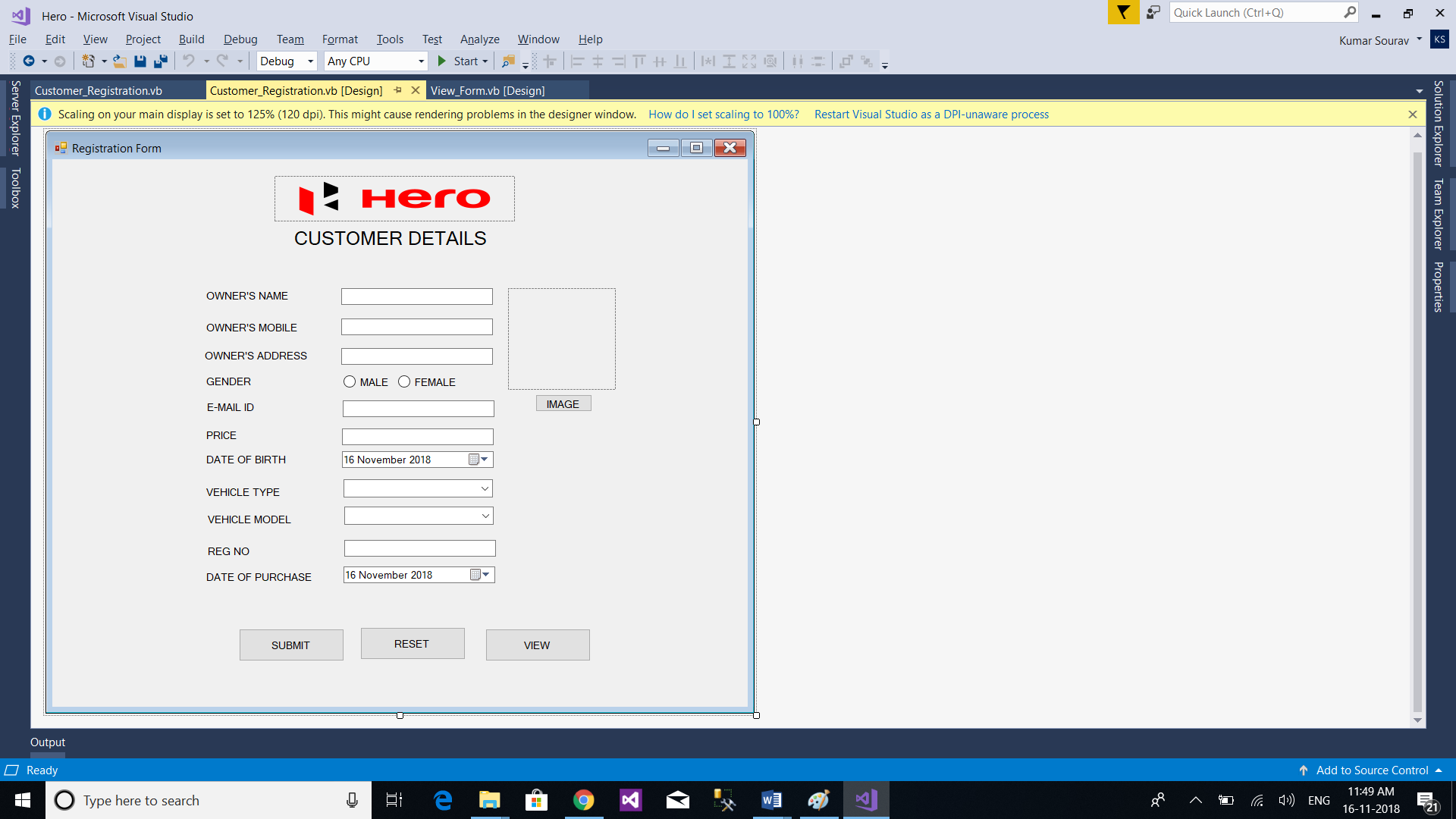
End If

Next

End Sub

End Class

**CUSTOMER\_REGISTRATION\_FORM.VB[DESIGN]**



**VIEW\_FORM.VB**

Imports System.Data.SqlClient

Imports System.IO

Public Class View\_Form

Dim connection As New SqlConnection("Server= SRV; Database = HERO; Integrated Security = true")

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Dim sqlcom As New SqlCommand("Select \* From Hero", connection)

Dim da As New SqlDataAdapter(sqlcom)

Dim dt As New DataTable

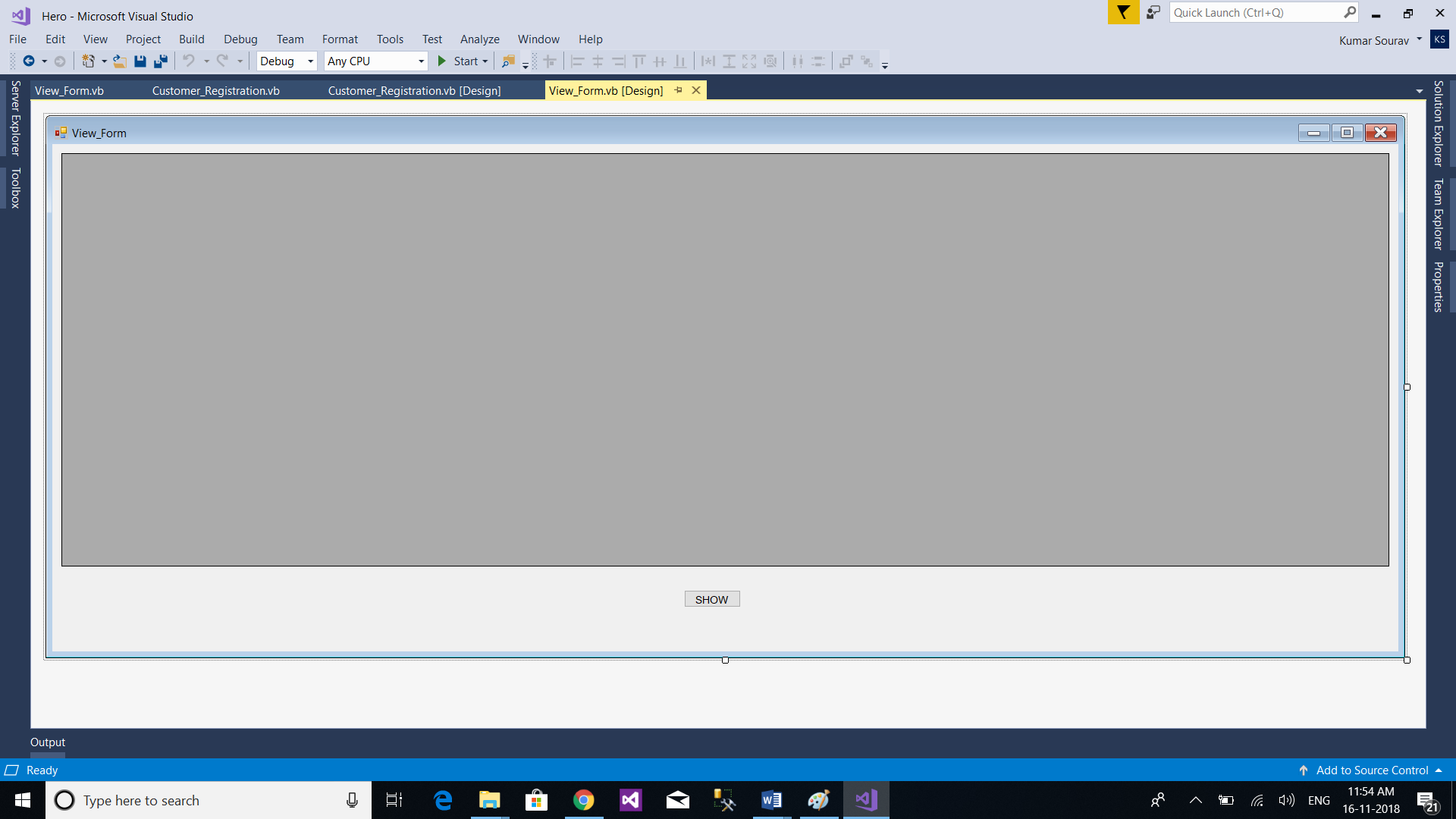
da.Fill(dt)

DataGridView1.DataSource = dt

End Sub

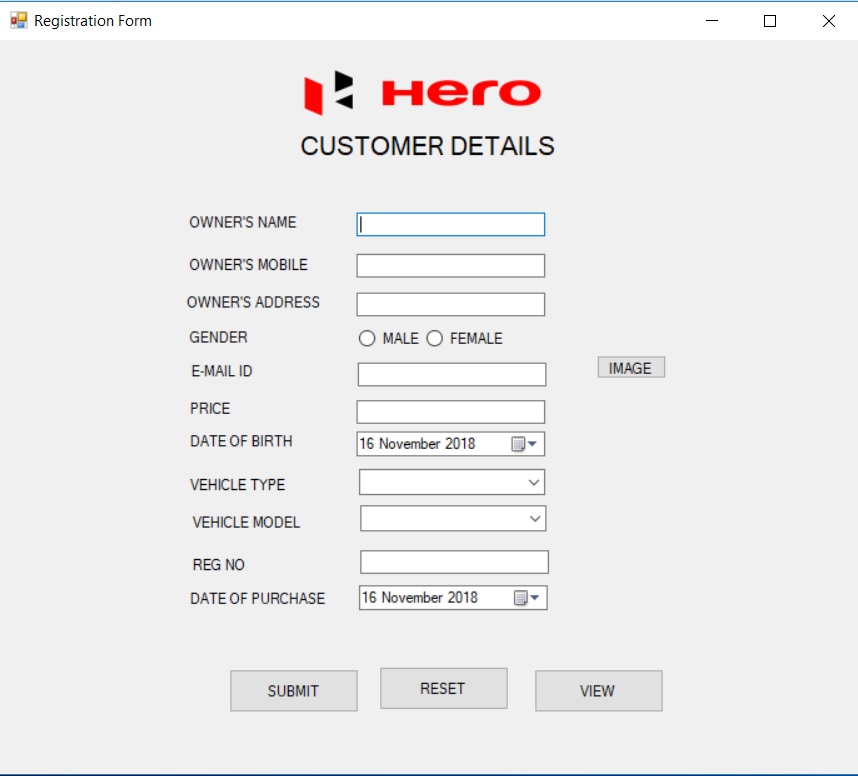
End Class

**VIEW\_FORM.VB[DESIGN]**

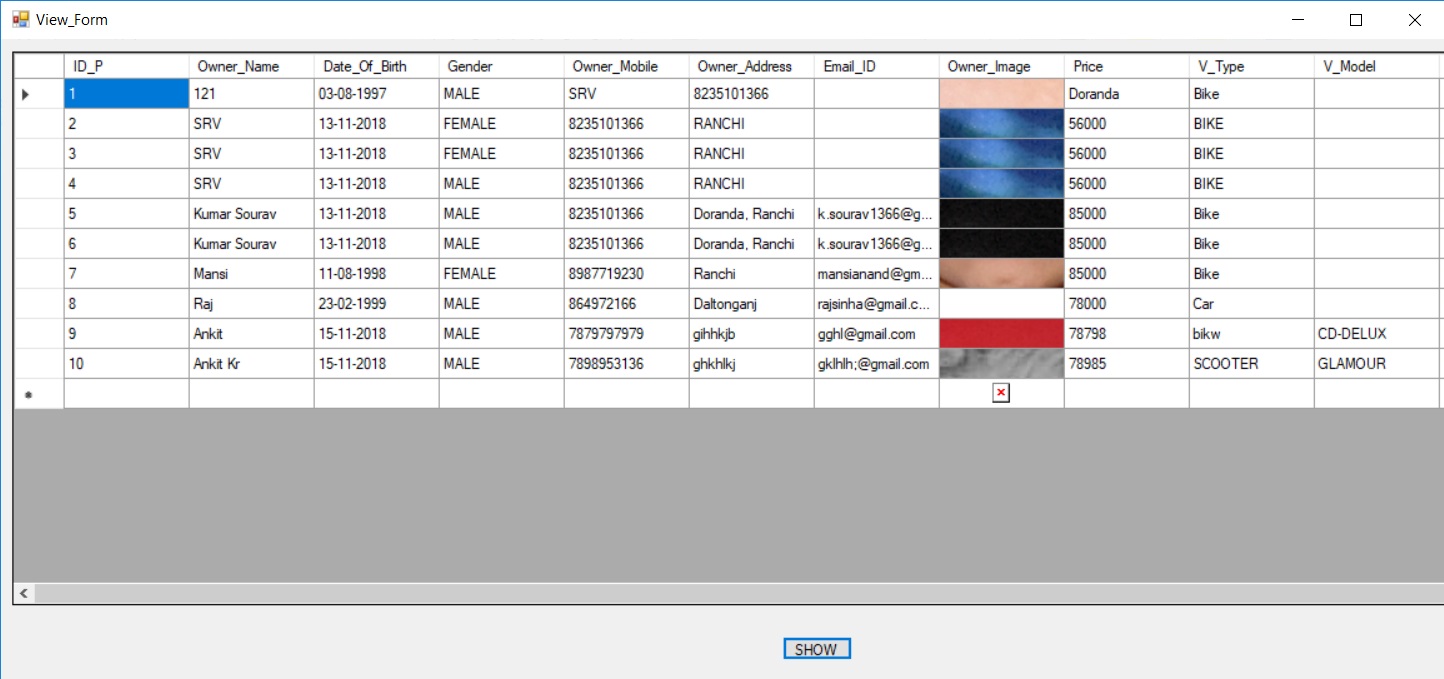


**OUTPUTS**

**Registration Form**

****

**View Form**

****