Vulnerability Management Tool

Technical Manual

Student Name: Síne Doheny
Supervised by: Richard Butler
Student ID: C00226237

Table of Contents

Introduction

The purpose of this document is to provide an appendix of the full Vulnerability Management application.

Database Configuration SQL Statements

Dbo.Asset

```
CREATE TABLE [dbo].[Asset] (
    [AssetID]                 INT          IDENTITY (1, 1) NOT NULL,
    [AssetName]               VARCHAR (50) NOT NULL,
    [DeviceCategory]          VARCHAR (50) NOT NULL,
    [Hardware]                VARCHAR (50) NOT NULL,
    [OperatingSystem]         VARCHAR (50) NOT NULL,
    [MacAddress]              VARCHAR (50) NOT NULL,
    [IPv4Address]             VARCHAR (50) NOT NULL,
    [IPv6Address]             VARCHAR (50) NOT NULL,
    [OfficeLocation]          VARCHAR (50) NOT NULL,
    [BusinessUnit]            VARCHAR (50) NOT NULL,
    [DataClassification]      VARCHAR (50) NOT NULL,
    [Risk]                    VARCHAR (50) NOT NULL,
    [OwnerName]               VARCHAR (50) NOT NULL,
    [TechnicalSupportContact] VARCHAR (50) NOT NULL,
    [DateCreated]             DATE         NOT NULL
);
```

Dbo.Assignment

```
CREATE TABLE [dbo].[Assignment] (
    [VulnerabilityID] INT          IDENTITY (1, 1) NOT NULL,
    [UserID]          VARCHAR (50) NOT NULL,
    [AssignmentID]    VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([VulnerabilityID] ASC)
);
```

Dbo.Contact

```
CREATE TABLE [dbo].[Contact] (
    [ContactID] INT          IDENTITY (1, 1) NOT NULL,
    [Name]      TEXT         NOT NULL,
    [Email]     VARCHAR (50) NOT NULL,
    [Telephone] INT          NOT NULL,
    PRIMARY KEY CLUSTERED ([ContactID] ASC)
);
```

Dbo.Credentials

```
CREATE TABLE [dbo].[Credentials] (
    [CredentialID]  INT          IDENTITY (1, 1) NOT NULL,
    [TestID]        VARCHAR (50) NOT NULL,
    [Username]      VARCHAR (50) NOT NULL,
    [Password]      VARCHAR (50) NOT NULL,
    [FunctionLevel] VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([CredentialID] ASC)
);
```

Dbo.Function

```
CREATE TABLE [dbo].[Function] (
    [FunctionID] INT          IDENTITY (1, 1) NOT NULL,
    [Tester]     VARCHAR (50) NOT NULL,
    [Developer]  VARCHAR (50) NOT NULL,
    [Admin]      VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([FunctionID] ASC)
);
```

Dbo.MitreTactic

```
CREATE TABLE [dbo].[MitreTactic] (
    [TacticID]    INT           IDENTITY (1, 1) NOT NULL,
    [TacticTag]   VARCHAR (50)  NOT NULL,
    [Tactic]      VARCHAR (50)  NOT NULL,
    [Description] VARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_MitreTactic] PRIMARY KEY CLUSTERED ([TacticID] ASC)
);
```

Dbo.MitreTechnique

```
CREATE TABLE [dbo].[MitreTechnique] (
    [TechniqueID] INT          IDENTITY (1, 1) NOT NULL,
    [TacticID]    INT          NOT NULL,
    [Tactic]      VARCHAR (50) NOT NULL,
    [Technique]   VARCHAR (50) NOT NULL,
    CONSTRAINT [PK_MitreTechnique] PRIMARY KEY CLUSTERED ([TechniqueID] ASC)
);
```

Dbo.Password

```
CREATE TABLE [dbo].[Password] (
    [UserID]   INT          IDENTITY (1, 1) NOT NULL,
    [Username] VARCHAR (50) NOT NULL,
    [Salt]     VARCHAR (50) NOT NULL,
    [Hash]     VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([UserID] ASC)
);
```

Dbo.Roles

```
CREATE TABLE [dbo].[Roles] (
    [RoleID]            INT          IDENTITY (1, 1) NOT NULL,
    [Administrator]     VARCHAR (50) NOT NULL,
    [PenetrationTester] VARCHAR (50) NOT NULL,
    [Engineer]          VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([RoleID] ASC)
);
```

Dbo.Session

```
CREATE TABLE [dbo].[Session] (
    [UserID]    INT          IDENTITY (1, 1) NOT NULL,
    [SessionID] VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([UserID] ASC)
);
```

Dbo.Status

```
CREATE TABLE [dbo].[Status] (
    [StatusID] INT IDENTITY (1, 1) NOT NULL,
    [Open]     BIT NOT NULL,
    [ReTest]   BIT NOT NULL,
    [Fixed]    BIT NOT NULL,
    PRIMARY KEY CLUSTERED ([StatusID] ASC)
);
```

Dbo.Test

```
CREATE TABLE [dbo].[Test] (
    [TestID]      INT           IDENTITY (1, 1) NOT NULL,
    [TestName]    VARCHAR (50)  NOT NULL,
    [AssetName]   VARCHAR (50)  NOT NULL,
    [StartDate]   DATE          NOT NULL,
    [EndDate]     DATE          NOT NULL,
    [SignOff]     BIT           NOT NULL,
```

```
    [URL]         VARCHAR (50)  NOT NULL,
    [Description] VARCHAR (MAX) NOT NULL,
    [ContactID]   VARCHAR (50)  NOT NULL,
    [TestStatus]  VARCHAR (50)  NULL,
    [TicketUser]  VARCHAR (50)  NULL,
    [TestNotes]   VARCHAR (50)  NULL,
    PRIMARY KEY CLUSTERED ([TestID] ASC)
);
```

Dbo.TestAssignment

```
CREATE TABLE [dbo].[TestAssignment] (
    [TestAssignmentID] INT          IDENTITY (1, 1) NOT NULL,
    [TestName]         VARCHAR (50) NOT NULL,
    [TicketUser]       VARCHAR (50) NOT NULL,
    [DateAssigned]     DATE         NOT NULL,
    [DateClosed]       DATE         NULL,
    [TicketStatus]     VARCHAR (50) NOT NULL,
    [TestTicketNotes]  VARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([TestAssignmentID] ASC)
);
```

Dbo.TestLog

```
CREATE TABLE [dbo].[TestLog] (
    [TestLogID]        INT            IDENTITY (1, 1) NOT NULL,
    [TestName]         VARCHAR (50)   NOT NULL,
    [TicketUser]       VARCHAR (500)  NOT NULL,
    [CurrentTimeStamp] DATETIME       NOT NULL,
    [TicketStatus]     VARCHAR (50)   NOT NULL,
    [TestTicketNotes]  VARCHAR (1000) NULL,
    PRIMARY KEY CLUSTERED ([TestLogID] ASC)
);
```

Dbo.TestStatus

```
CREATE TABLE [dbo].[TestStatus] (
    [StatusID]   INT          IDENTITY (1, 1) NOT NULL,
    [StatusType] VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([StatusID] ASC)
);
```

Dbo.Users

```
CREATE TABLE [dbo].[Users] (
    [UserID]     INT          IDENTITY (1, 1) NOT NULL,
    [UsersName]  VARCHAR (50) NOT NULL,
    [Username]   VARCHAR (50) NOT NULL,
    [Password]   VARCHAR (50) NOT NULL,
    [Role]       VARCHAR (50) NOT NULL,
    [Department] VARCHAR (50) NOT NULL,
    [Company]    VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([UserID] ASC)
);
```

Dbo.Vulnerability

```
CREATE TABLE [dbo].[Vulnerability] (
    [VulnerabilityID]   INT            IDENTITY (1, 1) NOT NULL,
    [TestName]          VARCHAR (50)   NOT NULL,
    [VulnerabilityType] VARCHAR (50)   NOT NULL,
    [Severity]          VARCHAR (50)   NOT NULL,
    [DateOpen]          DATE           NOT NULL,
```

```
    [DateClosed]        DATE           NOT NULL,
    [URL]               VARCHAR (50)   NOT NULL,
    [TacticID]          INT            NOT NULL,
    [TechniqueID]       INT            NOT NULL,
    [StepsToReproduce]  VARCHAR (MAX)  NULL,
    [Mitigation]        VARCHAR (MAX)  NULL,
    [StatusID]          VARCHAR (50)   NOT NULL,
    [Tactic]            VARCHAR (50)   NULL,
    [Technique]         VARCHAR (50)   NULL,
    [DamagePotential]   DECIMAL (9, 2) NULL,
    [Reproducability]   DECIMAL (9, 2) NULL,
    [Exploitability]    DECIMAL (9, 2) NULL,
    [AffectedUsers]     DECIMAL (9, 2) NULL,
    [Discoverability]   DECIMAL (9, 2) NULL,
    [DREADTotal]        DECIMAL (9, 2) NULL,
    [AssetName]         VARCHAR (100)  NULL,
    [TestID]            INT            NULL,
    [TicketNotes]       VARCHAR (MAX)  NULL,
    [AssignedUser]      VARCHAR (50)   NULL,
    PRIMARY KEY CLUSTERED ([VulnerabilityID] ASC)
);
```

Dbo.VulnerabilityAssignment

```
CREATE TABLE [dbo].[VulnerabilityAssignment] (
    [VulnerabilityAssignmentID] INT          IDENTITY (1, 1) NOT NULL,
    [VulnerabilityID]           INT          NOT NULL,
    [TicketUser]                VARCHAR (50) NOT NULL,
    [DateAssigned]              DATE         NOT NULL,
    [DateClosed]                DATE         NULL,
    [TicketStatus]              VARCHAR (50) NOT NULL,
    [VulnerabilityTicketNotes]  VARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([VulnerabilityAssignmentID] ASC)
);
```

Dbo.VulnerabilityLog

```
CREATE TABLE [dbo].[VulnerabilityLog] (
    [VulnerabilityLogID]       INT            IDENTITY (1, 1) NOT NULL,
    [VulnerabilityID]          INT            NOT NULL,
    [TicketUser]               VARCHAR (500)  NOT NULL,
    [CurrentTimeStamp]         DATETIME       NOT NULL,
    [TicketStatus]             VARCHAR (50)   NOT NULL,
    [VulnerabilityTicketNotes] VARCHAR (1000) NULL,
    PRIMARY KEY CLUSTERED ([VulnerabilityLogID] ASC)
);
```

Dbo.VulnerabilityStatusEngineer

```
CREATE TABLE [dbo].[VulnerabilityStatusEngineer] (
    [StatusID]   INT          IDENTITY (1, 1) NOT NULL,
    [StatusType] VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([StatusID] ASC)
);
```

Dbo.VulnerabilityStatusPenTester

```
CREATE TABLE [dbo].[VulnerabilityStatusPenTester] (
    [StatusID]   INT          IDENTITY (1, 1) NOT NULL,
    [StatusType] VARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([StatusID] ASC)
);
```

Code Implementation

RouteConfig.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Web;
5   using System.Web.Mvc;
6   using System.Web.Routing;
7
8   namespace Project
9   {
        1 reference
10      public class RouteConfig
11      {
            1 reference
12          public static void RegisterRoutes(RouteCollection routes)
13          {
14              routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
15
16              routes.MapRoute(
17                  name: "Private",
18                  url: "Private",
19                  defaults: new { controller = "Login", action = "onPrivateURL", id = UrlParameter.Optional }
20              );
21
22              routes.MapRoute(
23                  name: "Default",
24                  url: "{controller}/{action}/{id}",
25                  defaults: new { controller = "Dashboard", action = "Index", id = UrlParameter.Optional }
26              );
27          }
28      }
29  }
```

Asset Controller

AssetsController.cs

```
using Project.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Controllers
{
    0 references
    public class AssetsController : Controller
    {
        // GET: Assets
        [CustomAuthorization]
        0 references
        public ActionResult Index()
        {
            List<AssetModel> assets = new List<AssetModel>();
            AssetDAO assetDAO = new AssetDAO();
            assets = assetDAO.FetchAll();
            return View("Index", assets);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Details(int id)
        {
            AssetDAO assetDAO = new AssetDAO();
            AssetModel asset = assetDAO.FetchOne(id);
            return View("Details", asset);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Create()
        {
            return View("AddAssetForm");
        }
        [CustomAuthorization]
        0 references
        public ActionResult AddAsset(AssetModel assetModel)
        {
            //save it to the database
            AssetDAO assetDAO = new AssetDAO();
            assetDAO.Create(assetModel);
            return View("Details", assetModel);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Edit(int id)
        {
            AssetDAO assetDAO = new AssetDAO();
            AssetModel asset = assetDAO.FetchOne(id);
            return View("UpdateAssetForm", asset);
        }
        [CustomAuthorization]
        [HttpPost]
        0 references
        public ActionResult UpdateAsset(AssetModel assetModel)
        {
            //save it to the database
            AssetDAO assetDAO = new AssetDAO();

            assetDAO.Update(assetModel);
            return View("Details", assetModel);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Delete(int id)
        {
            AssetDAO assetDAO = new AssetDAO();
            assetDAO.Delete(id);
            List<AssetModel> assets = assetDAO.FetchAll();
            return View("Index", assets);
        }
    }
}
```

AssetDAO.cs

```
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Web;

namespace Project.Data
{
    12 references
    internal class AssetDAO
    {
        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
        //performs all operations




        2 references
        public List<AssetModel> FetchAll()
        {
            List<AssetModel> returnList = new List<AssetModel>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * from dbo.Asset";
                SqlCommand command = new SqlCommand(sqlQuery, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //create asset object
                        AssetModel asset = new AssetModel();
                        asset.AssetID = reader.GetInt32(0);
                        asset.AssetName = reader.GetString(1);
                        asset.DeviceCategory = reader.GetString(2);
                        asset.Hardware = reader.GetString(3);
                        asset.OperatingSystem = reader.GetString(4);
                        asset.MacAddress = reader.GetString(5);
                        asset.IPv4Address = reader.GetString(6);
                        asset.IPv6Address = reader.GetString(7);
                        asset.OfficeLocation = reader.GetString(8);
                        asset.BusinessUnit = reader.GetString(9);
                        asset.DataClassification = reader.GetString(10);
                        asset.Risk = reader.GetString(11);
                        asset.OwnerName = reader.GetString(12);
                        asset.TechnicalSupportContact = reader.GetString(13);
                        asset.DateCreated = reader.GetDateTime(14);

                        returnList.Add(asset);
                    }
                }


            }

                return returnList;
        }

```

```
60
                1 reference
61          internal int Delete(int id)
62          {
63              using (SqlConnection connection = new SqlConnection(connectionString))
64              {
65                  //prepared statement
66                  //update
67                  string sqlQuery = "DELETE FROM dbo.Asset WHERE AssetID = @id";
68
69
70
71
72                  //associate @id with Id parameter
73
74
75                  SqlCommand command = new SqlCommand(sqlQuery, connection);
76
77                  command.Parameters.Add("@Id", System.Data.SqlDbType.VarChar, 1000).Value = id;
78
79
80                  connection.Open();
81                  int deletedID = command.ExecuteNonQuery();
82
83                  return deletedID;
84              }
85          }
86
                2 references
87          public AssetModel FetchOne(int id)
88          {
89
90              //access database
91              using (SqlConnection connection = new SqlConnection(connectionString))
92              {
93                  string sqlQuery = "SELECT * FROM dbo.Asset WHERE AssetID = @id";
94
95                  SqlCommand command = new SqlCommand(sqlQuery, connection);
96                  command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;
97                  connection.Open();
98                  SqlDataReader reader = command.ExecuteReader();
99
100                 AssetModel asset = new AssetModel();
101                 if (reader.HasRows)
102                 {
103                     while (reader.Read())
104                     {
105                         //create new test object and add that to the list to be returned
106
107                         asset.AssetID = reader.GetInt32(0);
108                         asset.AssetName = reader.GetString(1);
109                         asset.DeviceCategory = reader.GetString(2);
110                         asset.Hardware = reader.GetString(3);
111                         asset.OperatingSystem = reader.GetString(4);
112                         asset.MacAddress = reader.GetString(5);
113                         asset.IPv4Address = reader.GetString(6);
114                         asset.IPv6Address = reader.GetString(7);
115                         asset.OfficeLocation = reader.GetString(8);
116                         asset.BusinessUnit = reader.GetString(9);
117                         asset.DataClassification = reader.GetString(10);
118                         asset.Risk = reader.GetString(11);
119                         asset.OwnerName = reader.GetString(12);
120                         asset.TechnicalSupportContact = reader.GetString(13);
121                         asset.DateCreated = reader.GetDateTime(14);
122
123
124                     }
125                 }
126                 return asset;
127             }
```

```
        }

    }
    1 reference
    public int Create(AssetModel assetModel)
    {

        //access database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            string sqlQuery = "INSERT INTO dbo.Asset Values(@AssetName, @DeviceCategory, @Hardware, @OperatingSystem, @MacAddress, @IPv4Address, @IPv6Address, @OfficeLocation,
              @BusinessUnit, @DataClassification, @Risk, @OwnerName, @TechnicalSupportContact, @DateCreated)";

            SqlCommand command = new SqlCommand(sqlQuery, connection);
            command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.AssetName;
            command.Parameters.Add("@DeviceCategory", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DeviceCategory;
            command.Parameters.Add("@Hardware", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Hardware;
            command.Parameters.Add("@OperatingSystem", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OperatingSystem;
            command.Parameters.Add("@MacAddress", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.MacAddress;
            command.Parameters.Add("@IPv4Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv4Address;
            command.Parameters.Add("@IPv6Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv6Address;
            command.Parameters.Add("@OfficeLocation", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OfficeLocation;
            command.Parameters.Add("@BusinessUnit", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.BusinessUnit;
            command.Parameters.Add("@DataClassification", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DataClassification;
            command.Parameters.Add("@Risk", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Risk;
            command.Parameters.Add("@OwnerName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OwnerName;
            command.Parameters.Add("@TechnicalSupportContact", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.TechnicalSupportContact;
            command.Parameters.Add("@DateCreated", System.Data.SqlDbType.DateTime).Value = assetModel.DateCreated;


            connection.Open();
            int newAssetID = command.ExecuteNonQuery();

            return newAssetID;
        }
    }

    1 reference
    public int Update(AssetModel assetModel)
    {


        //access the database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            //prepared statement
            //update
            string sqlQuery = "UPDATE dbo.Asset SET AssetName = @AssetName, DeviceCategory = @DeviceCategory, Hardware = @Hardware, OperatingSystem = @OperatingSystem, MacAddress =
              @MacAddress, IPv4Address = @IPv4Address, IPv6Address = @IPv6Address, OfficeLocation = @OfficeLocation, BusinessUnit = @BusinessUnit, DataClassification =
              @DataClassification, Risk = @Risk, OwnerName = @OwnerName, TechnicalSupportContact = @TechnicalSupportContact, DateCreated = @DateCreated  WHERE AssetID = @AssetID";

            //associate @id with Id parameter


            SqlCommand command = new SqlCommand(sqlQuery, connection);

            command.Parameters.Add("@AssetID", System.Data.SqlDbType.Int).Value = assetModel.AssetID;
            command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.AssetName;
            command.Parameters.Add("@DeviceCategory", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DeviceCategory;
            command.Parameters.Add("@Hardware", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Hardware;
            command.Parameters.Add("@OperatingSystem", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OperatingSystem;
            command.Parameters.Add("@MacAddress", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.MacAddress;
            command.Parameters.Add("@IPv4Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv4Address;
            command.Parameters.Add("@IPv6Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv6Address;
            command.Parameters.Add("@OfficeLocation", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OfficeLocation;
            command.Parameters.Add("@BusinessUnit", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.BusinessUnit;
            command.Parameters.Add("@DataClassification", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DataClassification;
            command.Parameters.Add("@Risk", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Risk;
            command.Parameters.Add("@OwnerName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OwnerName;
            command.Parameters.Add("@TechnicalSupportContact", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.TechnicalSupportContact;
            command.Parameters.Add("@DateCreated", System.Data.SqlDbType.DateTime).Value = assetModel.DateCreated;


            connection.Open();
            int newAssetID = command.ExecuteNonQuery();

            return newAssetID;
        }

    }
}
```

AssetModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    19 references
    public class AssetModel
    {
        [Display(Name = "Asset ID")]
        4 references
        public int AssetID { get; set; }
        [Display(Name = "Asset Name")]
        5 references
        public string AssetName { get; set; }
        [Display(Name = "Device Category")]
        5 references
        public string DeviceCategory { get; set; }
        [Display(Name = "Hardware")]
        5 references
        public string Hardware { get; set; }
        [Display(Name = "Operating System")]
        5 references
        public string OperatingSystem { get; set; }
        [Display(Name = "Mac Address")]
        5 references
        public string MacAddress { get; set; }
        [Display(Name = "IPV4 Address")]
        5 references
        public string IPv4Address { get; set; }
        [Display(Name = "IPV6 Address")]
        5 references
        public string IPv6Address { get; set; }
        [Display(Name = "Office Location")]
        5 references
        public string OfficeLocation { get; set; }
        [Display(Name = "Business Unit")]
        5 references
        public string BusinessUnit { get; set; }
        [Display(Name = "Data Classification")]
        5 references
        public string DataClassification { get; set; }
        [Display(Name = "Risk Level")]
        5 references
        public string Risk { get; set; }
        [Display(Name = "Owner Name")]
        5 references
        public string OwnerName { get; set; }
        [Display(Name = "Technical Support Contact")]
        5 references
        public string TechnicalSupportContact { get; set; }
        [Display(Name = "Date Created")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        5 references
        public DateTime DateCreated { get; set; }


        0 references
        public AssetModel(int assetID, string assetName, string deviceCategory, string hardware, string operatingSystem, string macAddress, string iPv4Address, string iPv6Address, string
          officeLocation, string businessUnit, string dataClassification, string risk, string ownerName, string technicalSupportContact, DateTime dateCreated)
        {
            AssetID = assetID;
            AssetName = assetName;
            DeviceCategory = deviceCategory;
            Hardware = hardware;
            OperatingSystem = operatingSystem;
            MacAddress = macAddress;
            IPv4Address = iPv4Address;
            IPv6Address = iPv6Address;
            OfficeLocation = officeLocation;
            BusinessUnit = businessUnit;
            DataClassification = dataClassification;
            Risk = risk;
            OwnerName = ownerName;
            TechnicalSupportContact = technicalSupportContact;
            DateCreated = dateCreated;
        }

        2 references
        public AssetModel()
        {

        }
    }
}
```

Asset Views

AddAssetForm.cshtml

```
@model Project.Models.AssetModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h4>Add Asset</h4>
<hr />
<body>
    @{ String Role;
        Role = Session["userRole"].ToString();
        if (Role == "Administrator")
        {


        }
        else if (Role == "PenetrationTester")
        {

        }
        else
        {
            Response.Redirect("~/Home/Index");
        }

    }
</body>

@using (Html.BeginForm("AddAsset", "Assets"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Add Asset</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(model => model.AssetID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.AssetID, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.AssetID, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.AssetName, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.AssetName, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DeviceCategory, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DeviceCategory, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.DeviceCategory, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Hardware, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Hardware, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Hardware, "", new { @class = "text-danger" })
        </div>
    </div>

```

```
71
72      <div class="form-group">
73          @Html.LabelFor(model => model.OperatingSystem, htmlAttributes: new { @class = "control-label col-md-2" })
74          <div class="col-md-10">
75              @Html.EditorFor(model => model.OperatingSystem, new { htmlAttributes = new { @class = "form-control" } })
76              @Html.ValidationMessageFor(model => model.OperatingSystem, "", new { @class = "text-danger" })
77          </div>
78      </div>
79
80      <div class="form-group">
81          @Html.LabelFor(model => model.MacAddress, htmlAttributes: new { @class = "control-label col-md-2" })
82          <div class="col-md-10">
83              @Html.EditorFor(model => model.MacAddress, new { htmlAttributes = new { @class = "form-control" } })
84              @Html.ValidationMessageFor(model => model.MacAddress, "", new { @class = "text-danger" })
85          </div>
86      </div>
87
88      <div class="form-group">
89          @Html.LabelFor(model => model.IPv4Address, htmlAttributes: new { @class = "control-label col-md-2" })
90          <div class="col-md-10">
91              @Html.EditorFor(model => model.IPv4Address, new { htmlAttributes = new { @class = "form-control" } })
92              @Html.ValidationMessageFor(model => model.IPv4Address, "", new { @class = "text-danger" })
93          </div>
94      </div>
95
96      <div class="form-group">
97          @Html.LabelFor(model => model.IPv6Address, htmlAttributes: new { @class = "control-label col-md-2" })
98          <div class="col-md-10">
99              @Html.EditorFor(model => model.IPv6Address, new { htmlAttributes = new { @class = "form-control" } })
100             @Html.ValidationMessageFor(model => model.IPv6Address, "", new { @class = "text-danger" })
101         </div>
102     </div>
103
104     <div class="form-group">
105         @Html.LabelFor(model => model.OfficeLocation, htmlAttributes: new { @class = "control-label col-md-2" })
106         <div class="col-md-10">
107             @Html.EditorFor(model => model.OfficeLocation, new { htmlAttributes = new { @class = "form-control" } })
108             @Html.ValidationMessageFor(model => model.OfficeLocation, "", new { @class = "text-danger" })
109         </div>
110     </div>
111
112     <div class="form-group">
113         @Html.LabelFor(model => model.BusinessUnit, htmlAttributes: new { @class = "control-label col-md-2" })
114         <div class="col-md-10">
115             @Html.EditorFor(model => model.BusinessUnit, new { htmlAttributes = new { @class = "form-control" } })
116             @Html.ValidationMessageFor(model => model.BusinessUnit, "", new { @class = "text-danger" })
117         </div>
118     </div>
119
120     <div class="form-group">
121         @Html.LabelFor(model => model.DataClassification, htmlAttributes: new { @class = "control-label col-md-2" })
122         <div class="col-md-10">
123             @Html.EditorFor(model => model.DataClassification, new { htmlAttributes = new { @class = "form-control" } })
124             @Html.ValidationMessageFor(model => model.DataClassification, "", new { @class = "text-danger" })
125         </div>
126     </div>
127
128     <div class="form-group">
129         @Html.LabelFor(model => model.Risk, htmlAttributes: new { @class = "control-label col-md-2" })
130         <div class="col-md-10">
131             @Html.EditorFor(model => model.Risk, new { htmlAttributes = new { @class = "form-control" } })
132             @Html.ValidationMessageFor(model => model.Risk, "", new { @class = "text-danger" })
133         </div>
134     </div>
135
136     <div class="form-group">
137         @Html.LabelFor(model => model.OwnerName, htmlAttributes: new { @class = "control-label col-md-2" })
138         <div class="col-md-10">
139             @Html.EditorFor(model => model.OwnerName, new { htmlAttributes = new { @class = "form-control" } })
140             @Html.ValidationMessageFor(model => model.OwnerName, "", new { @class = "text-danger" })
141         </div>
142     </div>
```

```
143
144         <div class="form-group">
145             @Html.LabelFor(model => model.TechnicalSupportContact, htmlAttributes: new { @class = "control-label col-md-2" })
146             <div class="col-md-10">
147                 @Html.EditorFor(model => model.TechnicalSupportContact, new { htmlAttributes = new { @class = "form-control" } })
148                 @Html.ValidationMessageFor(model => model.TechnicalSupportContact, "", new { @class = "text-danger" })
149             </div>
150         </div>
151
152         <div class="form-group">
153             @Html.LabelFor(model => model.DateCreated, htmlAttributes: new { @class = "control-label col-md-2" })
154             <div class="col-md-10">
155                 @Html.EditorFor(model => model.DateCreated, new { htmlAttributes = new { @class = "form-control" } })
156                 @Html.ValidationMessageFor(model => model.DateCreated, "", new { @class = "text-danger" })
157             </div>
158         </div>
159
160         <div class="form-group">
161             <div class="col-md-offset-2 col-md-10">
162                 <input type="submit" value="Save" class="btn btn-default" />
163             </div>
164         </div>
165     </div>
166     }
167
168     <div>
169         @Html.ActionLink("Back to List", "Index")
170     </div>
171
172     @section Scripts {
173         @Scripts.Render("~/bundles/jqueryval")
174     }
175
```

Details.cshtml

```
@model Project.Models.AssetModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h4> </h4>

<div>
    <br />
    <hr />
    <h4><b>Asset Intelligence</b></h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.AssetID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AssetID)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.AssetName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AssetName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DeviceCategory)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DeviceCategory)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Hardware)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Hardware)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.OperatingSystem)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.OperatingSystem)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.MacAddress)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.MacAddress)
        </dd>
```

```
79
80          <dt>
81              @Html.DisplayNameFor(model => model.IPv4Address)
82          </dt>
83
84          <dd>
85              @Html.DisplayFor(model => model.IPv4Address)
86          </dd>
87
88          <dt>
89              @Html.DisplayNameFor(model => model.IPv6Address)
90          </dt>
91
92          <dd>
93              @Html.DisplayFor(model => model.IPv6Address)
94          </dd>
95
96          <dt>
97              @Html.DisplayNameFor(model => model.OfficeLocation)
98          </dt>
99
100         <dd>
101             @Html.DisplayFor(model => model.OfficeLocation)
102         </dd>
103
104         <dt>
105             @Html.DisplayNameFor(model => model.BusinessUnit)
106         </dt>
107
108         <dd>
109             @Html.DisplayFor(model => model.BusinessUnit)
110         </dd>
111
112         <dt>
113             @Html.DisplayNameFor(model => model.DataClassification)
114         </dt>
115
116         <dd>
117             @Html.DisplayFor(model => model.DataClassification)
118         </dd>
119
120         <dt>
121             @Html.DisplayNameFor(model => model.Risk)
122         </dt>
123
124         <dd>
125             @Html.DisplayFor(model => model.Risk)
126         </dd>
127
128         <dt>
129             @Html.DisplayNameFor(model => model.OwnerName)
130         </dt>
131
132         <dd>
133             @Html.DisplayFor(model => model.OwnerName)
134         </dd>
135
136         <dt>
137             @Html.DisplayNameFor(model => model.TechnicalSupportContact)
138         </dt>
139
140         <dd>
141             @Html.DisplayFor(model => model.TechnicalSupportContact)
142         </dd>
143
144         <dt>
145             @Html.DisplayNameFor(model => model.DateCreated)
146         </dt>
147
148         <dd>
149             @Html.DisplayFor(model => model.DateCreated)
150         </dd>
151
152     </dl>
153 </div>
```

```
154     <p>
155         @Html.ActionLink("Edit", "Edit", new {  id = Model.AssetID }) |
156         @Html.ActionLink("Back to List", "Index")
157     </p>
158
```

Index.cshtml

```
@model IEnumerable<Project.Models.AssetModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2> </h2>

<p>
    <br />
    <hr />
    <h4><b>Asset Inventory</b></h4>
    <hr />
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.AssetID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.AssetName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DeviceCategory)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Hardware)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.OperatingSystem)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.MacAddress)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.IPv4Address)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.IPv6Address)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.OfficeLocation)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BusinessUnit)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DataClassification)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Risk)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.OwnerName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TechnicalSupportContact)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateCreated)
```

```
79              </th>
80              <th></th>
81          </tr>
82
83          @foreach (var item in Model)
84          {
85              <tr>
86                  <td>
87                      @Html.DisplayFor(modelItem => item.AssetID)
88                  </td>
89                  <td>
90                      @Html.DisplayFor(modelItem => item.AssetName)
91                  </td>
92                  <td>
93                      @Html.DisplayFor(modelItem => item.DeviceCategory)
94                  </td>
95                  <td>
96                      @Html.DisplayFor(modelItem => item.Hardware)
97                  </td>
98                  <td>
99                      @Html.DisplayFor(modelItem => item.OperatingSystem)
100                 </td>
101                 <td>
102                     @Html.DisplayFor(modelItem => item.MacAddress)
103                 </td>
104                 <td>
105                     @Html.DisplayFor(modelItem => item.IPv4Address)
106                 </td>
107                 <td>
108                     @Html.DisplayFor(modelItem => item.IPv6Address)
109                 </td>
110                 <td>
111                     @Html.DisplayFor(modelItem => item.OfficeLocation)
112                 </td>
113                 <td>
114                     @Html.DisplayFor(modelItem => item.BusinessUnit)
115                 </td>
116                 <td>
117                     @Html.DisplayFor(modelItem => item.DataClassification)
118                 </td>
119                 <td>
120                     @Html.DisplayFor(modelItem => item.Risk)
121                 </td>
122                 <td>
123                     @Html.DisplayFor(modelItem => item.OwnerName)
124                 </td>
125                 <td>
126                     @Html.DisplayFor(modelItem => item.TechnicalSupportContact)
127                 </td>
128                 <td>
129                     @Html.DisplayFor(modelItem => item.DateCreated)
130                 </td>
131                 <td>
132                     @Html.ActionLink("Edit", "Edit", new { id = item.AssetID }) |
133                     @Html.ActionLink("Details", "Details", new { id = item.AssetID }) |
134                     @Html.ActionLink("Delete", "Delete", new { id = item.AssetID })
135                 </td>
136             </tr>
137         }
138
139     </table>
140
```

UpdateAssetForm.cshtml

```
@model Project.Models.AssetModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2> </h2>


@using (Html.BeginForm("UpdateAsset", "Assets"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Update Asset</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(model => model.AssetID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.AssetID, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.AssetID, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.AssetName, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.AssetName, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DeviceCategory, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DeviceCategory, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.DeviceCategory, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Hardware, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Hardware, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Hardware, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.OperatingSystem, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.OperatingSystem, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.OperatingSystem, "", new { @class = "text-danger" })
        </div>
    </div>

```

```
<div class="form-group">
    @Html.LabelFor(model => model.MacAddress, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.MacAddress, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.MacAddress, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.IPv4Address, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.IPv4Address, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.IPv4Address, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.IPv6Address, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.IPv6Address, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.IPv6Address, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.OfficeLocation, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.OfficeLocation, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.OfficeLocation, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.BusinessUnit, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.BusinessUnit, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.BusinessUnit, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.DataClassification, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.DataClassification, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.DataClassification, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Risk, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Risk, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Risk, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.OwnerName, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.OwnerName, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.OwnerName, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.TechnicalSupportContact, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.TechnicalSupportContact, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.TechnicalSupportContact, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.DateCreated, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.DateCreated, new { htmlAttributes = new { @class = "form-control" } })
    </div>
```

```
154         </div>
155         <input type="hidden" name="AssetID" value="@Model.AssetID" />
156
157         <div class="form-group">
158             <div class="col-md-offset-2 col-md-10">
159                 <input type="submit" value="Create" class="btn btn-default" />
160             </div>
161         </div>
162     </div>
163     }
164
165     <div>
166         @Html.ActionLink("Back to List", "Index")
167     </div>
168
169     @section Scripts {
170         @Scripts.Render("~/bundles/jqueryval")
171     }
172
```

CustomAuthorizationAttribute

```
using Project.Models;
using Project.Services.Business;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Web.Mvc;

namespace Project.Controllers
{
    57 references
    internal class CustomAuthorizationAttribute : FilterAttribute, IAuthorizationFilter
    {
        0 references
        public void OnAuthorization(AuthorizationContext filterContext)
        {
            SecurityService securityService = new SecurityService();

            //get user from session variable
            UserModel user = (UserModel)filterContext.HttpContext.Session["user"];

            bool success = false;
            if (user != null)
            {



                success = securityService.Authenticate(user);

            }

            if (success)
            {


            }
            else
            { //Make Unauthorised page here
                filterContext.Result = new RedirectResult("~/Home/Index");
            }

            if (user == null)
            {

                filterContext.Result = new RedirectResult("/Home/Index");
            }




            }

        }

    }
```

SecurityService.cs

```
1   using Project.Models;
2   using Project.Services.Data;
3   using System;
4   using System.Collections.Generic;
5   using System.Linq;
6   using System.Web;
7
8   namespace Project.Services.Business
9   {
        4 references
10      public class SecurityService
11      {
12          SecurityDAO daoService = new SecurityDAO();
13
            2 references
14          public bool Authenticate(UserModel user)
15          {
16              return daoService.FindByUser(user);
17
18          }
19
20
21      }
22  }
```

SecurityDAO.cs

```
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Services.Data
{
    2 references
    public class SecurityDAO
    {
        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        1 reference
        internal bool FindByUser(UserModel user)
        {

            bool success = false;
            string sqlQuery = "SELECT * from dbo.Users WHERE Username = @Username AND Password = @Password";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                SqlCommand command = new SqlCommand(sqlQuery, connection);

                command.Parameters.Add("@Username", System.Data.SqlDbType.VarChar, 50).Value = user.Username;
                command.Parameters.Add("@Password", System.Data.SqlDbType.VarChar, 50).Value = user.Password;



                try
                {
                    connection.Open();
                    SqlDataReader reader = command.ExecuteReader();
                    if (reader.HasRows)
                    {
                        success = true;
                        while (reader.Read())
                        {
                            user.UsersName = reader.GetString(1);
                            user.Username = reader.GetString(2);
                            user.Password = reader.GetString(3);
                            user.Role = reader.GetString(4);
                            user.Department = reader.GetString(5);
                            user.Company = reader.GetString(6);



                        }
                    }
                    else
                    {
                        success = false;
                    }
                    reader.Close();
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                }

            }
            return success;
        }
    }
}
```

LoginController.cs

```
using Project.Models;
using Project.Services.Business;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace Project.Controllers
{
    1 reference
    public class LoginController : Controller
    {
        // GET: Login
        0 references
        public ActionResult Index()
        {
            return View("Login");
        }

        0 references
        public ActionResult Login(UserModel user)
        {
            // return "Results: Username = " + userModel.Username + "Password= " + userModel.Password;
            SecurityService securityService = new SecurityService();
            Boolean success = securityService.Authenticate(user);

            if (success)
            {

                Session["user"] = user;
               Session["userRole"] = user.Role;
                Session["username"] = user.Username;
                Session["usersname"] = user.UsersName;
                //Session["user"] = UserModel.LoggedInUser;
                return RedirectToAction("Index", "Dashboard", user);
               // return View("Dashboard", user);
            }
            else
            {
                Session.Clear();
                return View("LoginFailure");
            }
        }

        0 references
        public ActionResult Logout()
        {
            Session.Clear();
            Session["user"] = null;
            Session["userRole"] = null;
            return Redirect("~/Home/Index");
          // return RedirectToAction("Home","Index", null);

        }


        [CustomAuthorization]
        1 reference
        public ActionResult onPrivateURL()
        {
            return Content("Private information here only logged in user can see");
        }
    }
}
```

UserModel.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Web;
5
6   namespace Project.Models
7   {
        6 references
8       public class UserModel
9       {
            3 references
10          public string Username { get; set; }
            2 references
11          public string UsersName { get; set; }
            2 references
12          public string Password { get; set; }
            0 references
13          public string UserID { get; set; }
            2 references
14          public string Role { get; set; }
            1 reference
15          public string Department { get; set; }
            1 reference
16          public string Company { get; set; }
            0 references
17          public string RolePrivileges { get; set; }
            0 references
18          public string LoggedInUser { get; set; }
            0 references
19          public UserModel()
20          {
21
22          }
23      }
24
25
26  }
```

Login Views

Login.cshtml

```
@model Project.Models.UserModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>


@using (Html.BeginForm("Login", "Login", FormMethod.Post))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br/>
    <hr />
    <h4><b>Login</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(model => model.Username, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Username, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Username, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.PasswordFor(model => model.Password, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Login" class="btn btn-default" />
        </div>
    </div>
</div>
}


@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

```

LoginFailure.cs

```
@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>LoginFailure</h2>


```

LoginSuccess.cshtml

```
@model Project.Models.UserModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>LoginSuccess</h2>

<div>
    <h4>UserModel</h4>
    <hr />
    <div>

        @Session["user"] = UserModel.LoggedInUser;
        @Session["userRole"]

    </div>
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Username)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Username)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Password)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Password)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.UserID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UserID)
        </dd>

    </dl>
</div>

<p>
    @Html.ActionLink("Edit", "Edit", new { /* id = Model.PrimaryKey */ }) |
    @Html.ActionLink("Back to List", "Index")
</p>

```

DashboardController.cs

```
using Project.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Dynamic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Controllers
{
    0 references
    public class DashboardController : Controller
    {
        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        [CustomAuthorization]
        // GET: Dashboard
        0 references
        public ActionResult Index()
        {
            dynamic model = new ExpandoObject();
            model.OpenTestModel = GetOpenTests();
            model.ClosedTestModel = GetClosedTests();
            model.OpenVulnerabilityModel = GetOpenVulnerability();
            model.ClosedVulnerabilityModel = GetClosedVulnerability();
            model.HighSeverityModel = GetHighSeverity();
            model.MediumSeverityModel = GetMediumSeverity();
            model.LowSeverityModel = GetLowSeverity();
            return View(model);
        }
        [CustomAuthorization]
        1 reference
        private static List<OpenTestModel> GetOpenTests()
        {
            string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
              Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
            List<OpenTestModel> openTests = new List<OpenTestModel>();
            string sqlQuery = "SELECT  *  FROM dbo.TestAssignment WHERE TicketStatus = 'Open'";
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                SqlCommand command = new SqlCommand(sqlQuery, connection);
                {
                    connection.Open();
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            openTests.Add(new OpenTestModel
                            {
                                TestAssignmentID = (int)reader["TestAssignmentID"],
                                TestName = reader["TestName"].ToString(),
                                TicketUser = reader["TicketUser"].ToString(),
                                DateAssigned = (DateTime)reader["DateAssigned"],
                                DateClosed = (DateTime)reader["DateClosed"],
                                TicketStatus = reader["TicketStatus"].ToString(),
                                TestTicketNotes = reader["TestTicketNotes"].ToString(),
                            }) ;
                        }
                    }
                    connection.Close();
                    return openTests;
                }
            }
        }
```

```
63          [CustomAuthorization]
            1 reference
64          private static List<ClosedTestModel> GetClosedTests()
65          {
66              string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
                Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
67
68              List<ClosedTestModel> closedTests = new List<ClosedTestModel>();
69              string sqlQuery = "SELECT  *  FROM dbo.TestAssignment WHERE TicketStatus = 'Closed'";
70              using (SqlConnection connection = new SqlConnection(connectionString))
71              {
72                  SqlCommand command = new SqlCommand(sqlQuery, connection);
73                  {
74                      connection.Open();
75                      using (SqlDataReader reader = command.ExecuteReader())
76                      {
77                          while (reader.Read())
78                          {
79                              closedTests.Add(new ClosedTestModel
80                              {
81                                  TestAssignmentID = (int)reader["TestAssignmentID"],
82                                  TestName = reader["TestName"].ToString(),
83                                  TicketUser = reader["TicketUser"].ToString(),
84                                  DateAssigned = (DateTime)reader["DateAssigned"],
85                                  DateClosed = (DateTime)reader["DateClosed"],
86                                  TicketStatus = reader["TicketStatus"].ToString(),
87                                  TestTicketNotes = reader["TestTicketNotes"].ToString(),
88                              });
89                          }
90                      }
91                      connection.Close();
92                      return closedTests;
93                  }
94              }
95          }
96          [CustomAuthorization]
            1 reference
97          private static List<OpenVulnerabilityModel> GetOpenVulnerability()
98          {
99              string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
                Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
100
101             List<OpenVulnerabilityModel> openVulnerabilities = new List<OpenVulnerabilityModel>();
102             string sqlQuery = "SELECT  *  FROM dbo.VulnerabilityAssignment WHERE TicketStatus = 'Open'";
103             using (SqlConnection connection = new SqlConnection(connectionString))
104             {
105                 SqlCommand command = new SqlCommand(sqlQuery, connection);
106                 {
107                     connection.Open();
108                     using (SqlDataReader reader = command.ExecuteReader())
109                     {
110                         while (reader.Read())
111                         {
112                             openVulnerabilities.Add(new OpenVulnerabilityModel
113                             {
114                                 VulnerabilityAssignmentID = (int)reader["VulnerabilityAssignmentID"],
115                                 VulnerabilityID = (int)reader["VulnerabilityID"],
116                                 TicketUser = reader["TicketUser"].ToString(),
117                                 DateAssigned = (DateTime)reader["DateAssigned"],
118                                 DateClosed = (DateTime)reader["DateClosed"],
119                                 TicketStatus = reader["TicketStatus"].ToString(),
120                                 VulnerabilityTicketNotes = reader["VulnerabilityTicketNotes"].ToString(),
121                             });
122                         }
123                     }
124                     connection.Close();
125                     return openVulnerabilities;
126                 }
127             }
128         }
```

```
129         [CustomAuthorization]
            1 reference
130         private static List<ClosedVulnerabilityModel> GetClosedVulnerability()
131         {
132             string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
                Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
133
134             List<ClosedVulnerabilityModel> closedVulnerabilities = new List<ClosedVulnerabilityModel>();
135             string sqlQuery = "SELECT  *  FROM dbo.VulnerabilityAssignment WHERE TicketStatus = 'Closed'";
136             using (SqlConnection connection = new SqlConnection(connectionString))
137             {
138                 SqlCommand command = new SqlCommand(sqlQuery, connection);
139                 {
140                     connection.Open();
141                     using (SqlDataReader reader = command.ExecuteReader())
142                     {
143                         while (reader.Read())
144                         {
145                             closedVulnerabilities.Add(new ClosedVulnerabilityModel
146                             {
147                                 VulnerabilityAssignmentID = (int)reader["VulnerabilityAssignmentID"],
148                                 VulnerabilityID = (int)reader["VulnerabilityID"],
149                                 TicketUser = reader["TicketUser"].ToString(),
150                                 DateAssigned = (DateTime)reader["DateAssigned"],
151                                 DateClosed = (DateTime)reader["DateClosed"],
152                                 TicketStatus = reader["TicketStatus"].ToString(),
153                                 VulnerabilityTicketNotes = reader["VulnerabilityTicketNotes"].ToString(),
154                             });
155                         }
156                     }
157                     connection.Close();
158                     return closedVulnerabilities;
159                 }
160             }
161         }
162         [CustomAuthorization]
            1 reference
163         private static List<HighSeverityModel> GetHighSeverity()
164         {
165             string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
                Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
166
167             List<HighSeverityModel> highSeverities = new List<HighSeverityModel>();
168             string sqlQuery = "SELECT  *  FROM dbo.Vulnerability WHERE Severity = 'High Risk'";
169             using (SqlConnection connection = new SqlConnection(connectionString))
170             {
171                 SqlCommand command = new SqlCommand(sqlQuery, connection);
172                 {
173                     connection.Open();
174                     using (SqlDataReader reader = command.ExecuteReader())
175                     {
176                         while (reader.Read())
177                         {
178                             highSeverities.Add(new HighSeverityModel
179                             {
180                                 VulnerabilityID = (int)reader["VulnerabilityID"],
181                                 TestName = reader["TestName"].ToString(),
182                                 DateOpen = (DateTime)reader["DateOpen"],
183                                 DateClosed = (DateTime)reader["DateClosed"],
184                                 Severity = reader["Severity"].ToString(),
185                                 DREADTotal = (decimal)reader["DREADTotal"],
186                                 StatusID = reader["StatusID"].ToString(),
187                             });
188                         }
189                     }
190                     connection.Close();
191                     return highSeverities;
192                 }
```

```
            }
        }
    }
    [CustomAuthorization]
    1 reference
    private static List<MediumSeverityModel> GetMediumSeverity()
    {
        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        List<MediumSeverityModel> mediumSeverities = new List<MediumSeverityModel>();
        string sqlQuery = "SELECT  *  FROM dbo.Vulnerability WHERE Severity = 'Medium Risk'";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(sqlQuery, connection);
            {
                connection.Open();
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        mediumSeverities.Add(new MediumSeverityModel
                        {
                            VulnerabilityID = (int)reader["VulnerabilityID"],
                            TestName = reader["TestName"].ToString(),
                            DateOpen = (DateTime)reader["DateOpen"],
                            DateClosed = (DateTime)reader["DateClosed"],
                            Severity = reader["Severity"].ToString(),
                            DREADTotal = (decimal)reader["DREADTotal"],
                            StatusID = reader["StatusID"].ToString(),
                        });
                    }
                }
                connection.Close();
                return mediumSeverities;
            }
        }
    }
    [CustomAuthorization]
    1 reference
    private static List<LowSeverityModel> GetLowSeverity()
    {
        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        List<LowSeverityModel> lowSeverities = new List<LowSeverityModel>();
        string sqlQuery = "SELECT  *  FROM dbo.Vulnerability WHERE Severity = 'Low Risk'";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(sqlQuery, connection);
            {
                connection.Open();
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        lowSeverities.Add(new LowSeverityModel
                        {
                            VulnerabilityID = (int)reader["VulnerabilityID"],
                            TestName = reader["TestName"].ToString(),
                            DateOpen = (DateTime)reader["DateOpen"],
                            DateClosed = (DateTime)reader["DateClosed"],
                            Severity = reader["Severity"].ToString(),
                            DREADTotal = (decimal)reader["DREADTotal"],
                            StatusID = reader["StatusID"].ToString(),
                        });
                    }
                }
                connection.Close();
                return lowSeverities;
            }
        }
    }
}
```

ClosedTestModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    4 references
    public class ClosedTestModel
    {
        [Display(Name = "Test Assignment ID")]
        1 reference
        public int TestAssignmentID { get; set; }
        [Display(Name = "Test Name")]
        1 reference
        public string TestName { get; set; }
        [Display(Name = "Assignee")]
        1 reference
        public string TicketUser { get; set; }
        [Display(Name = "Date Assigned")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateAssigned { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        1 reference
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        1 reference
        public String TestTicketNotes { get; set; }
    }
}
```

ClosedVulnerabilityModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    4 references
    public class ClosedVulnerabilityModel
    {
        [Display(Name = "Vulnerability Assignment ID")]
        1 reference
        public int VulnerabilityAssignmentID { get; set; }
        [Display(Name = "Vulnerability ID")]
        1 reference
        public int VulnerabilityID { get; set; }
        [Display(Name = "Assignee")]
        1 reference
        public string TicketUser { get; set; }
        [Display(Name = "Date Asigned")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateAssigned { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        1 reference
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        1 reference
        public String VulnerabilityTicketNotes { get; set; }
    }
}
```

HighSeverityModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    4 references
    public class HighSeverityModel
    {
        [Display(Name = "Vulnerability ID")]
        1 reference
        public int VulnerabilityID { get; set; }
        [Display(Name = "Test Name")]
        1 reference
        public string TestName { get; set; }
        [Display(Name = "Date Open")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateOpen { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateClosed { get; set; }
        [Display(Name = "Severity")]
        1 reference
        public string Severity { get; set; }
        [Display(Name = "DREAD Score Rating")]
        1 reference
        public decimal DREADTotal { get; set; }
        [Display(Name = "Status")]
        1 reference
        public string StatusID { get; set; }

    }
}
```

LowSeverityModel.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel.DataAnnotations;
4   using System.Linq;
5   using System.Web;
6
7   namespace Project.Models
8   {
        4 references
9       public class LowSeverityModel
10      {
11          [Display(Name = "Vulnerability ID")]
            1 reference
12          public int VulnerabilityID { get; set; }
13          [Display(Name = "Test Name")]
            1 reference
14          public string TestName { get; set; }
15          [Display(Name = "Date Open")]
16          [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
            1 reference
17          public DateTime DateOpen { get; set; }
18          [Display(Name = "Date Closed")]
19          [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
            1 reference
20          public DateTime DateClosed { get; set; }
21          [Display(Name = "Severity")]
            1 reference
22          public string Severity { get; set; }
23          [Display(Name = "DREAD Score Rating")]
            1 reference
24          public decimal DREADTotal { get; set; }
25          [Display(Name = "Status")]
            1 reference
26          public string StatusID { get; set; }
27      }
28  }
```

MediumSeverityModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    4 references
    public class MediumSeverityModel
    {
        [Display(Name = "Vulnerability ID")]
        1 reference
        public int VulnerabilityID { get; set; }
        [Display(Name = "Test Name")]
        1 reference
        public string TestName { get; set; }
        [Display(Name = "Date Open")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateOpen { get; set; }
        [Display(Name = "Daate Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateClosed { get; set; }
        [Display(Name = "Severity")]
        1 reference
        public string Severity { get; set; }
        [Display(Name = "DREAD Score Rating")]
        1 reference
        public decimal DREADTotal { get; set; }
        [Display(Name = "Status")]
        1 reference
        public string StatusID { get; set; }
    }
}
```

OpenTestModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    4 references
    public class OpenTestModel
    {
        [Display(Name = "Test Assignment ID")]
        1 reference
        public int TestAssignmentID { get; set; }
        [Display(Name = "Test Name")]
        1 reference
        public string TestName { get; set; }
        [Display(Name = "Ticket User")]
        1 reference
        public string TicketUser { get; set; }
        [Display(Name = "Date Assigned")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateAssigned { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        1 reference
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        1 reference
        public String TestTicketNotes { get; set; }

    }
}
```

OpenVulnerabilityModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{
    4 references
    public class OpenVulnerabilityModel
    {
        [Display(Name = "Vulnerability Assignment ID")]
        1 reference
        public int VulnerabilityAssignmentID { get; set; }
        [Display(Name = "Vulnerability ID")]
        1 reference
        public int VulnerabilityID { get; set; }
        [Display(Name = "Assignee")]
        1 reference
        public string TicketUser { get; set; }
        [Display(Name = "Date Assigned")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateAssigned { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        1 reference
        public DateTime DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        1 reference
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        1 reference
        public String VulnerabilityTicketNotes { get; set; }
    }
}
```

Dashboard View

Index.cshtml

```
@using Project.Models
@model dynamic
@{
    Layout = null;
}
@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <br/>
    <hr/>
    <h2>Have a great day @Session["usersname"]!</h2>
    <hr />
    <h3>Open Tests</h3>
    <table class="table">
        <tr>
            <th>TestAssignmentID</th>
            <th>TestName</th>
            <th>TicketUser</th>
            <th>DateAssigned</th>
            <th>DateClosed</th>
            <th>TicketStatus</th>
            <th>TestTicketNotes</th>
            <th></th>
        </tr>
        @foreach (OpenTestModel openTests in Model.OpenTestModel)
        {
            <tr>
                <td>@openTests.TestAssignmentID</td>
                <td>@openTests.TestName</td>
                <td>@openTests.TicketUser</td>
                <td>@openTests.DateAssigned</td>
                <td>@openTests.DateClosed</td>
                <td>@openTests.TicketStatus</td>
                <td>@openTests.TestTicketNotes</td>
            </tr>
        }
    </table>
    <hr />
    <h3>Closed Tests</h3>
    <table class="table">
        <tr>
            <th>TestAssignmentID</th>
            <th>TestName</th>
            <th>TicketUser</th>
            <th>DateAssigned</th>
            <th>DateClosed</th>
            <th>TicketStatus</th>
            <th>TestTicketNotes</th>
            <th></th>
        </tr>
        @foreach (ClosedTestModel closedTests in Model.ClosedTestModel)
        {
            <tr>
                <td>@closedTests.TestAssignmentID</td>
                <td>@closedTests.TestName</td>
                <td>@closedTests.TicketUser</td>
                <td>@closedTests.DateAssigned</td>
                <td>@closedTests.DateClosed</td>
                <td>@closedTests.TicketStatus</td>
                <td>@closedTests.TestTicketNotes</td>
            </tr>
        }
    </table>
    <hr />
```

```
<h3>Open Vulnerabilities</h3>
<table class="table">
    <tr>
        <th>VulnerabilityAssignmentID</th>
        <th>VulnerabilityID</th>
        <th>TicketUser</th>
        <th>DateAssigned</th>
        <th>DateClosed</th>
        <th>TicketStatus</th>
        <th>VulnerabilityTicketNotes</th>
        <th></th>
    </tr>
    @foreach (OpenVulnerabilityModel openVulnerabilities in Model.OpenVulnerabilityModel)
    {
        <tr>
            <td>@openVulnerabilities.VulnerabilityAssignmentID</td>
            <td>@openVulnerabilities.VulnerabilityID</td>
            <td>@openVulnerabilities.TicketUser</td>
            <td>@openVulnerabilities.DateAssigned</td>
            <td>@openVulnerabilities.DateClosed</td>
            <td>@openVulnerabilities.TicketStatus</td>
            <td>@openVulnerabilities.VulnerabilityTicketNotes</td>
        </tr>
    }
</table>
<hr />

<h3>Closed Vulnerabilities</h3>
<table class="table">
    <tr>
        <th>VulnerabilityAssignmentID</th>
        <th>VulnerabilityID</th>
        <th>TicketUser</th>
        <th>DateAssigned</th>
        <th>DateClosed</th>
        <th>TicketStatus</th>
        <th>VulnerabilityTicketNotes</th>
        <th></th>
    </tr>
    @foreach (ClosedVulnerabilityModel closedVulnerabilities in Model.ClosedVulnerabilityModel)
    {
        <tr>
            <td>@closedVulnerabilities.VulnerabilityAssignmentID</td>
            <td>@closedVulnerabilities.VulnerabilityID</td>
            <td>@closedVulnerabilities.TicketUser</td>
            <td>@closedVulnerabilities.DateAssigned</td>
            <td>@closedVulnerabilities.DateClosed</td>
            <td>@closedVulnerabilities.TicketStatus</td>
            <td>@closedVulnerabilities.VulnerabilityTicketNotes</td>
        </tr>
    }
</table>
<hr />

<h3>High Severity Vulnerabilities</h3>
<table class="table">
    <tr>
        <th>VulnerabilityID</th>
        <th>TestName</th>
        <th>DateOpen</th>
        <th>DateClosed</th>
        <th>Severity</th>
        <th>DREADTotal</th>
        <th>StatusID</th>
        <th></th>
    </tr>
    @foreach (HighSeverityModel highSeverities in Model.HighSeverityModel)
    {
        <tr>
            <td>@highSeverities.VulnerabilityID</td>
            <td>@highSeverities.TestName</td>
            <td>@highSeverities.DateOpen</td>
            <td>@highSeverities.DateClosed</td>
            <td>@highSeverities.Severity</td>
            <td>@highSeverities.DREADTotal</td>
```

```
                <td>@highSeverities.StatusID</td>
            </tr>
        }
    </table>
    <hr />
    <h3>Medium Severity Vulnerabilities</h3>
    <table class="table">
        <tr>
            <th>VulnerabilityID</th>
            <th>TestName</th>
            <th>DateOpen</th>
            <th>DateClosed</th>
            <th>Severity</th>
            <th>DREADTotal</th>
            <th>StatusID</th>
            <th></th>
        </tr>
        @foreach (MediumSeverityModel mediumSeverities in Model.MediumSeverityModel)
        {
            <tr>
                <td>@mediumSeverities.VulnerabilityID</td>
                <td>@mediumSeverities.TestName</td>
                <td>@mediumSeverities.DateOpen</td>
                <td>@mediumSeverities.DateClosed</td>
                <td>@mediumSeverities.Severity</td>
                <td>@mediumSeverities.DREADTotal</td>
                <td>@mediumSeverities.StatusID</td>
            </tr>
        }
    </table>
    <hr />

    <h3>Low Severity Vulnerabilities</h3>
    <table class="table">
        <tr>
            <th>VulnerabilityID</th>
            <th>TestName</th>
            <th>DateOpen</th>
            <th>DateClosed</th>
            <th>Severity</th>
            <th>DREADTotal</th>
            <th>StatusID</th>
            <th></th>
        </tr>
        @foreach (LowSeverityModel lowSeverities in Model.LowSeverityModel)
        {
            <tr>
                <td>@lowSeverities.VulnerabilityID</td>
                <td>@lowSeverities.TestName</td>
                <td>@lowSeverities.DateOpen</td>
                <td>@lowSeverities.DateClosed</td>
                <td>@lowSeverities.Severity</td>
                <td>@lowSeverities.DREADTotal</td>
                <td>@lowSeverities.StatusID</td>
            </tr>
        }
    </table>
    <hr />
</body>
</html>
```

HomeController.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace Project.Controllers
8  {
       0 references
9      public class HomeController : Controller
10     {
           0 references
11         public ActionResult Index()
12         {
13             if(Session["user"] != null) {
14                 return RedirectToAction("Index", "Dashboard");
15
16             }
17             return View();
18         }
19
           0 references
20         public ActionResult About()
21         {
22             ViewBag.Message = "Your application description page.";
23
24             return View();
25         }
26
           0 references
27         public ActionResult Contact()
28         {
29             ViewBag.Message = "Your contact page.";
30
31             return View();
32         }
33     }
34 }
```

Home View

Index.cshtml

```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>Vulnerability Management</h1>
    <p class="lead">Putting the ease in Vulnerabilities.</a></p>
</div>

<div class="row">
    <div class="col-md-4">
        <h2>Getting started</h2>
        <p>Login to access your daily tasks</p>
    </div>
    <div class="col-md-4">
        <h2>Assistance</h2>
        <p>Should you require assistance please contact us at Assistance@VulnerabilityManagement.com</p>
    </div>
    <div class="col-md-4">
        <h2>Vulnerability Management Solutions</h2>
        <p>Asset Management</p>
        <p>Test Management</p>
        <p>Vulnerability Management</p>
        <p>Track A Vulnerability From Discovery and Assignment Through to Mitigation and Close </p>

    </div>
</div>
```

TestAssignmentController.cs

```
using Project.Controllers.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Controllers
{
    0 references
    public class TestAssignmentController : Controller
    {
        private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
        [CustomAuthorization]
        0 references
        public ActionResult Index()
        {
            List<TestAssignmentModel> testAssignments = new List<TestAssignmentModel>();
            TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
            testAssignments = testAssignmentDAO.FetchAll();

            return View("Index", testAssignments);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Details(int id)
        {
            TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
            TestAssignmentModel testAssignment = testAssignmentDAO.FetchOne(id);
            return View("Details", testAssignment);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Create()
        {
            TestAssignmentModel testAssignment = new TestAssignmentModel();

            //new
            testAssignment.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
            testAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role  = 'PenetrationTester' ORDER BY Username ", "Username", "Username");
            testAssignment.ListTestStatus = PopulateDropDown(" SELECT StatusType FROM dbo.TestStatus", "StatusType", "StatusType");
            return View("CreateTestAssignmentForm", testAssignment);
        }
        [CustomAuthorization]
        6 references
        private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
        {
            List<SelectListItem> items = new List<SelectListItem>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(query))
                {
                    cmd.Connection = connection;
                    connection.Open();
                    using (SqlDataReader sdr = cmd.ExecuteReader())
                    {
                        while (sdr.Read())
                        {
                            items.Add(new SelectListItem
                            {
                                Text = sdr[textColumn].ToString(),
                                Value = sdr[valueColumn].ToString()
                            });
                        }
                    }
                    connection.Close();
                }
            }

            return items;
        }
```

```
69          [CustomAuthorization]
            0 references
70          public ActionResult CreateTestAssignment(TestAssignmentModel testAssignmentModel)
71          {
72              //save it to the database
73
74              TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
75              testAssignmentDAO.Create(testAssignmentModel);
76              List<TestAssignmentModel> testAssignments = new List<TestAssignmentModel>();
77             //How to return full list of assignments instead of the one
78              testAssignments = testAssignmentDAO.FetchAll();
79              return View("Index",testAssignments);
80          }
81          [CustomAuthorization]
            0 references
82          public ActionResult Edit(int id)
83          {
84              //new
85
86              TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
87              TestAssignmentModel testAssignment = testAssignmentDAO.FetchOne(id);
88              testAssignment.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
89              testAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role  = 'PenetrationTester' ORDER BY Username ", "Username", "Username");
90              testAssignment.ListTestStatus = PopulateDropDown(" SELECT StatusType FROM dbo.TestStatus", "StatusType", "StatusType");
91              return View("UpdateTestAssignmentForm", testAssignment);
92          }
93          [CustomAuthorization]
94          [HttpPost]
            0 references
95          public ActionResult UpdateTestAssignment(TestAssignmentModel testAssignmentModel)
96          {
97              //save it to the database
98              TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
99
100             testAssignmentDAO.Update(testAssignmentModel);
101             return View("Details", testAssignmentModel);
102         }
103         [CustomAuthorization]
            0 references
104         public ActionResult Delete(int id)
105         {
106             TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
107             testAssignmentDAO.Delete(id);
108             List<TestAssignmentModel> testAssignments = testAssignmentDAO.FetchAll();
109             return View("Index", testAssignments);
110         }
111     }
112 }
```

TestAssignmentDAO.cs

```
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;

namespace Project.Controllers.Data
{
    14 references
    internal class TestAssignmentDAO
    {
        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        4 references
        public List<TestAssignmentModel> FetchAll()
        {
            List<TestAssignmentModel> returnList = new List<TestAssignmentModel>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * FROM dbo.TestAssignment";

                SqlCommand command = new SqlCommand(sqlQuery, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //creating new TestAssignment object and adding it to the list to be returned
                        TestAssignmentModel testAssignment = new TestAssignmentModel();
                        testAssignment.TestAssignmentID = reader.GetInt32(0);
                        testAssignment.TestName = reader.GetString(1);
                        testAssignment.TicketUser = reader.GetString(2);
                        testAssignment.DateAssigned = reader.GetDateTime(3);
                        testAssignment.DateClosed = reader.GetDateTime(4);
                        testAssignment.TicketStatus = reader.GetString(5);
                        testAssignment.TestTicketNotes = reader.GetString(6);
                        returnList.Add(testAssignment);
                    }
                }
            }

                return returnList;
        }

        2 references
        public TestAssignmentModel FetchOne(int id)
        {

            //access database
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * FROM dbo.TestAssignment WHERE TestAssignmentID = @id";

                SqlCommand command = new SqlCommand(sqlQuery, connection);
                command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();

                TestAssignmentModel testAssignment = new TestAssignmentModel();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //create new test object and add that to the list to be returned

                        testAssignment.TestAssignmentID = reader.GetInt32(0);
                        testAssignment.TestName = reader.GetString(1);
                        testAssignment.TicketUser = reader.GetString(2);
                        testAssignment.DateAssigned = reader.GetDateTime(3);
                        testAssignment.DateClosed = reader.GetDateTime(4);
                        testAssignment.TicketStatus = reader.GetString(5);
                        testAssignment.TestTicketNotes = reader.GetString(6);


```

```
                }
            }
            return testAssignment;
        }

    }

    1 reference
    internal int Delete(int id)
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            //prepared statement
            //update
            string sqlQuery = "DELETE FROM dbo.TestAssignment WHERE TestAssignmentID = @id";

            //associate @id with Id parameter

            SqlCommand command = new SqlCommand(sqlQuery, connection);

            command.Parameters.Add("@Id", System.Data.SqlDbType.VarChar, 1000).Value = id;


            connection.Open();
            int deletedID = command.ExecuteNonQuery();

            return deletedID;
        }
    }

    1 reference
    public int Create(TestAssignmentModel testAssignmentModel)
    {

        //access database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            string sqlQuery = "INSERT INTO dbo.TestAssignment Values(@TestName, @TicketUser, @DateAssigned, @DateClosed, @TicketStatus, @TestTicketNotes)";

            SqlCommand command = new SqlCommand(sqlQuery, connection);
            command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar,1000).Value = testAssignmentModel.TestName;
            command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketUser;
            command.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateAssigned;
            command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateClosed;
            command.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
            command.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar).Value = testAssignmentModel.TestTicketNotes;

            //Update Test that ticket is assigned
            string sqlQuery1 = "Update dbo.Test SET StartDate = @StartDate, EndDate = @EndDate, TestStatus = @TestStatus, TestNotes = @TestNotes  WHERE dbo.Test.TestName = @TestName";

            //associate @id with Id parameter
            SqlCommand command1 = new SqlCommand(sqlQuery1, connection);

            command1.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar,1000).Value = testAssignmentModel.TestName;
            command1.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateAssigned;
            command1.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateClosed;
            command1.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
            command1.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar).Value = testAssignmentModel.TestTicketNotes;

            //Insert Assignment To Vulnerability Log
            string sqlQuery2 = "INSERT INTO dbo.TestLog VALUES(@TestName, @TicketUser, @CurrentTimeStamp, @TicketStatus,  @TestTicketNotes)";

            //associate @id with Id parameter
            SqlCommand command2 = new SqlCommand(sqlQuery2, connection);

            command2.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
            command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.CurrentUser;
            command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
            command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
            command2.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestTicketNotes;

```

```
142
143                 connection.Open();
144                 int newTestAssignmentID = command.ExecuteNonQuery();
145                 command1.ExecuteNonQuery();
146                 command2.ExecuteNonQuery();
147                 return newTestAssignmentID;
148             }
149         }
150
151         1 reference
152         public int Update(TestAssignmentModel testAssignmentModel)
153         {
154             //access the database
155             using (SqlConnection connection = new SqlConnection(connectionString))
156             {
157                 //prepared statement
158                 //update
159                 string sqlQuery = "UPDATE dbo.TestAssignment SET TestName = @TestName, TicketUser = @TicketUser, DateAssigned = @DateAssigned, DateClosed = @DateClosed, TicketStatus =
                      @TicketStatus, TestTicketNotes = @TestTicketNotes WHERE TestAssignmentID = @TestAssignmentID";
160
161                 //associate @id with Id parameter
162
163
164                 SqlCommand command = new SqlCommand(sqlQuery, connection);
165
166                 command.Parameters.Add("@TestAssignmentID", System.Data.SqlDbType.Int).Value = testAssignmentModel.TestAssignmentID;
167                 command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
168                 command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketUser;
169                 command.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateAssigned;
170                 command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateClosed;
171                 command.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
172                 command.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar).Value = testAssignmentModel.TestTicketNotes;
173
174                 //Update Test that ticket is assigned
175                 string sqlQuery1 = "Update dbo.Test SET StartDate = @StartDate, EndDate = @EndDate, TestStatus = @TestStatus, TestNotes = @TestNotes WHERE dbo.Test.TestName = @TestName";
176
177                 //associate @id with Id parameter
178                 SqlCommand command1 = new SqlCommand(sqlQuery1, connection);
179
180                 command1.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
181                 command1.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateAssigned;
182                 command1.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateClosed;
183                 command1.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
184                 command1.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar).Value = testAssignmentModel.TestTicketNotes;
185
186                 //Insert Assignment To Vulnerability Log
187                 string sqlQuery2 = "INSERT INTO dbo.TestLog VALUES(@TestName, @TicketUser, @CurrentTimeStamp, @TicketStatus,  @TestTicketNotes)";
188
189                 //associate @id with Id parameter
190                 SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
191
192                 command2.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
193                 command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.CurrentUser;
194                 command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
195                 command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
196                 command2.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestTicketNotes;
197
198
199
200                 connection.Open();
201                 int newTestAssignmentID = command.ExecuteNonQuery();
202                 command1.ExecuteNonQuery();
203                 command2.ExecuteNonQuery();
204                 return newTestAssignmentID;
205             }
206
207         }
208     }
209 }
```

TestAssignmentModel.cs

```cs
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Models
{
    23 references
    public class TestAssignmentModel
    {
        [Display(Name = "Test Assignment ID")]
        4 references
        public int TestAssignmentID { get; set; }
        [Display(Name = "Test Name")]
        9 references
        public string TestName { get; set; }
        [Display(Name = "Assignee")]
        5 references
        public string TicketUser { get; set; }
        [Display(Name = "Current User")]
        2 references
        public string CurrentUser { get; set; }
        [Display(Name = "Date Assigned")]

        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        7 references
        public DateTime?  DateAssigned { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        7 references
        public DateTime? DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        9 references
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        9 references
        public string TestTicketNotes { get; set; }
        3 references
        public List<SelectListItem> ListTestName { get; set; }
        3 references
        public List<SelectListItem> ListTicketUser { get; set; }
        3 references
        public List<SelectListItem> ListTestStatus { get; set; }

        0 references
        public TestAssignmentModel(int testAssignmentID, string testName, string ticketUser, DateTime dateAssigned, DateTime dateClosed, string ticketStatus, string testTicketNotes)
        {
            TestAssignmentID = testAssignmentID;
            TestName = testName;
            TicketUser = ticketUser;
            DateAssigned = dateAssigned;
            DateClosed = dateClosed;
            TicketStatus = ticketStatus;
            TestTicketNotes = testTicketNotes;

        }

        3 references
        public TestAssignmentModel()
        {
            this.ListTestName = new List<SelectListItem>();
            this.ListTicketUser = new List<SelectListItem>();
            this.ListTestStatus = new List<SelectListItem>();
        }
    }
}
```

TestAssignment Views

CreateTestAssignmentForm.cshtml

```
@model Project.Models.TestAssignmentModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}

<h2> </h2>


@using (Html.BeginForm("CreateTestAssignment", "TestAssignment"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br/>
    <hr/>
    <h4><b>Assign Test</b></h4>
    <hr/>
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })


    <div class="form-group">
        @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TestName, Model.ListTestName, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
        </div>
    </div>

    <div hidden class="form-group">
        @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.DateAssigned, new { type = "date" })

        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.DateClosed, new { type = "date" })

        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TicketStatus, Model.ListTestStatus, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TestTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.TestTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.TestTicketNotes, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Details.cshtml

```
@model Project.Models.TestAssignmentModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2> </h2>

<div>
    <br />
    <hr />
    <h4><b>Test Assignment Details</b></h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.TestAssignmentID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TestAssignmentID)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TestName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TestName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TicketUser)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TicketUser)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateAssigned)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateAssigned)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateClosed)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateClosed)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TicketStatus)
        </dd>
```

```
79
80              <dt>
81                  @Html.DisplayNameFor(model => model.TestTicketNotes)
82              </dt>
83
84              <dd>
85                  @Html.DisplayFor(model => model.TestTicketNotes)
86              </dd>
87
88          </dl>
89      </div>
90      <p>
91          @Html.ActionLink("Edit", "Edit", new { id = Model.TestAssignmentID }) |
92          @Html.ActionLink("Back to List", "Index")
93      </p>
94
```

Index.cshtml

```
@model IEnumerable<Project.Models.TestAssignmentModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {
        <br />
        <hr />
        <h4><b>Assigned Tests</b></h4>
        <hr />


        @Html.ActionLink("Create New", "Create")

    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }

}



<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.TestAssignmentID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketUser)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateAssigned)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateClosed)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestTicketNotes)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.TestAssignmentID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TestName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TicketUser)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateAssigned)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateClosed)
```

```
78                  @Html.DisplayFor(modelItem => item.DateClosed)
79              </td>
80              <td>
81                  @Html.DisplayFor(modelItem => item.TicketStatus)
82              </td>
83              <td>
84                  @Html.DisplayFor(modelItem => item.TestTicketNotes)
85              </td>
86              <td>
87                  @Html.ActionLink("Edit", "Edit", new { id = item.TestAssignmentID }) |
88                  @Html.ActionLink("Details", "Details", new { id = item.TestAssignmentID }) |
89                  @Html.ActionLink("Delete", "Delete", new { id = item.TestAssignmentID })
90              </td>
91          </tr>
92      }
93
94  </table>
95
```

UpdateTestAssignmentForm.cshtml

```
@model Project.Models.TestAssignmentModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2> </h2>


@using (Html.BeginForm("UpdateTestAssignment", "TestAssignment"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Update Test Assignment</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div hidden class="form-group">
        @Html.LabelFor(model => model.TestAssignmentID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.TestAssignmentID, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.TestAssignmentID, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TestName, Model.ListTestName, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
        </div>
    </div>

    <div hidden class="form-group">
        @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateAssigned, new { htmlAttributes = new { @class = "form-control" } })

        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateClosed, new { htmlAttributes = new { @class = "form-control" } })
        </div>
```

```
78              </div>
79          </div>
80
81          <div class="form-group">
82              @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
83              <div class="col-md-10">
84                  @Html.DropDownListFor(m => m.TicketStatus, Model.ListTestStatus, "Please select")
85              </div>
86          </div>
87
88          <div class="form-group">
89              @Html.LabelFor(model => model.TestTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
90              <div class="col-md-10">
91                  @Html.EditorFor(model => model.TestTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
92                  @Html.ValidationMessageFor(model => model.TestTicketNotes, "", new { @class = "text-danger" })
93              </div>
94          </div>
95
96          <div class="form-group">
97              <div class="col-md-offset-2 col-md-10">
98                  <input type="submit" value="Create" class="btn btn-default" />
99              </div>
100         </div>
101     </div>
102     }
103
104 <div>
105     @Html.ActionLink("Back to List", "Index")
106 </div>
107
108
109
```

TestsController.cs

```
using Project.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Controllers
{
    public class TestsController : Controller
    {
        private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

    [CustomAuthorization]
        public ActionResult Index()
        {
            List<TestModel> tests = new List<TestModel>();
            TestDAO testDAO = new TestDAO();
            tests = testDAO.FetchAll();
            return View("Index", tests);
        }
        [CustomAuthorization]
        public ActionResult Details(int id)
        {
            TestDAO testDAO = new TestDAO();
            TestModel test = testDAO.FetchOne(id);
            return View("Details", test);
        }
        [CustomAuthorization]
        public ActionResult Create()
        {
            TestModel test = new TestModel();
            test.ListAssetName = PopulateDropDown(" SELECT AssetName FROM dbo.Asset", "AssetName", "AssetName");
            test.ListTestStatus = PopulateDropDown(" SELECT StatusType FROM dbo.TestStatus", "StatusType", "StatusType");
            return View("CreateTestForm", test);
        }
        [CustomAuthorization]
        private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
        {
            List<SelectListItem> items = new List<SelectListItem>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(query))
                {
                    cmd.Connection = connection;
                    connection.Open();
                    using (SqlDataReader sdr = cmd.ExecuteReader())
                    {
                        while (sdr.Read())
                        {
                            items.Add(new SelectListItem
                            {
                                Text = sdr[textColumn].ToString(),
                                Value = sdr[valueColumn].ToString()
                            });
                        }
                    }
                    connection.Close();
                }
            }

            return items;
        }
```

```
65          }
66          [CustomAuthorization]
            0 references
67          public ActionResult CreateProcess(TestModel testModel)
68          {
69              //save it to the database
70              TestDAO testDAO = new TestDAO();
71
72              testDAO.Create(testModel);
73              return View("Details", testModel);
74          }
75          [CustomAuthorization]
            0 references
76          public ActionResult Edit(int id)
77          {
78              TestDAO testDAO = new TestDAO();
79              TestModel test = testDAO.FetchOne(id);
80              test.ListAssetName = PopulateDropDown(" SELECT AssetName FROM dbo.Asset", "AssetName", "AssetName");
81              test.ListTestStatus = PopulateDropDown(" SELECT StatusType FROM dbo.TestStatus", "StatusType", "StatusType");
82              return View("UpdateTestForm", test);
83          }
84          [CustomAuthorization]
85          [HttpPost]
            0 references
86          public ActionResult UpdateTest(TestModel testModel)
87          {
88              //save it to the database
89              TestDAO testDAO = new TestDAO();
90
91              testDAO.Update(testModel);
92              //Back to index list
93              List<TestModel> tests = new List<TestModel>();
94              tests = testDAO.FetchAll();
95              return View("Index", tests);
96          }
97
98          [CustomAuthorization]
            0 references
99          public ActionResult Delete(int id)
100         {
101             TestDAO testDAO = new TestDAO();
102             testDAO.Delete(id);
103             List<TestModel> tests = testDAO.FetchAll();
104             return View("Index", tests);
105         }
106         [CustomAuthorization]
            0 references
107         public ActionResult TestEventLog(string name)
108         {
109             List<TestModel> test = new List<TestModel>();
110             TestDAO testDAO = new TestDAO();
111             test = testDAO.GetTestEventLog(name);
112             return View("TestEventLog", test);
113         }
114
115     }
116 }
```

TestDAO.cshtml

```
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;

namespace Project.Data
{
    14 references
    internal class TestDAO
    {
        //connect to database
        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        //Performs all operations on database, create, read, read one, update, delete

        3 references
        public List<TestModel> FetchAll()
        {
            List<TestModel> returnList = new List<TestModel>();

            //access database
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * FROM dbo.Test";
                SqlCommand command = new SqlCommand(sqlQuery, connection);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //create new test object and add that to the list to be returned
                        TestModel test = new TestModel();
                        test.TestID = reader.GetInt32(0);
                        test.TestName = reader.GetString(1);
                        test.AssetName = reader.GetString(2);
                        test.StartDate = reader.GetDateTime(3);
                        test.EndDate = reader.GetDateTime(4);
                        test.SignOff = reader.GetBoolean(5);
                        test.URL = reader.GetString(6);
                        test.Description = reader.GetString(7);
                        test.ContactID = reader.GetString(8);
                        test.TestStatus = reader.GetString(9);
                        test.TestNotes = reader.GetString(11);
                        //adding to list to be returned
                        returnList.Add(test);
                    }
                }
            }

            return returnList;

        }

        1 reference
        internal int Delete(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                //prepared statement
                //update
                string sqlQuery = "DELETE FROM dbo.Test WHERE TestID = @id";




                //associate @id with Id parameter


                SqlCommand command = new SqlCommand(sqlQuery, connection);

                command.Parameters.Add("@Id", System.Data.SqlDbType.VarChar, 1000).Value = id;

```

```
71
72
73              connection.Open();
74              int deletedID = command.ExecuteNonQuery();
75
76              return deletedID;
77          }
78      }
79
        2 references
80      public TestModel FetchOne(int id)
81      {
82
83          //access database
84          using (SqlConnection connection = new SqlConnection(connectionString))
85          {
86              string sqlQuery = "SELECT * FROM dbo.Test WHERE TestID = @id";
87
88              SqlCommand command = new SqlCommand(sqlQuery, connection);
89              command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;
90              connection.Open();
91              SqlDataReader reader = command.ExecuteReader();
92
93              TestModel test = new TestModel();
94              if (reader.HasRows)
95              {
96                  while (reader.Read())
97                  {
98                      //create new test object and add that to the list to be returned
99
100                     test.TestID = reader.GetInt32(0);
101                     test.TestName = reader.GetString(1);
102                     test.AssetName = reader.GetString(2);
103                     test.StartDate = reader.GetDateTime(3);
104                     test.EndDate = reader.GetDateTime(4);
105                     test.SignOff = reader.GetBoolean(5);
106                     test.URL = reader.GetString(6);
107                     test.Description = reader.GetString(7);
108                     test.ContactID = reader.GetString(8);
109                     test.TestStatus = reader.GetString(9);
110                     test.TestNotes = reader.GetString(11);
111
112
113                 }
114             }
115             return test;
116         }
117
118     }
119
        1 reference
120     public int Create(TestModel testModel)
121     {
122
123         //access database
124         using (SqlConnection connection = new SqlConnection(connectionString))
125         {
126             string sqlQuery = "INSERT INTO dbo.Test Values(@TestName, @AssetName, @StartDate, @EndDate, @SignOff, @URL, @Description, @ContactID, @TestStatus, @TicketUser, @TestNotes)";
127
128             SqlCommand command = new SqlCommand(sqlQuery, connection);
129             command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar,1000).Value = testModel.TestName;
130             command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.AssetName;
131             command.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testModel.StartDate;
132             command.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testModel.EndDate;
133             command.Parameters.Add("@SignOff", System.Data.SqlDbType.Bit).Value = testModel.SignOff;
134             command.Parameters.Add("@URL", System.Data.SqlDbType.VarChar, 1000).Value = testModel.URL;
135             command.Parameters.Add("@Description", System.Data.SqlDbType.VarChar, 1000).Value = testModel.Description;
136             command.Parameters.Add("@ContactID", System.Data.SqlDbType.VarChar, 1000).Value = testModel.ContactID;
137             command.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
138             command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
139             command.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;
140
141             connection.Open();
142             int newTestID = command.ExecuteNonQuery();
```

```
            int newTestID = command.ExecuteNonQuery();

            return newTestID;
        }
    }

    1 reference
    public int Update(TestModel testModel)
    {


        //access the database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            //prepared statement
            //update
            string sqlQuery = "UPDATE dbo.Test SET  TestName = @TestName, StartDate = @StartDate, EndDate = @EndDate, SignOff = @SignOff, TestStatus = @TestStatus, TicketUser =
              @TicketUser, TestNotes = @TestNotes WHERE TestID = @TestID";

            //associate @id with Id parameter

            SqlCommand command = new SqlCommand(sqlQuery, connection);
            command.Parameters.Add("@TestID", System.Data.SqlDbType.Int).Value = testModel.TestID;
            command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestName;
            command.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testModel.StartDate;
            command.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testModel.EndDate;
            command.Parameters.Add("@SignOff", System.Data.SqlDbType.Bit).Value = testModel.SignOff;
            command.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
            command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TicketUser;
            command.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;

            //Update Test Assignment
            string sqlQuery1 = "Update dbo.TestAssignment SET DateAssigned = @DateAssigned, DateClosed = @DateClosed, TicketStatus = @TicketStatus, TestTicketNotes = @TestTicketNotes
              WHERE dbo.TestAssignment.TestName = @TestName";

            //associate @id with Id parameter
            SqlCommand command1 = new SqlCommand(sqlQuery1, connection);

            command1.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar,1000).Value = testModel.TestName;
            command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = testModel.EndDate;
            command1.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = testModel.StartDate;
            command1.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
            command1.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;

            //Insert Edit To Test Log
            string sqlQuery2 = "INSERT INTO dbo.TestLog VALUES(@TestName, @TicketUser, @CurrentTimeStamp, @TicketStatus,  @TestTicketNotes)";

            //associate @id with Id parameter
            SqlCommand command2 = new SqlCommand(sqlQuery2, connection);

            command2.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar,1000).Value = testModel.TestName;
            command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testModel.CurrentUser;
            command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
            command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
            command2.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;


            connection.Open();
            int newTestID = command.ExecuteNonQuery();
            command1.ExecuteNonQuery();
            command2.ExecuteNonQuery();
            return newTestID;
        }

    }

    1 reference
    public List<TestModel> GetTestEventLog(string name)
    {
        List<TestModel> returnList = new List<TestModel>();
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            string sqlQuery = "SELECT * FROM dbo.TestLog WHERE TestName = @name";
            SqlCommand command = new SqlCommand(sqlQuery, connection);
```

```
                SqlCommand command = new SqlCommand(sqlQuery, connection);
                command.Parameters.Add("@name", System.Data.SqlDbType.VarChar,1000).Value = name;
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //creating new object and adding it to the list to be returned
                        TestModel test = new TestModel();
                        test.TestLogID = reader.GetInt32(0);
                        test.TestName = reader.GetString(1);
                        test.TicketUser = reader.GetString(2);
                        test.CurrentTimeStamp = reader.GetDateTime(3);
                        test.TicketStatus = reader.GetString(4);
                        test.TestTicketNotes = reader.GetString(5);
                        returnList.Add(test);
                    }

                }

            }
            return returnList;
        }

    }
}
```

TestModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Models
{
    30 references
    public class TestModel
    {
        [Display(Name = "Test ID")]
        4 references
        public int TestID { get; set; }
        [Display(Name = "Test Log ID")]
        1 reference
        public int TestLogID { get; set; }
        [Display(Name = "Test Name")]
        8 references
        public string TestName { get; set; }
        [Display(Name = "Asset")]
        3 references
        public string AssetName { get; set; }
        [Display(Name = "Ticket Status")]
        1 reference
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        1 reference
        public string TestTicketNotes { get; set; }
        [Display(Name = "Current TimeStamp")]
        1 reference
        public DateTime CurrentTimeStamp { get; set; }
        [Display(Name = "Start Date")]

        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        6 references
        public DateTime StartDate { get; set; }
        [Display(Name = "End Date")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        6 references
        public DateTime EndDate { get; set; }
        [Display(Name = "Sign Off Documents")]
        5 references
        public bool SignOff { get; set; }
        [Display(Name = "URL")]
        4 references
        public string URL { get; set; }
        [Display(Name = "Description")]
        4 references
        public string Description { get; set; }
        [Display(Name = "Contact ID")]
        4 references
        public string ContactID { get; set; }
        [Display(Name = "Test Status")]
        6 references
        public string TestStatus { get; set; }
        [Display(Name = "Assignee")]
        2 references
        public string TicketUser { get; set; }
        [Display(Name = "Curent User")]
        1 reference
        public string CurrentUser { get; set; }
        [Display(Name = "Test Notes")]
        6 references
        public string TestNotes { get; set; }
        3 references
        public List<SelectListItem> ListAssetName { get; set; }
        3 references
        public List<SelectListItem> ListTestStatus { get; set; }

        0 references
        public TestModel(int testID, string testName, DateTime startDate, DateTime endDate, bool signOff, string uRL, string description, string contactID)
        {
            TestID = testID;
            TestName = testName;
            StartDate = startDate;
            EndDate = endDate;
            SignOff = signOff;
            URL = uRL;
```

```
60              Description = description;
61              ContactID = contactID;
62          }
63
            4 references
64          public TestModel()
65          {
66              this.ListAssetName = new List<SelectListItem>();
67              this.ListTestStatus = new List<SelectListItem>();
68          }
69      }
70  }
```

Test Views

CreateTestForm.cshtml

```
@model Project.Models.TestModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2></h2>


@using (Html.BeginForm("CreateProcess", "tests"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Create Test</b></h4>
    <hr/>
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })

    <div class="form-group">
        @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.AssetName, Model.ListAssetName, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.TestName, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.TestName, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.StartDate, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.StartDate, new { type = "date" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.EndDate, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.EndDate, new { type = "date" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.SignOff, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            <div class="checkbox">
                @Html.EditorFor(model => model.SignOff)
                @Html.ValidationMessageFor(model => model.SignOff, "", new { @class = "text-danger" })
            </div>
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.URL, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
78              @Html.LabelFor(model => model.URL, htmlAttributes: new { @class = "control-label col-md-2" })
79              <div class="col-md-10">
80                  @Html.EditorFor(model => model.URL, new { htmlAttributes = new { @class = "form-control" } })
81                  @Html.ValidationMessageFor(model => model.URL, "", new { @class = "text-danger" })
82              </div>
83          </div>
84
85          <div class="form-group">
86              @Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-2" })
87              <div class="col-md-10">
88                  @Html.EditorFor(model => model.Description, new { htmlAttributes = new { @class = "form-control" } })
89                  @Html.ValidationMessageFor(model => model.Description, "", new { @class = "text-danger" })
90              </div>
91          </div>
92
93          <div class="form-group">
94              @Html.LabelFor(model => model.ContactID, htmlAttributes: new { @class = "control-label col-md-2" })
95              <div class="col-md-10">
96                  @Html.EditorFor(model => model.ContactID, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
97              </div>
98          </div>
99          <div class="form-group">
100             @Html.LabelFor(model => model.TestNotes, htmlAttributes: new { @class = "control-label col-md-2" })
101             <div class="col-md-10">
102                 @Html.EditorFor(model => model.TestNotes, new { htmlAttributes = new { @class = "form-control" } })
103             </div>
104         </div>
105         <div class="form-group">
106             @Html.LabelFor(model => model.TestStatus, htmlAttributes: new { @class = "control-label col-md-2" })
107             <div class="col-md-10">
108                 @Html.DropDownListFor(m => m.TestStatus, Model.ListTestStatus, "Please select")
109             </div>
110         </div>
111
112         <div hidden class="form-group">
113             @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
114             <div hidden class="col-md-10">
115                 @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = "Empty" } })
116             </div>
117         </div>
118
119         <div class="form-group">
120             <div class="col-md-offset-2 col-md-10">
121                 <input type="submit" value="Create" class="btn btn-default" />
122             </div>
123         </div>
124 </div>
125 }
126
127 <div>
128     @Html.ActionLink("Back to List", "Index")
129 </div>
130
131 @section Scripts {
132     @Scripts.Render("~/bundles/jqueryval")
133 }
134
```

Details.cshtml

```
@model Project.Models.TestModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2></h2>

<div>
    <br />
    <hr />
    <h4><b>Test Details</b></h4>
    <hr/>
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.TestName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TestName)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.AssetName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AssetName)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.StartDate)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.StartDate)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.EndDate)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.EndDate)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.SignOff)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.SignOff)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.URL)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.URL)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Description)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Description)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.ContactID)
        </dt>
```

```
78              </dt>
79              <dd>
80                  @Html.DisplayFor(model => model.ContactID)
81              </dd>
82              <dt>
83                  @Html.DisplayNameFor(model => model.TestNotes)
84              </dt>
85              <dd>
86                  @Html.DisplayFor(model => model.TestNotes)
87              </dd>
88              <dt>
89                  @Html.DisplayNameFor(model => model.TestStatus)
90              </dt>
91              <dd>
92                  @Html.DisplayFor(model => model.TestStatus)
93              </dd>
94
95          </dl>
96      </div>
97      <p>
98          @Html.ActionLink("Edit", "Edit", new { id = Model.TestID }) |
99          @Html.ActionLink("Back to List", "Index")
100     </p>
101     @Html.ActionLink("TestEventLog", "TestEventLog", new { name = Model.TestName })
```

Index.cshtml

```
@model IEnumerable<Project.Models.TestModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2> </h2>

<p>
    <br />
    <hr />
    <h4><b>Tests</b></h4>
    <hr />
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.TestName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.AssetName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.StartDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EndDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.SignOff)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.URL)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Description)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ContactID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestNotes)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestStatus)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.TestName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.AssetName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.StartDate)
```

```
77              @Html.DisplayFor(modelItem => item.StartDate)
78          </td>
79          <td>
80              @Html.DisplayFor(modelItem => item.EndDate)
81          </td>
82          <td>
83              @Html.DisplayFor(modelItem => item.SignOff)
84          </td>
85          <td>
86              @Html.DisplayFor(modelItem => item.URL)
87          </td>
88          <td>
89              @Html.DisplayFor(modelItem => item.Description)
90          </td>
91          <td>
92              @Html.DisplayFor(modelItem => item.ContactID)
93          </td>
94          <td>
95              @Html.DisplayFor(modelItem => item.TestNotes)
96          </td>
97          <td>
98              @Html.DisplayFor(modelItem => item.TestStatus)
99          </td>
100         <td>
101             @Html.ActionLink("Edit", "Edit", new { id = item.TestID }) |
102             @Html.ActionLink("Details", "Details", new { id = item.TestID }) |
103             @Html.ActionLink("Delete", "Delete", new { id = item.TestID })
104         </td>
105     </tr>
106     }
107
108 </table>
109
```

TestEventLog.cshtml

```
@model IEnumerable<Project.Models.TestModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>

<p>
    <br />
    <hr />
    <h4><b>Test Event Log</b></h4>
    <hr/>

</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.TestLogID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketUser)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.CurrentTimeStamp)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestTicketNotes)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.TestLogID)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.TestName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.TicketUser)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CurrentTimeStamp)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.TicketStatus)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.TestTicketNotes)
        </td>
    </tr>
}

</table>
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

UpdateTestForm.cshtml

```
@model Project.Models.TestModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@{
    String Role;
    Role = Session["userRole"].ToString();
    if (Role == "Administrator")
    {


    }
    else if (Role == "PenetrationTester")
    {

    }
    else
    {
        Response.Redirect("~/Home/Index");
    }
}
<h2></h2>


@using (Html.BeginForm("UpdateTest", "tests"))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
       <br/>
        <hr/>
        <h4><b>Update Test</b></h4>
        <hr/>
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })

        <div hidden class="form-group">
            @Html.LabelFor(model => model.TestID, htmlAttributes: new { @class = "control-label col-md-2" })
            <div hidden class="col-md-10">
                @Html.EditorFor(model => model.TestID, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TestID, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TestName, new { htmlAttributes = new { @class = "form-control" } })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.StartDate, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.StartDate, new { htmlAttributes = new { @class = "form-control" } })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.EndDate, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EndDate, new { htmlAttributes = new { @class = "form-control" } })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SignOff, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                <div class="checkbox">
                    @Html.EditorFor(model => model.SignOff)
                    @Html.ValidationMessageFor(model => model.SignOff, "", new { @class = "text-danger" })
                </div>
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TestNotes, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
```

```
            @Html.LabelFor(model => model.TestNotes, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TestNotes, new { htmlAttributes = new { @class = "form-control" } })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.TestStatus, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(m => m.TestStatus, Model.ListTestStatus, "Please select")
            </div>
        </div>
        <div hidden class="form-group">
            @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
            <div hidden class="col-md-10">
                @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
            </div>
        </div>

        <div hidden class="form-group">
            @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
            <div hidden class="col-md-10">
                @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
            </div>
        </div>

        <input type="hidden" name="TestID" value="TestID" />

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

VulnerabilitiesController.cshtml

```
using Project.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Web.Mvc;


namespace Project.Controllers
{


    0 references
    public class VulnerabilitiesController : Controller
    {
        private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";


        [CustomAuthorization]
        0 references
        public ActionResult Index()
        {
            List<VulnerabilityModel> vulnerabilities = new List<VulnerabilityModel>();
            VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
            vulnerabilities = vulnerabilityDAO.FetchAll();
            return View("Index", vulnerabilities);
        }

        [CustomAuthorization]
        0 references
        public ActionResult Details(int id)
        {
            VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
            vulnerabilityDAO.GetTactic(id);
            vulnerabilityDAO.GetAssetName(id);
            VulnerabilityModel vulnerability = vulnerabilityDAO.FetchOne(id);

            return View("Details", vulnerability);
        }

        [CustomAuthorization]
        0 references
        public ActionResult Create()
        {
            VulnerabilityModel vulnerability = new VulnerabilityModel();

            vulnerability.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
            vulnerability.MitreTactic = PopulateDropDown(" SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
            vulnerability.ListVulnerabilityStatusPenTester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPenTester", "StatusType", "StatusType");
            vulnerability.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
            return View("AddVulnerabilityForm", vulnerability);
        }

        [CustomAuthorization]
        [HttpPost]
        0 references
        public JsonResult AjaxMethod(string type, int value)
        {
            VulnerabilityModel vulnerability = new VulnerabilityModel();
            switch (type)
            {
                case "TacticID":
                    vulnerability.MitreTechnique = PopulateDropDown("SELECT TechniqueID, Technique FROM MitreTechnique WHERE TacticID= " + value, "Technique", "TechniqueID");
                    break;

            }
            return Json(vulnerability);
        }

        [CustomAuthorization]
        [HttpPost]
        0 references
        public ActionResult Create(int tacticID)
        {
```

```
{
    VulnerabilityModel vulnerability = new VulnerabilityModel();
    vulnerability.MitreTactic = PopulateDropDown("SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
    vulnerability.MitreTechnique = PopulateDropDown("SELECT TechniqueID, Technique FROM MitreTechnique WHERE TacticID = " + tacticID, "Technique", "TechniqueID");
    return View(vulnerability);
}

[CustomAuthorization]
13 references
private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
{
    List<SelectListItem> items = new List<SelectListItem>();
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand(query))
        {
            cmd.Connection = connection;
            connection.Open();
            using (SqlDataReader sdr = cmd.ExecuteReader())
            {
                while (sdr.Read())
                {
                    items.Add(new SelectListItem
                    {
                        Text = sdr[textColumn].ToString(),
                        Value = sdr[valueColumn].ToString()
                    });
                }
            }
            connection.Close();
        }
    }

    return items;
}

[CustomAuthorization]
0 references
public ActionResult AddVulnerability(VulnerabilityModel vulnerabilityModel)
{
    //save it to the database
    VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
    vulnerabilityDAO.Create(vulnerabilityModel);
    List<VulnerabilityModel> vulnerabilities = new List<VulnerabilityModel>();
   //Return full list of vulnerabilities
    vulnerabilities = vulnerabilityDAO.FetchAll();
    return View("Index", vulnerabilities);
}

[CustomAuthorization]
0 references
public ActionResult Edit(int id)
{
    VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
    VulnerabilityModel vulnerability = vulnerabilityDAO.FetchOne(id);
    vulnerability.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
    vulnerability.MitreTactic = PopulateDropDown(" SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
    vulnerability.ListVulnerabilityStatusPenTester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPenTester", "StatusType", "StatusType");
    vulnerability.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
    return View("UpdateVulnerabilityForm", vulnerability);
}
[CustomAuthorization]
[HttpPost]
0 references
public ActionResult Edit1(int tacticID)
{
    VulnerabilityModel vulnerability = new VulnerabilityModel();
    vulnerability.MitreTactic = PopulateDropDown("SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
    vulnerability.MitreTechnique = PopulateDropDown("SELECT TechniqueID, Technique FROM MitreTechnique WHERE TacticID = " + tacticID, "Technique", "TechniqueID");
    return View(vulnerability);
}
```

```
135         }
136
137         [CustomAuthorization]
138         [HttpPost]
            0 references
139         public ActionResult UpdateVulnerability(VulnerabilityModel vulnerabilityModel)
140         {
141             //save it to the database
142             VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
143             vulnerabilityDAO.Update(vulnerabilityModel);
144             List<VulnerabilityModel> vulnerabilities = new List<VulnerabilityModel>();
145             //Return full list of vulnerabilities
146             vulnerabilities = vulnerabilityDAO.FetchAll();
147             return View("Index", vulnerabilities);
148         }
149         [CustomAuthorization]
            0 references
150         public ActionResult Delete(int id)
151         {
152             VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
153             vulnerabilityDAO.Delete(id);
154             List<VulnerabilityModel> vulnerabilities = vulnerabilityDAO.FetchAll();
155             return View("Index", vulnerabilities);
156         }
157         [CustomAuthorization]
            0 references
158         public ActionResult VulnerabilityEventLog(int id)
159         {
160             List<VulnerabilityModel> vulnerability = new List<VulnerabilityModel>();
161             VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
162             vulnerability = vulnerabilityDAO.GetVulnerabilityEventLog(id);
163             return View("VulnerabilityEventLog", vulnerability);
164         }
165
166     }
167 }
```

VulnerabilityDAO.cs

```
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;

namespace Project.Data
{
    16 references
    internal class VulnerabilityDAO
    {
        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
        //performs all actions on vulnerability

        5 references
        public List<VulnerabilityModel> FetchAll()
        {
            List<VulnerabilityModel> returnList = new List<VulnerabilityModel>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * from dbo.Vulnerability";
                SqlCommand command = new SqlCommand(sqlQuery, connection);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //Create new vulnerability object and add it to list to be returned
                        VulnerabilityModel vulnerability = new VulnerabilityModel();
                        vulnerability.VulnerabilityID = reader.GetInt32(0);
                        vulnerability.TestName = reader.GetString(1);
                        vulnerability.VulnerabilityType = reader.GetString(2);
                        vulnerability.Severity = reader.GetString(3);
                        vulnerability.DateOpen = reader.GetDateTime(4);
                        vulnerability.DateClosed = reader.GetDateTime(5);
                        vulnerability.StatusID = reader.GetString(11);

                        returnList.Add(vulnerability);

                    }
                }

            }

            return returnList;

        }


        0 references
        public VulnerabilityModel GetAsset()
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                //SQL Statement Open Tests
                string sqlQuery = "SELECT dbo.Test.AssetName, dbo.Vulnerability.TestName, dbo.Vulnerability.VulnerabilityID FROM dbo.Vulnerability INNER JOIN dbo.Test ON
                  dbo.Vulnerability.TestName = dbo.Test.TestName INNER JOIN dbo.Asset ON dbo.Asset.AssetName = dbo.Test.AssetName ORDER BY dbo.Vulnerability.VulnerabilityID";

                SqlCommand command = new SqlCommand(sqlQuery, connection);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                VulnerabilityModel vulnerabilityModel = new VulnerabilityModel();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        vulnerabilityModel.AssetName = reader.GetString(0);
                    }


                }
                reader.Close();
                return vulnerabilityModel;
            }
```

```
        }
    }


    1 reference
    internal int Delete(int id)
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            //prepared statement
            //update
            string sqlQuery = "DELETE FROM dbo.Vulnerability WHERE VulnerabilityID = @id";




            //associate @id with Id parameter


            SqlCommand command = new SqlCommand(sqlQuery, connection);

            command.Parameters.Add("@Id", System.Data.SqlDbType.VarChar, 1000).Value = id;


            connection.Open();
            int deletedID = command.ExecuteNonQuery();

            return deletedID;
        }
    }

    1 reference
    public VulnerabilityModel GetTactic(int id)
    {
        String Tactic;
        String Technique;
        using (SqlConnection connection = new SqlConnection(connectionString))
        {

            string sqlQuery = "SELECT dbo.MitreTactic.Tactic, dbo.MitreTechnique.Technique,  dbo.Vulnerability.VulnerabilityID FROM  dbo.Vulnerability INNER JOIN dbo.MitreTactic ON
              dbo.Vulnerability.TacticID = dbo.MitreTactic.TacticID INNER JOIN dbo.MitreTechnique ON dbo.Vulnerability.TechniqueID = dbo.MitreTechnique.TechniqueID  WHERE
              dbo.Vulnerability.VulnerabilityID  = @id ORDER BY dbo.Vulnerability.VulnerabilityID";

            SqlCommand command = new SqlCommand(sqlQuery, connection);
            command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;


            connection.Open();
            //string result = (string)command.ExecuteReader();

            SqlDataReader reader = command.ExecuteReader();
            VulnerabilityModel vulnerabilityModel = new VulnerabilityModel();
            if (reader.HasRows) {
                while (reader.Read())
                {
                    vulnerabilityModel.Tactic = reader.GetString(0);
                    Tactic = vulnerabilityModel.Tactic;
                    vulnerabilityModel.Tactic = Tactic;
                    vulnerabilityModel.Technique = reader.GetString(1);
                    Technique = vulnerabilityModel.Technique;
                    vulnerabilityModel.Technique = Technique;
                    vulnerabilityModel.VulnerabilityID = reader.GetInt32(2);

                }

            }
            reader.Close();
            string Query = "UPDATE dbo.Vulnerability SET Tactic = @Tactic, Technique = @Technique WHERE VulnerabilityID = @VulnerabilityID";

            //associate @id with Id parameter

            SqlCommand command1 = new SqlCommand(Query, connection);
```

```
139
140             command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = id;
141             command1.Parameters.Add("@Tactic", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Tactic;
142             command1.Parameters.Add("@Technique", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Technique;
143
144
145             command1.ExecuteNonQuery();
146
147
148
149
150             connection.Close();
151             return vulnerabilityModel;
152         }
153     }
154
155     1 reference
    public VulnerabilityModel GetAssetName(int id)
156 {
157     String AssetName;
158     int TestID;
159
160     using (SqlConnection connection = new SqlConnection(connectionString))
161     {
162
163         string sqlQuery = "SELECT dbo.Test.AssetName, dbo.Test.TestID, dbo.Vulnerability.VulnerabilityID FROM dbo.Vulnerability INNER JOIN dbo.Test ON dbo.Vulnerability.TestName =
              dbo.Test.TestName WHERE dbo.Vulnerability.VulnerabilityID  = @id ORDER BY dbo.Vulnerability.VulnerabilityID";
164
165         SqlCommand command = new SqlCommand(sqlQuery, connection);
166         command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;
167
168
169         connection.Open();
170         //string result = (string)command.ExecuteReader();
171
172         SqlDataReader reader = command.ExecuteReader();
173         VulnerabilityModel vulnerabilityModel = new VulnerabilityModel();
174         if (reader.HasRows)
175         {
176             while (reader.Read())
177             {
178                 vulnerabilityModel.AssetName = reader.GetString(0);
179                 AssetName = vulnerabilityModel.AssetName;
180                 vulnerabilityModel.AssetName = AssetName;
181                 vulnerabilityModel.TestID = reader.GetInt32(1);
182                 TestID = vulnerabilityModel.TestID;
183                 vulnerabilityModel.TestID = TestID;
184                 vulnerabilityModel.VulnerabilityID = reader.GetInt32(2);
185
186
187             }
188
189         }
190         reader.Close();
191         string Query = "UPDATE dbo.Vulnerability SET AssetName = @AssetName, TestID = @TestID WHERE VulnerabilityID = @VulnerabilityID";
192
193         //associate @id with Id parameter
194
195         SqlCommand command1 = new SqlCommand(Query, connection);
196
197         command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = id;
198         command1.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.AssetName;
199         command1.Parameters.Add("@TestID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.TestID;
200
201
202         command1.ExecuteNonQuery();
203
204
205
206
207         connection.Close();
208         return vulnerabilityModel;
209     }
210 }
```

```
        }

        2 references
        public VulnerabilityModel FetchOne(int id)
        {
            //access database
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * FROM dbo.Vulnerability WHERE VulnerabilityID = @id";

                SqlCommand command = new SqlCommand(sqlQuery, connection);
                command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();

                VulnerabilityModel vulnerability = new VulnerabilityModel();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //create new test object and add that to the list to be returned

                        vulnerability.VulnerabilityID = reader.GetInt32(0);
                        vulnerability.AssetName = reader.GetString(20);
                        vulnerability.TestName = reader.GetString(1);
                        vulnerability.TestID = reader.GetInt32(21);
                        vulnerability.VulnerabilityType = reader.GetString(2);
                        vulnerability.Severity = reader.GetString(3);
                        vulnerability.DateOpen = reader.GetDateTime(4);
                        vulnerability.DateClosed = reader.GetDateTime(5);
                        vulnerability.URL = reader.GetString(6);
                        vulnerability.Tactic = reader.GetString(12);
                        vulnerability.Technique = reader.GetString(13);
                        vulnerability.StepsToReproduce = reader.GetString(9);
                        vulnerability.Mitigation = reader.GetString(10);
                        vulnerability.StatusID = reader.GetString(11);
                        vulnerability.DamagePotential = reader.GetDecimal(14);
                        vulnerability.Reproducability = reader.GetDecimal(15);
                        vulnerability.Exploitability = reader.GetDecimal(16);
                        vulnerability.AffectedUsers = reader.GetDecimal(17);
                        vulnerability.Discoverability = reader.GetDecimal(18);
                        vulnerability.DREADTotal = reader.GetDecimal(19);



                    }
                }
                return vulnerability;
            }

        }

        1 reference
        public int Create(VulnerabilityModel vulnerabilityModel)
        {

            //access database
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "INSERT INTO dbo.Vulnerability Values(@TestName, @VulnerabilityType, @Severity, @DateOpen, @DateClosed, @URL, @TacticID, @TechniqueID, @StepsToReproduce,
                  @Mitigation, @StatusID, @Tactic, @Technique, @DamagePotential, @Reproducability, @Exploitability, @AffectedUsers, @Discoverability, @DREADTotal, @AssetName, @TestID,
                  @TicketNotes, @AssignedUser)";

                SqlCommand command = new SqlCommand(sqlQuery, connection);
                command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TestName;
                command.Parameters.Add("@VulnerabilityType", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.VulnerabilityType;
                command.Parameters.Add("@Severity", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Severity;
                command.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateOpen;
                command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateClosed;
                command.Parameters.Add("@URL", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.URL;
                command.Parameters.Add("@TacticID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityModel.TacticID;
```

```
            command.Parameters.Add("@TacticID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityModel.TacticID;
            command.Parameters.Add("@TechniqueID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityModel.TechniqueID;
            command.Parameters.Add("@StepsToReproduce", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StepsToReproduce;
            command.Parameters.Add("@Mitigation", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Mitigation;
            command.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
            command.Parameters.Add("@Tactic", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
            command.Parameters.Add("@Technique", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
            command.Parameters.Add("@DamagePotential", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.DamagePotential;
            command.Parameters.Add("@Reproducability", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.Reproducability;
            command.Parameters.Add("@Exploitability", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.Exploitability;
            command.Parameters.Add("@AffectedUsers", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.AffectedUsers;
            command.Parameters.Add("@Discoverability", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.Discoverability;
            command.Parameters.Add("@DREADTotal", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.DREADTotal;
            command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
            command.Parameters.Add("@TestID", System.Data.SqlDbType.Int).Value = "0";
            command.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar,1000).Value = "Empty";
            command.Parameters.Add("@AssignedUser", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";









            connection.Open();
            int newVulnerabilityID = command.ExecuteNonQuery();

            return newVulnerabilityID;
        }
    }

    1 reference
    public int Update(VulnerabilityModel vulnerabilityModel)
    {
        //access the database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {

            //Update Vulnerability
            string sqlQuery = "UPDATE dbo.Vulnerability SET DateOpen = @DateOpen, DateClosed = @DateClosed, StepsToReproduce = @StepsToReproduce, Mitigation = @Mitigation, StatusID =
              @StatusID, TicketNotes = @TicketNotes WHERE VulnerabilityID = @VulnerabilityID";

            //associate @id with Id parameter
            SqlCommand command = new SqlCommand(sqlQuery, connection);

            command.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.VulnerabilityID;
            command.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateOpen;
            command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateClosed;
            command.Parameters.Add("@StepsToReproduce", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StepsToReproduce;
            command.Parameters.Add("@Mitigation", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Mitigation;
            command.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
            command.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar,1000).Value = vulnerabilityModel.TicketNotes;

            //Update Vulnerability Assignment
            string sqlQuery1 = "Update dbo.VulnerabilityAssignment SET DateAssigned = @DateAssigned, DateClosed = @DateClosed, TicketStatus = @TicketStatus, VulnerabilityTicketNotes =
              @VulnerabilityTicketNotes WHERE dbo.VulnerabilityAssignment.VulnerabilityID = @VulnerabilityID";

            //associate @id with Id parameter
            SqlCommand command1 = new SqlCommand(sqlQuery1, connection);

            command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.VulnerabilityID;
            command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateClosed;
            command1.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateOpen;
            command1.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
            command1.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TicketNotes;

            //Insert Edit To Vulnerability Log
            string sqlQuery2 = "INSERT INTO dbo.VulnerabilityLog VALUES(@VulnerabilityID, @TicketUser, @CurrentTimeStamp, @TicketStatus,  @VulnerabilityTicketNotes)";

            //associate @id with Id parameter
            SqlCommand command2 = new SqlCommand(sqlQuery2, connection);

            command2.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.VulnerabilityID;
```

```
344                 command2.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.VulnerabilityID;
345                 command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TicketUser;
346                 command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
347                 command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
348                 command2.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TicketNotes;
349                 connection.Open();
350
351                 int newVulnerabilityID = command.ExecuteNonQuery();
352                 command1.ExecuteNonQuery();
353                 command2.ExecuteNonQuery();
354                 return newVulnerabilityID;
355             }
356
357         }
358
359
                1 reference
360         public List<VulnerabilityModel> GetVulnerabilityEventLog(int id)
361         {
362             List<VulnerabilityModel> returnList = new List<VulnerabilityModel>();
363             using (SqlConnection connection = new SqlConnection(connectionString))
364             {
365                 string sqlQuery = "SELECT * FROM dbo.VulnerabilityLog WHERE VulnerabilityID = @id";
366                 SqlCommand command = new SqlCommand(sqlQuery, connection);
367                 command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;
368                 connection.Open();
369                 SqlDataReader reader = command.ExecuteReader();
370                 if (reader.HasRows)
371                 {
372                     while (reader.Read())
373                     {
374                         //creating new object and adding it to the list to be returned
375                         VulnerabilityModel vulnerability = new VulnerabilityModel();
376                         vulnerability.VulnerabilityLogID = reader.GetInt32(0);
377                         vulnerability.VulnerabilityID = reader.GetInt32(1);
378                         vulnerability.TicketUser = reader.GetString(2);
379                         vulnerability.CurrentTimeStamp = reader.GetDateTime(3);
380                         vulnerability.TicketStatus = reader.GetString(4);
381                         vulnerability.VulnerabilityTicketNotes = reader.GetString(5);
382                         returnList.Add(vulnerability);
383                     }
384
385                 }
386
387             }
388             return returnList;
389         }
390
391     }
392 }
```

VulnerabilityModel.cs

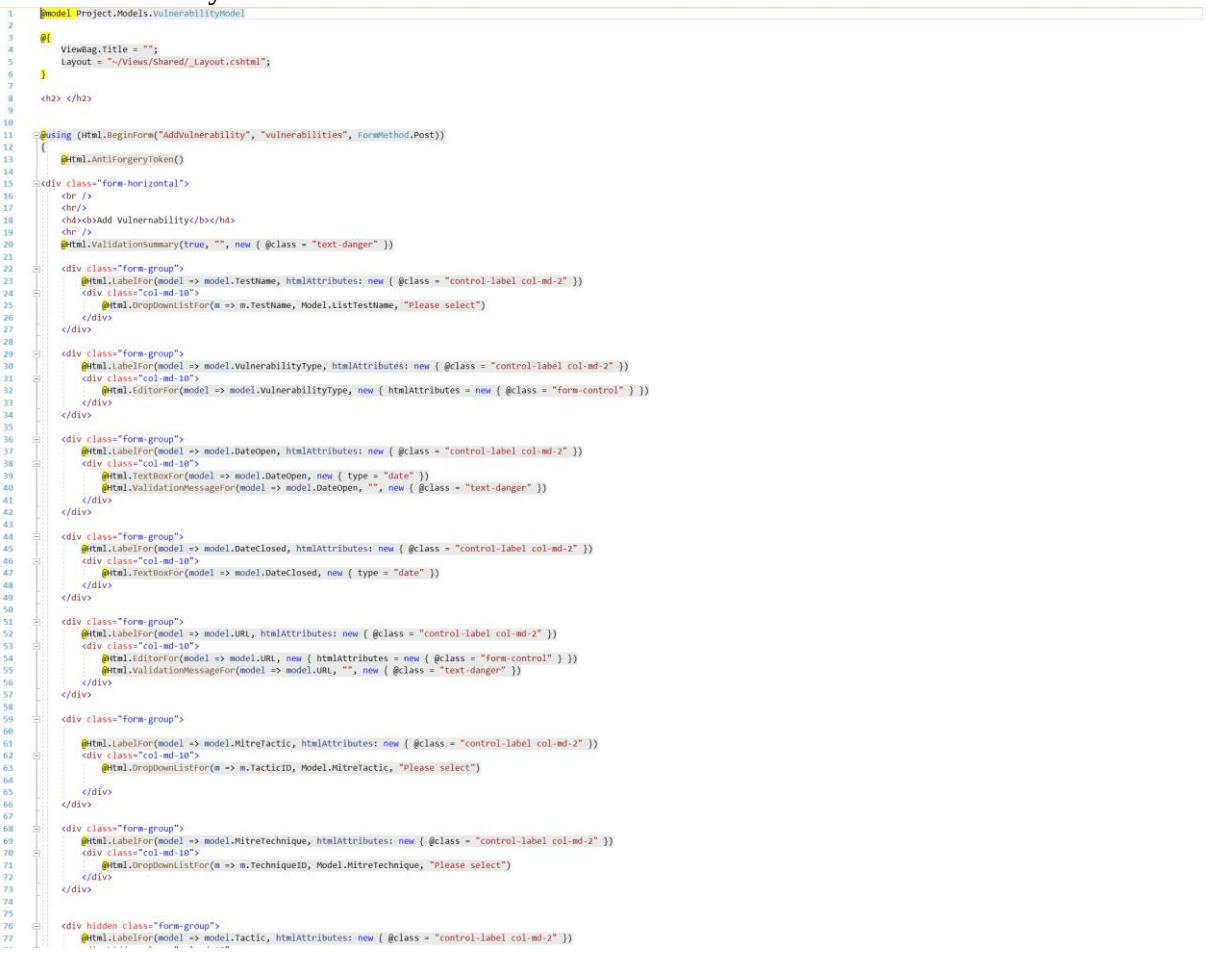
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Configuration;
using System.Data.SqlClient;

namespace Project.Models
{
    48 references
    public class VulnerabilityModel
    {
        0 references
        public List<SelectListItem> Vulnerabilities { get; set; }
        0 references
        public List<SelectListItem> MitreTactics { set; get; }
        0 references
        public List<SelectListItem> MitreTechniques { get; set; }
        [Display(Name = "Vulnerability ID")]
        9 references
        public int? VulnerabilityID { get; set; }
        [Display(Name = "Vulnerability Log ID")]
        1 reference
        public int VulnerabilityLogID { get; set; }
        [Display(Name = "Test Name")]
        4 references
        public string TestName { get; set; }
        [Display(Name = "Ticket Status")]
        1 reference
        public string TicketStatus { get; set; }
        [Display(Name = "Vulnerability Ticket Notes")]
        1 reference
        public string VulnerabilityTicketNotes { get; set; }
        [Display(Name = "Vulnerability Type")]
        4 references
        public string VulnerabilityType { get; set; }
        [Display(Name = "Severity")]
        4 references
        public string Severity { get; set; }
        [Display(Name = "Date Open")]

        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        6 references
        public DateTime DateOpen { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        6 references
        public DateTime DateClosed { get; set; }
        [Display(Name = "URL")]
        3 references
        public string URL { get; set; }

        [Required(ErrorMessage = "Please select a category")]
        [Display(Name = "Steps To Reproduce")]
        4 references
        public string StepsToReproduce { get; set; }
        [Display(Name = "Mitigation")]
        4 references
        public string Mitigation { get; set; }
        [Display(Name = "Status")]
        7 references
        public string StatusID { get; set; }
        [Display(Name = "Mitre Tactic")]
        6 references
        public string Tactic { get; set; }
        [Display(Name = "Mitre Technique")]
        6 references
        public string Technique { get; set; }
        3 references
        public List<SelectListItem> ListTestName { get; set; }
        1 reference
        public List<SelectListItem> ListAssetName { get; set; }
        3 references
        public List<SelectListItem> ListVulnerabilityStatusPenTester { get; set; }
```

```
3 references
public List<SelectListItem> ListVulnerabilityStatusEngineer { get; set; }
[Display(Name = "Damage Potential Score")]
2 references
public decimal DamagePotential { get; set; }
[Display(Name = "Reproducability Score")]
2 references
public decimal Reproducability { get; set; }
[Display(Name = "Exploitability Score")]
2 references
public decimal Exploitability { get; set; }
[Display(Name = "Affected Users Score")]
2 references
public decimal AffectedUsers { get; set; }
[Display(Name = "Discoverability Score")]
2 references
public decimal Discoverability { get; set; }
[Display(Name = "DREAD Total Threat Rating")]
2 references
public decimal DREADTotal { get; set; }
[Display(Name = "Asset")]
6 references
public string AssetName { get; set; }
[Display(Name = "Test ID")]
5 references
public int TestID { get; set; }
[Display(Name = "Ticket Notes")]
4 references
public string TicketNotes { get; set; }
[Display(Name = "Assignee")]
2 references
public string TicketUser { get; set; }
[Display(Name = "Assigned User")]
0 references
public string AssignedUser { get; set; }
1 reference
public DateTime CurrentTimeStamp { get; set; }

0 references
public VulnerabilityModel(int vulnerabilityID, string testName, string vulnerabilityType, string severity, DateTime dateOpen, DateTime dateClosed, string uRL, string tactic, string
  technique, string stepsToReproduce, string mitigation, string statusID, string ticketNotes)
{
    VulnerabilityID = vulnerabilityID;
    TestName = testName;
    VulnerabilityType = vulnerabilityType;
    Severity = severity;
    DateOpen = dateOpen;
    DateClosed = dateClosed;
    URL = uRL;
    Tactic = tactic;
   Technique = technique;
    StepsToReproduce = stepsToReproduce;
    Mitigation = mitigation;
    StatusID = statusID;
    TicketNotes = ticketNotes;
}

10 references
public VulnerabilityModel()
{
    //new
    this.ListTestName = new List<SelectListItem>();
    this.ListAssetName = new List<SelectListItem>();
    this.ListVulnerabilityStatusPenTester = new List<SelectListItem>();
    this.ListVulnerabilityStatusEngineer = new List<SelectListItem>();
    this.MitreTactic = new List<SelectListItem>();
    this.MitreTechnique = new List<SelectListItem>();
}

5 references
public List<SelectListItem> MitreTactic { get; set; }
4 references
public List<SelectListItem> MitreTechnique { get; set; }

1 reference
public int TacticID { get; set; }
1 reference
public int TechniqueID { get; set; }
1 reference
```

```
        1 reference
114     public int TechniqueID { get; set; }
        1 reference
115     public string selectedTactic { get; set; }
116 }
117 }
```

Vulnerability Views

AddVulnerabilityForm.css



```
@model Project.Models.VulnerabilityModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>


@using (Html.BeginForm("AddVulnerability", "vulnerabilities", FormMethod.Post))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr/>
    <h4><b>Add Vulnerability</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })

    <div class="form-group">
        @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TestName, Model.ListTestName, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.VulnerabilityType, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.VulnerabilityType, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateOpen, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.DateOpen, new { type = "date" })
            @Html.ValidationMessageFor(model => model.DateOpen, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.DateClosed, new { type = "date" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.URL, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.URL, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.URL, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">

        @Html.LabelFor(model => model.MitreTactic, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TacticID, Model.MitreTactic, "Please select")

        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.MitreTechnique, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TechniqueID, Model.MitreTechnique, "Please select")
        </div>
    </div>


    <div hidden class="form-group">
        @Html.LabelFor(model => model.Tactic, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
        @Html.LabelFor(model => model.Tactic, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.Tactic, new { htmlAttributes = new { @class = "form-control", value = "Empty" } })

        </div>
    </div>

    <div hidden class="form-group">
        @Html.LabelFor(model => model.Technique, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.Technique, new { htmlAttributes = new { @class = "form-control", value = "Empty" } })
        </div>
    </div>
    <div hidden class="form-group">
        @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.AssetName, new { htmlAttributes = new { @class = "form-control", value = "Empty" } })

        </div>
    </div>
    <div hidden class="form-group">
        @Html.LabelFor(model => model.TestID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.TestID, new { htmlAttributes = new { @class = "form-control", value = "0" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DamagePotential, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DamagePotential, new { htmlAttributes = new { @class = "form-control", placeholder = "", type = "text", id = "DamagePotential", onchange = "CalculateDamagePotentialTotal()" } })
            @Html.ValidationMessageFor(model => model.DamagePotential, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Reproducability, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Reproducability, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Reproducability", onchange = "CalculateReproducabilityTotal()" } })
            @Html.ValidationMessageFor(model => model.Reproducability, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Exploitability, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Exploitability, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Exploitability", onchange = "CalculateExploitabilityTotal()" } })
            @Html.ValidationMessageFor(model => model.Exploitability, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.AffectedUsers, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.AffectedUsers, new { htmlAttributes = new { @class = "form-control", type = "text", id = "AffectedUsers", onchange = "CalculateAffectedUsersTotal()" } })
            @Html.ValidationMessageFor(model => model.AffectedUsers, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Discoverability, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Discoverability, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Discoverability", onchange = "CalculateDiscoverabilityTotal()" } })
            @Html.ValidationMessageFor(model => model.Discoverability, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DREADTotal, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DREADTotal, new { htmlAttributes = new { @class = "form-control", type = "text", id = "DREADTotal", onfocus = "CalculateDREADTotal()" } })
            @Html.ValidationMessageFor(model => model.DREADTotal, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.Severity, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Severity, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Severity", onfocus = "CalculateSeverity()" } })
            @Html.ValidationMessageFor(model => model.Severity, "", new { @class = "text-danger" })
        </div>
```

```
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.StepsToReproduce, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.StepsToReproduce, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.StepsToReproduce, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Mitigation, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Mitigation, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Mitigation, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.TicketNotes, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>

    @{
        String Role;
        Role = Session["userRole"].ToString();
        if (Role == "PenetrationTester")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }
        else if (Role == "SecurityEngineer")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusEngineer, "Please select")
                </div>
            </div>
        }
        else if (Role == "Administrator")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }

    }

    <div hidden class="form-group">
        @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
        </div>
    </div>

    <div hidden class="form-group">
        @Html.LabelFor(model => model.AssignedUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.AssignedUser, new { htmlAttributes = new { @class = "form-control", Value = "Empty" } })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-default" />
        </div>
```

```
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index", new { id = Model.VulnerabilityID })
</div>

@section Scripts {
    <script src="~/Scripts/jquery-1.10.2.min.js"></script>
    <script type="text/javascript">
        $(function () {


            $("select").change(function () {
                var value = 0;
                if ($(this).val() != "") {
                    value = $(this).val();
                }
                var id = $(this).attr("id");
                $.ajax({
                    type: "POST",
                    url: "/Vulnerabilities/AjaxMethod",
                    data: '{type: "' + id + '", value: ' + value + '}',
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    success: function (response) {
                        var dropDownId;
                        var list;
                        switch (id) {
                            case "TacticID":
                                list = response.MitreTechnique;
                                DisableDropDown("#TechniqueID");

                                PopulateDropDown("#TechniqueID", list);
                                break;
                        }

                    }
                });
            });
        });

        function DisableDropDown(dropDownId) {
            $(dropDownId).attr("disabled", "disabled");
            $(dropDownId).empty().append('<option selected="selected" value="0">Please select</option>');
        }

        function PopulateDropDown(dropDownId, list) {
            if (list != null && list.length > 0) {
                $(dropDownId).removeAttr("disabled");
                $.each(list, function () {
                    $(dropDownId).append($("<option></option>").val(this['Value']).html(this['Text']));
                });
            }
        }
    </script>
    <script type="text/javascript">
        function CalculateDamagePotentialTotal() {

            var DamagePotential = document.getElementById('DamagePotential').value;
            var DamageTotal = ((DamagePotential *40) /100);
            //alert(DamageTotal);
            $('#DamagePotential').val(DamageTotal);
            var FinalVal = document.getElementById('DamagePotential').value;

        }
        function CalculateReproducabilityTotal() {
             var Reproducability = document.getElementById('Reproducability').value;
             var ReproducabilityTotal = ((Reproducability * 5) / 100);
            // alert(ReproducabilityTotal);
             $('#Reproducability').val(ReproducabilityTotal);
            var FinalVal = document.getElementById('Reproducability').value;


```

```
300
301         }
302         function CalculateExploitabilityTotal() {
303               var Exploitability = document.getElementById('Exploitability').value;
304               var ExploitabilityTotal = ((Exploitability * 10) / 100);
305              // alert(ExploitabilityTotal);
306               $('#Exploitability').val(ExploitabilityTotal);
307             var FinalVal = document.getElementById('Exploitability').value;
308
309
310         }
311
312         function CalculateAffectedUsersTotal() {
313             var AffectedUsers = document.getElementById('AffectedUsers').value;
314             var AffectedUsersTotal = ((AffectedUsers * 40) / 100);
315            // alert(AffectedUsersTotal);
316             $('#AffectedUsers').val(AffectedUsersTotal);
317             var FinalVal = document.getElementById('AffectedUsers').value;
318
319
320         }
321
322         function CalculateDiscoverabilityTotal() {
323             var Discoverability = document.getElementById('Discoverability').value;
324             var DiscoverabilityTotal = ((Discoverability * 5) / 100);
325             //alert(DiscoverabilityTotal);
326             $('#Discoverability').val(DiscoverabilityTotal);
327             var FinalVal = document.getElementById('Discoverability').value;
328
329
330         }
331
332         function CalculateDREADTotal() {
333             var DamagePotentialVal = document.getElementById('DamagePotential').value;
334             var ReproducabilityVal = document.getElementById('Reproducability').value;
335             var ExploitabilityVal = document.getElementById('Exploitability').value;
336             var AffectedUsersVal = document.getElementById('AffectedUsers').value;
337             var DiscoverabilityVal = document.getElementById('Discoverability').value;
338
339             var DREADTotal = +DamagePotentialVal + +ReproducabilityVal + +ExploitabilityVal + +AffectedUsersVal + +DiscoverabilityVal;
340             $('#DREADTotal').val(DREADTotal);
341             var FinalVal = document.getElementById('DREADTotal').value;
342
343
344         }
345
346         function CalculateSeverity() {
347             var DREADTotal = document.getElementById('DREADTotal').value;
348             if (DREADTotal == 0.0) {
349                 var Severity = "Observation";
350                 $('#Severity').val(Severity);
351             }
352             else if (DREADTotal >= 0.1 && DREADTotal <= 4.0) {
353                 var Severity1 = "Low Risk";
354                 $('#Severity').val(Severity1);
355             }
356             else if (DREADTotal >= 4.1 && DREADTotal <= 8.0) {
357                 var Severity2 = "Medium Risk";
358                 $('#Severity').val(Severity2);
359             }
360             else if (DREADTotal >= 8.1 && DREADTotal <= 10.0) {
361                 var Severity3 = "High Risk";
362                 $('#Severity').val(Severity3);
363             }
364
365
366         }
367
368     </script>
369 }
370
```

Details.cshtml

```
@model Project.Models.VulnerabilityModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2></h2>
<div>
    <br/>
    <hr/>
    <h4><b>Vulnerability Intelligence</b></h4>
    <hr />

    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.VulnerabilityID)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TestName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TestName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TestID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TestID)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.AssetName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AssetName)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.VulnerabilityType)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.VulnerabilityType)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Severity)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Severity)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateOpen)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateOpen)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateClosed)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateClosed)
        </dd>
```

```
77          </dd>
78
79          <dt>
80              @Html.DisplayNameFor(model => model.URL)
81          </dt>
82
83          <dd>
84              @Html.DisplayFor(model => model.URL)
85          </dd>
86          <hr />
87          <h4><b>Vulnerability Mitre Threat Intelligence</b></h4>
88          <hr />
89          <dt>
90              @Html.DisplayNameFor(model => model.Tactic)
91          </dt>
92          <dd>
93              @Html.DisplayFor(model => model.Tactic)
94          </dd>
95          <dt>
96              @Html.DisplayNameFor(model => model.Technique)
97          </dt>
98          <dd>
99              @Html.DisplayFor(model => model.Technique)
100         </dd>
101         <hr />
102         <h4><b>Vulnerability DREAD Threat Rating</b></h4>
103         <hr />
104         <dt>
105             @Html.DisplayNameFor(model => model.DamagePotential)
106         </dt>
107         <dd>
108             @Html.DisplayFor(model => model.DamagePotential)
109         </dd>
110         <dt>
111             @Html.DisplayNameFor(model => model.Reproducability)
112         </dt>
113
114         <dd>
115             @Html.DisplayFor(model => model.Reproducability)
116         </dd>
117
118         <dt>
119             @Html.DisplayNameFor(model => model.Exploitability)
120         </dt>
121         <dd>
122             @Html.DisplayFor(model => model.Exploitability)
123         </dd>
124         <dt>
125             @Html.DisplayNameFor(model => model.AffectedUsers)
126         </dt>
127         <dd>
128             @Html.DisplayFor(model => model.AffectedUsers)
129         </dd>
130         <dt>
131             @Html.DisplayNameFor(model => model.Discoverability)
132         </dt>
133         <dd>
134             @Html.DisplayFor(model => model.Discoverability)
135         </dd>
136         <dt>
137             @Html.DisplayNameFor(model => model.DREADTotal)
138         </dt>
139         <dd>
140             @Html.DisplayFor(model => model.DREADTotal)
141         </dd>
142         <hr />
143         <h4><b>Vulnerability Mitigation</b></h4>
144         <hr />
145         <dt>
146             @Html.DisplayNameFor(model => model.StepsToReproduce)
147         </dt>
148         <dd>
149             @Html.DisplayFor(model => model.StepsToReproduce)
150         </dd>
```

```
150             </dd>
151             <dt>
152                 @Html.DisplayNameFor(model => model.Mitigation)
153             </dt>
154             <dd>
155                 @Html.DisplayFor(model => model.Mitigation)
156             </dd>
157             <hr />
158             <h4><b>Vulnerability Status</b></h4>
159             <hr />
160             <dt>
161                 @Html.DisplayNameFor(model => model.StatusID)
162             </dt>
163             <dd>
164                 @Html.DisplayFor(model => model.StatusID)
165             </dd>
166         </dl>
167     </div>
168     <p>
169         @Html.ActionLink("Edit", "Edit", new { id = Model.VulnerabilityID }) |
170         @Html.ActionLink("Back to List", "Index")
171         @Html.ActionLink("VulnerabilityEventLog", "VulnerabilityEventLog", new { id = Model.VulnerabilityID })
172     </p>
173
174
```

Index.cshtml

```
@model IEnumerable<Project.Models.VulnerabilityModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>

<p>
    <br />
    <hr />
    <h4><b>Vulnerabilities</b></h4>
    <hr />
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityType)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Severity)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateOpen)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateClosed)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.StatusID)
        </th>
        <th>

        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TestName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityType)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Severity)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateOpen)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateClosed)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.StatusID)
    </td>
    <td>

    </td>
    <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.VulnerabilityID }) |
        @Html.ActionLink("Details", "Details", new { id = item.VulnerabilityID }) |
        @Html.ActionLink("Delete", "Delete", new { id = item.VulnerabilityID })
    </td>
</tr>

</tr>
}

</table>
```

UpdateVulnerabilityForm.cshtml

```
@model Project.Models.VulnerabilityModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>UpdateVulnerabilityForm</h2>


@using (Html.BeginForm("UpdateVulnerability", "vulnerabilities"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Update Vulnernability</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })




    <div class="form-group">
        @Html.LabelFor(model => model.DateOpen, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateOpen, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateClosed, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.StepsToReproduce, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.StepsToReproduce, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Mitigation, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Mitigation, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.TicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.TicketNotes, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>
    @{
        String Role;
        Role = Session["userRole"].ToString();
        if (Role == "PenetrationTester")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }
        else if (Role == "SecurityEngineer")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusEngineer, "Please select")
                </div>
            </div>
        }
        else if (Role == "Administrator")
        {
```

```
        {
            <div class="form-group">
                @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }
    }
    <div hidden class="form-group">
        @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
        </div>
    </div>



    <input type="hidden" name="VulnerabilityID" value="@Model.VulnerabilityID" />

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
@section Scripts {
    <script src="~/Scripts/jquery-1.10.2.min.js"></script>
    <script type="text/javascript">
        $(function () {


            $("select").change(function () {
                var value = 0;
                if ($(this).val() != "") {
                    value = $(this).val();
                }
                var id = $(this).attr("id");
                $.ajax({
                    type: "POST",
                    url: "/Vulnerabilities/AjaxMethod",
                    data: '{type: "' + id + '", value: ' + value + '}',
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    success: function (response) {
                        var dropDownId;
                        var list;
                        switch (id) {
                            case "TacticID":
                                list = response.MitreTechnique;
                                DisableDropDown("#TechniqueID");

                                PopulateDropDown("#TechniqueID", list);
                                break;
                        }

                    }
                });
            });
        });

        function DisableDropDown(dropDownId) {
            $(dropDownId).attr("disabled", "disabled");
            $(dropDownId).empty().append('<option selected="selected" value="0">Please select</option>');
        }

        function PopulateDropDown(dropDownId, list) {
            if (list != null && list.length > 0) {
                $(dropDownId).removeAttr("disabled");
```

```
                $(dropDownId).removeAttr("disabled");
                $.each(list, function () {
                    $(dropDownId).append($("<option></option>").val(this['Value']).html(this['Text']));
                });
            }
        }
    </script>
    <script type="text/javascript">
        function CalculateDamagePotentialTotal() {

            var DamagePotential = document.getElementById('DamagePotential').value;
            var DamageTotal = ((DamagePotential * 40) / 100);
            //alert(DamageTotal);
            $('#DamagePotential').val(DamageTotal);
            var FinalVal = document.getElementById('DamagePotential').value;

        }
        function CalculateReproducabilityTotal() {
            var Reproducability = document.getElementById('Reproducability').value;
            var ReproducabilityTotal = ((Reproducability * 5) / 100);
            // alert(ReproducabilityTotal);
            $('#Reproducability').val(ReproducabilityTotal);
            var FinalVal = document.getElementById('Reproducability').value;


        }
        function CalculateExploitabilityTotal() {
            var Exploitability = document.getElementById('Exploitability').value;
            var ExploitabilityTotal = ((Exploitability * 10) / 100);
            // alert(ExploitabilityTotal);
            $('#Exploitability').val(ExploitabilityTotal);
            var FinalVal = document.getElementById('Exploitability').value;


        }

        function CalculateAffectedUsersTotal() {
            var AffectedUsers = document.getElementById('AffectedUsers').value;
            var AffectedUsersTotal = ((AffectedUsers * 40) / 100);
            // alert(AffectedUsersTotal);
            $('#AffectedUsers').val(AffectedUsersTotal);
            var FinalVal = document.getElementById('AffectedUsers').value;


        }

        function CalculateDiscoverabilityTotal() {
            var Discoverability = document.getElementById('Discoverability').value;
            var DiscoverabilityTotal = ((Discoverability * 5) / 100);
            //alert(DiscoverabilityTotal);
            $('#Discoverability').val(DiscoverabilityTotal);
            var FinalVal = document.getElementById('Discoverability').value;


        }

        function CalculateDREADTotal() {
            var DamagePotentialVal = document.getElementById('DamagePotential').value;
            var ReproducabilityVal = document.getElementById('Reproducability').value;
            var ExploitabilityVal = document.getElementById('Exploitability').value;
            var AffectedUsersVal = document.getElementById('AffectedUsers').value;
            var DiscoverabilityVal = document.getElementById('Discoverability').value;

            var DREADTotal = +DamagePotentialVal + +ReproducabilityVal + +ExploitabilityVal + +AffectedUsersVal + +DiscoverabilityVal;
            $('#DREADTotal').val(DREADTotal);
            var FinalVal = document.getElementById('DREADTotal').value;


        }

        function CalculateSeverity() {
            var DREADTotal = document.getElementById('DREADTotal').value;
            if (DREADTotal == 0.0) {
                var Severity = "Observation";
```

```
225                 var Severity = "Observation";
226                 $('#Severity').val(Severity);
227             }
228             else if (DREADTotal >= 0.1 && DREADTotal <= 4.0) {
229                 var Severity1 = "Low Risk";
230                 $('#Severity').val(Severity1);
231             }
232             else if (DREADTotal >= 4.1 && DREADTotal <= 8.0) {
233                 var Severity2 = "Medium Risk";
234                 $('#Severity').val(Severity2);
235             }
236             else if (DREADTotal >= 8.1 && DREADTotal <= 10.0) {
237                 var Severity3 = "High Risk";
238                 $('#Severity').val(Severity3);
239             }
240
241
242         }
243
244     </script>
245 }
246
```

VulnerabilityEventLog.cshtml

```
@model IEnumerable<Project.Models.VulnerabilityModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>
<br />
<hr />
<h4><b>Vulnerability Event Log</b></h4>
<hr />

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityLogID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketUser)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.CurrentTimeStamp)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.VulnerabilityLogID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.VulnerabilityID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TicketUser)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.CurrentTimeStamp)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TicketStatus)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
            </td>
        </tr>
    }

</table>
@Html.ActionLink("Back to List", "Index")
```

VulnerabilityAssignmentsController.cs

```
using Project.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Controllers
{
    0 references
    public class VulnerabilityAssignmentController : Controller
    {
        private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
        [CustomAuthorization]
        0 references
        public ActionResult Index()
        {

            List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();

            vulnerabilityAssignments = vulnerabilityAssignmentDAO.FetchAll();
            return View("Index", vulnerabilityAssignments);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Details(int id)
        {

            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
            VulnerabilityAssignmentModel vulnerabilityAssignment = vulnerabilityAssignmentDAO.FetchOne(id);
            return View("Details", vulnerabilityAssignment);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Create()
        {
            VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
            vulnerabilityAssignment.ListVulnerabilityID = PopulateDropDown(" SELECT VulnerabilityID FROM dbo.Vulnerability", "VulnerabilityID", "VulnerabilityID");
            vulnerabilityAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role  = 'SecurityEngineer' ORDER BY Username ", "Username", "Username");
            vulnerabilityAssignment.ListVulnerabilityStatusPenTester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPenTester", "StatusType", "StatusType");
            vulnerabilityAssignment.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
            return View("CreateVulnerabilityAssignmentForm", vulnerabilityAssignment);
        }
        [CustomAuthorization]
        0 references
        private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
        {
            List<SelectListItem> items = new List<SelectListItem>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(query))
                {
                    cmd.Connection = connection;
                    connection.Open();
                    using (SqlDataReader sdr = cmd.ExecuteReader())
                    {
                        while (sdr.Read())
                        {
                            items.Add(new SelectListItem
                            {
                                Text = sdr[textColumn].ToString(),
                                Value = sdr[valueColumn].ToString()
                            });
                        }
                    }
                    connection.Close();
                }
            }

            return items;
        }
        [CustomAuthorization]
```

```
        }
        [CustomAuthorization]
        0 references
        public ActionResult CreateVulnerabilityAssignment(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
        {
            //save it to the database
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
            vulnerabilityAssignmentDAO.Create(vulnerabilityAssignmentModel);
            List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
            //How to return full list of assignments instead of the one
            vulnerabilityAssignments = vulnerabilityAssignmentDAO.FetchAll();
            return View("Index", vulnerabilityAssignments);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Edit(int id)
        {
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
            VulnerabilityAssignmentModel vulnerabilityAssignment = vulnerabilityAssignmentDAO.FetchOne(id);
            vulnerabilityAssignment.ListVulnerabilityID = PopulateDropDown(" SELECT VulnerabilityID FROM dbo.Vulnerability", "VulnerabilityID", "VulnerabilityID");
            vulnerabilityAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role  = 'SecurityEngineer' ORDER BY Username ", "Username", "Username");
            vulnerabilityAssignment.ListVulnerabilityStatusPenTester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPenTester", "StatusType", "StatusType");
            vulnerabilityAssignment.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
            return View("UpdateVulnerabilityAssignmentForm", vulnerabilityAssignment);
        }
        [CustomAuthorization]
        [HttpPost]
        0 references
        public ActionResult UpdateVulnerabilityAssignment(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
        {
            //save it to the database
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();

            vulnerabilityAssignmentDAO.Update(vulnerabilityAssignmentModel);
            return View("Details", vulnerabilityAssignmentModel);
        }
        [CustomAuthorization]
        0 references
        public ActionResult Delete(int id)
        {
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
            vulnerabilityAssignmentDAO.Delete(id);
            List<VulnerabilityAssignmentModel> vulnerabilityAssignments = vulnerabilityAssignmentDAO.FetchAll();
            return View("Index", vulnerabilityAssignments);
        }
        [CustomAuthorization]
        0 references
        public ActionResult VulnerabilityAssignmentsEngineer(string session)
        {
            List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();

            vulnerabilityAssignments = vulnerabilityAssignmentDAO.GetVulnerabilityAssignmentsEngineer(session);
            return View("VulnerabilityAssignmentsEngineer", vulnerabilityAssignments);

        }
        [CustomAuthorization]
        0 references
        public ActionResult TestAssignmentsPenTester(string session)
        {
            List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
            vulnerabilityAssignments = vulnerabilityAssignmentDAO.GetTestAssignmentsPenTester(session);
            return View("TestAssignmentsPenTester", vulnerabilityAssignments);

        }
        [CustomAuthorization]
        0 references
        public ActionResult VulnerabilityAssignmentsPenTester()
        {
            List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
            VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
            vulnerabilityAssignments = vulnerabilityAssignmentDAO.GetVulnerabilityAssignmentsPenTester();
            return View("VulnerabilityAssignmentsPenTester", vulnerabilityAssignments);

        }

    }
}
```

VulnerabilityAssignmentDAO.cs

```
using Project.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;

namespace Project.Data
{
    20 references
    internal class VulnerabilityAssignmentDAO
    {

        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        4 references
        public List<VulnerabilityAssignmentModel> FetchAll()
        {
            List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment";

                SqlCommand command = new SqlCommand(sqlQuery, connection);

                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //creating new TestAssignment object and adding it to the list to be returned
                        VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
                        vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
                        vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
                        vulnerabilityAssignment.TicketUser = reader.GetString(2);
                        vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
                        vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
                        vulnerabilityAssignment.TicketStatus = reader.GetString(5);
                        vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
                        returnList.Add(vulnerabilityAssignment);
                    }
                }
            }

            return returnList;
        }

        2 references
        public VulnerabilityAssignmentModel FetchOne(int id)
        {

            //access database
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE VulnerabilityAssignmentID = @id";

                SqlCommand command = new SqlCommand(sqlQuery, connection);
                command.Parameters.Add("@Id", System.Data.SqlDbType.Int).Value = id;
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();

                VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        //create new test object and add that to the list to be returned

                        vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
                        vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
                        vulnerabilityAssignment.TicketUser = reader.GetString(2);
                        vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
                        vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
                        vulnerabilityAssignment.TicketStatus = reader.GetString(5);
                        vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);

```

```
72
73
74
75                  }
76              }
77              return vulnerabilityAssignment;
78          }
79
80      }
81
                1 reference
82      internal int Delete(int id)
83      {
84          using (SqlConnection connection = new SqlConnection(connectionString))
85          {
86              //prepared statement
87              //update
88              string sqlQuery = "DELETE FROM dbo.VulnerabilityAssignment WHERE VulnerabilityAssignmentID = @id";
89
90              //associate @id with Id parameter
91
92              SqlCommand command = new SqlCommand(sqlQuery, connection);
93
94              command.Parameters.Add("@Id", System.Data.SqlDbType.VarChar, 1000).Value = id;
95
96
97              connection.Open();
98              int deletedID = command.ExecuteNonQuery();
99
100             return deletedID;
101         }
102     }
103
                1 reference
104     public int Create(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
105     {
106
107         //access database
108         using (SqlConnection connection = new SqlConnection(connectionString))
109         {
110             string sqlQuery = "INSERT INTO dbo.VulnerabilityAssignment Values(@VulnerabilityID, @TicketUser, @DateAssigned, @DateClosed, @TicketStatus, @VulnerabilityTicketNotes)";
111
112             SqlCommand command = new SqlCommand(sqlQuery, connection);
113             command.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityID;
114             command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketUser;
115             command.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateAssigned;
116             command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateClosed;
117             command.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
118             command.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
119
120             //Update Vulnerability that it has been assigned
121             string sqlQuery1 = "Update dbo.Vulnerability SET DateOpen = @DateOpen, DateClosed = @DateClosed, StatusID = @StatusID, TicketNotes = @TicketNotes, AssignedUser =
                  @AssignedUser WHERE dbo.Vulnerability.VulnerabilityID = @VulnerabilityID";
122
123             //associate @id with Id parameter
124             SqlCommand command1 = new SqlCommand(sqlQuery1, connection);
125
126             command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityAssignmentModel.VulnerabilityID;
127             command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateClosed;
128             command1.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateAssigned;
129             command1.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
130             command1.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
131             command1.Parameters.Add("@AssignedUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketUser;
132
133             //Insert Assignment To Vulnerability Log
134             string sqlQuery2 = "INSERT INTO dbo.VulnerabilityLog VALUES(@VulnerabilityID, @TicketUser, @CurrentTimeStamp, @TicketStatus,  @VulnerabilityTicketNotes)";
135
136             //associate @id with Id parameter
137             SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
138
139             command2.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityAssignmentModel.VulnerabilityID;
140             command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.CurrentUser;
141             command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
```

```
            command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
            command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
            command2.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;


            connection.Open();
            int newVulnerabilityAssignmentID = command.ExecuteNonQuery();
            command1.ExecuteNonQuery();
            command2.ExecuteNonQuery();
            return newVulnerabilityAssignmentID;
        }
    }

    1 reference
    public int Update(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
    {
        //access the database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            //prepared statement
            //update
            string sqlQuery = "UPDATE dbo.VulnerabilityAssignment SET VulnerabilityID = @VulnerabilityID, TicketUser = @TicketUser, DateAssigned = @DateAssigned, DateClosed =
              @DateClosed, TicketStatus = @TicketStatus, VulnerabilityTicketNotes = @VulnerabilityTicketNotes WHERE VulnerabilityAssignmentID = @VulnerabilityAssignmentID";

            //associate @id with Id parameter


            SqlCommand command = new SqlCommand(sqlQuery, connection);

            command.Parameters.Add("@VulnerabilityAssignmentID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityAssignmentID;
            command.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityID;
            command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketUser;
            command.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateAssigned;
            command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateClosed;
            command.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
            command.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;

            //Update Vulnerability
            string sqlQuery1 = "Update dbo.Vulnerability SET DateOpen = @DateOpen, DateClosed = @DateClosed, StatusID = @StatusID, TicketNotes = @TicketNotes WHERE
              dbo.Vulnerability.VulnerabilityID = @VulnerabilityID";

            //associate @id with Id parameter
            SqlCommand command1 = new SqlCommand(sqlQuery1, connection);

            command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityAssignmentModel.VulnerabilityID;
            command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateClosed;
            command1.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateAssigned;
            command1.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
            command1.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
            command1.Parameters.Add("@AssignedUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketUser;

            //Insert Edit To Vulnerability Log
            string sqlQuery2 = "INSERT INTO dbo.VulnerabilityLog VALUES(@VulnerabilityID, @TicketUser, @CurrentTimeStamp, @TicketStatus,  @VulnerabilityTicketNotes)";

            //associate @id with Id parameter
            SqlCommand command2 = new SqlCommand(sqlQuery2, connection);

            command2.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityAssignmentModel.VulnerabilityID;
            command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.CurrentUser;
            command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
            command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
            command2.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;

            connection.Open();
            int newVulnerabilityAssignmentID = command.ExecuteNonQuery();
            command1.ExecuteNonQuery();
            command2.ExecuteNonQuery();
            return newVulnerabilityAssignmentID;
        }

    }
    1 reference
    public List<VulnerabilityAssignmentModel> GetVulnerabilityAssignmentsEngineer(string session)
```

```
1 reference
public List<VulnerabilityAssignmentModel> GetVulnerabilityAssignmentsEngineer(string session)
{
    List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketUser = @session AND TicketStatus = 'Open'";

        SqlCommand command = new SqlCommand(sqlQuery, connection);
        command.Parameters.Add("@session", System.Data.SqlDbType.VarChar).Value = session;

        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                //creating new TestAssignment object and adding it to the list to be returned
                VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
                vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
                vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
                vulnerabilityAssignment.TicketUser = reader.GetString(2);
                vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
                vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
                vulnerabilityAssignment.TicketStatus = reader.GetString(5);
                vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
                returnList.Add(vulnerabilityAssignment);
            }
        }
    }

    return returnList;
}

1 reference
public List<VulnerabilityAssignmentModel> GetVulnerabilityAssignmentsPenTester()
{
    List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketStatus = 'Re-Test'";

        SqlCommand command = new SqlCommand(sqlQuery, connection);

        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                //creating new TestAssignment object and adding it to the list to be returned
                VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
                vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
                vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
                vulnerabilityAssignment.TicketUser = reader.GetString(2);
                vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
                vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
                vulnerabilityAssignment.TicketStatus = reader.GetString(5);
                vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
                returnList.Add(vulnerabilityAssignment);
            }

        }

    }
    return returnList;
}

1 reference
public List<VulnerabilityAssignmentModel> GetTestAssignmentsPenTester(string session)
{
    List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        string sqlQuery2 = "SELECT TestAssignmentID, TestName, DateAssigned, DateClosed, TicketStatus, TestTicketNotes  FROM dbo.TestAssignment WHERE TicketStatus = 'Open' AND
          TicketUser = @session";
        SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
        command2.Parameters.Add("@session", System.Data.SqlDbType.VarChar).Value = session;
        connection.Open();
        SqlDataReader reader2 = command2.ExecuteReader();
        if (reader2.HasRows)
        {
            while (reader2.Read())
            {
                //creating new TestAssignment object and adding it to the list to be returned
                VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
                vulnerabilityAssignment.TestAssignmentID = reader2.GetInt32(0);
                vulnerabilityAssignment.TestName1 = reader2.GetString(1);
                vulnerabilityAssignment.DateAssigned1 = reader2.GetDateTime(2);
                vulnerabilityAssignment.DateClosed1 = reader2.GetDateTime(3);
                vulnerabilityAssignment.TicketStatus1 = reader2.GetString(4);
                vulnerabilityAssignment.TestTicketNotes1 = reader2.GetString(5);
                returnList.Add(vulnerabilityAssignment);
            }

        }

    }
    return returnList;
}
}
```

VulnerabilityAssignmentModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Models
{

    public class VulnerabilityAssignmentModel
    {
        [Display(Name = "Vulnerability Assignment ID")]

        public int VulnerabilityAssignmentID { get; set; }
        [Display(Name = "Test Assignment ID")]

        public int TestAssignmentID { get; set; }
        [Display(Name = "Vulnerability ID")]

        public int VulnerabilityID { get; set; }
        [Display(Name = "Ticket User")]

        public string TicketUser { get; set; }
        [Display(Name = "Current User")]

        public string CurrentUser { get; set; }
        [Display(Name = "Test Name")]

        public string TestName { get; set; }
        [Display(Name = "Severity")]

        public string Severity { get; set; }
        [Display(Name = "Test Name")]

        public string TestName1 { get; set; }
        [Display(Name = "Date Assigned")]

        public DateTime DateAssigned1 { get; set; }
        [Display(Name = "Date Closed")]

        public DateTime DateClosed1 { get; set; }
        [Display(Name = "Ticket Notes")]

        public string TicketNotes { get; set; }
        [Display(Name = "Ticket Notes")]

        public string TestTicketNotes { get; set; }
        [Display(Name = "Ticket Status")]

        public string TicketStatus1 { get; set; }
        [Display(Name = "Ticket Notes")]
        1 reference
        public string TestTicketNotes1 { get; set; }
        [Display(Name = "Status")]
        0 references
        public string StatusID { get; set; }
        [Display(Name = "Date Assigned")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        9 references
        public DateTime DateAssigned { get; set; }
        [Display(Name = "Date Open")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        0 references
        public DateTime DateOpen { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        9 references
        public DateTime DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        11 references
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        11 references
        public string VulnerabilityTicketNotes { get; set; }

        3 references
        public List<SelectListItem> ListVulnerabilityID { get; set; }
        3 references
        public List<SelectListItem> ListTicketUser { get; set; }
        3 references
        public List<SelectListItem> ListVulnerabilityStatusPenTester { get; set; }
        3 references
        public List<SelectListItem> ListVulnerabilityStatusEngineer { get; set; }

        0 references
        public VulnerabilityAssignmentModel(int vulnerabilityAssignmentID, int vulnerabilityID, string ticketUser, DateTime dateAssigned, DateTime dateClosed, string ticketStatus, string
            vulnerabilityTicketNotes)
        {
            VulnerabilityAssignmentID = vulnerabilityAssignmentID;
            VulnerabilityID = vulnerabilityID;
            TicketUser = ticketUser;
            DateAssigned = dateAssigned;
            DateClosed = dateClosed;
            TicketStatus = ticketStatus;
            VulnerabilityTicketNotes = vulnerabilityTicketNotes;
        }

        6 references
        public VulnerabilityAssignmentModel()
        {
            this.ListVulnerabilityID = new List<SelectListItem>();
            this.ListTicketUser = new List<SelectListItem>();
            this.ListVulnerabilityStatusPenTester = new List<SelectListItem>();
            this.ListVulnerabilityStatusEngineer = new List<SelectListItem>();
        }
    }
}
```

VulnerabilityAssignment Views

CreateVulnerabilityAssignmentForm.cshtml

```
@model Project.Models.VulnerabilityAssignmentModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>


@using (Html.BeginForm("CreateVulnerabilityAssignment", "VulnerabilityAssignment"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Assign Vulnernability</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div hidden class="form-group">
        @Html.LabelFor(model => model.VulnerabilityAssignmentID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.VulnerabilityAssignmentID, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.VulnerabilityAssignmentID, "", new { @class = "text-danger" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.VulnerabilityID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.VulnerabilityID, Model.ListVulnerabilityID, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
        </div>
    </div>
    <div hidden class="form-group">
        @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.DateAssigned, new { type = "date" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBoxFor(model => model.DateClosed, new { type = "date" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.VulnerabilityTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.VulnerabilityTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.VulnerabilityTicketNotes, "", new { @class = "text-danger" })
        </div>
    </div>
    @{
        String Role;
        Role = Session["userRole"].ToString();
        if (Role == "PenetrationTester")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }
```

```
78                  </div>
79              }
80              else if (Role == "SecurityEngineer")
81              {
82                  <div class="form-group">
83                      @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
84                      <div class="col-md-10">
85                          @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusEngineer, "Please select")
86                      </div>
87                  </div>
88                  Response.Redirect("~/Home/Index");
89              }
90              else if (Role == "Administrator")
91              {
92                  <div class="form-group">
93                      @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
94                      <div class="col-md-10">
95                          @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
96                      </div>
97                  </div>
98              }
99          }
100
101         <div class="form-group">
102             <div class="col-md-offset-2 col-md-10">
103                 <input type="submit" value="Create" class="btn btn-default" />
104             </div>
105         </div>
106     </div>
107     }
108
109     <div>
110         @Html.ActionLink("Back to List", "Index")
111     </div>
112
113     @section Scripts {
114         @Scripts.Render("~/bundles/jqueryval")
115     }
116
```

Details.cshtml

```
@model Project.Models.VulnerabilityAssignmentModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2></h2>

<div>
    <br />
    <hr />
    <h4><b>Assigned Vulnerability Information</b></h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.VulnerabilityAssignmentID)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.VulnerabilityID)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TicketUser)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TicketUser)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateAssigned)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateAssigned)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateClosed)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateClosed)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.VulnerabilityTicketNotes)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TicketStatus)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.VulnerabilityAssignmentID }) |
    @Html.ActionLink("Back to List", "Index")
</p>

```

Index.cshtml

```
@model IEnumerable<Project.Models.VulnerabilityAssignmentModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2></h2>

<p>
    @{
        String Role;
        Role = Session["userRole"].ToString();
        if (Role == "Administrator")
        {
            <br />
            <hr />
            <h4><b>Assigned Vulnernabilities</b></h4>
            <hr />
            @Html.ActionLink("Create New", "Create")
        }
        else if (Role == "PenetrationTester")
        {
            <br />
            <hr />
            <h4><b>Assigned Vulnernabilities</b></h4>
            <hr />
            @Html.ActionLink("Create New", "Create")
        }
        else if (Role == "Engineer")
        {
            Response.Redirect("~/Home/Index");
        }
    }

</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketUser)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateAssigned)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateClosed)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.VulnerabilityAssignmentID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.VulnerabilityID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TicketUser)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateAssigned)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateClosed)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TicketStatus)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.VulnerabilityAssignmentID }, new { session = Session["Username"] }) |
                @Html.ActionLink("Details", "Details", new { id = item.VulnerabilityAssignmentID }, new { session = Session["Username"] }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.VulnerabilityAssignmentID }, new { session = Session["Username"] })
            </td>
        </tr>
    }

</table>
```

TestAssignmentsPenTester.cshtml

```
@model IEnumerable<Project.Models.VulnerabilityAssignmentModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<br />
<hr />
<h4><b></b></h4>
<hr />
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.TestAssignmentID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestName1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateAssigned1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateClosed1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TestTicketNotes1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus1)
        </th>

        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.TestAssignmentID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TestName1)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateAssigned1)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateClosed1)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TestTicketNotes1)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TicketStatus1)
    </td>

</tr>
    }

</table>

```

UpdateVulnerabilityAssignment.cshtml

```
@model Project.Models.VulnerabilityAssignmentModel

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2> </h2>


@using (Html.BeginForm("UpdateVulnerabilityAssignment", "VulnerabilityAssignment"))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <br />
    <hr />
    <h4><b>Update Vulnerability Assignment</b></h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div hidden class="form-group">
        @Html.LabelFor(model => model.VulnerabilityAssignmentID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.VulnerabilityAssignmentID, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.VulnerabilityAssignmentID, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.VulnerabilityID, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.VulnerabilityID, Model.ListVulnerabilityID, "Please select")
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
        </div>
    </div>

    <div hidden class="form-group">
        @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
        <div hidden class="col-md-10">
            @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateAssigned, new { htmlAttributes = new { @class = "form-control" } })


        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.DateClosed, new { htmlAttributes = new { @class = "form-control" } })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(model => model.VulnerabilityTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.VulnerabilityTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.VulnerabilityTicketNotes, "", new { @class = "text-danger" })
        </div>
    </div>
    @{
        String Role;
        Role = Session["userRole"].ToString();
        if (Role == "PenetrationTester")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
                @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }
        else if (Role == "SecurityEngineer")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusEngineer, "Please select")
                </div>
            </div>
            Response.Redirect("~/Home/Index");
        }
        else if (Role == "Administrator")
        {
            <div class="form-group">
                @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
                </div>
            </div>
        }
    }


    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

VulnerabilityAssignmentsEngineer.cshtml

```
@model IEnumerable<Project.Models.VulnerabilityAssignmentModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<br />
<hr />
<h4><b>Tasks</b></h4>
<hr />
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketUser)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateAssigned)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateClosed)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityAssignmentID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TicketUser)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateAssigned)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateClosed)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TicketStatus)
    </td>
</tr>
    }

</table>

```

VulnerabilityAssignmentsPenTester.cshtml

```
@model IEnumerable<Project.Models.VulnerabilityAssignmentModel>

@{
    ViewBag.Title = "";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<br />
<hr />
<h4><b>Vulnernabilities Ready For Re-Test</b></h4>
<hr />
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketUser)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateAssigned)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateClosed)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TicketStatus)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityAssignmentID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TicketUser)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateAssigned)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.DateClosed)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.TicketStatus)
    </td>
</tr>
    }

</table>

```

Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Vulnerability Management</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Vulnerability Management", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    @{ if (Session["userRole"] != null)
                        {
                            String Role;
                            Role = Session["userRole"].ToString();
                            if (Role == "Administrator")
                            {
                                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                                <li>@Html.ActionLink("Assign Test", "Index", "TestAssignment")</li>
                                <li>@Html.ActionLink("Tests", "Index", "Tests")</li>
                                <li>@Html.ActionLink("Assign Vulnerability", "Index", "VulnerabilityAssignment")</li>
                                <li>@Html.ActionLink("Vulnerabilities", "Index", "Vulnerabilities")</li>
                                <li>@Html.ActionLink("Assets", "Index", "Assets")</li>
                                if (Session["userRole"] != null)
                                {
                                    <li> <a href="#"> @Session["username"]</a></li>
                                }
                                else
                                {
                                    <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
                                }
                                <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>


                            }
                            else if (Role == "PenetrationTester")
                            {
                                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                                <li>@Html.ActionLink("Tasks", "TestAssignmentsPenTester", "VulnerabilityAssignment", new { session = Session["Username"] }, null)</li>
                                <li>@Html.ActionLink("Ready For Re-Test", "VulnerabilityAssignmentsPenTester", "VulnerabilityAssignment")</li>
                                <li>@Html.ActionLink("Assign Test", "Index", "TestAssignment")</li>
                                <li>@Html.ActionLink("Tests", "Index", "Tests")</li>
                                <li>@Html.ActionLink("Assign Vulnerability", "Index", "VulnerabilityAssignment")</li>
                                <li>@Html.ActionLink("Vulnerabilities", "Index", "Vulnerabilities")</li>
                                <li>@Html.ActionLink("Assets", "Index", "Assets")</li>
                                if (Session["userRole"] != null)
                                {
                                    <li> <a href="#"> @Session["username"]</a></li>
                                }
                                else
                                {
                                    <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
                                }
                                <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>

                            }
                            else if (Role == "SecurityEngineer")
                            {
                                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                                <li>@Html.ActionLink("Tasks", "VulnerabilityAssignmentsEngineer", "VulnerabilityAssignment", new { session = Session["Username"] }, null)</li>
                                <li>@Html.ActionLink("Assign Vulnerability", "Index", "VulnerabilityAssignment")</li>
                                <li>@Html.ActionLink("Vulnerabilities", "Index", "Vulnerabilities")</li>
                                if (Session["userRole"] != null)
                                {
                                    <li> <a href="#"> @Session["username"]</a></li>
                                }
                                else
```

```
                                else
                                {
                                    <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
                                }
                                <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>
                            }
                        }
                        else
                        {
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
                            <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>

                        }

                    }

                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Vulnerability Management</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>

```

Error.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Error</title>
</head>
<body>
    <hgroup>
        <h1>Error.</h1>
        <h2>An error occurred while processing your request.</h2>
    </hgroup>
</body>
</html>

```

VulnerabilityTasksController.cshtml

```
using Project.Data;
using Project.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Project.Controllers
{
    0 references
    public class VulnerabilityTasksController : Controller
    {
        [CustomAuthorization]
        // GET: VulnerabilityTasks
        0 references
        public ActionResult Index()
        {
            List<VulnerabilityTasksModel> vulnerabilityTasksModel = new List<VulnerabilityTasksModel>();
            //Session["usersName"] = vulnerabilityTasksModel.UsersName;
            VulnerabilityTasksDAO vulnerabilityTasksDAO = new VulnerabilityTasksDAO();
          //  vulnerabilityTasksModel = vulnerabilityTasksDAO.GetTasks();
            return View("VulnerabilityTasks", vulnerabilityTasksModel);
        }
    }
}
```

VulnerabilityTasksDAO.cs

```
using Project.Models;
using System.Data.SqlClient;

namespace Project.Data
{
    2 references
    internal class VulnerabilityTasksDAO
    {
        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
          Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

        0 references
        public VulnerabilityTasksModel GetTasks()
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                //SQL Statement: Get all vulnerability tasks for user logged in
                string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketUser = @UsersName AND TicketStatus = 'Open'";
                VulnerabilityTasksModel vulnerabilityTasksModel = new VulnerabilityTasksModel();
                SqlCommand command = new SqlCommand(sqlQuery, connection);
                command.Parameters.Add("@UsersName", System.Data.SqlDbType.VarChar, 50).Value = vulnerabilityTasksModel.UsersName;
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        vulnerabilityTasksModel.VulnerabilityAssignmentID = reader.GetInt32(0);
                        vulnerabilityTasksModel.VulnerabilityID = reader.GetInt32(1);
                        vulnerabilityTasksModel.TicketUser = reader.GetString(2);
                        vulnerabilityTasksModel.DateAssigned = reader.GetDateTime(3);
                        vulnerabilityTasksModel.DateClosed = reader.GetDateTime(4);
                        vulnerabilityTasksModel.TicketStatus = reader.GetString(5);
                        vulnerabilityTasksModel.VulnerabilityTicketNotes = reader.GetString(6);
                    }


                }
                reader.Close();
                connection.Close();

                return vulnerabilityTasksModel;
            }
        }
    }
}
```

VulnerabilityTaskModel.cs

```cs
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Project.Models
{

    public class VulnerabilityTasksModel
    {
        //public List<SelectListItem> vulnerabilityTasksModel { get; set; }
        [Display(Name = "Vulnerability Assignment ID")]
        public int VulnerabilityAssignmentID { get; set; }
        [Display(Name = "Vulnerability ID")]
        public int VulnerabilityID { get; set; }
        [Display(Name = "Assignee")]
        public string TicketUser { get; set; }
        [Display(Name = "Date Assigned")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        public DateTime DateAssigned { get; set; }
        [Display(Name = "Date Closed")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
        public DateTime DateClosed { get; set; }
        [Display(Name = "Ticket Status")]
        public string TicketStatus { get; set; }
        [Display(Name = "Ticket Notes")]
        public string VulnerabilityTicketNotes { get; set; }
        [Display(Name = "User")]
        public string UsersName { get; set; }


        public VulnerabilityTasksModel(int vulnerabilityAssignmentID, int vulnerabilityID, string ticketUser, DateTime dateAssigned, DateTime dateClosed, string ticketStatus, string
            vulnerabilityTicketNotes)
        {
            VulnerabilityAssignmentID = vulnerabilityAssignmentID;
            VulnerabilityID = vulnerabilityID;
            TicketUser = ticketUser;
            DateAssigned = dateAssigned;
            DateClosed = dateClosed;
            TicketStatus = ticketStatus;
            VulnerabilityTicketNotes = vulnerabilityTicketNotes;
        }


        public VulnerabilityTasksModel()
        {



        }
    }
}
```