



Vulnerability Management Tool

Technical Manual

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

Table of Contents

Introduction	5
Database Configuration SQL Statements	6
Dbó.Asset.....	6
Dbó.Assignment.....	6
Dbó.Contact	6
Dbó.Credentials.....	6
Dbó.Function.....	6
Dbó.MitreTactic	7
Dbó.MitreTechnique.....	7
Dbó.Password.....	7
Dbó.Roles.....	7
Dbó.Session.....	7
Dbó.Status	7
Dbó.Test.....	7
Dbó.TestAssignment	8
Dbó.TestLog.....	8
Dbó.TestStatus	8
Dbó.Users.....	8
Dbó.Vulnerability	8
Dbó.VulnerabilityAssignment	9
Dbó.VulnerabilityLog.....	9
Dbó.VulnerabilityStatusEngineer.....	9
Dbó.VulnerabilityStatusPenTester	9
Code Implementation	10
RouteConfig.cs.....	10
AssetsController.cs.....	11
AssetDAO.cs.....	12
AssetModel.cs	15
AddAssetForm.cshtml	16
Details.cshtml.....	19
Index.cshtml.....	22
UpdateAssetForm.cshtml.....	24
CustomAuthorizationAttribute.....	27
SecurityService.cs	28

SecurityDAO.cs.....	29
LoginController.cs.....	30
UserModel.cs	31
Login.cshtml.....	32
LoginFailure.cs.....	32
LoginSuccess.cshtml	33
DashboardController.cs	34
ClosedTestModel.cs	38
ClosedVulnerabilityModel.cs.....	39
HighSeverityModel.cs	40
LowSeverityModel.cs.....	41
MediumSeverityModel.cs.....	42
OpenTestModel.cs.....	43
OpenVulnerabilityModel.cs.....	44
Index.cshtml	45
HomeController.cs.....	48
Index.cshtml.....	49
TestAssignmentController.cs	50
TestAssignmentDAO.cs	52
TestAssignmentModel.cs.....	55
CreateTestAssignmentForm.cshtml	56
Details.cshtml.....	57
Index.cshtml.....	59
UpdateTestAssignmentForm.cshtml	61
TestsController.cs.....	63
TestDAO.cshtml.....	65
TestModel.cs	68
CreateTestForm.cshtml.....	70
Details.cshtml.....	72
Index.cshtml.....	74
TestEventLog.cshtml.....	76
UpdateTestForm.cshtml.....	77
VulnerabilitiesController.cshtml	79
VulnerabilityDAO.cs.....	82
VulnerabilityModel.cs	88

AddVulnerabilityForm.css.....	90
Details.cshtml.....	95
Index.cshtml.....	98
UpdateVulnerabilityForm.cshtml.....	99
VulnerabilityEventLog.cshtml.....	103
VulnerabilityAssignmentsController.cs.....	104
VulnerabilityAssignmentDAO.cs.....	106
VulnerabilityAssignmentModel.cs.....	110
CreateVulnerabilityAssignmentForm.cshtml.....	111
Details.cshtml.....	113
Index.cshtml.....	114
TestAssignmentsPenTester.cshtml.....	115
UpdateVulnerabilityAssignment.cshtml.....	116
VulnerabilityAssignmentsEngineer.cshtml.....	118
VulnerabilityAssignmentsPenTester.cshtml.....	119
Layout.cshtml.....	120
Error.cshtml.....	121
VulnerabilityTasksController.cshtml.....	122
VulnerabilityTasksDAO.cs.....	122
VulnerabilityTaskModel.cs.....	123

Introduction

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

Database Configuration SQL Statements

Dbó.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  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.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)  
);
```

Dbó.Session

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

Dbó.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)  
);
```

Dbó.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)
);

```

Dbó.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)
);

```

Dbó.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)
);

```

Dbó.TestStatus

```

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

```

Dbó.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)
);

```

Dbó.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)
);

```

Dbv.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)
);

```

Dbv.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)
);

```

Dbv.VulnerabilityStatusEngineer

```

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

```

Dbv.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  {
10     1 reference
11     public class RouteConfig
12     {
13         1 reference
14         public static void RegisterRoutes(RouteCollection routes)
15         {
16             routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
17
18             routes.MapRoute(
19                 name: "Private",
20                 url: "Private",
21                 defaults: new { controller = "Login", action = "onPrivateURL", id = UrlParameter.Optional }
22             );
23
24             routes.MapRoute(
25                 name: "Default",
26                 url: "{controller}/{action}/{id}",
27                 defaults: new { controller = "Dashboard", action = "Index", id = UrlParameter.Optional }
28             );
29         }
30     }
31 }
```

Asset Controller

AssetsController.cs

```
1  using Project.Data;
2  using Project.Models;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Web;
7  using System.Web.Mvc;
8
9  namespace Project.Controllers
10 {
11     0 references
12     public class AssetsController : Controller
13     {
14         // GET: Assets
15         [CustomAuthorization]
16         0 references
17         public ActionResult Index()
18         {
19             List<AssetModel> assets = new List<AssetModel>();
20             AssetDAO assetDAO = new AssetDAO();
21             assets = assetDAO.FetchAll();
22             return View("Index", assets);
23         }
24         [CustomAuthorization]
25         0 references
26         public ActionResult Details(int id)
27         {
28             AssetDAO assetDAO = new AssetDAO();
29             AssetModel asset = assetDAO.FetchOne(id);
30             return View("Details", asset);
31         }
32         [CustomAuthorization]
33         0 references
34         public ActionResult Create()
35         {
36             return View("AddAssetForm");
37         }
38         [CustomAuthorization]
39         0 references
40         public ActionResult AddAsset(AssetModel assetModel)
41         {
42             //save it to the database
43             AssetDAO assetDAO = new AssetDAO();
44             assetDAO.Create(assetModel);
45             return View("Details", assetModel);
46         }
47         [CustomAuthorization]
48         0 references
49         public ActionResult Edit(int id)
50         {
51             AssetDAO assetDAO = new AssetDAO();
52             AssetModel asset = assetDAO.FetchOne(id);
53             return View("UpdateAssetForm", asset);
54         }
55         [CustomAuthorization]
56         [HttpPost]
57         0 references
58         public ActionResult UpdateAsset(AssetModel assetModel)
59         {
60             //save it to the database
61             AssetDAO assetDAO = new AssetDAO();
62
63             assetDAO.Update(assetModel);
64             return View("Details", assetModel);
65         }
66         [CustomAuthorization]
67         0 references
68         public ActionResult Delete(int id)
69         {
70             AssetDAO assetDAO = new AssetDAO();
71             assetDAO.Delete(id);
72             List<AssetModel> assets = assetDAO.FetchAll();
73             return View("Index", assets);
74         }
75     }
76 }
```

AssetDAO.cs

```
1 using Project.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Data.SqlClient;
5 using System.Web;
6
7 namespace Project.Data
8 {
9     12 references
10    internal class AssetDAO
11    {
12        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
13        Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
14        //performs all operations
15
16
17
18        2 references
19        public List<AssetModel> FetchAll()
20        {
21            List<AssetModel> returnList = new List<AssetModel>();
22            using (SqlConnection connection = new SqlConnection(connectionString))
23            {
24                string sqlQuery = "SELECT * from dbo.Asset";
25                SqlCommand command = new SqlCommand(sqlQuery, connection);
26
27                connection.Open();
28                SqlDataReader reader = command.ExecuteReader();
29
30                if (reader.HasRows)
31                {
32                    while (reader.Read())
33                    {
34                        //create asset object
35                        AssetModel asset = new AssetModel();
36                        asset.AssetId = reader.GetInt32(0);
37                        asset.AssetName = reader.GetString(1);
38                        asset.DeviceCategory = reader.GetString(2);
39                        asset.Hardware = reader.GetString(3);
40                        asset.OperatingSystem = reader.GetString(4);
41                        asset.MacAddress = reader.GetString(5);
42                        asset.IPv4Address = reader.GetString(6);
43                        asset.IPv6Address = reader.GetString(7);
44                        asset.OfficeLocation = reader.GetString(8);
45                        asset.BusinessUnit = reader.GetString(9);
46                        asset.DataClassification = reader.GetString(10);
47                        asset.Risk = reader.GetString(11);
48                        asset.OwnerName = reader.GetString(12);
49                        asset.TechnicalSupportContact = reader.GetString(13);
50                        asset.DateCreated = reader.GetDateTime(14);
51
52                        returnList.Add(asset);
53                    }
54                }
55            }
56
57            return returnList;
58        }
59    }
60 }
```

```

60
61 1 reference
62  internal int Delete(int id)
63  {
64      using (SqlConnection connection = new SqlConnection(connectionString))
65      {
66          //prepared statement
67          //update
68          string sqlQuery = "DELETE FROM dbo.Asset WHERE AssetID = @id";
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
87  2 references
88  public AssetModel FetchOne(int id)
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         return asset;
126     }
127 }
128

```

```

127     }
128 }
129 }
130 1 reference public int Create(AssetModel assetModel)
131 {
132     //access database
133     using (SqlConnection connection = new SqlConnection(connectionString))
134     {
135         string sqlQuery = "INSERT INTO dbo.Asset Values(@AssetName, @DeviceCategory, @Hardware, @OperatingSystem, @MacAddress, @IPv4Address, @IPv6Address, @OfficeLocation,
136             @BusinessUnit, @DataClassification, @Risk, @OwnerName, @TechnicalSupportContact, @DateCreated)";
137
138         SqlCommand command = new SqlCommand(sqlQuery, connection);
139         command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.AssetName;
140         command.Parameters.Add("@DeviceCategory", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DeviceCategory;
141         command.Parameters.Add("@Hardware", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Hardware;
142         command.Parameters.Add("@OperatingSystem", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OperatingSystem;
143         command.Parameters.Add("@MacAddress", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.MacAddress;
144         command.Parameters.Add("@IPv4Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv4Address;
145         command.Parameters.Add("@IPv6Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv6Address;
146         command.Parameters.Add("@OfficeLocation", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OfficeLocation;
147         command.Parameters.Add("@BusinessUnit", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.BusinessUnit;
148         command.Parameters.Add("@DataClassification", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DataClassification;
149         command.Parameters.Add("@Risk", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Risk;
150         command.Parameters.Add("@OwnerName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OwnerName;
151         command.Parameters.Add("@TechnicalSupportContact", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.TechnicalSupportContact;
152         command.Parameters.Add("@DateCreated", System.Data.SqlDbType.DateTime).Value = assetModel.DateCreated;
153
154         connection.Open();
155         int newAssetID = command.ExecuteNonQuery();
156
157         return newAssetID;
158     }
159 }
160 }
161 }

```

```

162 1 reference public int Update(AssetModel assetModel)
163 {
164     //access the database
165     using (SqlConnection connection = new SqlConnection(connectionString))
166     {
167         //prepared statement
168         //update
169         string sqlQuery = "UPDATE dbo.Asset SET AssetName = @AssetName, DeviceCategory = @DeviceCategory, Hardware = @Hardware, OperatingSystem = @OperatingSystem, MacAddress =
170             @MacAddress, IPv4Address = @IPv4Address, IPv6Address = @IPv6Address, OfficeLocation = @OfficeLocation, BusinessUnit = @BusinessUnit, DataClassification =
171             @DataClassification, Risk = @Risk, OwnerName = @OwnerName, TechnicalSupportContact = @TechnicalSupportContact, DateCreated = @DateCreated WHERE AssetID = @AssetID";
172
173         //associate @id with Id parameter
174
175         SqlCommand command = new SqlCommand(sqlQuery, connection);
176
177         command.Parameters.Add("@AssetID", System.Data.SqlDbType.Int).Value = assetModel.AssetID;
178         command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.AssetName;
179         command.Parameters.Add("@DeviceCategory", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DeviceCategory;
180         command.Parameters.Add("@Hardware", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Hardware;
181         command.Parameters.Add("@OperatingSystem", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OperatingSystem;
182         command.Parameters.Add("@MacAddress", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.MacAddress;
183         command.Parameters.Add("@IPv4Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv4Address;
184         command.Parameters.Add("@IPv6Address", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.IPv6Address;
185         command.Parameters.Add("@OfficeLocation", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OfficeLocation;
186         command.Parameters.Add("@BusinessUnit", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.BusinessUnit;
187         command.Parameters.Add("@DataClassification", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.DataClassification;
188         command.Parameters.Add("@Risk", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.Risk;
189         command.Parameters.Add("@OwnerName", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.OwnerName;
190         command.Parameters.Add("@TechnicalSupportContact", System.Data.SqlDbType.VarChar, 1000).Value = assetModel.TechnicalSupportContact;
191         command.Parameters.Add("@DateCreated", System.Data.SqlDbType.DateTime).Value = assetModel.DateCreated;
192
193         connection.Open();
194         int newAssetID = command.ExecuteNonQuery();
195
196         return newAssetID;
197     }
198 }
199 }
200 }
201 }
202 }
203 }

```

AssetModel.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 {
9     3D references
10    public class AssetModel
11    {
12        [Display(Name = "Asset ID")]
13        4 references
14        public int AssetID { get; set; }
15        [Display(Name = "Asset Name")]
16        5 references
17        public string AssetName { get; set; }
18        [Display(Name = "Device Category")]
19        5 references
20        public string DeviceCategory { get; set; }
21        [Display(Name = "Hardware")]
22        5 references
23        public string Hardware { get; set; }
24        [Display(Name = "Operating System")]
25        5 references
26        public string OperatingSystem { get; set; }
27        [Display(Name = "Mac Address")]
28        5 references
29        public string MacAddress { get; set; }
30        [Display(Name = "IPv4 Address")]
31        5 references
32        public string IPv4Address { get; set; }
33        [Display(Name = "IPv6 Address")]
34        5 references
35        public string IPv6Address { get; set; }
36        [Display(Name = "Office Location")]
37        5 references
38        public string OfficeLocation { get; set; }
39        [Display(Name = "Business Unit")]
40        5 references
41        public string BusinessUnit { get; set; }
42        [Display(Name = "Data Classification")]
43        5 references
44        public string DataClassification { get; set; }
45        [Display(Name = "Risk Level")]
46        5 references
47        public string Risk { get; set; }
48        [Display(Name = "Owner Name")]
49        5 references
50        public string OwnerName { get; set; }
51        [Display(Name = "Technical Support Contact")]
52        5 references
53        public string TechnicalSupportContact { get; set; }
54        [Display(Name = "Date Created")]
55        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy}")]
56        5 references
57        public DateTime DateCreated { get; set; }
58    }
59
60    0 references
61    public AssetModel(int assetID, string assetName, string deviceCategory, string hardware, string operatingSystem, string macAddress, string ipv4Address, string ipv6Address, string
62    officeLocation, string businessUnit, string dataClassification, string risk, string ownerName, string technicalSupportContact, DateTime dateCreated)
63    {
64        AssetID = assetID;
65        AssetName = assetName;
66        DeviceCategory = deviceCategory;
67        Hardware = hardware;
68        OperatingSystem = operatingSystem;
69        MacAddress = macAddress;
70        IPv4Address = ipv4Address;
71        IPv6Address = ipv6Address;
72        OfficeLocation = officeLocation;
73        BusinessUnit = businessUnit;
74        DataClassification = dataClassification;
75        Risk = risk;
76        OwnerName = ownerName;
77        TechnicalSupportContact = technicalSupportContact;
78        DateCreated = dateCreated;
79    }
80
81    2 references
82    public AssetModel()
83    {
84    }
85 }
```

Asset Views

AddAssetForm.cshtml

```
1  @model Project.Models.AssetModel
2
3  @{
4  ViewBag.Title = "";
5  Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <h4>Add Asset</h4>
9  <hr />
10 <body>
11     @if (String.IsNullOrEmpty(Role))
12     {
13         Role = Session["userRole"].ToString();
14         if (Role == "Administrator")
15         {
16         }
17     }
18     else if (Role == "PenetrationTester")
19     {
20     }
21     else
22     {
23         Response.Redirect("~/Home/Index");
24     }
25 }
26
27 </body>
28
29
30 @using (Html.BeginForm("AddAsset", "Assets"))
31 {
32     @Html.AntiForgeryToken()
33
34     <div class="form-horizontal">
35         <br />
36         <hr />
37         <h4><b>Add Asset</b></h4>
38         <hr />
39         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
40         <div class="form-group">
41             @Html.LabelFor(model => model.AssetId, htmlAttributes: new { @class = "control-label col-md-2" })
42             <div class="col-md-10">
43                 @Html.EditorFor(model => model.AssetId, new { htmlAttributes = new { @class = "form-control" } })
44                 @Html.ValidationMessageFor(model => model.AssetId, "", new { @class = "text-danger" })
45             </div>
46         </div>
47
48         <div class="form-group">
49             @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
50             <div class="col-md-10">
51                 @Html.EditorFor(model => model.AssetName, new { htmlAttributes = new { @class = "form-control" } })
52                 @Html.ValidationMessageFor(model => model.AssetName, "", new { @class = "text-danger" })
53             </div>
54         </div>
55
56         <div class="form-group">
57             @Html.LabelFor(model => model.DeviceCategory, htmlAttributes: new { @class = "control-label col-md-2" })
58             <div class="col-md-10">
59                 @Html.EditorFor(model => model.DeviceCategory, new { htmlAttributes = new { @class = "form-control" } })
60                 @Html.ValidationMessageFor(model => model.DeviceCategory, "", new { @class = "text-danger" })
61             </div>
62         </div>
63
64         <div class="form-group">
65             @Html.LabelFor(model => model.Hardware, htmlAttributes: new { @class = "control-label col-md-2" })
66             <div class="col-md-10">
67                 @Html.EditorFor(model => model.Hardware, new { htmlAttributes = new { @class = "form-control" } })
68                 @Html.ValidationMessageFor(model => model.Hardware, "", new { @class = "text-danger" })
69             </div>
70         </div>
71     </div>
```



```

71
72
73 <div class="form-group">
74   @Html.LabelFor(model => model.OperatingSystem, htmlAttributes: new { @class = "control-label col-md-2" })
75   <div class="col-md-10">
76     @Html.EditorFor(model => model.OperatingSystem, new { htmlAttributes = new { @class = "form-control" } })
77     @Html.ValidationMessageFor(model => model.OperatingSystem, "", new { @class = "text-danger" })
78   </div>
79 </div>
80
81 <div class="form-group">
82   @Html.LabelFor(model => model.MacAddress, htmlAttributes: new { @class = "control-label col-md-2" })
83   <div class="col-md-10">
84     @Html.EditorFor(model => model.MacAddress, new { htmlAttributes = new { @class = "form-control" } })
85     @Html.ValidationMessageFor(model => model.MacAddress, "", new { @class = "text-danger" })
86   </div>
87 </div>
88
89 <div class="form-group">
90   @Html.LabelFor(model => model.Ipv4Address, htmlAttributes: new { @class = "control-label col-md-2" })
91   <div class="col-md-10">
92     @Html.EditorFor(model => model.Ipv4Address, new { htmlAttributes = new { @class = "form-control" } })
93     @Html.ValidationMessageFor(model => model.Ipv4Address, "", new { @class = "text-danger" })
94   </div>
95 </div>
96
97 <div class="form-group">
98   @Html.LabelFor(model => model.Ipv6Address, htmlAttributes: new { @class = "control-label col-md-2" })
99   <div class="col-md-10">
100     @Html.EditorFor(model => model.Ipv6Address, new { htmlAttributes = new { @class = "form-control" } })
101     @Html.ValidationMessageFor(model => model.Ipv6Address, "", new { @class = "text-danger" })
102   </div>
103 </div>
104
105 <div class="form-group">
106   @Html.LabelFor(model => model.OfficeLocation, htmlAttributes: new { @class = "control-label col-md-2" })
107   <div class="col-md-10">
108     @Html.EditorFor(model => model.OfficeLocation, new { htmlAttributes = new { @class = "form-control" } })
109     @Html.ValidationMessageFor(model => model.OfficeLocation, "", new { @class = "text-danger" })
110   </div>
111 </div>
112
113 <div class="form-group">
114   @Html.LabelFor(model => model.BusinessUnit, htmlAttributes: new { @class = "control-label col-md-2" })
115   <div class="col-md-10">
116     @Html.EditorFor(model => model.BusinessUnit, new { htmlAttributes = new { @class = "form-control" } })
117     @Html.ValidationMessageFor(model => model.BusinessUnit, "", new { @class = "text-danger" })
118   </div>
119 </div>
120
121 <div class="form-group">
122   @Html.LabelFor(model => model.DataClassification, htmlAttributes: new { @class = "control-label col-md-2" })
123   <div class="col-md-10">
124     @Html.EditorFor(model => model.DataClassification, new { htmlAttributes = new { @class = "form-control" } })
125     @Html.ValidationMessageFor(model => model.DataClassification, "", new { @class = "text-danger" })
126   </div>
127 </div>
128
129 <div class="form-group">
130   @Html.LabelFor(model => model.Risk, htmlAttributes: new { @class = "control-label col-md-2" })
131   <div class="col-md-10">
132     @Html.EditorFor(model => model.Risk, new { htmlAttributes = new { @class = "form-control" } })
133     @Html.ValidationMessageFor(model => model.Risk, "", new { @class = "text-danger" })
134   </div>
135 </div>
136
137 <div class="form-group">
138   @Html.LabelFor(model => model.OwnerName, htmlAttributes: new { @class = "control-label col-md-2" })
139   <div class="col-md-10">
140     @Html.EditorFor(model => model.OwnerName, new { htmlAttributes = new { @class = "form-control" } })
141     @Html.ValidationMessageFor(model => model.OwnerName, "", new { @class = "text-danger" })
142   </div>
143 </div>

```

```

143
144
145     <div class="form-group">
146         @Html.LabelFor(model => model.TechnicalSupportContact, htmlAttributes: new { @class = "control-label col-md-2" })
147         <div class="col-md-10">
148             @Html.EditorFor(model => model.TechnicalSupportContact, new { htmlAttributes = new { @class = "form-control" } })
149             @Html.ValidationMessageFor(model => model.TechnicalSupportContact, "", new { @class = "text-danger" })
150         </div>
151     </div>
152
153     <div class="form-group">
154         @Html.LabelFor(model => model.DateCreated, htmlAttributes: new { @class = "control-label col-md-2" })
155         <div class="col-md-10">
156             @Html.EditorFor(model => model.DateCreated, new { htmlAttributes = new { @class = "form-control" } })
157             @Html.ValidationMessageFor(model => model.DateCreated, "", new { @class = "text-danger" })
158         </div>
159     </div>
160
161     <div class="form-group">
162         <div class="col-md-offset-2 col-md-10">
163             <input type="submit" value="Save" class="btn btn-default" />
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

```
1 @model Project.Models.AssetModel
2
3 @f
4 ViewBag.Title = "";
5 Layout = "~/Views/Shared/_Layout.cshtml";
6
7 @f
8 String Role;
9 Role = Session["userRole"].ToString();
10 if (Role == "Administrator")
11 {
12
13
14 }
15 else if (Role == "PenetrationTester")
16 {
17
18 }
19 else
20 {
21     Response.Redirect("~/Home/Index");
22 }
23 }
24 <h4> </h4>
25
26 <div>
27     <br />
28     <hr />
29     <h4><b>Asset Intelligence</b></h4>
30     <hr />
31     <dl class="dl-horizontal">
32         <dt>
33             @Html.DisplayNameFor(model => model.AssetId)
34         </dt>
35         <dd>
36             @Html.DisplayFor(model => model.AssetId)
37         </dd>
38     </dl>
39     <dl>
40         <dt>
41             @Html.DisplayNameFor(model => model.AssetName)
42         </dt>
43         <dd>
44             @Html.DisplayFor(model => model.AssetName)
45         </dd>
46     </dl>
47     <dl>
48         <dt>
49             @Html.DisplayNameFor(model => model.DeviceCategory)
50         </dt>
51         <dd>
52             @Html.DisplayFor(model => model.DeviceCategory)
53         </dd>
54     </dl>
55     <dl>
56         <dt>
57             @Html.DisplayNameFor(model => model.Hardware)
58         </dt>
59         <dd>
60             @Html.DisplayFor(model => model.Hardware)
61         </dd>
62     </dl>
63     <dl>
64         <dt>
65             @Html.DisplayNameFor(model => model.OperatingSystem)
66         </dt>
67         <dd>
68             @Html.DisplayFor(model => model.OperatingSystem)
69         </dd>
70     </dl>
71     <dl>
72         <dt>
73             @Html.DisplayNameFor(model => model.MacAddress)
74         </dt>
75         <dd>
76             @Html.DisplayFor(model => model.MacAddress)
77         </dd>
78     </dl>
79 </div>
```

```

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

```
1  @model IEnumerable<Project.Models.AssetModel>
2
3  @{
4  ViewBag.Title = "";
5  Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7  @{
8  String Role;
9  Role = Session["userRole"].ToString();
10 if (Role == "Administrator")
11 {
12
13
14 }
15 else if (Role == "PenetrationTester")
16 {
17
18 }
19 else
20 {
21 Response.Redirect("~/Home/Index");
22 }
23 }
24 <h2> </h2>
25
26 <p>
27 <br />
28 <hr />
29 <h4><b>Asset Inventory</b></h4>
30 <hr />
31 @Html.ActionLink("Create New", "Create")
32 </p>
33 <table class="table">
34 <tr>
35 <th>
36 @Html.DisplayNameFor(model => model.AssetID)
37 </th>
38 <th>
39 @Html.DisplayNameFor(model => model.AssetName)
40 </th>
41 <th>
42 @Html.DisplayNameFor(model => model.DeviceCategory)
43 </th>
44 <th>
45 @Html.DisplayNameFor(model => model.Hardware)
46 </th>
47 <th>
48 @Html.DisplayNameFor(model => model.OperatingSystem)
49 </th>
50 <th>
51 @Html.DisplayNameFor(model => model.MacAddress)
52 </th>
53 <th>
54 @Html.DisplayNameFor(model => model.IPV4Address)
55 </th>
56 <th>
57 @Html.DisplayNameFor(model => model.IPV6Address)
58 </th>
59 <th>
60 @Html.DisplayNameFor(model => model.OfficeLocation)
61 </th>
62 <th>
63 @Html.DisplayNameFor(model => model.BusinessUnit)
64 </th>
65 <th>
66 @Html.DisplayNameFor(model => model.DataClassification)
67 </th>
68 <th>
69 @Html.DisplayNameFor(model => model.Risk)
70 </th>
71 <th>
72 @Html.DisplayNameFor(model => model.OwnerName)
73 </th>
74 <th>
75 @Html.DisplayNameFor(model => model.TechnicalSupportContact)
76 </th>
77 <th>
78 @Html.DisplayNameFor(model => model.DateCreated)
79 ...
--
```

```

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

```
1 @model Project.Models.AssetModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 @{
9     String Role;
10    Role = Session["userRole"].ToString();
11    if (Role == "Administrator")
12    {
13    }
14
15    else if (Role == "PenetrationTester")
16    {
17    }
18
19    else
20    {
21        Response.Redirect("~/Home/Index");
22    }
23 }
24 <h2> </h2>
25
26
27 @using (Html.BeginForm("UpdateAsset", "Assets"))
28 {
29     @Html.AntiForgeryToken()
30
31     <div class="form-horizontal">
32         <br />
33         <hr />
34         <h4><b>Update Asset</b></h4>
35         <hr />
36         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
37         <div class="form-group">
38             @Html.LabelFor(model => model.AssetID, htmlAttributes: new { @class = "control-label col-md-2" })
39             <div class="col-md-10">
40                 @Html.EditorFor(model => model.AssetID, new { htmlAttributes = new { @class = "form-control" } })
41                 @Html.ValidationMessageFor(model => model.AssetID, "", new { @class = "text-danger" })
42             </div>
43         </div>
44
45         <div class="form-group">
46             @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
47             <div class="col-md-10">
48                 @Html.EditorFor(model => model.AssetName, new { htmlAttributes = new { @class = "form-control" } })
49                 @Html.ValidationMessageFor(model => model.AssetName, "", new { @class = "text-danger" })
50             </div>
51         </div>
52
53         <div class="form-group">
54             @Html.LabelFor(model => model.DeviceCategory, htmlAttributes: new { @class = "control-label col-md-2" })
55             <div class="col-md-10">
56                 @Html.EditorFor(model => model.DeviceCategory, new { htmlAttributes = new { @class = "form-control" } })
57                 @Html.ValidationMessageFor(model => model.DeviceCategory, "", new { @class = "text-danger" })
58             </div>
59         </div>
60
61         <div class="form-group">
62             @Html.LabelFor(model => model.Hardware, htmlAttributes: new { @class = "control-label col-md-2" })
63             <div class="col-md-10">
64                 @Html.EditorFor(model => model.Hardware, new { htmlAttributes = new { @class = "form-control" } })
65                 @Html.ValidationMessageFor(model => model.Hardware, "", new { @class = "text-danger" })
66             </div>
67         </div>
68
69         <div class="form-group">
70             @Html.LabelFor(model => model.OperatingSystem, htmlAttributes: new { @class = "control-label col-md-2" })
71             <div class="col-md-10">
72                 @Html.EditorFor(model => model.OperatingSystem, new { htmlAttributes = new { @class = "form-control" } })
73                 @Html.ValidationMessageFor(model => model.OperatingSystem, "", new { @class = "text-danger" })
74             </div>
75         </div>
76
```



```

77 <div class="form-group">
78     @Html.LabelFor(model => model.MacAddress, htmlAttributes: new { @class = "control-label col-md-2" })
79     <div class="col-md-10">
80         @Html.EditorFor(model => model.MacAddress, new { htmlAttributes = new { @class = "form-control" } })
81         @Html.ValidationMessageFor(model => model.MacAddress, "", new { @class = "text-danger" })
82     </div>
83 </div>
84
85 <div class="form-group">
86     @Html.LabelFor(model => model.Ipv4Address, htmlAttributes: new { @class = "control-label col-md-2" })
87     <div class="col-md-10">
88         @Html.EditorFor(model => model.Ipv4Address, new { htmlAttributes = new { @class = "form-control" } })
89         @Html.ValidationMessageFor(model => model.Ipv4Address, "", new { @class = "text-danger" })
90     </div>
91 </div>
92
93 <div class="form-group">
94     @Html.LabelFor(model => model.Ipv6Address, htmlAttributes: new { @class = "control-label col-md-2" })
95     <div class="col-md-10">
96         @Html.EditorFor(model => model.Ipv6Address, new { htmlAttributes = new { @class = "form-control" } })
97         @Html.ValidationMessageFor(model => model.Ipv6Address, "", new { @class = "text-danger" })
98     </div>
99 </div>
100
101 <div class="form-group">
102     @Html.LabelFor(model => model.OfficeLocation, htmlAttributes: new { @class = "control-label col-md-2" })
103     <div class="col-md-10">
104         @Html.EditorFor(model => model.OfficeLocation, new { htmlAttributes = new { @class = "form-control" } })
105         @Html.ValidationMessageFor(model => model.OfficeLocation, "", new { @class = "text-danger" })
106     </div>
107 </div>
108
109 <div class="form-group">
110     @Html.LabelFor(model => model.BusinessUnit, htmlAttributes: new { @class = "control-label col-md-2" })
111     <div class="col-md-10">
112         @Html.EditorFor(model => model.BusinessUnit, new { htmlAttributes = new { @class = "form-control" } })
113         @Html.ValidationMessageFor(model => model.BusinessUnit, "", new { @class = "text-danger" })
114     </div>
115 </div>
116
117 <div class="form-group">
118     @Html.LabelFor(model => model.DataClassification, htmlAttributes: new { @class = "control-label col-md-2" })
119     <div class="col-md-10">
120         @Html.EditorFor(model => model.DataClassification, new { htmlAttributes = new { @class = "form-control" } })
121         @Html.ValidationMessageFor(model => model.DataClassification, "", new { @class = "text-danger" })
122     </div>
123 </div>
124
125 <div class="form-group">
126     @Html.LabelFor(model => model.Risk, htmlAttributes: new { @class = "control-label col-md-2" })
127     <div class="col-md-10">
128         @Html.EditorFor(model => model.Risk, new { htmlAttributes = new { @class = "form-control" } })
129         @Html.ValidationMessageFor(model => model.Risk, "", new { @class = "text-danger" })
130     </div>
131 </div>
132
133 <div class="form-group">
134     @Html.LabelFor(model => model.OwnerName, htmlAttributes: new { @class = "control-label col-md-2" })
135     <div class="col-md-10">
136         @Html.EditorFor(model => model.OwnerName, new { htmlAttributes = new { @class = "form-control" } })
137         @Html.ValidationMessageFor(model => model.OwnerName, "", new { @class = "text-danger" })
138     </div>
139 </div>
140
141 <div class="form-group">
142     @Html.LabelFor(model => model.TechnicalSupportContact, htmlAttributes: new { @class = "control-label col-md-2" })
143     <div class="col-md-10">
144         @Html.EditorFor(model => model.TechnicalSupportContact, new { htmlAttributes = new { @class = "form-control" } })
145         @Html.ValidationMessageFor(model => model.TechnicalSupportContact, "", new { @class = "text-danger" })
146     </div>
147 </div>
148
149 <div class="form-group">
150     @Html.LabelFor(model => model.DateCreated, htmlAttributes: new { @class = "control-label col-md-2" })
151     <div class="col-md-10">
152         @Html.EditorFor(model => model.DateCreated, new { htmlAttributes = new { @class = "form-control" } })
153     </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

```
1  using Project.Models;
2  using Project.Services.Business;
3  using System;
4  using System.Collections.Generic;
5  using System.Data.SqlClient;
6  using System.Web.Mvc;
7
8  namespace Project.Controllers
9  {
10     57 references
11     internal class CustomAuthorizationAttribute : FilterAttribute, IAuthorizationFilter
12     {
13         0 references
14         public void OnAuthorization(AuthorizationContext filterContext)
15         {
16             SecurityService securityService = new SecurityService();
17
18             //get user from session variable
19             UserModel user = (UserModel)filterContext.HttpContext.Session["user"];
20
21             bool success = false;
22             if (user != null)
23             {
24
25                 success = securityService.Authenticate(user);
26
27             }
28
29             if (success)
30             {
31
32             }
33         }
34         else
35         { //Make Unauthorised page here
36             filterContext.Result = new RedirectResult("~/Home/Index");
37         }
38
39         if (user == null)
40         {
41             filterContext.Result = new RedirectResult("~/Home/Index");
42         }
43     }
44
45
46
47
48     }
49
50 }
51
52 }
53
54
55
56
57
```

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  {
10     4 references
11     public class SecurityService
12     {
13         SecurityDAO daoService = new SecurityDAO();
14
15         2 references
16         public bool Authenticate(UserModel user)
17         {
18             return daoService.FindByUser(user);
19         }
20     }
21 }
22
```

SecurityDAO.cs

```
1 using Project.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Data.SqlClient;
5 using System.Linq;
6 using System.Web;
7 using System.Web.Mvc;
8
9 namespace Project.Services.Data
10 {
11     public class SecurityDAO
12     {
13         string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
14             Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
15
16         internal bool FindByUser(UserModel user)
17         {
18             bool success = false;
19             string sqlQuery = "SELECT * from dbo.Users WHERE Username = @Username AND Password = @Password";
20
21             using (SqlConnection connection = new SqlConnection(connectionString))
22             {
23                 SqlCommand command = new SqlCommand(sqlQuery, connection);
24
25                 command.Parameters.Add("@Username", System.Data.SqlDbType.VarChar, 50).Value = user.Username;
26                 command.Parameters.Add("@Password", System.Data.SqlDbType.VarChar, 50).Value = user.Password;
27
28
29
30                 try
31                 {
32                     connection.Open();
33                     SqlDataReader reader = command.ExecuteReader();
34                     if (reader.HasRows)
35                     {
36                         success = true;
37                         while (reader.Read())
38                         {
39                             user.UserName = reader.GetString(1);
40                             user.Username = reader.GetString(2);
41                             user.Password = reader.GetString(3);
42                             user.Role = reader.GetString(4);
43                             user.Department = reader.GetString(5);
44                             user.Company = reader.GetString(6);
45
46
47                         }
48                     }
49                     else
50                     {
51                         success = false;
52                     }
53                     reader.Close();
54                 }
55                 catch (Exception e)
56                 {
57                     Console.WriteLine(e.Message);
58                 }
59             }
60
61             return success;
62         }
63     }
64 }
65
```

LoginController.cs

```
1 using Project.Models;
2 using Project.Services.Business;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Web;
7 using System.Web.Mvc;
8 using System.Web.Security;
9
10 namespace Project.Controllers
11 {
12     1 reference
13     public class LoginController : Controller
14     {
15         // GET: Login
16         0 references
17         public ActionResult Index()
18         {
19             return View("Login");
20         }
21
22         0 references
23         public ActionResult Login(UserModel user)
24         {
25             // return "Results: Username = " + userModel.Username + "Password= " + userModel.Password;
26             SecurityService securityService = new SecurityService();
27             Boolean success = securityService.Authenticate(user);
28
29             if (success)
30             {
31                 Session["user"] = user;
32                 Session["userRole"] = user.Role;
33                 Session["username"] = user.Username;
34                 Session["usersname"] = user.UsersName;
35                 //Session["user"] = userModel.LoggedInUser;
36                 return RedirectToAction("Index", "Dashboard", user);
37                 // return View("Dashboard", user);
38             }
39             else
40             {
41                 Session.Clear();
42                 return View("LoginFailure");
43             }
44         }
45
46         0 references
47         public ActionResult Logout()
48         {
49             Session.Clear();
50             Session["user"] = null;
51             Session["userRole"] = null;
52             return Redirect("~/Home/Index");
53             // return RedirectToAction("Home", "Index", null);
54         }
55
56         [CustomAuthorization]
57         1 reference
58         public ActionResult onPrivateURL()
59         {
60             return Content("Private information here only logged in user can see");
61         }
62     }
63 }
```

UserModel.cs

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

Login Views

Login.cshtml

```
1  @model Project.Models.UserModel
2
3  @{
4      ViewBag.Title = "";
5      Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <h2> </h2>
9
10
11  @using (Html.BeginForm("Login", "Login", FormMethod.Post))
12  {
13      @Html.AntiForgeryToken()
14
15      <div class="form-horizontal">
16          <br />
17          <hr />
18          <h4><b>Login</b></h4>
19          <hr />
20          @Html.ValidationSummary(true, "", new { @class = "text-danger" })
21          <div class="form-group">
22              @Html.LabelFor(model => model.Username, htmlAttributes: new { @class = "control-label col-md-2" })
23              <div class="col-md-10">
24                  @Html.EditorFor(model => model.Username, new { htmlAttributes = new { @class = "form-control" } })
25                  @Html.ValidationMessageFor(model => model.Username, "", new { @class = "text-danger" })
26              </div>
27          </div>
28
29          <div class="form-group">
30              @Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })
31              <div class="col-md-10">
32                  @Html.PasswordFor(model => model.Password, new { htmlAttributes = new { @class = "form-control" } })
33                  @Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })
34              </div>
35          </div>
36
37          <div class="form-group">
38              <div class="col-md-offset-2 col-md-10">
39                  <input type="submit" value="Login" class="btn btn-default" />
40              </div>
41          </div>
42      </div>
43  }
44
45
46
47  @section Scripts {
48      @Scripts.Render("~/bundles/jqueryval")
49  }
50
```

LoginFailure.cs

```
1  |
2  @{|
3      ViewBag.Title = "";
4      Layout = "~/Views/Shared/_Layout.cshtml";
5  }
6
7  <h2>LoginFailure</h2>
8
9
```


LoginSuccess.cshtml

```
1  @model Project.Models.UserModel
2
3  @{
4  ViewBag.Title = "";
5  Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <h2>LoginSuccess</h2>
9
10 <div>
11     <h4>UserModel</h4>
12     <hr />
13     <div>
14
15         @Session["user"] = UserModel.LoggedInUser;
16         @Session["userRole"]
17
18     </div>
19     <dl class="dl-horizontal">
20         <dt>
21             @Html.DisplayNameFor(model => model.Username)
22         </dt>
23
24         <dd>
25             @Html.DisplayFor(model => model.Username)
26         </dd>
27
28         <dt>
29             @Html.DisplayNameFor(model => model.Password)
30         </dt>
31
32         <dd>
33             @Html.DisplayFor(model => model.Password)
34         </dd>
35         <dt>
36             @Html.DisplayNameFor(model => model.UserID)
37         </dt>
38
39         <dd>
40             @Html.DisplayFor(model => model.UserID)
41         </dd>
42
43     </dl>
44 </div>
45
46 <p>
47     @Html.ActionLink("Edit", "Edit", new { /* id = Model.PrimaryKey */ }) |
48     @Html.ActionLink("Back to List", "Index")
49 </p>
50
```

DashboardController.cs

```
1 using Project.Data;
2 using Project.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Data.SqlClient;
6 using System.Dynamic;
7 using System.Linq;
8 using System.Web;
9 using System.Web.Mvc;
10
11 namespace Project.Controllers
12 {
13     0 references
14     public class DashboardController : Controller
15     {
16         string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
17             Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
18
19         [CustomAuthorization]
20         // GET: Dashboard
21         0 references
22         public ActionResult Index()
23         {
24             dynamic model = new ExpandoObject();
25             model.OpenTestModel = GetOpenTests();
26             model.ClosedTestModel = GetClosedTests();
27             model.OpenVulnerabilityModel = GetOpenVulnerability();
28             model.ClosedVulnerabilityModel = GetClosedVulnerability();
29             model.HighSeverityModel = GetHighSeverity();
30             model.MediumSeverityModel = GetMediumSeverity();
31             model.LowSeverityModel = GetLowSeverity();
32             return View(model);
33         }
34
35         [CustomAuthorization]
36         1 reference
37         private static List<OpenTestModel> GetOpenTests()
38         {
39             string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
40                 Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
41             List<OpenTestModel> openTests = new List<OpenTestModel>();
42             string sqlQuery = "SELECT * FROM dbo.TestAssignment WHERE TicketStatus = 'Open'";
43             using (SqlConnection connection = new SqlConnection(connectionString))
44             {
45                 SqlCommand command = new SqlCommand(sqlQuery, connection);
46                 {
47                     connection.Open();
48                     using (SqlDataReader reader = command.ExecuteReader())
49                     {
50                         while (reader.Read())
51                         {
52                             openTests.Add(new OpenTestModel
53                             {
54                                 TestAssignmentID = (int)reader["TestAssignmentID"],
55                                 TestName = reader["TestName"].ToString(),
56                                 TicketUser = reader["TicketUser"].ToString(),
57                                 DateAssigned = (DateTime)reader["DateAssigned"],
58                                 DateClosed = (DateTime)reader["DateClosed"],
59                                 TicketStatus = reader["TicketStatus"].ToString(),
60                                 TestTicketNotes = reader["TestTicketNotes"].ToString(),
61                             });
62                         }
63                     }
64                 }
65                 connection.Close();
66                 return openTests;
67             }
68         }
69     }
70 }
```

```

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

```

```

129 [CustomAuthorization]
130 1 reference
131 private static List<ClosedVulnerabilityModel> GetClosedVulnerability()
132 {
133     string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
134     Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
135
136     List<ClosedVulnerabilityModel> closedVulnerabilities = new List<ClosedVulnerabilityModel>();
137     string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketStatus = 'Closed'";
138     using (SqlConnection connection = new SqlConnection(connectionString))
139     {
140         SqlCommand command = new SqlCommand(sqlQuery, connection);
141         {
142             connection.Open();
143             using (SqlDataReader reader = command.ExecuteReader())
144             {
145                 while (reader.Read())
146                 {
147                     closedVulnerabilities.Add(new ClosedVulnerabilityModel
148                     {
149                         VulnerabilityAssignmentID = (int)reader["VulnerabilityAssignmentID"],
150                         VulnerabilityID = (int)reader["VulnerabilityID"],
151                         TicketUser = reader["TicketUser"].ToString(),
152                         DateAssigned = (DateTime)reader["DateAssigned"],
153                         DateClosed = (DateTime)reader["DateClosed"],
154                         TicketStatus = reader["TicketStatus"].ToString(),
155                         VulnerabilityTicketNotes = reader["VulnerabilityTicketNotes"].ToString(),
156                     });
157                 }
158             }
159             connection.Close();
160             return closedVulnerabilities;
161         }
162     }
163 }
164 [CustomAuthorization]
165 1 reference
166 private static List<HighSeverityModel> GetHighSeverity()
167 {
168     string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
169     Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
170
171     List<HighSeverityModel> highSeverities = new List<HighSeverityModel>();
172     string sqlQuery = "SELECT * FROM dbo.Vulnerability WHERE Severity = 'High Risk'";
173     using (SqlConnection connection = new SqlConnection(connectionString))
174     {
175         SqlCommand command = new SqlCommand(sqlQuery, connection);
176         {
177             connection.Open();
178             using (SqlDataReader reader = command.ExecuteReader())
179             {
180                 while (reader.Read())
181                 {
182                     highSeverities.Add(new HighSeverityModel
183                     {
184                         VulnerabilityID = (int)reader["VulnerabilityID"],
185                         TestName = reader["TestName"].ToString(),
186                         DateOpen = (DateTime)reader["DateOpen"],
187                         DateClosed = (DateTime)reader["DateClosed"],
188                         Severity = reader["Severity"].ToString(),
189                         DREADTotal = (decimal)reader["DREADTotal"],
190                         StatusID = reader["StatusID"].ToString(),
191                     });
192                 }
193             }
194             connection.Close();
195             return highSeverities;
196         }
197     }
198 }

```

```

192     }
193     }
194 }
195 [CustomAuthorization]
196 private static List<MediumSeverityModel> GetMediumSeverity()
197 {
198     string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
199     Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
200
201     List<MediumSeverityModel> mediumSeverities = new List<MediumSeverityModel>();
202     string sqlQuery = "SELECT * FROM dbo.Vulnerability WHERE Severity = 'Medium Risk'";
203     using (SqlConnection connection = new SqlConnection(connectionString))
204     {
205         SqlCommand command = new SqlCommand(sqlQuery, connection);
206         {
207             connection.Open();
208             using (SqlDataReader reader = command.ExecuteReader())
209             {
210                 while (reader.Read())
211                 {
212                     mediumSeverities.Add(new MediumSeverityModel
213                     {
214                         VulnerabilityID = (int)reader["VulnerabilityID"],
215                         TestName = reader["TestName"].ToString(),
216                         DateOpen = (DateTime)reader["DateOpen"],
217                         DateClosed = (DateTime)reader["DateClosed"],
218                         Severity = reader["Severity"].ToString(),
219                         DREADTotal = (decimal)reader["DREADTotal"],
220                         StatusID = reader["StatusID"].ToString(),
221                     });
222                 }
223             }
224             connection.Close();
225             return mediumSeverities;
226         }
227     }
228 }
229 [CustomAuthorization]
230 private static List<LowSeverityModel> GetLowSeverity()
231 {
232     string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
233     Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
234
235     List<LowSeverityModel> lowSeverities = new List<LowSeverityModel>();
236     string sqlQuery = "SELECT * FROM dbo.Vulnerability WHERE Severity = 'Low Risk'";
237     using (SqlConnection connection = new SqlConnection(connectionString))
238     {
239         SqlCommand command = new SqlCommand(sqlQuery, connection);
240         {
241             connection.Open();
242             using (SqlDataReader reader = command.ExecuteReader())
243             {
244                 while (reader.Read())
245                 {
246                     lowSeverities.Add(new LowSeverityModel
247                     {
248                         VulnerabilityID = (int)reader["VulnerabilityID"],
249                         TestName = reader["TestName"].ToString(),
250                         DateOpen = (DateTime)reader["DateOpen"],
251                         DateClosed = (DateTime)reader["DateClosed"],
252                         Severity = reader["Severity"].ToString(),
253                         DREADTotal = (decimal)reader["DREADTotal"],
254                         StatusID = reader["StatusID"].ToString(),
255                     });
256                 }
257             }
258             connection.Close();
259             return lowSeverities;
260         }
261     }
262 }

```

ClosedTestModel.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 {
9     4 references
10    public class ClosedTestModel
11    {
12        [Display(Name = "Test Assignment ID")]
13        1 reference
14        public int TestAssignmentID { get; set; }
15        [Display(Name = "Test Name")]
16        1 reference
17        public string TestName { get; set; }
18        [Display(Name = "Assignee")]
19        1 reference
20        public string TicketUser { get; set; }
21        [Display(Name = "Date Assigned")]
22        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
23        1 reference
24        public DateTime DateAssigned { get; set; }
25        [Display(Name = "Date Closed")]
26        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
27        1 reference
28        public DateTime DateClosed { get; set; }
29        [Display(Name = "Ticket Status")]
30        1 reference
31        public string TicketStatus { get; set; }
32        [Display(Name = "Ticket Notes")]
33        1 reference
34        public String TestTicketNotes { get; set; }
35    }
36 }
```

ClosedVulnerabilityModel.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 {
9     4 references
10    public class ClosedVulnerabilityModel
11    {
12        [Display(Name = "Vulnerability Assignment ID")]
13        1 reference
14        public int VulnerabilityAssignmentID { get; set; }
15        [Display(Name = "Vulnerability ID")]
16        1 reference
17        public int VulnerabilityID { get; set; }
18        [Display(Name = "Assignee")]
19        1 reference
20        public string TicketUser { get; set; }
21        [Display(Name = "Date Assigned")]
22        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
23        1 reference
24        public DateTime DateAssigned { get; set; }
25        [Display(Name = "Date Closed")]
26        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
27        1 reference
28        public DateTime DateClosed { get; set; }
29        [Display(Name = "Ticket Status")]
30        1 reference
31        public string TicketStatus { get; set; }
32        [Display(Name = "Ticket Notes")]
33        1 reference
34        public string VulnerabilityTicketNotes { get; set; }
35    }
36 }
```

HighSeverityModel.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 {
9     4 references
10    public class HighSeverityModel
11    {
12        1 reference
13        [Display(Name = "Vulnerability ID")]
14        public int VulnerabilityID { get; set; }
15        1 reference
16        [Display(Name = "Test Name")]
17        public string TestName { get; set; }
18        1 reference
19        [Display(Name = "Date Open")]
20        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
21        public DateTime DateOpen { get; set; }
22        1 reference
23        [Display(Name = "Date Closed")]
24        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
25        public DateTime DateClosed { get; set; }
26        1 reference
27        [Display(Name = "Severity")]
28        public string Severity { get; set; }
29        1 reference
30        [Display(Name = "DREAD Score Rating")]
31        public decimal DREADTotal { get; set; }
32        1 reference
33        [Display(Name = "Status")]
34        public string StatusID { get; set; }
35    }
36 }
```


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 {
9     4 references
10    public class LowSeverityModel
11    {
12        [Display(Name = "Vulnerability ID")]
13        1 reference
14        public int VulnerabilityID { get; set; }
15        [Display(Name = "Test Name")]
16        1 reference
17        public string TestName { get; set; }
18        [Display(Name = "Date Open")]
19        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
20        1 reference
21        public DateTime DateOpen { get; set; }
22        [Display(Name = "Date Closed")]
23        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
24        1 reference
25        public DateTime DateClosed { get; set; }
26        [Display(Name = "Severity")]
27        1 reference
28        public string Severity { get; set; }
29        [Display(Name = "DREAD Score Rating")]
30        1 reference
31        public decimal DREADTotal { get; set; }
32        [Display(Name = "Status")]
33        1 reference
34        public string StatusID { get; set; }
35    }
36 }
```

MediumSeverityModel.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 {
9     4 references
10    public class MediumSeverityModel
11    {
12        [Display(Name = "Vulnerability ID")]
13        1 reference
14        public int vulnerabilityID { get; set; }
15        [Display(Name = "Test Name")]
16        1 reference
17        public string TestName { get; set; }
18        [Display(Name = "Date Open")]
19        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
20        1 reference
21        public DateTime DateOpen { get; set; }
22        [Display(Name = "Daate Closed")]
23        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
24        1 reference
25        public DateTime DateClosed { get; set; }
26        [Display(Name = "Severity")]
27        1 reference
28        public string Severity { get; set; }
29        [Display(Name = "DREAD Score Rating")]
30        1 reference
31        public decimal DREADTotal { get; set; }
32        [Display(Name = "Status")]
33        1 reference
34        public string StatusID { get; set; }
35    }
36 }
```

OpenTestModel.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 {
9     public class OpenTestModel
10    {
11        [Display(Name = "Test Assignment ID")]
12        public int TestAssignmentID { get; set; }
13        [Display(Name = "Test Name")]
14        public string TestName { get; set; }
15        [Display(Name = "Ticket User")]
16        public string TicketUser { get; set; }
17        [Display(Name = "Date Assigned")]
18        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
19        public DateTime DateAssigned { get; set; }
20        [Display(Name = "Date Closed")]
21        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
22        public DateTime DateClosed { get; set; }
23        [Display(Name = "Ticket Status")]
24        public string TicketStatus { get; set; }
25        [Display(Name = "Ticket Notes")]
26        public String TestTicketNotes { get; set; }
27    }
28 }
29 }
```

OpenVulnerabilityModel.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 {
9     4 references
10    public class OpenVulnerabilityModel
11    {
12        [Display(Name = "Vulnerability Assignment ID")]
13        1 reference
14        public int VulnerabilityAssignmentID { get; set; }
15        [Display(Name = "Vulnerability ID")]
16        1 reference
17        public int VulnerabilityID { get; set; }
18        [Display(Name = "Assignee")]
19        1 reference
20        public string TicketUser { get; set; }
21        [Display(Name = "Date Assigned")]
22        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
23        1 reference
24        public DateTime DateAssigned { get; set; }
25        [Display(Name = "Date Closed")]
26        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
27        1 reference
28        public DateTime DateClosed { get; set; }
29        [Display(Name = "Ticket Status")]
30        1 reference
31        public string TicketStatus { get; set; }
32        [Display(Name = "Ticket Notes")]
33        1 reference
34        public string VulnerabilityTicketNotes { get; set; }
35    }
36 }
```

Dashboard View

Index.cshtml

```
1  @using Project.Models
2  @model dynamic
3  @{
4      Layout = null;
5  }
6  @{
7      ViewBag.Title = "";
8      Layout = "~/Views/Shared/_Layout.cshtml";
9  }
10
11 <h2> </h2>
12 <!DOCTYPE html>
13
14 <html>
15 <head>
16     <meta name="viewport" content="width=device-width" />
17     <title>Index</title>
18 </head>
19 <body>
20     <br/>
21     <hr/>
22     <h2>Have a great day @Session["username"]!</h2>
23     <hr />
24     <h3>Open Tests</h3>
25     <table class="table">
26         <tr>
27             <th>TestAssignmentID</th>
28             <th>TestName</th>
29             <th>TicketUser</th>
30             <th>DateAssigned</th>
31             <th>DateClosed</th>
32             <th>TicketStatus</th>
33             <th>TestTicketNotes</th>
34             <th></th>
35         </tr>
36         @foreach (OpenTestModel openTests in Model.OpenTestModel)
37         {
38             <tr>
39                 <td>@openTests.TestAssignmentID</td>
40                 <td>@openTests.TestName</td>
41                 <td>@openTests.TicketUser</td>
42                 <td>@openTests.DateAssigned</td>
43                 <td>@openTests.DateClosed</td>
44                 <td>@openTests.TicketStatus</td>
45                 <td>@openTests.TestTicketNotes</td>
46             </tr>
47         }
48     </table>
49     <hr />
50     <h3>Closed Tests</h3>
51     <table class="table">
52         <tr>
53             <th>TestAssignmentID</th>
54             <th>TestName</th>
55             <th>TicketUser</th>
56             <th>DateAssigned</th>
57             <th>DateClosed</th>
58             <th>TicketStatus</th>
59             <th>TestTicketNotes</th>
60             <th></th>
61         </tr>
62         @foreach (ClosedTestModel closedTests in Model.ClosedTestModel)
63         {
64             <tr>
65                 <td>@closedTests.TestAssignmentID</td>
66                 <td>@closedTests.TestName</td>
67                 <td>@closedTests.TicketUser</td>
68                 <td>@closedTests.DateAssigned</td>
69                 <td>@closedTests.DateClosed</td>
70                 <td>@closedTests.TicketStatus</td>
71                 <td>@closedTests.TestTicketNotes</td>
72             </tr>
73         }
74     </table>
75     <hr />
```

```

76 <h3>Open Vulnerabilities</h3>
77 <table class="table">
78   <tr>
79     <th>VulnerabilityAssignmentID</th>
80     <th>VulnerabilityID</th>
81     <th>TicketUser</th>
82     <th>DateAssigned</th>
83     <th>DateClosed</th>
84     <th>TicketStatus</th>
85     <th>VulnerabilityTicketNotes</th>
86   </th></tr>
87 </tr>
88 @foreach (OpenVulnerabilityModel openVulnerabilities in Model.OpenVulnerabilityModel)
89 {
90   <tr>
91     <td>@openVulnerabilities.VulnerabilityAssignmentID</td>
92     <td>@openVulnerabilities.VulnerabilityID</td>
93     <td>@openVulnerabilities.TicketUser</td>
94     <td>@openVulnerabilities.DateAssigned</td>
95     <td>@openVulnerabilities.DateClosed</td>
96     <td>@openVulnerabilities.TicketStatus</td>
97     <td>@openVulnerabilities.VulnerabilityTicketNotes</td>
98   </tr>
99 }
100 </table>
101 <hr />
102
103 <h3>Closed Vulnerabilities</h3>
104 <table class="table">
105   <tr>
106     <th>VulnerabilityAssignmentID</th>
107     <th>VulnerabilityID</th>
108     <th>TicketUser</th>
109     <th>DateAssigned</th>
110     <th>DateClosed</th>
111     <th>TicketStatus</th>
112     <th>VulnerabilityTicketNotes</th>
113   </th></tr>
114 </tr>
115 @foreach (ClosedVulnerabilityModel closedVulnerabilities in Model.ClosedVulnerabilityModel)
116 {
117   <tr>
118     <td>@closedVulnerabilities.VulnerabilityAssignmentID</td>
119     <td>@closedVulnerabilities.VulnerabilityID</td>
120     <td>@closedVulnerabilities.TicketUser</td>
121     <td>@closedVulnerabilities.DateAssigned</td>
122     <td>@closedVulnerabilities.DateClosed</td>
123     <td>@closedVulnerabilities.TicketStatus</td>
124     <td>@closedVulnerabilities.VulnerabilityTicketNotes</td>
125   </tr>
126 }
127 </table>
128 <hr />
129
130 <h3>High Severity Vulnerabilities</h3>
131 <table class="table">
132   <tr>
133     <th>VulnerabilityID</th>
134     <th>TestName</th>
135     <th>DateOpen</th>
136     <th>DateClosed</th>
137     <th>Severity</th>
138     <th>DREADTotal</th>
139     <th>StatusID</th>
140   </th></tr>
141 </tr>
142 @foreach (HighSeverityModel highSeverities in Model.HighSeverityModel)
143 {
144   <tr>
145     <td>@highSeverities.VulnerabilityID</td>
146     <td>@highSeverities.TestName</td>
147     <td>@highSeverities.DateOpen</td>
148     <td>@highSeverities.DateClosed</td>
149     <td>@highSeverities.Severity</td>
150     <td>@highSeverities.DREADTotal</td>

```

```

151         <td>@highSeverities.StatusID</td>
152     </tr>
153     }
154 </table>
155 <hr />
156 <h3>Medium Severity Vulnerabilities</h3>
157 <table class="table">
158     <tr>
159         <th>VulnerabilityID</th>
160         <th>TestName</th>
161         <th>DateOpen</th>
162         <th>DateClosed</th>
163         <th>Severity</th>
164         <th>DREADTotal</th>
165         <th>StatusID</th>
166         <th></th>
167     </tr>
168     @foreach (MediumSeverityModel mediumSeverities in Model.MediumSeverityModel)
169     {
170         <tr>
171             <td>@mediumSeverities.VulnerabilityID</td>
172             <td>@mediumSeverities.TestName</td>
173             <td>@mediumSeverities.DateOpen</td>
174             <td>@mediumSeverities.DateClosed</td>
175             <td>@mediumSeverities.Severity</td>
176             <td>@mediumSeverities.DREADTotal</td>
177             <td>@mediumSeverities.StatusID</td>
178         </tr>
179     }
180 </table>
181 <hr />
182
183 <h3>Low Severity Vulnerabilities</h3>
184 <table class="table">
185     <tr>
186         <th>VulnerabilityID</th>
187         <th>TestName</th>
188         <th>DateOpen</th>
189         <th>DateClosed</th>
190         <th>Severity</th>
191         <th>DREADTotal</th>
192         <th>StatusID</th>
193         <th></th>
194     </tr>
195     @foreach (LowSeverityModel lowSeverities in Model.LowSeverityModel)
196     {
197         <tr>
198             <td>@lowSeverities.VulnerabilityID</td>
199             <td>@lowSeverities.TestName</td>
200             <td>@lowSeverities.DateOpen</td>
201             <td>@lowSeverities.DateClosed</td>
202             <td>@lowSeverities.Severity</td>
203             <td>@lowSeverities.DREADTotal</td>
204             <td>@lowSeverities.StatusID</td>
205         </tr>
206     }
207 </table>
208 <hr />
209 </body>
210 </html>
211
212
213
214

```

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  {
9      0 references
10     public class HomeController : Controller
11     {
12         0 references
13         public ActionResult Index()
14         {
15             if(Session["user"] != null) {
16                 return RedirectToAction("Index", "Dashboard");
17             }
18             return View();
19         }
20         0 references
21         public ActionResult About()
22         {
23             ViewBag.Message = "Your application description page.";
24             return View();
25         }
26         0 references
27         public ActionResult Contact()
28         {
29             ViewBag.Message = "Your contact page.";
30             return View();
31         }
32     }
33 }
34
```


Home View

Index.cshtml

```
1  @{
2      ViewBag.Title = "Home Page";
3  }
4
5  <div class="jumbotron">
6      <h1>Vulnerability Management</h1>
7      <p class="lead">Putting the ease in Vulnerabilities.</a></p>
8  </div>
9
10 <div class="row">
11     <div class="col-md-4">
12         <h2>Getting started</h2>
13         <p>Login to access your daily tasks</p>
14     </div>
15     <div class="col-md-4">
16         <h2>Assistance</h2>
17         <p>Should you require assistance please contact us at Assistance@VulnerabilityManagement.com</p>
18     </div>
19     <div class="col-md-4">
20         <h2>Vulnerability Management Solutions</h2>
21         <p>Asset Management</p>
22         <p>Test Management</p>
23         <p>Vulnerability Management</p>
24         <p>Track A Vulnerability From Discovery and Assignment Through to Mitigation and Close </p>
25     </div>
26 </div>
27 </div>
```

TestAssignmentController.cs

```
1 using Project.Controllers.Data;
2 using Project.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Data.SqlClient;
6 using System.Linq;
7 using System.Web;
8 using System.Web.Mvc;
9
10 namespace Project.Controllers
11 {
12     [D.references]
13     public class TestAssignmentController : Controller
14     {
15         private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
16             Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
17         [CustomAuthorization]
18         public ActionResult Index()
19         {
20             [D.references]
21             List<TestAssignmentModel> testAssignments = new List<TestAssignmentModel>();
22             TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
23             testAssignments = testAssignmentDAO.FetchAll();
24
25             return View("Index", testAssignments);
26         }
27         [CustomAuthorization]
28         public ActionResult Details(int id)
29         {
30             [D.references]
31             TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
32             TestAssignmentModel testAssignment = testAssignmentDAO.FetchOne(id);
33             return View("Details", testAssignment);
34         }
35         [CustomAuthorization]
36         public ActionResult Create()
37         {
38             [D.references]
39             TestAssignmentModel testAssignment = new TestAssignmentModel();
40
41             //new
42             testAssignment.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
43             testAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role = 'PenetrationTester' ORDER BY Username ", "Username", "Username");
44             testAssignment.ListTestStatus = PopulateDropDown(" SELECT StatusType FROM dbo.TestStatus", "StatusType", "StatusType");
45             return View("CreateTestAssignmentForm", testAssignment);
46         }
47         [CustomAuthorization]
48         private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
49         {
50             [D.references]
51             List<SelectListItem> items = new List<SelectListItem>();
52             using (SqlConnection connection = new SqlConnection(connectionString))
53             {
54                 using (SqlCommand cmd = new SqlCommand(query))
55                 {
56                     cmd.Connection = connection;
57                     connection.Open();
58                     using (SqlDataReader sdr = cmd.ExecuteReader())
59                     {
60                         while (sdr.Read())
61                         {
62                             items.Add(new SelectListItem
63                             {
64                                 Text = sdr[textColumn].ToString(),
65                                 Value = sdr[valueColumn].ToString()
66                             });
67                         }
68                     }
69                     connection.Close();
70                 }
71             }
72             return items;
73         }
74     }
75 }
```

```

69 [CustomAuthorization]
70 @references
71 public ActionResult CreateTestAssignment(TestAssignmentModel testAssignmentModel)
72 {
73     //save it to the database
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]
82 @references
83 public ActionResult Edit(int id)
84 {
85     //new
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]
95 @references
96 public ActionResult UpdateTestAssignment(TestAssignmentModel testAssignmentModel)
97 {
98     //save it to the database
99     TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
100     testAssignmentDAO.Update(testAssignmentModel);
101     return View("Details", testAssignmentModel);
102 }
103 [CustomAuthorization]
104 @references
105 public ActionResult Delete(int id)
106 {
107     TestAssignmentDAO testAssignmentDAO = new TestAssignmentDAO();
108     testAssignmentDAO.Delete(id);
109     List<TestAssignmentModel> testAssignments = testAssignmentDAO.FetchAll();
110     return View("Index", testAssignments);
111 }
112 }

```

TestAssignmentDAO.cs

```
1 using Project.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Data.SqlClient;
5
6 namespace Project.Controllers.Data
7 {
8     14 references
9     internal class TestAssignmentDAO
10    {
11        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
12        Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
13
14        4 references
15        public List<TestAssignmentModel> FetchAll()
16        {
17            List<TestAssignmentModel> returnList = new List<TestAssignmentModel>();
18            using (SqlConnection connection = new SqlConnection(connectionString))
19            {
20                string sqlQuery = "SELECT * FROM dbo.TestAssignment";
21                SqlCommand command = new SqlCommand(sqlQuery, connection);
22
23                connection.Open();
24                SqlDataReader reader = command.ExecuteReader();
25                if (reader.HasRows)
26                {
27                    while (reader.Read())
28                    {
29                        //creating new TestAssignment object and adding it to the list to be returned
30                        TestAssignmentModel testAssignment = new TestAssignmentModel();
31                        testAssignment.TestAssignmentID = reader.GetInt32(0);
32                        testAssignment.TestName = reader.GetString(1);
33                        testAssignment.TicketUser = reader.GetString(2);
34                        testAssignment.DateAssigned = reader.GetDateTime(3);
35                        testAssignment.DateClosed = reader.GetDateTime(4);
36                        testAssignment.TicketStatus = reader.GetString(5);
37                        testAssignment.TestTicketNotes = reader.GetString(6);
38                        returnList.Add(testAssignment);
39                    }
40                }
41            }
42            return returnList;
43        }
44
45        2 references
46        public TestAssignmentModel FetchOne(int id)
47        {
48            //access database
49            using (SqlConnection connection = new SqlConnection(connectionString))
50            {
51                string sqlQuery = "SELECT * FROM dbo.TestAssignment WHERE TestAssignmentID = @id";
52                SqlCommand command = new SqlCommand(sqlQuery, connection);
53                command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
54                connection.Open();
55                SqlDataReader reader = command.ExecuteReader();
56
57                TestAssignmentModel testAssignment = new TestAssignmentModel();
58                if (reader.HasRows)
59                {
60                    while (reader.Read())
61                    {
62                        //create new test object and add that to the list to be returned
63
64                        testAssignment.TestAssignmentID = reader.GetInt32(0);
65                        testAssignment.TestName = reader.GetString(1);
66                        testAssignment.TicketUser = reader.GetString(2);
67                        testAssignment.DateAssigned = reader.GetDateTime(3);
68                        testAssignment.DateClosed = reader.GetDateTime(4);
69                        testAssignment.TicketStatus = reader.GetString(5);
70                        testAssignment.TestTicketNotes = reader.GetString(6);
71                    }
72                }
73            }
74        }
75    }
76 }
```

```

73     }
74     }
75     }
76     return testAssignment;
77     }
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.TestAssignment WHERE TestAssignmentID = @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             connection.Open();
97             int deletedID = command.ExecuteNonQuery();
98
99             return deletedID;
100        }
101    }
102
103     1 reference
104     public int Create(TestAssignmentModel testAssignmentModel)
105     {
106         //access database
107         using (SqlConnection connection = new SqlConnection(connectionString))
108         {
109             string sqlQuery = "INSERT INTO dbo.TestAssignment Values(@TestName, @TicketUser, @DateAssigned, @DateClosed, @TicketStatus, @TestTicketNotes)";
110
111             SqlCommand command = new SqlCommand(sqlQuery, connection);
112             command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
113             command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketUser;
114             command.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateAssigned;
115             command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateClosed;
116             command.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
117             command.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar).Value = testAssignmentModel.TestTicketNotes;
118
119             //Update Test that ticket is assigned
120             string sqlQuery1 = "Update dbo.Test SET StartDate = @StartDate, EndDate = @EndDate, TestStatus = @TestStatus, TestNotes = @TestNotes WHERE dbo.Test.TestName = @TestName";
121
122             //associate @id with Id parameter
123             SqlCommand command1 = new SqlCommand(sqlQuery1, connection);
124
125             command1.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
126             command1.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateAssigned;
127             command1.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testAssignmentModel.DateClosed;
128             command1.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
129             command1.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar).Value = testAssignmentModel.TestTicketNotes;
130
131             //Insert Assignment To Vulnerability Log
132             string sqlQuery2 = "INSERT INTO dbo.TestLog Values(@TestName, @TicketUser, @CurrentTimeStamp, @TicketStatus, @TestTicketNotes)";
133
134             //associate @id with Id parameter
135             SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
136
137             command2.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestName;
138             command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.CurrentUser;
139             command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
140             command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TicketStatus;
141             command2.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testAssignmentModel.TestTicketNotes;
142

```

```

142
143
144         connection.Open();
145         int newTestAssignmentID = command.ExecuteNonQuery();
146         command1.ExecuteNonQuery();
147         command2.ExecuteNonQuery();
148         return newTestAssignmentID;
149     }
150 }
151
152 1 reference 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 =
160             @TicketStatus, TestTicketNotes = @TestTicketNotes WHERE TestAssignmentID = @TestAssignmentID";
161
162         //associate @id with Id parameter
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

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Web;
6 using System.Web.Mvc;
7
8 namespace Project.Models
9 {
10     23 references
11     public class TestAssignmentModel
12     {
13         4 references
14         [Display(Name = "Test Assignment ID")]
15         public int TestAssignmentID { get; set; }
16         9 references
17         [Display(Name = "Test Name")]
18         public string TestName { get; set; }
19         5 references
20         [Display(Name = "Assignee")]
21         public string TicketUser { get; set; }
22         2 references
23         [Display(Name = "Current User")]
24         public string CurrentUser { get; set; }
25         7 references
26         [Display(Name = "Date Assigned")]
27         public DateTime? DateAssigned { get; set; }
28         7 references
29         [Display(Name = "Date Closed")]
30         public DateTime? DateClosed { get; set; }
31         9 references
32         [Display(Name = "Ticket Status")]
33         public string TicketStatus { get; set; }
34         9 references
35         [Display(Name = "Ticket Notes")]
36         public string TestTicketNotes { get; set; }
37         2 references
38         public List<SelectListItem> ListTestName { get; set; }
39         3 references
40         public List<SelectListItem> ListTicketUser { get; set; }
41         3 references
42         public List<SelectListItem> ListTestStatus { get; set; }
43
44         0 references
45         public TestAssignmentModel(int testAssignmentID, string testName, string ticketUser, DateTime dateAssigned, DateTime dateClosed, string ticketStatus, string testTicketNotes)
46         {
47             TestAssignmentID = testAssignmentID;
48             TestName = testName;
49             TicketUser = ticketUser;
50             DateAssigned = dateAssigned;
51             DateClosed = dateClosed;
52             TicketStatus = ticketStatus;
53             TestTicketNotes = testTicketNotes;
54         }
55
56         3 references
57         public TestAssignmentModel()
58         {
59             this.ListTestName = new List<SelectListItem>();
60             this.ListTicketUser = new List<SelectListItem>();
61             this.ListTestStatus = new List<SelectListItem>();
62         }
63     }
64 }
```

TestAssignment Views

CreateTestAssignmentForm.cshtml

```
1 @model Project.Models.TestAssignmentModel
2
3 @if
4 {
5     ViewBag.Title = "";
6     Layout = "~/Views/Shared/_Layout.cshtml";
7 }
8 @if
9 {
10     String Role;
11     Role = Session["userRole"].ToString();
12     if (Role == "Administrator")
13     {
14     }
15     else if (Role == "PenetrationTester")
16     {
17     }
18     else
19     {
20     }
21     Response.Redirect("~/Home/Index");
22 }
23
24 <h2> </h2>
25
26
27
28 @using (Html.BeginForm("CreateTestAssignment", "TestAssignment"))
29 {
30     @Html.AntiForgeryToken()
31
32     <div class="form-horizontal">
33         <br/>
34         <br/>
35         <h4><b>Assign Test</b></h4>
36         <br/>
37         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
38
39         <div class="form-group">
40             @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
41             <div class="col-md-10">
42                 @Html.DropDownListFor(m => m.TestName, Model.ListTestName, "Please select")
43             </div>
44         </div>
45
46         <div class="form-group">
47             @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
48             <div class="col-md-10">
49                 @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
50             </div>
51         </div>
52
53         <div hidden class="form-group">
54             @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
55             <div hidden class="col-md-10">
56                 @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = session["username"] } })
57             </div>
58         </div>
59
60         <div class="form-group">
61             @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
62             <div class="col-md-10">
63                 @Html.TextBoxFor(model => model.DateAssigned, new { type = "date" })
64             </div>
65         </div>
66
67         <div class="form-group">
68             @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
69             <div class="col-md-10">
70                 @Html.TextBoxFor(model => model.DateClosed, new { type = "date" })
71             </div>
72         </div>
73
74         <div class="form-group">
75             @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
76             <div class="col-md-10">
77                 @Html.DropDownListFor(m => m.TicketStatus, Model.ListTestStatus, "Please select")
78             </div>
79         </div>
80
81         <div class="form-group">
82             @Html.LabelFor(model => model.TestTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
83             <div class="col-md-10">
84                 @Html.EditorFor(model => model.TestTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
85                 @Html.ValidationMessageFor(model => model.TestTicketNotes, "", new { @class = "text-danger" })
86             </div>
87         </div>
88
89         <div class="form-group">
90             <div class="col-md-offset-2 col-md-10">
91                 <input type="submit" value="create" class="btn btn-default" />
92             </div>
93         </div>
94     </div>
95
96     <div>
97         @Html.ActionLink("Back to List", "Index")
98     </div>
99
100 </div>
```


Details.cshtml

```
1 @model Project.Models.TestAssignmentModel
2
3
4 ViewBag.Title = "";
5 Layout = "~/Views/Shared/_Layout.cshtml";
6
7
8 @if
9 {
10     String Role;
11     Role = Session["userRole"].ToString();
12     if (Role == "Administrator")
13     {
14     }
15     else if (Role == "PenetrationTester")
16     {
17     }
18     }
19     else
20     {
21     Response.Redirect("~/Home/Index");
22     }
23 }
24 <h2> </h2>
25
26 <div>
27     <br />
28     <hr />
29     <h4><b>Test Assignment Details</b></h4>
30     <hr />
31     <dl class="dl-horizontal">
32         <dt>
33             @Html.DisplayNameFor(model => model.TestAssignmentID)
34         </dt>
35         <dd>
36             @Html.DisplayFor(model => model.TestAssignmentID)
37         </dd>
38     <dt>
39         @Html.DisplayNameFor(model => model.TestName)
40     </dt>
41     <dd>
42         @Html.DisplayFor(model => model.TestName)
43     </dd>
44     <dt>
45         @Html.DisplayNameFor(model => model.TicketUser)
46     </dt>
47     <dd>
48         @Html.DisplayFor(model => model.TicketUser)
49     </dd>
50     <dt>
51         @Html.DisplayNameFor(model => model.DateAssigned)
52     </dt>
53     <dd>
54         @Html.DisplayFor(model => model.DateAssigned)
55     </dd>
56     <dt>
57         @Html.DisplayNameFor(model => model.DateClosed)
58     </dt>
59     <dd>
60         @Html.DisplayFor(model => model.DateClosed)
61     </dd>
62     <dt>
63         @Html.DisplayNameFor(model => model.TicketStatus)
64     </dt>
65     <dd>
66         @Html.DisplayFor(model => model.TicketStatus)
67     </dd>
68 </dl>
69 </div>
```

```
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

```
1 @model IEnumerable<Project.Models.TestAssignmentModel>
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2> </h2>
9 @{
10     String Role;
11     Role = Session["userRole"].ToString();
12     if (Role == "Administrator")
13     {
14         <br />
15         <hr />
16         <h4><b>Assigned Tests</b></h4>
17         <hr />
18
19         @Html.ActionLink("Create New", "Create")
20     }
21     else if (Role == "PenetrationTester")
22     {
23     }
24     else
25     {
26         Response.Redirect("~/Home/Index");
27     }
28 }
29
30
31 }
32
33
34
35
36 <table class="table">
37 <tr>
38 <th>
39     @Html.DisplayNameFor(model => model.TestAssignmentID)
40 </th>
41 <th>
42     @Html.DisplayNameFor(model => model.TestName)
43 </th>
44 <th>
45     @Html.DisplayNameFor(model => model.TicketUser)
46 </th>
47 <th>
48     @Html.DisplayNameFor(model => model.DateAssigned)
49 </th>
50 <th>
51     @Html.DisplayNameFor(model => model.DateClosed)
52 </th>
53 <th>
54     @Html.DisplayNameFor(model => model.TicketStatus)
55 </th>
56 <th>
57     @Html.DisplayNameFor(model => model.TestTicketNotes)
58 </th>
59 <th></th>
60 </tr>
61
62 @foreach (var item in Model)
63 {
64 <tr>
65 <td>
66     @Html.DisplayFor(modelItem => item.TestAssignmentID)
67 </td>
68 <td>
69     @Html.DisplayFor(modelItem => item.TestName)
70 </td>
71 <td>
72     @Html.DisplayFor(modelItem => item.TicketUser)
73 </td>
74 <td>
75     @Html.DisplayFor(modelItem => item.DateAssigned)
76 </td>
77 <td>
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 </tr>
87 }
```

```

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

```
1 @model Project.Models.TestAssignmentModel
2
3 @f
4 ViewBag.Title = "";
5 Layout = "~/Views/Shared/_Layout.cshtml";
6
7 @f
8 String Role;
9 Role = Session["userRole"].ToString();
10 if (Role == "Administrator")
11 {
12
13
14
15 }
16 else if (Role == "PenetrationTester")
17 {
18
19
20
21 }
22 else
23 {
24     Response.Redirect("~/Home/Index");
25 }
26
27 <h2> </h2>
28
29 @using (Html.BeginForm("UpdateTestAssignment", "TestAssignment"))
30 {
31     @Html.AntiForgeryToken()
32
33     <div class="form-horizontal">
34         <br />
35         <hr />
36         <h4><b>Update Test Assignment</b></h4>
37         <hr />
38         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
39         <div hidden class="form-group">
40             @Html.LabelFor(model => model.TestAssignmentID, htmlAttributes: new { @class = "control-label col-md-2" })
41             <div hidden class="col-md-10">
42                 @Html.EditorFor(model => model.TestAssignmentID, new { htmlAttributes = new { @class = "form-control" } })
43                 @Html.ValidationMessageFor(model => model.TestAssignmentID, "", new { @class = "text-danger" })
44             </div>
45         </div>
46         <div class="form-group">
47             @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
48             <div class="col-md-10">
49                 @Html.DropDownListFor(m => m.TestName, Model.ListTestName, "Please select")
50             </div>
51         </div>
52         <div class="form-group">
53             @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
54             <div class="col-md-10">
55                 @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
56             </div>
57         </div>
58         <div hidden class="form-group">
59             @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
60             <div hidden class="col-md-10">
61                 @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
62             </div>
63         </div>
64         <div class="form-group">
65             @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
66             <div class="col-md-10">
67                 @Html.EditorFor(model => model.DateAssigned, new { htmlAttributes = new { @class = "form-control" } })
68             </div>
69         </div>
70         <div class="form-group">
71             @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
72             <div class="col-md-10">
73                 @Html.EditorFor(model => model.DateClosed, new { htmlAttributes = new { @class = "form-control" } })
74             </div>
75         </div>
76     </div>
77 }
78
```

```

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

```
1 using Project.Data;
2 using Project.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Data.SqlClient;
6 using System.Linq;
7 using System.Web;
8 using System.Web.Mvc;
9
10 namespace Project.Controllers
11 {
12     0 references
13     public class TestsController : Controller
14     {
15         private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
16             Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
17
18         [CustomAuthorization]
19         0 references
20         public ActionResult Index()
21         {
22             List<TestModel> tests = new List<TestModel>();
23             TestDAO testDAO = new TestDAO();
24             tests = testDAO.FetchAll();
25             return View("Index", tests);
26         }
27
28         [CustomAuthorization]
29         0 references
30         public ActionResult Details(int id)
31         {
32             TestDAO testDAO = new TestDAO();
33             TestModel test = testDAO.FetchOne(id);
34             return View("Details", test);
35         }
36
37         [CustomAuthorization]
38         0 references
39         public ActionResult Create()
40         {
41             TestModel test = new TestModel();
42             test.ListAssetName = PopulateDropDown(" SELECT AssetName FROM dbo.Asset", "AssetName", "AssetName");
43             test.ListTestStatus = PopulateDropDown(" SELECT StatusType FROM dbo.TestStatus", "StatusType", "StatusType");
44             return View("CreateTestForm", test);
45         }
46
47         [CustomAuthorization]
48         4 references
49         private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
50         {
51             List<SelectListItem> items = new List<SelectListItem>();
52             using (SqlConnection connection = new SqlConnection(connectionString))
53             {
54                 using (SqlCommand cmd = new SqlCommand(query))
55                 {
56                     cmd.Connection = connection;
57                     connection.Open();
58                     using (SqlDataReader sdr = cmd.ExecuteReader())
59                     {
60                         while (sdr.Read())
61                         {
62                             items.Add(new SelectListItem
63                             {
64                                 Text = sdr[textColumn].ToString(),
65                                 Value = sdr[valueColumn].ToString()
66                             });
67                         }
68                     }
69                 }
70             }
71             connection.Close();
72
73             return items;
74         }
75     }
76 }
```

```

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

```


TestDAO.cshtml

```
1 using Project.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Data.SqlClient;
5
6 namespace Project.Data
7 {
8     14 references
9     internal class TestDAO
10    {
11        //connect to database
12        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
13        Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
14
15        //Performs all operations on database, create, read, read one, update, delete
16
17        2 references
18        public List<TestModel> FetchAll()
19        {
20            List<TestModel> returnList = new List<TestModel>();
21
22            //access database
23            using (SqlConnection connection = new SqlConnection(connectionString))
24            {
25                string sqlQuery = "SELECT * FROM dbo.Test";
26                SqlCommand command = new SqlCommand(sqlQuery, connection);
27                connection.Open();
28                SqlDataReader reader = command.ExecuteReader();
29
30                if (reader.HasRows)
31                {
32                    while (reader.Read())
33                    {
34                        //create new test object and add that to the list to be returned
35                        TestModel test = new TestModel();
36                        test.TestID = reader.GetInt32(0);
37                        test.TestName = reader.GetString(1);
38                        test.AssetName = reader.GetString(2);
39                        test.StartDate = reader.GetDateTime(3);
40                        test.EndDate = reader.GetDateTime(4);
41                        test.SignOff = reader.GetBoolean(5);
42                        test.URL = reader.GetString(6);
43                        test.Description = reader.GetString(7);
44                        test.ContactID = reader.GetString(8);
45                        test.TestStatus = reader.GetString(9);
46                        test.TestNotes = reader.GetString(11);
47                        //adding to list to be returned
48                        returnList.Add(test);
49                    }
50                }
51            }
52
53            return returnList;
54        }
55
56        1 reference
57        internal int Delete(int id)
58        {
59            using (SqlConnection connection = new SqlConnection(connectionString))
60            {
61                //prepared statement
62                //update
63                string sqlQuery = "DELETE FROM dbo.Test WHERE TestID = @id";
64
65                //associate @id with Id parameter
66
67
68                SqlCommand command = new SqlCommand(sqlQuery, connection);
69
70                command.Parameters.Add("@Id", System.Data.SqlDbType.VarChar, 1000).Value = id;
71
72            }
73        }
74    }
75 }
```

```

71
72
73
74         connection.Open();
75         int deletedID = command.ExecuteNonQuery();
76
77         return deletedID;
78     }
79
80     2 references
81     public TestModel FetchOne(int id)
82     {
83
84         //access database
85         using (SqlConnection connection = new SqlConnection(connectionString))
86         {
87             string sqlQuery = "SELECT * FROM dbo.Test WHERE TestID = @id";
88
89             SqlCommand command = new SqlCommand(sqlQuery, connection);
90             command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
91             connection.Open();
92             SqlDataReader reader = command.ExecuteReader();
93
94             TestModel test = new TestModel();
95             if (reader.HasRows)
96             {
97                 while (reader.Read())
98                 {
99                     //create new test object and add that to the list to be returned
100
101                     test.TestID = reader.GetInt32(0);
102                     test.TestName = reader.GetString(1);
103                     test.AssetName = reader.GetString(2);
104                     test.StartDate = reader.GetDateTime(3);
105                     test.EndDate = reader.GetDateTime(4);
106                     test.SignOff = reader.GetBoolean(5);
107                     test.URL = reader.GetString(6);
108                     test.Description = reader.GetString(7);
109                     test.ContactID = reader.GetString(8);
110                     test.TestStatus = reader.GetString(9);
111                     test.TestNotes = reader.GetString(11);
112                 }
113             }
114             return test;
115         }
116     }
117
118
119
120     1 reference
121     public int Create(TestModel testModel)
122     {
123
124         //access database
125         using (SqlConnection connection = new SqlConnection(connectionString))
126         {
127             string sqlQuery = "INSERT INTO dbo.Test Values(@TestName, @AssetName, @StartDate, @EndDate, @SignOff, @URL, @Description, @ContactID, @TestStatus, @TicketUser, @TestNotes)";
128
129             SqlCommand command = new SqlCommand(sqlQuery, connection);
130             command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestName;
131             command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.AssetName;
132             command.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testModel.StartDate;
133             command.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testModel.EndDate;
134             command.Parameters.Add("@SignOff", System.Data.SqlDbType.Bit).Value = testModel.SignOff;
135             command.Parameters.Add("@URL", System.Data.SqlDbType.VarChar, 1000).Value = testModel.URL;
136             command.Parameters.Add("@Description", System.Data.SqlDbType.VarChar, 1000).Value = testModel.Description;
137             command.Parameters.Add("@ContactID", System.Data.SqlDbType.VarChar, 1000).Value = testModel.ContactID;
138             command.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
139             command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
140             command.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;
141
142             connection.Open();
143             int newTestID = command.ExecuteNonQuery();
144
145         }

```

```

142         int newTestID = command.ExecuteNonQuery();
143     }
144     return newTestID;
145 }
146
147
148 1 reference
149 public int Update(TestModel testModel)
150 {
151     //access the database
152     using (SqlConnection connection = new SqlConnection(connectionString))
153     {
154         //prepared statement
155         //update
156         string sqlQuery = "UPDATE dbo.Test SET TestName = @TestName, StartDate = @StartDate, EndDate = @EndDate, SignOff = @SignOff, TestStatus = @TestStatus, TicketUser =
157             @TicketUser, TestNotes = @TestNotes WHERE TestID = @TestID";
158
159         //associate @id with Id parameter
160
161         SqlCommand command = new SqlCommand(sqlQuery, connection);
162         command.Parameters.Add("@TestID", System.Data.SqlDbType.Int).Value = testModel.TestID;
163         command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestName;
164         command.Parameters.Add("@StartDate", System.Data.SqlDbType.DateTime).Value = testModel.StartDate;
165         command.Parameters.Add("@EndDate", System.Data.SqlDbType.DateTime).Value = testModel.EndDate;
166         command.Parameters.Add("@SignOff", System.Data.SqlDbType.Bit).Value = testModel.SignOff;
167         command.Parameters.Add("@TestStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
168         command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TicketUser;
169         command.Parameters.Add("@TestNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;
170
171         //Update Test Assignment
172         string sqlQuery1 = "Update dbo.TestAssignment SET DateAssigned = @DateAssigned, DateClosed = @DateClosed, TicketStatus = @TicketStatus, TestTicketNotes = @TestTicketNotes
173             WHERE dbo.TestAssignment.TestName = @TestName";
174
175         //associate @id with Id parameter
176         SqlCommand command1 = new SqlCommand(sqlQuery1, connection);
177
178         command1.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestName;
179         command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = testModel.EndDate;
180         command1.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = testModel.StartDate;
181         command1.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
182         command1.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;
183
184         //Insert Edit To Test Log
185         string sqlQuery2 = "INSERT INTO dbo.TestLog VALUES(@TestName, @TicketUser, @CurrentTimeStamp, @TicketStatus, @TestTicketNotes)";
186
187         //associate @id with Id parameter
188         SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
189
190         command2.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestName;
191         command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = testModel.CurrentUser;
192         command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
193         command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestStatus;
194         command2.Parameters.Add("@TestTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = testModel.TestNotes;
195
196         connection.Open();
197         int newTestID = command.ExecuteNonQuery();
198         command1.ExecuteNonQuery();
199         command2.ExecuteNonQuery();
200         return newTestID;
201     }
202 }
203
204
205 1 reference
206 public List<TestModel> GetTestEventLog(string name)
207 {
208     List<TestModel> returnList = new List<TestModel>();
209     using (SqlConnection connection = new SqlConnection(connectionString))
210     {
211         string sqlQuery = "SELECT * FROM dbo.TestLog WHERE TestName = @name";
212         SqlCommand command = new SqlCommand(sqlQuery, connection);
213
214         SqlCommand command = new SqlCommand(sqlQuery, connection);
215         command.Parameters.Add("@name", System.Data.SqlDbType.VarChar, 1000).Value = name;
216         connection.Open();
217         SqlDataReader reader = command.ExecuteReader();
218         if (reader.HasRows)
219         {
220             while (reader.Read())
221             {
222                 //creating new object and adding it to the list to be returned
223                 TestModel test = new TestModel();
224                 test.TestLogID = reader.GetInt32(0);
225                 test.TestName = reader.GetString(1);
226                 test.TicketUser = reader.GetString(2);
227                 test.CurrentTimeStamp = reader.GetDateTime(3);
228                 test.TicketStatus = reader.GetString(4);
229                 test.TestTicketNotes = reader.GetString(5);
230                 returnList.Add(test);
231             }
232         }
233         return returnList;
234     }
235 }
236
237
238

```

TestModel.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Web;
6 using System.Web.Mvc;
7
8 namespace Project.Models
9 {
10     30 references
11     public class TestModel
12     {
13         4 references
14         [Display(Name = "Test ID")]
15         public int TestID { get; set; }
16         1 reference
17         [Display(Name = "Test Log ID")]
18         public int TestLogID { get; set; }
19         8 references
20         [Display(Name = "Test Name")]
21         public string TestName { get; set; }
22         3 references
23         [Display(Name = "Asset")]
24         public string AssetName { get; set; }
25         1 reference
26         [Display(Name = "Ticket Status")]
27         public string TicketStatus { get; set; }
28         1 reference
29         [Display(Name = "Ticket Notes")]
30         public string TestTicketNotes { get; set; }
31         1 reference
32         [Display(Name = "Current TimeStamp")]
33         public DateTime CurrentTimeStamp { get; set; }
34         1 reference
35         [Display(Name = "Start Date")]
36         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy}")]
37         public DateTime StartDate { get; set; }
38         6 references
39         [Display(Name = "End Date")]
40         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy}")]
41         public DateTime EndDate { get; set; }
42         6 references
43         [Display(Name = "Sign Off Documents")]
44         public bool SignOff { get; set; }
45         5 references
46         [Display(Name = "URL")]
47         public string URL { get; set; }
48         4 references
49         [Display(Name = "Description")]
50         public string Description { get; set; }
51         4 references
52         [Display(Name = "Contact ID")]
53         public string ContactID { get; set; }
54         4 references
55         [Display(Name = "Test Status")]
56         public string TestStatus { get; set; }
57         6 references
58         [Display(Name = "Assignee")]
59         public string TicketUser { get; set; }
60         2 references
61         [Display(Name = "Current User")]
62         public string CurrentUser { get; set; }
63         1 reference
64         [Display(Name = "Test Notes")]
65         public string TestNotes { get; set; }
66         6 references
67         public List<SelectListItem> ListAssetName { get; set; }
68         3 references
69         public List<SelectListItem> ListTestStatus { get; set; }
70
71     }
72     0 references
73     public TestModel(int testID, string testName, DateTime startDate, DateTime endDate, bool signOff, string url, string description, string contactID)
74     {
75         TestID = testID;
76         TestName = testName;
77         StartDate = startDate;
78         EndDate = endDate;
79         SignOff = signOff;
80         URL = url;
81     }
82 }
```

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

Test Views

CreateTestForm.cshtml

```
1 @model Project.Models.TestModel
2
3
4 @{
5     ViewBag.Title = "";
6     Layout = "~/Views/Shared/_Layout.cshtml";
7 }
8 @{
9     String Role;
10    Role = Session["userRole"].ToString();
11    if (Role == "Administrator")
12    {
13    }
14
15    else if (Role == "PenetrationTester")
16    {
17    }
18
19    else
20    {
21        Response.Redirect("~/Home/Index");
22    }
23 }
24 <h2></h2>
25
26
27 @using (Html.BeginForm("CreateProcess", "tests"))
28 {
29     @Html.AntiForgeryToken()
30
31     <div class="form-horizontal">
32         <br />
33         <hr />
34         <h4><b>Create Test</b></h4>
35         <hr />
36         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
37
38         <div class="form-group">
39             @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
40             <div class="col-md-10">
41                 @Html.DropDownListFor(m => m.AssetName, Model.ListAssetName, "Please select")
42             </div>
43         </div>
44
45         <div class="form-group">
46             @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
47             <div class="col-md-10">
48                 @Html.EditorFor(model => model.TestName, new { htmlAttributes = new { @class = "form-control" } })
49                 @Html.ValidationMessageFor(model => model.TestName, "", new { @class = "text-danger" })
50             </div>
51         </div>
52
53         <div class="form-group">
54             @Html.LabelFor(model => model.StartDate, htmlAttributes: new { @class = "control-label col-md-2" })
55             <div class="col-md-10">
56                 @Html.TextBoxFor(model => model.StartDate, new { type = "date" })
57             </div>
58         </div>
59
60         <div class="form-group">
61             @Html.LabelFor(model => model.EndDate, htmlAttributes: new { @class = "control-label col-md-2" })
62             <div class="col-md-10">
63                 @Html.TextBoxFor(model => model.EndDate, new { type = "date" })
64             </div>
65         </div>
66
67         <div class="form-group">
68             @Html.LabelFor(model => model.SignOff, htmlAttributes: new { @class = "control-label col-md-2" })
69             <div class="col-md-10">
70                 <div class="checkbox">
71                     @Html.EditorFor(model => model.SignOff)
72                     @Html.ValidationMessageFor(model => model.SignOff, "", new { @class = "text-danger" })
73                 </div>
74             </div>
75         </div>
76
77         <div class="form-group">
78             @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

```
1  @model Project.Models.TestModel
2
3  @{
4  ViewBag.Title = "";
5  Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7  @{
8  String Role;
9  Role = Session["userRole"].ToString();
10 if (Role == "Administrator")
11 {
12
13
14 }
15 else if (Role == "PenetrationTester")
16 {
17
18 }
19 else
20 {
21     Response.Redirect("~/Home/Index");
22 }
23 }
24 <h2></h2>
25
26 <div>
27     <br />
28     <hr />
29     <h4><b>Test Details</b></h4>
30     <hr />
31     <dl class="dl-horizontal">
32         <dt>
33             @Html.DisplayNameFor(model => model.TestName)
34         </dt>
35
36         <dd>
37             @Html.DisplayFor(model => model.TestName)
38         </dd>
39         <dt>
40             @Html.DisplayNameFor(model => model.AssetName)
41         </dt>
42
43         <dd>
44             @Html.DisplayFor(model => model.AssetName)
45         </dd>
46         <dt>
47             @Html.DisplayNameFor(model => model.StartDate)
48         </dt>
49         <dd>
50             @Html.DisplayFor(model => model.StartDate)
51         </dd>
52         <dt>
53             @Html.DisplayNameFor(model => model.EndDate)
54         </dt>
55         <dd>
56             @Html.DisplayFor(model => model.EndDate)
57         </dd>
58         <dt>
59             @Html.DisplayNameFor(model => model.SignOff)
60         </dt>
61         <dd>
62             @Html.DisplayFor(model => model.SignOff)
63         </dd>
64         <dt>
65             @Html.DisplayNameFor(model => model.URL)
66         </dt>
67         <dd>
68             @Html.DisplayFor(model => model.URL)
69         </dd>
70         <dt>
71             @Html.DisplayNameFor(model => model.Description)
72         </dt>
73         <dd>
74             @Html.DisplayFor(model => model.Description)
75         </dd>
76         <dt>
77             @Html.DisplayNameFor(model => model.ContactID)
78         </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 </dl>
95 </div>
96 <p>
97     @Html.ActionLink("Edit", "Edit", new { id = Model.TestID }) |
98     @Html.ActionLink("Back to List", "Index")
99 </p>
100 @Html.ActionLink("TestEventLog", "TestEventLog", new { name = Model.TestName })
101

```

Index.cshtml

```
1  @model IEnumerable<Project.Models.TestModel>
2
3  @{
4  ViewBag.Title = "";
5  Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7  @{
8  String Role;
9  Role = Session["userRole"].ToString();
10 if (Role == "Administrator")
11 {
12
13
14 }
15 else if (Role == "PenetrationTester")
16 {
17
18 }
19 else
20 {
21     Response.Redirect("~/Home/Index");
22 }
23 }
24 <h2> </h2>
25
26 <p>
27     <br />
28     <hr />
29     <h4><b>Tests</b></h4>
30     <hr />
31     @Html.ActionLink("Create New", "Create")
32 </p>
33 <table class="table">
34     <tr>
35         <th>
36             @Html.DisplayNameFor(model => model.TestName)
37         </th>
38         <th>
39             @Html.DisplayNameFor(model => model.AssetName)
40         </th>
41         <th>
42             @Html.DisplayNameFor(model => model.StartDate)
43         </th>
44         <th>
45             @Html.DisplayNameFor(model => model.EndDate)
46         </th>
47         <th>
48             @Html.DisplayNameFor(model => model.SignOff)
49         </th>
50         <th>
51             @Html.DisplayNameFor(model => model.URL)
52         </th>
53         <th>
54             @Html.DisplayNameFor(model => model.Description)
55         </th>
56         <th>
57             @Html.DisplayNameFor(model => model.ContactID)
58         </th>
59         <th>
60             @Html.DisplayNameFor(model => model.TestNotes)
61         </th>
62         <th>
63             @Html.DisplayNameFor(model => model.TestStatus)
64         </th>
65     <th></th>
66 </tr>
67
68 @foreach (var item in Model) {
69     <tr>
70         <td>
71             @Html.DisplayFor(modelItem => item.TestName)
72         </td>
73         <td>
74             @Html.DisplayFor(modelItem => item.AssetName)
75         </td>
76         <td>
77             @Html.DisplayFor(modelItem => item.StartDate)
78         </td>
```

```

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

```
1  @model IEnumerable<Project.Models.TestModel>
2
3  @{
4      ViewBag.Title = "";
5      Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <h2> </h2>
9
10 <p>
11     <br />
12     <hr />
13     <h4><b>Test Event Log</b></h4>
14     <hr />
15 </p>
16
17 <table class="table">
18     <tr>
19         <th>
20             @Html.DisplayNameFor(model => model.TestLogID)
21         </th>
22         <th>
23             @Html.DisplayNameFor(model => model.TestName)
24         </th>
25         <th>
26             @Html.DisplayNameFor(model => model.TicketUser)
27         </th>
28         <th>
29             @Html.DisplayNameFor(model => model.CurrentTimeStamp)
30         </th>
31         <th>
32             @Html.DisplayNameFor(model => model.TicketStatus)
33         </th>
34         <th>
35             @Html.DisplayNameFor(model => model.TestTicketNotes)
36         </th>
37     </tr>
38
39
40     @foreach (var item in Model) {
41         <tr>
42             <td>
43                 @Html.DisplayFor(modelItem => item.TestLogID)
44             </td>
45             <td>
46                 @Html.DisplayFor(modelItem => item.TestName)
47             </td>
48             <td>
49                 @Html.DisplayFor(modelItem => item.TicketUser)
50             </td>
51             <td>
52                 @Html.DisplayFor(modelItem => item.CurrentTimeStamp)
53             </td>
54             <td>
55                 @Html.DisplayFor(modelItem => item.TicketStatus)
56             </td>
57             <td>
58                 @Html.DisplayFor(modelItem => item.TestTicketNotes)
59             </td>
60         </tr>
61     }
62
63 </table>
64 <div>
65     @Html.ActionLink("Back to List", "Index")
66 </div>
```

UpdateTestForm.cshtml

```
1 @model Project.Models.TestModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7 @{
8     String Role;
9     Role = Session["userRole"].ToString();
10    if (Role == "Administrator")
11    {
12
13    }
14
15    else if (Role == "PenetrationTester")
16    {
17
18    }
19    else
20    {
21        Response.Redirect("~/Home/Index");
22    }
23 }
24 <h2></h2>
25
26
27 @using (Html.BeginForm("UpdateTest", "tests"))
28 {
29     @Html.AntiForgeryToken()
30
31     <div class="form-horizontal">
32         <br/>
33         <hr/>
34         <h4><b>Update Test</b></h4>
35         <hr/>
36         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
37
38         <div hidden class="form-group">
39             @Html.LabelFor(model => model.TestID, htmlAttributes: new { @class = "control-label col-md-2" })
40             <div hidden class="col-md-10">
41                 @Html.EditorFor(model => model.TestID, new { htmlAttributes = new { @class = "form-control" } })
42                 @Html.ValidationMessageFor(model => model.TestID, "", new { @class = "text-danger" })
43             </div>
44         </div>
45         <div class="form-group">
46             @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
47             <div class="col-md-10">
48                 @Html.EditorFor(model => model.TestName, new { htmlAttributes = new { @class = "form-control" } })
49             </div>
50         </div>
51
52         <div class="form-group">
53             @Html.LabelFor(model => model.StartDate, htmlAttributes: new { @class = "control-label col-md-2" })
54             <div class="col-md-10">
55                 @Html.EditorFor(model => model.StartDate, new { htmlAttributes = new { @class = "form-control" } })
56             </div>
57         </div>
58
59         <div class="form-group">
60             @Html.LabelFor(model => model.EndDate, htmlAttributes: new { @class = "control-label col-md-2" })
61             <div class="col-md-10">
62                 @Html.EditorFor(model => model.EndDate, new { htmlAttributes = new { @class = "form-control" } })
63             </div>
64         </div>
65
66         <div class="form-group">
67             @Html.LabelFor(model => model.SignOff, htmlAttributes: new { @class = "control-label col-md-2" })
68             <div class="col-md-10">
69                 <div class="checkbox">
70                     @Html.EditorFor(model => model.SignOff)
71                     @Html.ValidationMessageFor(model => model.SignOff, "", new { @class = "text-danger" })
72                 </div>
73             </div>
74         </div>
75
76         <div class="form-group">
77             @Html.LabelFor(model => model.TestNotes, htmlAttributes: new { @class = "control-label col-md-2" })
78             <div class="col-md-10">
```

```

77     @Html.LabelFor(model => model.TestNotes, htmlAttributes: new { @class = "control-label col-md-2" })
78     <div class="col-md-10">
79         @Html.EditorFor(model => model.TestNotes, new { htmlAttributes = new { @class = "form-control" } })
80     </div>
81 </div>
82 <div class="form-group">
83     @Html.LabelFor(model => model.TestStatus, htmlAttributes: new { @class = "control-label col-md-2" })
84     <div class="col-md-10">
85         @Html.DropDownListFor(m => m.TestStatus, Model.ListTestStatus, "Please select")
86     </div>
87 </div>
88 <div hidden class="form-group">
89     @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
90     <div hidden class="col-md-10">
91         @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
92     </div>
93 </div>
94
95 <div hidden class="form-group">
96     @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
97     <div hidden class="col-md-10">
98         @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
99     </div>
100 </div>
101
102 <input type="hidden" name="TestID" value="TestID" />
103
104 <div class="form-group">
105     <div class="col-md-offset-2 col-md-10">
106         <input type="submit" value="Save" class="btn btn-default" />
107     </div>
108 </div>
109 </div>
110 }
111
112 <div>
113     @Html.ActionLink("Back to List", "Index")
114 </div>
115
116 @section Scripts {
117     @Scripts.Render("~/bundles/jqueryval")
118 }
119

```

VulnerabilitiesController.cshtml

```
1 using Project.Data;
2 using Project.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Configuration;
6 using System.Data.SqlClient;
7 using System.Web.Mvc;
8
9
10 namespace Project.Controllers
11 {
12
13
14     0 references
15     public class VulnerabilitiesController : Controller
16     {
17         private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
18         Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
19
20         [CustomAuthorization]
21         0 references
22         public ActionResult Index()
23         {
24             List<VulnerabilityModel> vulnerabilities = new List<VulnerabilityModel>();
25             VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
26             vulnerabilities = vulnerabilityDAO.FetchAll();
27             return View("Index", vulnerabilities);
28         }
29
30         [CustomAuthorization]
31         0 references
32         public ActionResult Details(int id)
33         {
34             VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
35             vulnerabilityDAO.GetTactic(id);
36             vulnerabilityDAO.GetAssetName(id);
37             VulnerabilityModel vulnerability = vulnerabilityDAO.FetchOne(id);
38             return View("Details", vulnerability);
39         }
40
41         [CustomAuthorization]
42         0 references
43         public ActionResult Create()
44         {
45             VulnerabilityModel vulnerability = new VulnerabilityModel();
46             vulnerability.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
47             vulnerability.MitreTactic = PopulateDropDown(" SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
48             vulnerability.ListVulnerabilityStatusPentester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPentester", "StatusType", "StatusType");
49             vulnerability.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
50             return View("AddVulnerabilityForm", vulnerability);
51         }
52
53         [CustomAuthorization]
54         [HttpPost]
55         0 references
56         public JsonResult AjaxMethod(string type, int value)
57         {
58             VulnerabilityModel vulnerability = new VulnerabilityModel();
59             switch (type)
60             {
61                 case "TacticID":
62                     vulnerability.MitreTechnique = PopulateDropDown("SELECT TechniqueID, Technique FROM MitreTechnique WHERE TacticID= " + value, "Technique", "TechniqueID");
63                     break;
64             }
65             return Json(vulnerability);
66         }
67
68         [CustomAuthorization]
69         [HttpPost]
70         0 references
71         public ActionResult Create(int tacticID)
72         {
73         }
```

```

69 {
70     VulnerabilityModel vulnerability = new VulnerabilityModel();
71     vulnerability.MitreTactic = PopulateDropDown("SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
72     vulnerability.MitreTechnique = PopulateDropDown("SELECT TechniqueID, Technique FROM MitreTechnique WHERE TacticID = " + tacticID, "Technique", "TechniqueID");
73     return View(vulnerability);
74 }
75
76 [CustomAuthorization]
77 13 references
78 private static List<SelectListItem> PopulateDropDown(string query, string textColumn, string valueColumn)
79 {
80     List<SelectListItem> items = new List<SelectListItem>();
81     using (SqlConnection connection = new SqlConnection(connectionString))
82     {
83         using (SqlCommand cmd = new SqlCommand(query))
84         {
85             cmd.Connection = connection;
86             connection.Open();
87             using (SqlDataReader sdr = cmd.ExecuteReader())
88             {
89                 while (sdr.Read())
90                 {
91                     items.Add(new SelectListItem
92                     {
93                         Text = sdr[textColumn].ToString(),
94                         Value = sdr[valueColumn].ToString()
95                     });
96                 }
97             }
98             connection.Close();
99         }
100     }
101     return items;
102 }
103
104 [CustomAuthorization]
105 0 references
106 public ActionResult AddVulnerability(VulnerabilityModel vulnerabilityModel)
107 {
108     //save it to the database
109     VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
110     vulnerabilityDAO.Create(vulnerabilityModel);
111     List<VulnerabilityModel> vulnerabilities = new List<VulnerabilityModel>();
112     //Return full list of vulnerabilities
113     vulnerabilities = vulnerabilityDAO.FetchAll();
114     return View("Index", vulnerabilities);
115 }
116
117 [CustomAuthorization]
118 0 references
119 public ActionResult Edit(int id)
120 {
121     VulnerabilityDAO vulnerabilityDAO = new VulnerabilityDAO();
122     VulnerabilityModel vulnerability = vulnerabilityDAO.FetchOne(id);
123     vulnerability.ListTestName = PopulateDropDown(" SELECT TestName FROM dbo.Test", "TestName", "TestName");
124     vulnerability.MitreTactic = PopulateDropDown(" SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
125     vulnerability.ListVulnerabilityStatusPentester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPentester", "StatusType", "StatusType");
126     vulnerability.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
127     return View("UpdateVulnerabilityForm", vulnerability);
128 }
129 [CustomAuthorization]
130 [HttpPost]
131 0 references
132 public ActionResult Edit1(int tacticID)
133 {
134     VulnerabilityModel vulnerability = new VulnerabilityModel();
135     vulnerability.MitreTactic = PopulateDropDown("SELECT TacticID, Tactic FROM MitreTactic", "Tactic", "TacticID");
136     vulnerability.MitreTechnique = PopulateDropDown("SELECT TechniqueID, Technique FROM MitreTechnique WHERE TacticID = " + tacticID, "Technique", "TechniqueID");
137     return View(vulnerability);
138 }

```



```

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

```

VulnerabilityDAO.cs

```
1 using Project.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Data.SqlClient;
5
6 namespace Project.Data
7 {
8     16 references
9     internal class VulnerabilityDAO
10    {
11        string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
12        Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFallback=False";
13        //performs all actions on vulnerability
14
15        5 references
16        public List<VulnerabilityModel> FetchAll()
17        {
18            List<VulnerabilityModel> returnList = new List<VulnerabilityModel>();
19
20            using (SqlConnection connection = new SqlConnection(connectionString))
21            {
22                string sqlQuery = "SELECT * from dbo.Vulnerability";
23                SqlCommand command = new SqlCommand(sqlQuery, connection);
24                connection.Open();
25                SqlDataReader reader = command.ExecuteReader();
26                if (reader.HasRows)
27                {
28                    while (reader.Read())
29                    {
30                        //Create new vulnerability object and add it to list to be returned
31                        VulnerabilityModel vulnerability = new VulnerabilityModel();
32                        vulnerability.VulnerabilityID = reader.GetInt32(0);
33                        vulnerability.TestName = reader.GetString(1);
34                        vulnerability.VulnerabilityType = reader.GetString(2);
35                        vulnerability.Severity = reader.GetString(3);
36                        vulnerability.DateOpen = reader.GetDateTime(4);
37                        vulnerability.DateClosed = reader.GetDateTime(5);
38                        vulnerability.StatusID = reader.GetString(11);
39
40                        returnList.Add(vulnerability);
41                    }
42                }
43            }
44
45            return returnList;
46        }
47
48
49        0 references
50        public VulnerabilityModel GetAsset()
51        {
52            using (SqlConnection connection = new SqlConnection(connectionString))
53            {
54                //SQL Statement Open Tests
55                string sqlQuery = "SELECT dbo.Test.AssetName, dbo.Vulnerability.TestName, dbo.Vulnerability.VulnerabilityID FROM dbo.Vulnerability INNER JOIN dbo.Test ON
56                dbo.Vulnerability.TestName = dbo.Test.TestName INNER JOIN dbo.Asset ON dbo.Asset.AssetName = dbo.Test.AssetName ORDER BY dbo.Vulnerability.VulnerabilityID";
57
58                SqlCommand command = new SqlCommand(sqlQuery, connection);
59                connection.Open();
60                SqlDataReader reader = command.ExecuteReader();
61                VulnerabilityModel vulnerabilityModel = new VulnerabilityModel();
62                if (reader.HasRows)
63                {
64                    while (reader.Read())
65                    {
66                        vulnerabilityModel.AssetName = reader.GetString(0);
67                    }
68                }
69                reader.Close();
70                return vulnerabilityModel;
71            }
72        }
73    }
74 }
```

```

71     }
72 }
73
74
75 1 reference
76 internal int Delete(int id)
77 {
78     using (SqlConnection connection = new SqlConnection(connectionString))
79     {
80         //prepared statement
81         //update
82         string sqlQuery = "DELETE FROM dbo.Vulnerability WHERE VulnerabilityID = @id";
83
84
85
86
87         //associate @id with Id parameter
88
89         SqlCommand command = new SqlCommand(sqlQuery, connection);
90
91         command.Parameters.Add("@id", System.Data.SqlDbType.VarChar, 1000).Value = id;
92
93
94         connection.Open();
95         int deletedID = command.ExecuteNonQuery();
96
97         return deletedID;
98     }
99 }
100
101 1 reference
102 public VulnerabilityModel GetTactic(int id)
103 {
104     string Tactic;
105     string Technique;
106     using (SqlConnection connection = new SqlConnection(connectionString))
107     {
108         string sqlQuery = "SELECT dbo.MitreTactic.Tactic, dbo.MitreTechnique.Technique, dbo.Vulnerability.VulnerabilityID FROM dbo.Vulnerability INNER JOIN dbo.MitreTactic ON
109             dbo.Vulnerability.TacticID = dbo.MitreTactic.TacticID INNER JOIN dbo.MitreTechnique ON dbo.Vulnerability.TechniqueID = dbo.MitreTechnique.TechniqueID WHERE
110             dbo.Vulnerability.VulnerabilityID = @id ORDER BY dbo.Vulnerability.VulnerabilityID";
111
112         SqlCommand command = new SqlCommand(sqlQuery, connection);
113         command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
114
115         connection.Open();
116         //string result = (string)command.ExecuteReader();
117
118         SqlDataReader reader = command.ExecuteReader();
119         VulnerabilityModel vulnerabilityModel = new VulnerabilityModel();
120         if (reader.HasRows) {
121             while (reader.Read())
122             {
123                 vulnerabilityModel.Tactic = reader.GetString(0);
124                 Tactic = vulnerabilityModel.Tactic;
125                 vulnerabilityModel.Tactic = Tactic;
126                 vulnerabilityModel.Technique = reader.GetString(1);
127                 Technique = vulnerabilityModel.Technique;
128                 vulnerabilityModel.Technique = Technique;
129                 vulnerabilityModel.VulnerabilityID = reader.GetInt32(2);
130             }
131         }
132         reader.Close();
133         string Query = "UPDATE dbo.Vulnerability SET Tactic = @Tactic, Technique = @Technique WHERE VulnerabilityID = @VulnerabilityID";
134
135         //associate @id with Id parameter
136
137         SqlCommand command1 = new SqlCommand(Query, connection);
138
139

```

```

139
140
141 command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = id;
142 command1.Parameters.Add("@Tactic", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Tactic;
143 command1.Parameters.Add("@Technique", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Technique;
144
145
146 command1.ExecuteNonQuery();
147
148
149
150 connection.Close();
151 return vulnerabilityModel;
152 }
153
154
155 1 reference
156 public VulnerabilityModel GetAssetName(int id)
157 {
158     string AssetName;
159     int TestID;
160
161     using (SqlConnection connection = new SqlConnection(connectionString))
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 =
164             dbo.Test.TestName WHERE dbo.Vulnerability.VulnerabilityID = @id ORDER BY dbo.Vulnerability.VulnerabilityID";
165
166         SqlCommand command = new SqlCommand(sqlQuery, connection);
167         command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
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     reader.Close();
190     string Query = "UPDATE dbo.Vulnerability SET AssetName = @AssetName, TestID = @TestID WHERE VulnerabilityID = @VulnerabilityID";
191
192     //associate @id with Id parameter
193
194     SqlCommand command1 = new SqlCommand(Query, connection);
195
196     command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = id;
197     command1.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.AssetName;
198     command1.Parameters.Add("@TestID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.TestID;
199
200
201     command1.ExecuteNonQuery();
202
203
204
205
206
207 connection.Close();
208 return vulnerabilityModel;
209 }
210

```

```

210     }
211 }
212
213 2 references
214 public VulnerabilityModel FetchOne(int id)
215 {
216     //access database
217     using (SqlConnection connection = new SqlConnection(connectionString))
218     {
219         string sqlQuery = "SELECT * FROM dbo.Vulnerability WHERE VulnerabilityID = @id";
220
221         SqlCommand command = new SqlCommand(sqlQuery, connection);
222         command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
223         connection.Open();
224         SqlDataReader reader = command.ExecuteReader();
225
226         VulnerabilityModel vulnerability = new VulnerabilityModel();
227         if (reader.HasRows)
228         {
229             while (reader.Read())
230             {
231                 //create new test object and add that to the list to be returned
232
233                 vulnerability.VulnerabilityID = reader.GetInt32(0);
234                 vulnerability.AssetName = reader.GetString(20);
235                 vulnerability.TestName = reader.GetString(1);
236                 vulnerability.TestID = reader.GetInt32(21);
237                 vulnerability.VulnerabilityType = reader.GetString(2);
238                 vulnerability.Severity = reader.GetString(3);
239                 vulnerability.DateOpen = reader.GetDateTime(4);
240                 vulnerability.DateClosed = reader.GetDateTime(5);
241                 vulnerability.URL = reader.GetString(6);
242                 vulnerability.Tactic = reader.GetString(12);
243                 vulnerability.Technique = reader.GetString(13);
244                 vulnerability.StepsToReproduce = reader.GetString(9);
245                 vulnerability.Mitigation = reader.GetString(10);
246                 vulnerability.StatusID = reader.GetString(11);
247                 vulnerability.DamagePotential = reader.GetDecimal(14);
248                 vulnerability.Reproducibility = reader.GetDecimal(15);
249                 vulnerability.Exploitability = reader.GetDecimal(16);
250                 vulnerability.AffectedUsers = reader.GetDecimal(17);
251                 vulnerability.Discoverability = reader.GetDecimal(18);
252                 vulnerability.DREADTotal = reader.GetDecimal(19);
253
254             }
255         }
256         return vulnerability;
257     }
258 }
259
260
261 1 reference
262 public int Create(VulnerabilityModel vulnerabilityModel)
263 {
264
265     //access database
266     using (SqlConnection connection = new SqlConnection(connectionString))
267     {
268         string sqlQuery = "INSERT INTO dbo.Vulnerability Values(@TestName, @VulnerabilityType, @Severity, @DateOpen, @DateClosed, @URL, @TacticID, @TechniqueID, @StepsToReproduce,
269             @Mitigation, @StatusID, @Tactic, @Technique, @DamagePotential, @Reproducibility, @Exploitability, @AffectedUsers, @Discoverability, @DREADTotal, @AssetName, @TestID,
270             @TicketNotes, @AssignedUser)";
271
272         SqlCommand command = new SqlCommand(sqlQuery, connection);
273         command.Parameters.Add("@TestName", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TestName;
274         command.Parameters.Add("@VulnerabilityType", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.VulnerabilityType;
275         command.Parameters.Add("@Severity", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Severity;
276         command.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateOpen;
277         command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateClosed;
278         command.Parameters.Add("@URL", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.URL;
279         command.Parameters.Add("@TacticID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityModel.TacticID;

```

```

277 command.Parameters.Add("@TacticID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityModel.TacticID;
278 command.Parameters.Add("@TechniqueID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityModel.TechniqueID;
279 command.Parameters.Add("@StepsToReproduce", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StepsToReproduce;
280 command.Parameters.Add("@Mitigation", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Mitigation;
281 command.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
282 command.Parameters.Add("@Tactic", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
283 command.Parameters.Add("@Technique", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
284 command.Parameters.Add("@DamagePotential", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.DamagePotential;
285 command.Parameters.Add("@Reproducibility", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.Reproducibility;
286 command.Parameters.Add("@Exploitability", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.Exploitability;
287 command.Parameters.Add("@AffectUsers", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.AffectedUsers;
288 command.Parameters.Add("@Discoverability", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.Discoverability;
289 command.Parameters.Add("@DREADTotal", System.Data.SqlDbType.Decimal, 100).Value = vulnerabilityModel.DREADTotal;
290 command.Parameters.Add("@AssetName", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
291 command.Parameters.Add("@TestID", System.Data.SqlDbType.Int).Value = "0";
292 command.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
293 command.Parameters.Add("@AssignedUser", System.Data.SqlDbType.VarChar, 1000).Value = "Empty";
294
295
296
297
298
299 connection.Open();
300 int newVulnerabilityID = command.ExecuteNonQuery();
301
302 return newVulnerabilityID;
303 }
304 }
305
306 1 reference
307 public int Update(VulnerabilityModel vulnerabilityModel)
308 {
309     //access the database
310     using (SqlConnection connection = new SqlConnection(connectionString))
311     {
312         //Update Vulnerability
313         string sqlQuery = "UPDATE dbo.Vulnerability SET DateOpen = @DateOpen, DateClosed = @DateClosed, StepsToReproduce = @StepsToReproduce, Mitigation = @Mitigation, StatusID =
314             @StatusID, TicketNotes = @TicketNotes WHERE VulnerabilityID = @VulnerabilityID";
315
316         //associate @id with Id parameter
317         SqlCommand command = new SqlCommand(sqlQuery, connection);
318
319         command.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.VulnerabilityID;
320         command.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateOpen;
321         command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateClosed;
322         command.Parameters.Add("@StepsToReproduce", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StepsToReproduce;
323         command.Parameters.Add("@Mitigation", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.Mitigation;
324         command.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
325         command.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TicketNotes;
326
327         //Update Vulnerability Assignment
328         string sqlQuery1 = "Update dbo.VulnerabilityAssignment SET DateAssigned = @DateAssigned, DateClosed = @DateClosed, TicketStatus = @TicketStatus, VulnerabilityTicketNotes =
329             @VulnerabilityTicketNotes WHERE dbo.vulnerabilityAssignment.VulnerabilityID = @VulnerabilityID";
330
331         //associate @id with Id parameter
332         SqlCommand command1 = new SqlCommand(sqlQuery1, connection);
333
334         command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityModel.VulnerabilityID;
335         command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateClosed;
336         command1.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = vulnerabilityModel.DateOpen;
337         command1.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.StatusID;
338         command1.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityModel.TicketNotes;
339
340         //Insert Edit To Vulnerability Log
341         string sqlQuery2 = "INSERT INTO dbo.VulnerabilityLog VALUES(@VulnerabilityID, @TicketUser, @CurrentTimeStamp, @TicketStatus, @VulnerabilityTicketNotes)";
342
343         //associate @id with Id parameter
344         SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
345
346         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
360 1 reference
361 public List<VulnerabilityModel> GetVulnerabilityEventLog(int id)
362 {
363     List<VulnerabilityModel> returnList = new List<VulnerabilityModel>();
364     using (SqlConnection connection = new SqlConnection(connectionString))
365     {
366         string sqlQuery = "SELECT * FROM dbo.VulnerabilityLog WHERE VulnerabilityID = @id";
367         SqlCommand command = new SqlCommand(sqlQuery, connection);
368         command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
369         connection.Open();
370         SqlDataReader reader = command.ExecuteReader();
371         if (reader.HasRows)
372         {
373             while (reader.Read())
374             {
375                 //creating new object and adding it to the list to be returned
376                 VulnerabilityModel vulnerability = new VulnerabilityModel();
377                 vulnerability.VulnerabilityLogID = reader.GetInt32(0);
378                 vulnerability.VulnerabilityID = reader.GetInt32(1);
379                 vulnerability.TicketUser = reader.GetString(2);
380                 vulnerability.CurrentTimeStamp = reader.GetDateTime(3);
381                 vulnerability.TicketStatus = reader.GetString(4);
382                 vulnerability.VulnerabilityTicketNotes = reader.GetString(5);
383                 returnList.Add(vulnerability);
384             }
385         }
386     }
387     return returnList;
388 }
389
390 }
391
392 }

```

VulnerabilityModel.cs

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.ComponentModel.DataAnnotations;
5 using System.Data.Entity;
6 using System.Linq;
7 using System.Web;
8 using System.Web.Mvc;
9 using System.Configuration;
10 using System.Data.SqlClient;
11
12 namespace Project.Models
13 {
14     48 references
15     public class VulnerabilityModel
16     {
17         0 references
18         public List<SelectListItem> Vulnerabilities { get; set; }
19         0 references
20         public List<SelectListItem> MitreTactics { get; set; }
21         0 references
22         public List<SelectListItem> MitreTechniques { get; set; }
23         [Display(Name = "Vulnerability ID")]
24         9 references
25         public int? VulnerabilityID { get; set; }
26         [Display(Name = "Vulnerability Log ID")]
27         1 reference
28         public int VulnerabilityLogID { get; set; }
29         [Display(Name = "Test Name")]
30         4 references
31         public string TestName { get; set; }
32         [Display(Name = "Ticket Status")]
33         1 reference
34         public string TicketStatus { get; set; }
35         [Display(Name = "Vulnerability Ticket Notes")]
36         1 reference
37         public string VulnerabilityTicketNotes { get; set; }
38         [Display(Name = "Vulnerability Type")]
39         4 references
40         public string VulnerabilityType { get; set; }
41         [Display(Name = "Severity")]
42         4 references
43         public string Severity { get; set; }
44         [Display(Name = "Date Open")]
45         6 references
46         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy}")]
47         public DateTime DateOpen { get; set; }
48         [Display(Name = "Date Closed")]
49         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy}")]
50         6 references
51         public DateTime DateClosed { get; set; }
52         [Display(Name = "URL")]
53         3 references
54         public string URL { get; set; }
55
56         [Required(ErrorMessage = "Please select a category")]
57         [Display(Name = "Steps To Reproduce")]
58         4 references
59         public string StepsToReproduce { get; set; }
60         [Display(Name = "Mitigation")]
61         4 references
62         public string Mitigation { get; set; }
63         [Display(Name = "Status")]
64         7 references
65         public string StatusID { get; set; }
66         [Display(Name = "Mitre Tactic")]
67         6 references
68         public string Tactic { get; set; }
69         [Display(Name = "Mitre Technique")]
70         6 references
71         public string Technique { get; set; }
72         3 references
73         public List<SelectListItem> ListTestName { get; set; }
74         1 reference
75         public List<SelectListItem> ListAssetName { get; set; }
76         3 references
77         public List<SelectListItem> ListVulnerabilityStatusPenTester { get; set; }
```



```
57 3 references
58 public List<SelectListItem> ListVulnerabilityStatusEngineer { get; set; }
59 [Display(Name = "Damage Potential Score")]
60 2 references
61 public decimal DamagePotential { get; set; }
62 [Display(Name = "Reproducibility Score")]
63 2 references
64 public decimal Reproducibility { get; set; }
65 [Display(Name = "Exploitability Score")]
66 2 references
67 public decimal Exploitability { get; set; }
68 [Display(Name = "Affected Users Score")]
69 2 references
70 public decimal AffectedUsers { get; set; }
71 [Display(Name = "Discoverability Score")]
72 2 references
73 public decimal Discoverability { get; set; }
74 [Display(Name = "DREAD Total Threat Rating")]
75 2 references
76 public decimal DREADTotal { get; set; }
77 [Display(Name = "Asset")]
78 6 references
79 public string AssetName { get; set; }
80 [Display(Name = "Test ID")]
81 5 references
82 public int TestID { get; set; }
83 [Display(Name = "Ticket Notes")]
84 4 references
85 public string TicketNotes { get; set; }
86 [Display(Name = "Assignee")]
87 2 references
88 public string TicketUser { get; set; }
89 [Display(Name = "Assigned User")]
90 0 references
91 public string AssignedUser { get; set; }
92 1 reference
93 public DateTime CurrentTimeStamp { get; set; }
94
95 0 references
96 public VulnerabilityModel(int vulnerabilityID, string testName, string vulnerabilityType, string severity, DateTime dateOpen, DateTime dateClosed, string url, string tactic, string
97 technique, string stepsToReproduce, string mitigation, string statusID, string ticketNotes)
98 {
99     VulnerabilityID = vulnerabilityID;
100     TestName = testName;
101     VulnerabilityType = vulnerabilityType;
102     Severity = severity;
103     DateOpen = dateOpen;
104     DateClosed = dateClosed;
105     URL = url;
106     Tactic = tactic;
107     Technique = technique;
108     StepsToReproduce = stepsToReproduce;
109     Mitigation = mitigation;
110     StatusID = statusID;
111     TicketNotes = ticketNotes;
112 }
113 30 references
114 public VulnerabilityModel()
115 {
116     //new
117     this.ListTestName = new List<SelectListItem>();
118     this.ListAssetName = new List<SelectListItem>();
119     this.ListVulnerabilityStatusPenTester = new List<SelectListItem>();
120     this.ListVulnerabilityStatusEngineer = new List<SelectListItem>();
121     this.MitreTactic = new List<SelectListItem>();
122     this.MitreTechnique = new List<SelectListItem>();
123 }
124 5 references
125 public List<SelectListItem> MitreTactic { get; set; }
126 4 references
127 public List<SelectListItem> MitreTechnique { get; set; }
128
129 1 reference
130 public int TacticID { get; set; }
131 1 reference
132 public int TechniqueID { get; set; }
133 1 reference
```

```
114 | }
115 | }
116 | }
117 | }
```

```
1 reference
public int TechniqueID { get; set; }
1 reference
public string selectedTactic { get; set; }
```

Vulnerability Views

AddVulnerabilityForm.css

```
1 @model Project.Models.VulnerabilityModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2> </h2>
9
10
11 @using (Html.BeginForm("AddVulnerability", "vulnerabilities", FormMethod.Post))
12 {
13     @Html.AntiForgeryToken()
14
15     <div class="form-horizontal">
16         <hr />
17         <hr />
18         <h3><b>Add Vulnerability</b></h3>
19         <hr />
20         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
21
22         <div class="form-group">
23             @Html.LabelFor(model => model.TestName, htmlAttributes: new { @class = "control-label col-md-2" })
24             <div class="col-md-10">
25                 @Html.DropDownListFor(m => m.TestName, Model.ListTestName, "Please select")
26             </div>
27         </div>
28
29         <div class="form-group">
30             @Html.LabelFor(model => model.VulnerabilityType, htmlAttributes: new { @class = "control-label col-md-2" })
31             <div class="col-md-10">
32                 @Html.EditorFor(model => model.VulnerabilityType, new { htmlAttributes = new { @class = "form-control" } })
33             </div>
34         </div>
35
36         <div class="form-group">
37             @Html.LabelFor(model => model.DateOpen, htmlAttributes: new { @class = "control-label col-md-2" })
38             <div class="col-md-10">
39                 @Html.TextBoxFor(model => model.DateOpen, new { type = "date" })
40                 @Html.ValidationMessageFor(model => model.DateOpen, "", new { @class = "text-danger" })
41             </div>
42         </div>
43
44         <div class="form-group">
45             @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
46             <div class="col-md-10">
47                 @Html.TextBoxFor(model => model.DateClosed, new { type = "date" })
48             </div>
49         </div>
50
51         <div class="form-group">
52             @Html.LabelFor(model => model.URL, htmlAttributes: new { @class = "control-label col-md-2" })
53             <div class="col-md-10">
54                 @Html.EditorFor(model => model.URL, new { htmlAttributes = new { @class = "form-control" } })
55                 @Html.ValidationMessageFor(model => model.URL, "", new { @class = "text-danger" })
56             </div>
57         </div>
58
59         <div class="form-group">
60             @Html.LabelFor(model => model.MitreTactic, htmlAttributes: new { @class = "control-label col-md-2" })
61             <div class="col-md-10">
62                 @Html.DropDownListFor(m => m.TacticID, Model.MitreTactic, "Please select")
63             </div>
64         </div>
65
66         <div class="form-group">
67             @Html.LabelFor(model => model.MitreTechnique, htmlAttributes: new { @class = "control-label col-md-2" })
68             <div class="col-md-10">
69                 @Html.DropDownListFor(m => m.TechniqueID, Model.MitreTechnique, "Please select")
70             </div>
71         </div>
72
73         <div hidden class="form-group">
74             @Html.LabelFor(model => model.Tactic, htmlAttributes: new { @class = "control-label col-md-2" })
75         </div>
76
77     </div>
78 }
```

```

77     @Html.LabelFor(model => model.Tactic, htmlAttributes: new { @class = "control-label col-md-2" })
78     <div hidden class="col-md-10">
79         @Html.EditorFor(model => model.Tactic, new { htmlAttributes = new { @class = "form-control", value = "Empty" } })
80     </div>
81 </div>
82 </div>
83
84 <div hidden class="form-group">
85     @Html.LabelFor(model => model.Technique, htmlAttributes: new { @class = "control-label col-md-2" })
86     <div hidden class="col-md-10">
87         @Html.EditorFor(model => model.Technique, new { htmlAttributes = new { @class = "form-control", value = "Empty" } })
88     </div>
89 </div>
90 <div hidden class="form-group">
91     @Html.LabelFor(model => model.AssetName, htmlAttributes: new { @class = "control-label col-md-2" })
92     <div hidden class="col-md-10">
93         @Html.EditorFor(model => model.AssetName, new { htmlAttributes = new { @class = "form-control", value = "Empty" } })
94     </div>
95 </div>
96 </div>
97 <div hidden class="form-group">
98     @Html.LabelFor(model => model.TestID, htmlAttributes: new { @class = "control-label col-md-2" })
99     <div hidden class="col-md-10">
100         @Html.EditorFor(model => model.TestID, new { htmlAttributes = new { @class = "form-control", value = "0" } })
101     </div>
102 </div>
103 <div class="form-group">
104     @Html.LabelFor(model => model.DamagePotential, htmlAttributes: new { @class = "control-label col-md-2" })
105     <div class="col-md-10">
106         @Html.EditorFor(model => model.DamagePotential, new { htmlAttributes = new { @class = "form-control", placeholder = "", type = "text", id = "DamagePotential", onchange = "CalculateDamagePotentialTotal()" } })
107         @Html.ValidationMessageFor(model => model.DamagePotential, "", new { @class = "text-danger" })
108     </div>
109 </div>
110 <div class="form-group">
111     @Html.LabelFor(model => model.Reproducibility, htmlAttributes: new { @class = "control-label col-md-2" })
112     <div class="col-md-10">
113         @Html.EditorFor(model => model.Reproducibility, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Reproducibility", onchange = "CalculateReproducibilityTotal()" } })
114         @Html.ValidationMessageFor(model => model.Reproducibility, "", new { @class = "text-danger" })
115     </div>
116 </div>
117 <div class="form-group">
118     @Html.LabelFor(model => model.Exploitability, htmlAttributes: new { @class = "control-label col-md-2" })
119     <div class="col-md-10">
120         @Html.EditorFor(model => model.Exploitability, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Exploitability", onchange = "CalculateExploitabilityTotal()" } })
121         @Html.ValidationMessageFor(model => model.Exploitability, "", new { @class = "text-danger" })
122     </div>
123 </div>
124 <div class="form-group">
125     @Html.LabelFor(model => model.AffectedUsers, htmlAttributes: new { @class = "control-label col-md-2" })
126     <div class="col-md-10">
127         @Html.EditorFor(model => model.AffectedUsers, new { htmlAttributes = new { @class = "form-control", type = "text", id = "AffectedUsers", onchange = "CalculateAffectedUsersTotal()" } })
128         @Html.ValidationMessageFor(model => model.AffectedUsers, "", new { @class = "text-danger" })
129     </div>
130 </div>
131 <div class="form-group">
132     @Html.LabelFor(model => model.Discoverability, htmlAttributes: new { @class = "control-label col-md-2" })
133     <div class="col-md-10">
134         @Html.EditorFor(model => model.Discoverability, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Discoverability", onchange = "CalculateDiscoverabilityTotal()" } })
135         @Html.ValidationMessageFor(model => model.Discoverability, "", new { @class = "text-danger" })
136     </div>
137 </div>
138 <div class="form-group">
139     @Html.LabelFor(model => model.DREADTotal, htmlAttributes: new { @class = "control-label col-md-2" })
140     <div class="col-md-10">
141         @Html.EditorFor(model => model.DREADTotal, new { htmlAttributes = new { @class = "form-control", type = "text", id = "DREADTotal", onfocus = "CalculateDREADTotal()" } })
142         @Html.ValidationMessageFor(model => model.DREADTotal, "", new { @class = "text-danger" })
143     </div>
144 </div>
145 <div class="form-group">
146     @Html.LabelFor(model => model.Severity, htmlAttributes: new { @class = "control-label col-md-2" })
147     <div class="col-md-10">
148         @Html.EditorFor(model => model.Severity, new { htmlAttributes = new { @class = "form-control", type = "text", id = "Severity", onfocus = "CalculateSeverity()" } })
149         @Html.ValidationMessageFor(model => model.Severity, "", new { @class = "text-danger" })
150     </div>

```

```

150     </div>
151 </div>
152 <div class="form-group">
153     @Html.LabelFor(model => model.StepsToReproduce, htmlAttributes: new { @class = "control-label col-md-2" })
154     <div class="col-md-10">
155         @Html.EditorFor(model => model.StepsToReproduce, new { htmlAttributes = new { @class = "form-control" } })
156         @Html.ValidationMessageFor(model => model.StepsToReproduce, "", new { @class = "text-danger" })
157     </div>
158 </div>
159
160 <div class="form-group">
161     @Html.LabelFor(model => model.Mitigation, htmlAttributes: new { @class = "control-label col-md-2" })
162     <div class="col-md-10">
163         @Html.EditorFor(model => model.Mitigation, new { htmlAttributes = new { @class = "form-control" } })
164         @Html.ValidationMessageFor(model => model.Mitigation, "", new { @class = "text-danger" })
165     </div>
166 </div>
167
168 <div class="form-group">
169     @Html.LabelFor(model => model.TicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
170     <div class="col-md-10">
171         @Html.EditorFor(model => model.TicketNotes, new { htmlAttributes = new { @class = "form-control" } })
172     </div>
173 </div>
174
175 @if
176 {
177     string Role;
178     Role = Session["UserRole"].ToString();
179     if (Role == "PenetrationTester")
180     {
181         <div class="form-group">
182             @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
183             <div class="col-md-10">
184                 @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
185             </div>
186         </div>
187     }
188     else if (Role == "SecurityEngineer")
189     {
190         <div class="form-group">
191             @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
192             <div class="col-md-10">
193                 @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusEngineer, "Please select")
194             </div>
195         </div>
196     }
197     else if (Role == "Administrator")
198     {
199         <div class="form-group">
200             @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
201             <div class="col-md-10">
202                 @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
203             </div>
204         </div>
205     }
206 }
207
208 <div hidden class="form-group">
209     @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
210     <div hidden class="col-md-10">
211         @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
212     </div>
213 </div>
214
215 <div hidden class="form-group">
216     @Html.LabelFor(model => model.AssignedUser, htmlAttributes: new { @class = "control-label col-md-2" })
217     <div hidden class="col-md-10">
218         @Html.EditorFor(model => model.AssignedUser, new { htmlAttributes = new { @class = "form-control", Value = "Empty" } })
219     </div>
220 </div>
221
222 <div class="form-group">
223     <div class="col-md offset-2 col-md-10">
224         <input type="submit" value="Create" class="btn btn-default" />
225     </div>

```

```

225     </div>
226 </div>
227 </div>
228 }
229
230 <div>
231 @Html.ActionLink("Back to List", "Index", new { id = Model.VulnerabilityID })
232 </div>
233
234 <section Scripts >
235 <script src="/Scripts/jquery-1.10.2.min.js"></script>
236 <script type="text/javascript">
237     $(function () {
238
239
240         $("select").change(function () {
241             var value = 0;
242             if ($(this).val() != "") {
243                 value = $(this).val();
244             }
245             var id = $(this).attr("id");
246             $.ajax({
247                 type: "POST",
248                 url: "/Vulnerabilities/AjaxMethod",
249                 data: '{type: "' + id + '", value: ' + value + '}',
250                 contentType: "application/json; charset=utf-8",
251                 dataType: "json",
252                 success: function (response) {
253                     var dropdownId;
254                     var list;
255                     switch (id) {
256                         case "TacticID":
257                             list = response.MitreTechnique;
258                             DisableDropDown("#TechniqueID");
259
260                             PopulateDropDown("#TechniqueID", list);
261                             break;
262                     }
263                 }
264             });
265         });
266     });
267
268     function DisableDropDown(dropdownId) {
269         $(dropdownId).attr("disabled", "disabled");
270         $(dropdownId).empty().append('<option selected="selected" value="0">Please select</option>');
271     }
272
273     function PopulateDropDown(dropdownId, list) {
274         if (list != null && list.length > 0) {
275             $(dropdownId).removeAttr("disabled");
276             $.each(list, function () {
277                 $(dropdownId).append('<option></option>'.val(this['Value']).html(this['Text']));
278             });
279         }
280     }
281 }
282 </script>
283 <script type="text/javascript">
284     function CalculateDamagePotentialTotal() {
285
286         var DamagePotential = document.getElementById('DamagePotential').value;
287         var DamageTotal = ((DamagePotential * 40) / 100);
288         //alert(DamageTotal);
289         $("#DamagePotential").val(DamageTotal);
290         var FinalVal = document.getElementById('DamagePotential').value;
291     }
292
293     function CalculateReproducibilityTotal() {
294         var Reproducibility = document.getElementById('Reproducibility').value;
295         var ReproducibilityTotal = ((Reproducibility * 5) / 100);
296         // alert(ReproducibilityTotal);
297         $("#Reproducibility").val(ReproducibilityTotal);
298         var FinalVal = document.getElementById('Reproducibility').value;
299     }
300

```

```

300
301
302 }
303 function CalculateExploitabilityTotal() {
304     var Exploitability = document.getElementById('Exploitability').value;
305     var ExploitabilityTotal = ((Exploitability * 10) / 100);
306     // alert(ExploitabilityTotal);
307     $('#Exploitability').val(ExploitabilityTotal);
308     var FinalVal = document.getElementById('Exploitability').value;
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 ReproducibilityVal = document.getElementById('Reproducibility').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 + +ReproducibilityVal + +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

```
1 @model Project.Models.VulnerabilityModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2></h2>
9 <div>
10     <br/>
11     <hr/>
12     <h4><b>Vulnerability Intelligence</b></h4>
13     <hr />
14
15     <dl class="dl-horizontal">
16         <dt>
17             @Html.DisplayNameFor(model => model.VulnerabilityID)
18         </dt>
19
20         <dd>
21             @Html.DisplayFor(model => model.VulnerabilityID)
22         </dd>
23
24         <dt>
25             @Html.DisplayNameFor(model => model.TestName)
26         </dt>
27
28         <dd>
29             @Html.DisplayFor(model => model.TestName)
30         </dd>
31
32         <dt>
33             @Html.DisplayNameFor(model => model.TestID)
34         </dt>
35
36         <dd>
37             @Html.DisplayFor(model => model.TestID)
38         </dd>
39
40         <dt>
41             @Html.DisplayNameFor(model => model.AssetName)
42         </dt>
43
44         <dd>
45             @Html.DisplayFor(model => model.AssetName)
46         </dd>
47
48         <dt>
49             @Html.DisplayNameFor(model => model.VulnerabilityType)
50         </dt>
51
52         <dd>
53             @Html.DisplayFor(model => model.VulnerabilityType)
54         </dd>
55
56         <dt>
57             @Html.DisplayNameFor(model => model.Severity)
58         </dt>
59
60         <dd>
61             @Html.DisplayFor(model => model.Severity)
62         </dd>
63
64         <dt>
65             @Html.DisplayNameFor(model => model.DateOpen)
66         </dt>
67
68         <dd>
69             @Html.DisplayFor(model => model.DateOpen)
70         </dd>
71
72         <dt>
73             @Html.DisplayNameFor(model => model.DateClosed)
74         </dt>
75
76         <dd>
77             @Html.DisplayFor(model => model.DateClosed)
78         </dd>
79     </dl>
80 </div>
```

```

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.Reproducibility)
112 </dt>
113 <dd>
114 @Html.DisplayFor(model => model.Reproducibility)
115 </dd>
116 <dt>
117 @Html.DisplayNameFor(model => model.Exploitability)
118 </dt>
119 <dd>
120 @Html.DisplayFor(model => model.Exploitability)
121 </dd>
122 <dt>
123 @Html.DisplayNameFor(model => model.AffectedUsers)
124 </dt>
125 <dd>
126 @Html.DisplayFor(model => model.AffectedUsers)
127 </dd>
128 <dt>
129 @Html.DisplayNameFor(model => model.Discoverability)
130 </dt>
131 <dd>
132 @Html.DisplayFor(model => model.Discoverability)
133 </dd>
134 <dt>
135 @Html.DisplayNameFor(model => model.DREADTotal)
136 </dt>
137 <dd>
138 @Html.DisplayFor(model => model.DREADTotal)
139 </dd>
140 <hr />
141 <h4><b>Vulnerability Mitigation</b></h4>
142 <hr />
143 <dt>
144 @Html.DisplayNameFor(model => model.StepsToReproduce)
145 </dt>
146 <dd>
147 @Html.DisplayFor(model => model.StepsToReproduce)
148 </dd>
149
150

```



```

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

```
1 @model IEnumerable<Project.Models.VulnerabilityModel>
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2> </h2>
9
10 <p>
11     <br />
12     <hr />
13     <h4><b>Vulnerabilities</b></h4>
14     <hr />
15     @Html.ActionLink("Create New", "Create")
16 </p>
17 <table class="table">
18 <tr>
19 <th>
20     @Html.DisplayNameFor(model => model.VulnerabilityID)
21 </th>
22 <th>
23     @Html.DisplayNameFor(model => model.TestName)
24 </th>
25 <th>
26     @Html.DisplayNameFor(model => model.VulnerabilityType)
27 </th>
28 <th>
29     @Html.DisplayNameFor(model => model.Severity)
30 </th>
31 <th>
32     @Html.DisplayNameFor(model => model.DateOpen)
33 </th>
34 <th>
35     @Html.DisplayNameFor(model => model.DateClosed)
36 </th>
37 <th>
38     @Html.DisplayNameFor(model => model.StatusID)
39 </th>
40 <th>
41
42 </th>
43 <th></th>
44 </tr>
45
46 @foreach (var item in Model) {
47 <tr>
48 <td>
49     @Html.DisplayFor(modelItem => item.VulnerabilityID)
50 </td>
51 <td>
52     @Html.DisplayFor(modelItem => item.TestName)
53 </td>
54 <td>
55     @Html.DisplayFor(modelItem => item.VulnerabilityType)
56 </td>
57 <td>
58     @Html.DisplayFor(modelItem => item.Severity)
59 </td>
60 <td>
61     @Html.DisplayFor(modelItem => item.DateOpen)
62 </td>
63 <td>
64     @Html.DisplayFor(modelItem => item.DateClosed)
65 </td>
66 <td>
67     @Html.DisplayFor(modelItem => item.StatusID)
68 </td>
69 <td>
70
71 </td>
72 <td>
73     @Html.ActionLink("Edit", "Edit", new { id = item.VulnerabilityID }) |
74     @Html.ActionLink("Details", "Details", new { id = item.VulnerabilityID }) |
75     @Html.ActionLink("Delete", "Delete", new { id = item.VulnerabilityID })
76 </td>
77 </tr>
78
79 </tr>
80 </table>
81
```

UpdateVulnerabilityForm.cshtml

```
1 @model Project.Models.VulnerabilityModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2>UpdateVulnerabilityForm</h2>
9
10
11 @using (Html.BeginForm("UpdateVulnerability", "vulnerabilities"))
12 {
13     @Html.AntiForgeryToken()
14
15     <div class="form-horizontal">
16         <br />
17         <hr />
18         <h4><b>Update Vulnerability</b></h4>
19         <hr />
20         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
21
22
23
24
25         <div class="form-group">
26             @Html.LabelFor(model => model.DateOpen, htmlAttributes: new { @class = "control-label col-md-2" })
27             <div class="col-md-10">
28                 @Html.EditorFor(model => model.DateOpen, new { htmlAttributes = new { @class = "form-control" } })
29             </div>
30         </div>
31
32         <div class="form-group">
33             @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
34             <div class="col-md-10">
35                 @Html.EditorFor(model => model.DateClosed, new { htmlAttributes = new { @class = "form-control" } })
36             </div>
37         </div>
38         <div class="form-group">
39             @Html.LabelFor(model => model.StepsToReproduce, htmlAttributes: new { @class = "control-label col-md-2" })
40             <div class="col-md-10">
41                 @Html.EditorFor(model => model.StepsToReproduce, new { htmlAttributes = new { @class = "form-control" } })
42             </div>
43         </div>
44
45         <div class="form-group">
46             @Html.LabelFor(model => model.Mitigation, htmlAttributes: new { @class = "control-label col-md-2" })
47             <div class="col-md-10">
48                 @Html.EditorFor(model => model.Mitigation, new { htmlAttributes = new { @class = "form-control" } })
49             </div>
50         </div>
51         <div class="form-group">
52             @Html.LabelFor(model => model.TicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
53             <div class="col-md-10">
54                 @Html.EditorFor(model => model.TicketNotes, new { htmlAttributes = new { @class = "form-control" } })
55             </div>
56         </div>
57         @{
58             String Role;
59             Role = Session["userRole"].ToString();
60             if (Role == "PenetrationTester")
61             {
62                 <div class="form-group">
63                     @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
64                     <div class="col-md-10">
65                         @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
66                     </div>
67                 </div>
68             }
69             else if (Role == "SecurityEngineer")
70             {
71                 <div class="form-group">
72                     @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
73                     <div class="col-md-10">
74                         @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusEngineer, "Please select")
75                     </div>
76                 </div>
77             }
78             else if (Role == "Administrator")
79             {
80             }
81         }
```

```

79     {
80         <div class="form-group">
81             @Html.LabelFor(model => model.StatusID, htmlAttributes: new { @class = "control-label col-md-2" })
82             <div class="col-md-10">
83                 @Html.DropDownListFor(m => m.StatusID, Model.ListVulnerabilityStatusPenTester, "Please select")
84             </div>
85         </div>
86     }
87 }
88 <div hidden class="form-group">
89     @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
90     <div hidden class="col-md-10">
91         @Html.EditorFor(model => model.TicketUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
92     </div>
93 </div>
94
95
96 <input type="hidden" name="VulnerabilityID" value="@Model.VulnerabilityID" />
97
98 <div class="form-group">
99     <div class="col-md-offset-2 col-md-10">
100         <input type="submit" value="Create" class="btn btn-default" />
101     </div>
102 </div>
103 </div>
104 </div>
105 }
106
107 <div>
108     @Html.ActionLink("Back to List", "Index")
109 </div>
110 <section Scripts >
111     <script src="/Scripts/jquery-1.10.2.min.js"></script>
112     <script type="text/javascript">
113         $(function () {
114
115
116             $("select").change(function () {
117                 var value = 0;
118                 if ($(this).val() != "") {
119                     value = $(this).val();
120                 }
121                 var id = $(this).attr("id");
122                 $.ajax({
123                     type: "post",
124                     url: "/Vulnerabilities/AjaxMethod",
125                     data: '{type: "' + id + '", value: ' + value + '\''}',
126                     contentType: "application/json; charset=utf-8",
127                     dataType: "json",
128                     success: function (response) {
129                         var dropdownId;
130                         var list;
131                         switch (id) {
132                             case "TacticID":
133                                 list = response.MitreTechnique;
134                                 DisableDropDown("#TechniqueID");
135
136                                 PopulateDropDown("#TechniqueID", list);
137                                 break;
138                         }
139                     }
140                 });
141             });
142         });
143     });
144
145     function DisableDropDown(dropdownId) {
146         $(dropdownId).attr("disabled", "disabled");
147         $(dropdownId).empty().append('<option selected="selected" value="0">Please select</option>');
148     }
149
150     function PopulateDropDown(dropdownId, list) {
151         if (list != null && list.length > 0) {
152             $(dropdownId).removeAttr("disabled");

```

```

152     $(dropDownId).removeAttr("disabled");
153     $.each(list, function () {
154         $(dropDownId).append($("<option></option>").val(this['Value']).html(this['Text']));
155     });
156     }
157 }
158 </script>
159 <script type="text/javascript">
160     function CalculateDamagePotentialTotal() {
161
162         var DamagePotential = document.getElementById('DamagePotential').value;
163         var DamageTotal = ((DamagePotential * 40) / 100);
164         //alert(DamageTotal);
165         $('#DamagePotential').val(DamageTotal);
166         var FinalVal = document.getElementById('DamagePotential').value;
167     }
168
169     function CalculateReproducibilityTotal() {
170         var Reproducibility = document.getElementById('Reproducibility').value;
171         var ReproducibilityTotal = ((Reproducibility * 5) / 100);
172         // alert(ReproducibilityTotal);
173         $('#Reproducibility').val(ReproducibilityTotal);
174         var FinalVal = document.getElementById('Reproducibility').value;
175     }
176
177
178     function CalculateExploitabilityTotal() {
179         var Exploitability = document.getElementById('Exploitability').value;
180         var ExploitabilityTotal = ((Exploitability * 10) / 100);
181         // alert(ExploitabilityTotal);
182         $('#Exploitability').val(ExploitabilityTotal);
183         var FinalVal = document.getElementById('Exploitability').value;
184     }
185
186
187
188     function CalculateAffectedUsersTotal() {
189         var AffectedUsers = document.getElementById('AffectedUsers').value;
190         var AffectedUsersTotal = ((AffectedUsers * 40) / 100);
191         // alert(AffectedUsersTotal);
192         $('#AffectedUsers').val(AffectedUsersTotal);
193         var FinalVal = document.getElementById('AffectedUsers').value;
194     }
195
196
197
198     function CalculateDiscoverabilityTotal() {
199         var Discoverability = document.getElementById('Discoverability').value;
200         var DiscoverabilityTotal = ((Discoverability * 5) / 100);
201         //alert(DiscoverabilityTotal);
202         $('#Discoverability').val(DiscoverabilityTotal);
203         var FinalVal = document.getElementById('Discoverability').value;
204     }
205
206
207
208     function CalculateDREADTotal() {
209         var DamagePotentialVal = document.getElementById('DamagePotential').value;
210         var ReproducibilityVal = document.getElementById('Reproducibility').value;
211         var ExploitabilityVal = document.getElementById('Exploitability').value;
212         var AffectedUsersVal = document.getElementById('AffectedUsers').value;
213         var DiscoverabilityVal = document.getElementById('Discoverability').value;
214
215         var DREADTotal = +DamagePotentialVal + +ReproducibilityVal + +ExploitabilityVal + +AffectedUsersVal + +DiscoverabilityVal;
216         $('#DREADTotal').val(DREADTotal);
217         var FinalVal = document.getElementById('DREADTotal').value;
218     }
219
220
221
222     function CalculateSeverity() {
223         var DREADTotal = document.getElementById('DREADTotal').value;
224         if (DREADTotal == 0.0) {
225             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

```
1 @model IEnumerable<Project.Models.VulnerabilityModel>
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2> </h2>
9 <br />
10 <hr />
11 <h4><b>Vulnerability Event Log</b></h4>
12 <hr />
13
14 <table class="table">
15     <tr>
16         <th>
17             @Html.DisplayNameFor(model => model.VulnerabilityLogID)
18         </th>
19         <th>
20             @Html.DisplayNameFor(model => model.VulnerabilityID)
21         </th>
22         <th>
23             @Html.DisplayNameFor(model => model.TicketUser)
24         </th>
25         <th>
26             @Html.DisplayNameFor(model => model.CurrentTimeStamp)
27         </th>
28         <th>
29             @Html.DisplayNameFor(model => model.TicketStatus)
30         </th>
31         <th>
32             @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
33         </th>
34     </tr>
35 </table>
36
37 @foreach (var item in Model)
38 {
39     <tr>
40         <td>
41             @Html.DisplayFor(modelItem => item.VulnerabilityLogID)
42         </td>
43         <td>
44             @Html.DisplayFor(modelItem => item.VulnerabilityID)
45         </td>
46         <td>
47             @Html.DisplayFor(modelItem => item.TicketUser)
48         </td>
49         <td>
50             @Html.DisplayFor(modelItem => item.CurrentTimeStamp)
51         </td>
52         <td>
53             @Html.DisplayFor(modelItem => item.TicketStatus)
54         </td>
55         <td>
56             @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
57         </td>
58     </tr>
59 }
60
61 </table>
62 @Html.ActionLink("Back to List", "Index")
```

VulnerabilityAssignmentsController.cs

```
1 using Project.Data;
2 using Project.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Data.SqlClient;
6 using System.Linq;
7 using System.Web;
8 using System.Web.Mvc;
9
10 namespace Project.Controllers
11 {
12     [CustomAuthorization]
13     public class VulnerabilityAssignmentController : Controller
14     {
15         private static string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;connect
16         Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
17         [CustomAuthorization]
18         public ActionResult Index()
19         {
20             List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
21             VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
22             vulnerabilityAssignments = vulnerabilityAssignmentDAO.FetchAll();
23             return View("Index", vulnerabilityAssignments);
24         }
25         [CustomAuthorization]
26         public ActionResult Details(int id)
27         {
28             VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
29             VulnerabilityAssignmentModel vulnerabilityAssignment = vulnerabilityAssignmentDAO.FetchOne(id);
30             return View("Details", vulnerabilityAssignment);
31         }
32         [CustomAuthorization]
33         public ActionResult create()
34         {
35             VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
36             vulnerabilityAssignment.ListVulnerabilityID = PopulateDropDown(" SELECT VulnerabilityID FROM dbo.Vulnerability", "VulnerabilityID", "VulnerabilityID");
37             vulnerabilityAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role = 'SecurityEngineer' ORDER BY Username", "Username", "Username");
38             vulnerabilityAssignment.ListVulnerabilityStatusPenTester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPenTester", "StatusType", "StatusType");
39             vulnerabilityAssignment.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
40             return View("CreateVulnerabilityAssignmentForm", vulnerabilityAssignment);
41         }
42     }
43     [CustomAuthorization]
44     private static List<SelectListItem> PopulateDropDown(string query, string textcolumn, string valuecolumn)
45     {
46         List<SelectListItem> items = new List<SelectListItem>();
47         using (SqlConnection connection = new SqlConnection(connectionString))
48         {
49             using (SqlCommand cmd = new SqlCommand(query))
50             {
51                 cmd.Connection = connection;
52                 connection.Open();
53                 using (SqlDataReader sdr = cmd.ExecuteReader())
54                 {
55                     while (sdr.Read())
56                     {
57                         items.Add(new SelectListItem
58                         {
59                             Text = sdr[textcolumn].ToString(),
60                             Value = sdr[valuecolumn].ToString()
61                         });
62                     }
63                 }
64                 connection.Close();
65             }
66         }
67         return items;
68     }
69 }
70 [CustomAuthorization]
```



```

70 [CustomAuthorization]
71 @references
72 public ActionResult CreateVulnerabilityAssignment(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
73 {
74     //save it to the database
75     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
76     vulnerabilityAssignmentDAO.Create(vulnerabilityAssignmentModel);
77     List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
78     //How to return full list of assignments instead of the one
79     vulnerabilityAssignments = vulnerabilityAssignmentDAO.FetchAll();
80     return View("Index", vulnerabilityAssignments);
81 }
82 [CustomAuthorization]
83 @references
84 public ActionResult Edit(int id)
85 {
86     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
87     VulnerabilityAssignmentModel vulnerabilityAssignment = vulnerabilityAssignmentDAO.FetchOne(id);
88     vulnerabilityAssignment.ListVulnerabilityID = PopulateDropDown(" SELECT VulnerabilityID FROM dbo.Vulnerability", "VulnerabilityID", "VulnerabilityID");
89     vulnerabilityAssignment.ListTicketUser = PopulateDropDown(" SELECT Username, Role FROM dbo.Users WHERE Role = 'SecurityEngineer' ORDER BY Username ", "Username", "Username");
90     vulnerabilityAssignment.ListVulnerabilityStatusPenTester = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusPenTester", "StatusType", "StatusType");
91     vulnerabilityAssignment.ListVulnerabilityStatusEngineer = PopulateDropDown(" SELECT StatusType FROM dbo.VulnerabilityStatusEngineer", "StatusType", "StatusType");
92     return View("UpdateVulnerabilityAssignmentForm", vulnerabilityAssignment);
93 }
94 [CustomAuthorization]
95 [HttpPost]
96 @references
97 public ActionResult UpdateVulnerabilityAssignment(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
98 {
99     //save it to the database
100     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
101     vulnerabilityAssignmentDAO.Update(vulnerabilityAssignmentModel);
102     return View("Details", vulnerabilityAssignmentModel);
103 }
104 [CustomAuthorization]
105 @references
106 public ActionResult Delete(int id)
107 {
108     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
109     vulnerabilityAssignmentDAO.Delete(id);
110     List<VulnerabilityAssignmentModel> vulnerabilityAssignments = vulnerabilityAssignmentDAO.FetchAll();
111     return View("Index", vulnerabilityAssignments);
112 }
113 [CustomAuthorization]
114 @references
115 public ActionResult VulnerabilityAssignmentsEngineer(string session)
116 {
117     List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
118     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
119     vulnerabilityAssignments = vulnerabilityAssignmentDAO.GetVulnerabilityAssignmentsEngineer(session);
120     return View("VulnerabilityAssignmentsEngineer", vulnerabilityAssignments);
121 }
122 [CustomAuthorization]
123 @references
124 public ActionResult TestAssignmentsPenTester(string session)
125 {
126     List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
127     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
128     vulnerabilityAssignments = vulnerabilityAssignmentDAO.GetTestAssignmentsPenTester(session);
129     return View("TestAssignmentsPenTester", vulnerabilityAssignments);
130 }
131 [CustomAuthorization]
132 @references
133 public ActionResult VulnerabilityAssignmentsPenTester()
134 {
135     List<VulnerabilityAssignmentModel> vulnerabilityAssignments = new List<VulnerabilityAssignmentModel>();
136     VulnerabilityAssignmentDAO vulnerabilityAssignmentDAO = new VulnerabilityAssignmentDAO();
137     vulnerabilityAssignments = vulnerabilityAssignmentDAO.GetVulnerabilityAssignmentsPenTester();
138     return View("VulnerabilityAssignmentsPenTester", vulnerabilityAssignments);
139 }

```

VulnerabilityAssignmentDAO.cs

```
1 using Project.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Data.SqlClient;
5
6 namespace Project.Data
7 {
8     20 references
9     internal class VulnerabilityAssignmentDAO
10    {
11        private string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
12        Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
13
14        4 references
15        public List<VulnerabilityAssignmentModel> FetchAll()
16        {
17            List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
18            using (SqlConnection connection = new SqlConnection(connectionString))
19            {
20                string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment";
21
22                SqlCommand command = new SqlCommand(sqlQuery, connection);
23
24                connection.Open();
25                SqlDataReader reader = command.ExecuteReader();
26                if (reader.HasRows)
27                {
28                    while (reader.Read())
29                    {
30                        //creating new TestAssignment object and adding it to the list to be returned
31                        VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
32                        vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
33                        vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
34                        vulnerabilityAssignment.TicketUser = reader.GetString(2);
35                        vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
36                        vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
37                        vulnerabilityAssignment.TicketStatus = reader.GetString(5);
38                        vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
39                        returnList.Add(vulnerabilityAssignment);
40                    }
41                }
42            }
43
44            return returnList;
45        }
46
47        2 references
48        public VulnerabilityAssignmentModel FetchOne(int id)
49        {
50            //access database
51            using (SqlConnection connection = new SqlConnection(connectionString))
52            {
53                string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE VulnerabilityAssignmentID = @id";
54
55                SqlCommand command = new SqlCommand(sqlQuery, connection);
56                command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
57                connection.Open();
58                SqlDataReader reader = command.ExecuteReader();
59
60                VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
61                if (reader.HasRows)
62                {
63                    while (reader.Read())
64                    {
65                        //create new test object and add that to the list to be returned
66
67                        vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
68                        vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
69                        vulnerabilityAssignment.TicketUser = reader.GetString(2);
70                        vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
71                        vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
72                        vulnerabilityAssignment.TicketStatus = reader.GetString(5);
73                        vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
74                    }
75                }
76            }
77
78            return vulnerabilityAssignment;
79        }
80    }
81 }
```

```

72     }
73     }
74     }
75     }
76     }
77     return vulnerabilityAssignment;
78 }
79 }
80 }
81 }
82
83 1 reference
84 internal int Delete(int id)
85 {
86     using (SqlConnection connection = new SqlConnection(connectionString))
87     {
88         //prepared statement
89         //update
90         string sqlQuery = "DELETE FROM dbo.VulnerabilityAssignment WHERE VulnerabilityAssignmentID = @id";
91
92         //associate @id with Id parameter
93         SqlCommand command = new SqlCommand(sqlQuery, connection);
94
95         command.Parameters.Add("@id", System.Data.SqlDbType.VarChar, 1000).Value = id;
96
97         connection.Open();
98         int deletedID = command.ExecuteNonQuery();
99
100        return deletedID;
101    }
102 }
103
104 1 reference
105 public int Create(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
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;
142     }
143 }

```

```

141     command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
142     command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
143     command2.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
144
145
146     connection.Open();
147     int newVulnerabilityAssignmentID = command.ExecuteNonQuery();
148     command1.ExecuteNonQuery();
149     command2.ExecuteNonQuery();
150     return newVulnerabilityAssignmentID;
151 }
152 }
153
154 1 reference
155 public int Update(VulnerabilityAssignmentModel vulnerabilityAssignmentModel)
156 {
157     //access the database
158     using (SqlConnection connection = new SqlConnection(connectionString))
159     {
160         //prepared statement
161         //update
162         string sqlQuery = "UPDATE dbo.VulnerabilityAssignment SET VulnerabilityID = @VulnerabilityID, TicketUser = @TicketUser, DateAssigned = @DateAssigned, DateClosed = @DateClosed, TicketStatus = @TicketStatus, VulnerabilityTicketNotes = @VulnerabilityTicketNotes WHERE VulnerabilityAssignmentID = @VulnerabilityAssignmentID";
163
164         //associate @id with Id parameter
165
166         SqlCommand command = new SqlCommand(sqlQuery, connection);
167
168         command.Parameters.Add("@VulnerabilityAssignmentID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityAssignmentID;
169         command.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityID;
170         command.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketUser;
171         command.Parameters.Add("@DateAssigned", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateAssigned;
172         command.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateClosed;
173         command.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
174         command.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
175
176         //Update Vulnerability
177         string sqlQuery1 = "Update dbo.Vulnerability SET DateOpen = @DateOpen, DateClosed = @DateClosed, StatusID = @StatusID, TicketNotes = @TicketNotes WHERE dbo.Vulnerability.VulnerabilityID = @VulnerabilityID";
178
179         //associate @id with Id parameter
180         SqlCommand command1 = new SqlCommand(sqlQuery1, connection);
181
182         command1.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityAssignmentModel.VulnerabilityID;
183         command1.Parameters.Add("@DateClosed", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateClosed;
184         command1.Parameters.Add("@DateOpen", System.Data.SqlDbType.DateTime).Value = vulnerabilityAssignmentModel.DateAssigned;
185         command1.Parameters.Add("@StatusID", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
186         command1.Parameters.Add("@TicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
187         command1.Parameters.Add("@AssignedUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketUser;
188
189         //Insert Edit To Vulnerability Log
190         string sqlQuery2 = "INSERT INTO dbo.VulnerabilityLog VALUES(@VulnerabilityID, @TicketUser, @CurrentTimeStamp, @TicketStatus, @VulnerabilityTicketNotes)";
191
192         //associate @id with Id parameter
193         SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
194
195         command2.Parameters.Add("@VulnerabilityID", System.Data.SqlDbType.Int).Value = vulnerabilityAssignmentModel.VulnerabilityID;
196         command2.Parameters.Add("@TicketUser", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.CurrentUser;
197         command2.Parameters.Add("@CurrentTimeStamp", System.Data.SqlDbType.DateTime).Value = DateTime.Now;
198         command2.Parameters.Add("@TicketStatus", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.TicketStatus;
199         command2.Parameters.Add("@VulnerabilityTicketNotes", System.Data.SqlDbType.VarChar, 1000).Value = vulnerabilityAssignmentModel.VulnerabilityTicketNotes;
200
201         connection.Open();
202         int newVulnerabilityAssignmentID = command.ExecuteNonQuery();
203         command1.ExecuteNonQuery();
204         command2.ExecuteNonQuery();
205         return newVulnerabilityAssignmentID;
206     }
207 }
208
209 1 reference
210 public List<VulnerabilityAssignmentModel> GetVulnerabilityAssignmentsEngineer(string session)

```

```

209 1 reference
210 public List<VulnerabilityAssignmentModel> GetVulnerabilityAssignmentsEngineer(string session)
211 {
212     List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
213     using (SqlConnection connection = new SqlConnection(connectionString))
214     {
215         string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketUser = @session AND TicketStatus = 'Open'";
216
217         SqlCommand command = new SqlCommand(sqlQuery, connection);
218         command.Parameters.Add("@session", System.Data.SqlDbType.VarChar).Value = session;
219
220         connection.Open();
221         SqlDataReader reader = command.ExecuteReader();
222         if (reader.HasRows)
223         {
224             while (reader.Read())
225             {
226                 //creating new TestAssignment object and adding it to the list to be returned
227                 VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
228                 vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
229                 vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
230                 vulnerabilityAssignment.TicketUser = reader.GetString(2);
231                 vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
232                 vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
233                 vulnerabilityAssignment.TicketStatus = reader.GetString(5);
234                 vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
235                 returnList.Add(vulnerabilityAssignment);
236             }
237         }
238     }
239     return returnList;
240 }
241
242 1 reference
243 public List<VulnerabilityAssignmentModel> GetVulnerabilityAssignmentsPenTester()
244 {
245     List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
246     using (SqlConnection connection = new SqlConnection(connectionString))
247     {
248         string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketStatus = 'Re-Test'";
249
250         SqlCommand command = new SqlCommand(sqlQuery, connection);
251
252         connection.Open();
253         SqlDataReader reader = command.ExecuteReader();
254         if (reader.HasRows)
255         {
256             while (reader.Read())
257             {
258                 //creating new TestAssignment object and adding it to the list to be returned
259                 VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
260                 vulnerabilityAssignment.VulnerabilityAssignmentID = reader.GetInt32(0);
261                 vulnerabilityAssignment.VulnerabilityID = reader.GetInt32(1);
262                 vulnerabilityAssignment.TicketUser = reader.GetString(2);
263                 vulnerabilityAssignment.DateAssigned = reader.GetDateTime(3);
264                 vulnerabilityAssignment.DateClosed = reader.GetDateTime(4);
265                 vulnerabilityAssignment.TicketStatus = reader.GetString(5);
266                 vulnerabilityAssignment.VulnerabilityTicketNotes = reader.GetString(6);
267                 returnList.Add(vulnerabilityAssignment);
268             }
269         }
270     }
271     return returnList;
272 }
273
274 1 reference
275 public List<VulnerabilityAssignmentModel> GetTestAssignmentsPenTester(string session)
276 {
277     List<VulnerabilityAssignmentModel> returnList = new List<VulnerabilityAssignmentModel>();
278     using (SqlConnection connection = new SqlConnection(connectionString))
279     {
280         string sqlQuery2 = "SELECT TestAssignmentID, TestName, DateAssigned, DateClosed, TicketStatus, TestTicketNotes FROM dbo.TestAssignment WHERE TicketStatus = 'Open' AND
281         TicketUser = @session";
282
283         SqlCommand command2 = new SqlCommand(sqlQuery2, connection);
284         command2.Parameters.Add("@session", System.Data.SqlDbType.VarChar).Value = session;
285         connection.Open();
286         SqlDataReader reader2 = command2.ExecuteReader();
287         if (reader2.HasRows)
288         {
289             while (reader2.Read())
290             {
291                 //creating new TestAssignment object and adding it to the list to be returned
292                 VulnerabilityAssignmentModel vulnerabilityAssignment = new VulnerabilityAssignmentModel();
293                 vulnerabilityAssignment.TestAssignmentID = reader2.GetInt32(0);
294                 vulnerabilityAssignment.TestName1 = reader2.GetString(1);
295                 vulnerabilityAssignment.DateAssigned1 = reader2.GetDateTime(2);
296                 vulnerabilityAssignment.DateClosed1 = reader2.GetDateTime(3);
297                 vulnerabilityAssignment.TicketStatus1 = reader2.GetString(4);
298                 vulnerabilityAssignment.TestTicketNotes1 = reader2.GetString(5);
299                 returnList.Add(vulnerabilityAssignment);
300             }
301         }
302     }
303     return returnList;
304 }
305 }
306

```

VulnerabilityAssignmentModel.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Web;
6 using System.Web.Mvc;
7
8 namespace Project.Models
9 {
10     public class VulnerabilityAssignmentModel
11     {
12         [Display(Name = "Vulnerability Assignment ID")]
13         public int VulnerabilityAssignmentID { get; set; }
14         [Display(Name = "Test Assignment ID")]
15         public int TestAssignmentID { get; set; }
16         [Display(Name = "Vulnerability ID")]
17         public int VulnerabilityID { get; set; }
18         [Display(Name = "Ticket User")]
19         public string TicketUser { get; set; }
20         [Display(Name = "Current User")]
21         public string CurrentUser { get; set; }
22         [Display(Name = "Test Name")]
23         public string TestName { get; set; }
24         [Display(Name = "Severity")]
25         public string Severity { get; set; }
26         [Display(Name = "Test Name")]
27         public string TestName1 { get; set; }
28         [Display(Name = "Date Assigned")]
29         public DateTime DateAssigned1 { get; set; }
30         [Display(Name = "Date Closed")]
31         public DateTime DateClosed1 { get; set; }
32         [Display(Name = "Ticket Notes")]
33         public string TicketNotes { get; set; }
34         [Display(Name = "Ticket Notes")]
35         public string TestTicketNotes { get; set; }
36         [Display(Name = "Ticket Status")]
37         public string TicketStatus1 { get; set; }
38         [Display(Name = "Ticket Notes")]
39         public string TestTicketNotes1 { get; set; }
40         [Display(Name = "Status")]
41         public string StatusID { get; set; }
42         [Display(Name = "Date Assigned")]
43         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
44         public DateTime DateAssigned { get; set; }
45         [Display(Name = "Date Open")]
46         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
47         public DateTime DateOpen { get; set; }
48         [Display(Name = "Date Closed")]
49         [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MMM/yyyy}")]
50         public DateTime DateClosed { get; set; }
51         [Display(Name = "Ticket Status")]
52         public string TicketStatus { get; set; }
53         [Display(Name = "Ticket Notes")]
54         public string vulnerabilityTicketNotes { get; set; }
55
56         [3 references]
57         public List<SelectListItem> ListVulnerabilityID { get; set; }
58         [3 references]
59         public List<SelectListItem> ListTicketUser { get; set; }
60
61         [2 references]
62         public List<SelectListItem> ListTicketUser { get; set; }
63         [3 references]
64         public List<SelectListItem> ListVulnerabilityStatusPenTester { get; set; }
65         [3 references]
66         public List<SelectListItem> ListVulnerabilityStatusEngineer { get; set; }
67
68         [0 references]
69         public VulnerabilityAssignmentModel(int vulnerabilityAssignmentID, int vulnerabilityID, string ticketUser, DateTime dateAssigned, DateTime dateClosed, string ticketStatus, string vulnerabilityTicketNotes)
70         {
71             VulnerabilityAssignmentID = vulnerabilityAssignmentID;
72             vulnerabilityID = vulnerabilityID;
73             TicketUser = ticketUser;
74             DateAssigned = dateAssigned;
75             DateClosed = dateClosed;
76             TicketStatus = ticketStatus;
77             VulnerabilityTicketNotes = vulnerabilityTicketNotes;
78         }
79
80         [8 references]
81         public VulnerabilityAssignmentModel()
82         {
83             this.ListVulnerabilityID = new List<SelectListItem>();
84             this.ListTicketUser = new List<SelectListItem>();
85             this.ListVulnerabilityStatusPenTester = new List<SelectListItem>();
86             this.ListVulnerabilityStatusEngineer = new List<SelectListItem>();
87         }
88     }
89 }
```

VulnerabilityAssignment Views

CreateVulnerabilityAssignmentForm.cshtml

```
1 @model Project.Models.VulnerabilityAssignmentModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2> </h2>
9
10
11 @using (Html.BeginForm("CreateVulnerabilityAssignment", "VulnerabilityAssignment"))
12 {
13     @Html.AntiForgeryToken()
14
15     <div class="form-horizontal">
16         <br />
17         <hr />
18         <h4><b>Assign Vulnerability</b></h4>
19         <hr />
20         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
21         <div hidden class="form-group">
22             @Html.LabelFor(model => model.VulnerabilityAssignmentID, htmlAttributes: new { @class = "control-label col-md-2" })
23             <div hidden class="col-md-10">
24                 @Html.EditorFor(model => model.VulnerabilityAssignmentID, new { htmlAttributes = new { @class = "form-control" } })
25                 @Html.ValidationMessageFor(model => model.VulnerabilityAssignmentID, "", new { @class = "text-danger" })
26             </div>
27         </div>
28         <div class="form-group">
29             @Html.LabelFor(model => model.VulnerabilityID, htmlAttributes: new { @class = "control-label col-md-2" })
30             <div class="col-md-10">
31                 @Html.DropDownListFor(m => m.VulnerabilityID, Model.ListVulnerabilityID, "Please select")
32             </div>
33         </div>
34
35         <div class="form-group">
36             @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
37             <div class="col-md-10">
38                 @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
39             </div>
40         </div>
41         <div hidden class="form-group">
42             @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
43             <div hidden class="col-md-10">
44                 @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
45             </div>
46         </div>
47
48         <div class="form-group">
49             @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
50             <div class="col-md-10">
51                 @Html.TextBoxFor(model => model.DateAssigned, new { type = "date" })
52             </div>
53         </div>
54
55         <div class="form-group">
56             @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
57             <div class="col-md-10">
58                 @Html.TextBoxFor(model => model.DateClosed, new { type = "date" })
59             </div>
60         </div>
61         <div class="form-group">
62             @Html.LabelFor(model => model.VulnerabilityTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
63             <div class="col-md-10">
64                 @Html.EditorFor(model => model.VulnerabilityTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
65                 @Html.ValidationMessageFor(model => model.VulnerabilityTicketNotes, "", new { @class = "text-danger" })
66             </div>
67         </div>
68     }
69     String Role;
70     Role = Session["userRole"].ToString();
71     if (Role == "PenetrationTester")
72     {
73         <div class="form-group">
74             @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
75             <div class="col-md-10">
76                 @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
77             </div>
78         </div>
79     }
80 }
```

```

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

```
1 @model Project.Models.VulnerabilityAssignmentModel
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <h2></h2>
9
10 <div>
11     <br />
12     <hr />
13     <h4><b>Assigned Vulnerability Information</b></h4>
14     <hr />
15     <dl class="dl-horizontal">
16         <dt>
17             @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
18         </dt>
19         <dd>
20             @Html.DisplayFor(model => model.VulnerabilityAssignmentID)
21         </dd>
22     </dl>
23     <dl>
24         <dt>
25             @Html.DisplayNameFor(model => model.VulnerabilityID)
26         </dt>
27         <dd>
28             @Html.DisplayFor(model => model.VulnerabilityID)
29         </dd>
30     </dl>
31     <dl>
32         <dt>
33             @Html.DisplayNameFor(model => model.TicketUser)
34         </dt>
35         <dd>
36             @Html.DisplayFor(model => model.TicketUser)
37         </dd>
38     </dl>
39     <dl>
40         <dt>
41             @Html.DisplayNameFor(model => model.DateAssigned)
42         </dt>
43         <dd>
44             @Html.DisplayFor(model => model.DateAssigned)
45         </dd>
46     </dl>
47     <dl>
48         <dt>
49             @Html.DisplayNameFor(model => model.DateClosed)
50         </dt>
51         <dd>
52             @Html.DisplayFor(model => model.DateClosed)
53         </dd>
54     </dl>
55     <dl>
56         <dt>
57             @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
58         </dt>
59         <dd>
60             @Html.DisplayFor(model => model.VulnerabilityTicketNotes)
61         </dd>
62     </dl>
63     <dl>
64         <dt>
65             @Html.DisplayNameFor(model => model.TicketStatus)
66         </dt>
67         <dd>
68             @Html.DisplayFor(model => model.TicketStatus)
69         </dd>
70     </dl>
71 </div>
72 <p>
73     @Html.ActionLink("Edit", "Edit", new { id = Model.VulnerabilityAssignmentID }) |
74     @Html.ActionLink("Back to List", "Index")
75 </p>
76
```

Index.cshtml

```
1 @model IEnumerable<Project.Models.VulnerabilityAssignmentModel>
2
3
4 @{
5     ViewBag.Title = "";
6     Layout = "~/Views/Shared/_Layout.cshtml";
7 }
8 <h2></h2>
9
10 <p>
11     @{
12         String Role;
13         Role = Session["UserRole"].ToString();
14         if (Role == "Administrator")
15         {
16             <br />
17             <br />
18             <h4><b>Assigned Vulnerabilities</b></h4>
19             <br />
20             @Html.ActionLink("Create New", "Create")
21         }
22         else if (Role == "PenetrationTester")
23         {
24             <br />
25             <br />
26             <h4><b>Assigned Vulnerabilities</b></h4>
27             <br />
28             @Html.ActionLink("Create New", "Create")
29         }
30         else if (Role == "Engineer")
31         {
32             Response.Redirect("~/Home/Index");
33         }
34     }
35 </p>
36
37 <table class="table">
38     <tr>
39         <th>
40             @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
41         </th>
42         <th>
43             @Html.DisplayNameFor(model => model.VulnerabilityID)
44         </th>
45         <th>
46             @Html.DisplayNameFor(model => model.TicketUser)
47         </th>
48         <th>
49             @Html.DisplayNameFor(model => model.DateAssigned)
50         </th>
51         <th>
52             @Html.DisplayNameFor(model => model.DateClosed)
53         </th>
54         <th>
55             @Html.DisplayNameFor(model => model.TicketStatus)
56         </th>
57         <th>
58             @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
59         </th>
60     </tr>
61
62     <tr>
63         @foreach (var item in Model)
64         {
65             <tr>
66                 <td>
67                     @Html.DisplayFor(modelItem => item.VulnerabilityAssignmentID)
68                 </td>
69                 <td>
70                     @Html.DisplayFor(modelItem => item.VulnerabilityID)
71                 </td>
72                 <td>
73                     @Html.DisplayFor(modelItem => item.TicketUser)
74                 </td>
75                 <td>
76                     @Html.DisplayFor(modelItem => item.DateAssigned)
77                 </td>
78                 <td>
79                     @Html.DisplayFor(modelItem => item.DateClosed)
80                 </td>
81                 <td>
82                     @Html.DisplayFor(modelItem => item.TicketStatus)
83                 </td>
84                 <td>
85                     @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
86                 </td>
87                 <td>
88                     @Html.ActionLink("Edit", "Edit", new { id = item.VulnerabilityAssignmentID }, new { session = Session["Username"] }) |
89                     @Html.ActionLink("Details", "Details", new { id = item.VulnerabilityAssignmentID }, new { session = Session["Username"] }) |
90                     @Html.ActionLink("Delete", "Delete", new { id = item.VulnerabilityAssignmentID }, new { session = Session["Username"] })
91                 </td>
92             </tr>
93         }
94     </table>
95
96
```

TestAssignmentsPenTester.cshtml

```
1 @model IEnumerable<Project.Models.VulnerabilityAssignmentModel>
2
3 @{
4     ViewBag.Title = "";
5     Layout = "~/Views/Shared/_Layout.cshtml";
6 }
7
8 <br />
9 <hr />
10 <h4><b></b></h4>
11 <hr />
12 <table class="table">
13     <tr>
14         <th>
15             @Html.DisplayNameFor(model => model.TestAssignmentID)
16         </th>
17         <th>
18             @Html.DisplayNameFor(model => model.TestName1)
19         </th>
20         <th>
21             @Html.DisplayNameFor(model => model.DateAssigned1)
22         </th>
23         <th>
24             @Html.DisplayNameFor(model => model.DateClosed1)
25         </th>
26         <th>
27             @Html.DisplayNameFor(model => model.TestTicketNotes1)
28         </th>
29         <th>
30             @Html.DisplayNameFor(model => model.TicketStatus1)
31         </th>
32     </tr>
33     <tr>
34     </tr>
35
36     @foreach (var item in Model)
37     {
38     <tr>
39     <td>
40         @Html.DisplayFor(modelItem => item.TestAssignmentID)
41     </td>
42     <td>
43         @Html.DisplayFor(modelItem => item.TestName1)
44     </td>
45     <td>
46         @Html.DisplayFor(modelItem => item.DateAssigned1)
47     </td>
48     <td>
49         @Html.DisplayFor(modelItem => item.DateClosed1)
50     </td>
51     <td>
52         @Html.DisplayFor(modelItem => item.TestTicketNotes1)
53     </td>
54     <td>
55         @Html.DisplayFor(modelItem => item.TicketStatus1)
56     </td>
57     </tr>
58     </tr>
59     }
60
61 </table>
62
```

UpdateVulnerabilityAssignment.cshtml

```
1 @model Project.Models.VulnerabilityAssignmentModel
2
3 @f
4 ViewBag.Title = "";
5 Layout = "~/Views/Shared/_Layout.cshtml";
6
7
8 <h2> </h2>
9
10
11 @using (Html.BeginForm("UpdateVulnerabilityAssignment", "VulnerabilityAssignment"))
12 {
13     @Html.AntiForgeryToken()
14
15     <div class="form-horizontal">
16         <br />
17         <hr />
18         <h4><b>Update Vulnerability Assignment</b></h4>
19         <hr />
20         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
21         <div hidden class="form-group">
22             @Html.LabelFor(model => model.VulnerabilityAssignmentID, htmlAttributes: new { @class = "control-label col-md-2" })
23             <div hidden class="col-md-10">
24                 @Html.EditorFor(model => model.VulnerabilityAssignmentID, new { htmlAttributes = new { @class = "form-control" } })
25                 @Html.ValidationMessageFor(model => model.VulnerabilityAssignmentID, "", new { @class = "text-danger" })
26             </div>
27         </div>
28
29         <div class="form-group">
30             @Html.LabelFor(model => model.VulnerabilityID, htmlAttributes: new { @class = "control-label col-md-2" })
31             <div class="col-md-10">
32                 @Html.DropDownListFor(m => m.VulnerabilityID, Model.ListVulnerabilityID, "Please select")
33             </div>
34         </div>
35
36         <div class="form-group">
37             @Html.LabelFor(model => model.TicketUser, htmlAttributes: new { @class = "control-label col-md-2" })
38             <div class="col-md-10">
39                 @Html.DropDownListFor(m => m.TicketUser, Model.ListTicketUser, "Please select")
40             </div>
41         </div>
42
43         <div hidden class="form-group">
44             @Html.LabelFor(model => model.CurrentUser, htmlAttributes: new { @class = "control-label col-md-2" })
45             <div hidden class="col-md-10">
46                 @Html.EditorFor(model => model.CurrentUser, new { htmlAttributes = new { @class = "form-control", Value = Session["username"] } })
47             </div>
48         </div>
49         <div class="form-group">
50             @Html.LabelFor(model => model.DateAssigned, htmlAttributes: new { @class = "control-label col-md-2" })
51             <div class="col-md-10">
52                 @Html.EditorFor(model => model.DateAssigned, new { htmlAttributes = new { @class = "form-control" } })
53             </div>
54         </div>
55
56         <div class="form-group">
57             @Html.LabelFor(model => model.DateClosed, htmlAttributes: new { @class = "control-label col-md-2" })
58             <div class="col-md-10">
59                 @Html.EditorFor(model => model.DateClosed, new { htmlAttributes = new { @class = "form-control" } })
60             </div>
61         </div>
62
63         <div class="form-group">
64             @Html.LabelFor(model => model.VulnerabilityTicketNotes, htmlAttributes: new { @class = "control-label col-md-2" })
65             <div class="col-md-10">
66                 @Html.EditorFor(model => model.VulnerabilityTicketNotes, new { htmlAttributes = new { @class = "form-control" } })
67                 @Html.ValidationMessageFor(model => model.VulnerabilityTicketNotes, "", new { @class = "text-danger" })
68             </div>
69         </div>
70     </div>
71     @f
72     String Role;
73     Role = Session["userRole"].ToString();
74     if (Role == "PenetrationTester")
75     {
76         <div class="form-group">
77             @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
78             <div class="col-md-10">
```

```

77     @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
78     <div class="col-md-10">
79         @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
80     </div>
81 </div>
82 }
83 else if (Role == "SecurityEngineer")
84 {
85     <div class="form-group">
86         @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
87         <div class="col-md-10">
88             @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusEngineer, "Please select")
89         </div>
90     </div>
91     Response.Redirect("~/Home/Index");
92 }
93 else if (Role == "Administrator")
94 {
95     <div class="form-group">
96         @Html.LabelFor(model => model.TicketStatus, htmlAttributes: new { @class = "control-label col-md-2" })
97         <div class="col-md-10">
98             @Html.DropDownListFor(m => m.TicketStatus, Model.ListVulnerabilityStatusPenTester, "Please select")
99         </div>
100     </div>
101 }
102 }
103
104
105 <div class="form-group">
106     <div class="col-md-offset-2 col-md-10">
107         <input type="submit" value="Create" class="btn btn-default" />
108     </div>
109 </div>
110 </div>
111 }
112
113 <div>
114     @Html.ActionLink("Back to List", "Index")
115 </div>
116
117 @section Scripts {
118     @Scripts.Render("~/bundles/jqueryval")
119 }
120

```

VulnerabilityAssignmentsEngineer.cshtml

```
1  @model IEnumerable<Project.Models.VulnerabilityAssignmentModel>
2
3  @{
4  ViewBag.Title = "";
5  Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <br />
9  <hr />
10 <h4><b>Tasks</b></h4>
11 <hr />
12 <table class="table">
13 <tr>
14 <th>
15     @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
16 </th>
17 <th>
18     @Html.DisplayNameFor(model => model.VulnerabilityID)
19 </th>
20 <th>
21     @Html.DisplayNameFor(model => model.TicketUser)
22 </th>
23 <th>
24     @Html.DisplayNameFor(model => model.DateAssigned)
25 </th>
26 <th>
27     @Html.DisplayNameFor(model => model.DateClosed)
28 </th>
29 <th>
30     @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
31 </th>
32 <th>
33     @Html.DisplayNameFor(model => model.TicketStatus)
34 </th>
35 <th></th>
36 </tr>
37
38 @foreach (var item in Model)
39 {
40 <tr>
41 <td>
42     @Html.DisplayFor(modelItem => item.VulnerabilityAssignmentID)
43 </td>
44 <td>
45     @Html.DisplayFor(modelItem => item.VulnerabilityID)
46 </td>
47 <td>
48     @Html.DisplayFor(modelItem => item.TicketUser)
49 </td>
50 <td>
51     @Html.DisplayFor(modelItem => item.DateAssigned)
52 </td>
53 <td>
54     @Html.DisplayFor(modelItem => item.DateClosed)
55 </td>
56 <td>
57     @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
58 </td>
59 <td>
60     @Html.DisplayFor(modelItem => item.TicketStatus)
61 </td>
62 </tr>
63 }
64
65 </table>
66
```

VulnerabilityAssignmentsPenTester.cshtml

```
1  @model IEnumerable<Project.Models.VulnerabilityAssignmentModel>
2
3  @{
4      ViewBag.Title = "";
5      Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <br />
9  <hr />
10 <h4><b>Vulnerabilities Ready For Re-Test</b></h4>
11 <hr />
12 <table class="table">
13     <tr>
14         <th>
15             @Html.DisplayNameFor(model => model.VulnerabilityAssignmentID)
16         </th>
17         <th>
18             @Html.DisplayNameFor(model => model.VulnerabilityID)
19         </th>
20         <th>
21             @Html.DisplayNameFor(model => model.TicketUser)
22         </th>
23         <th>
24             @Html.DisplayNameFor(model => model.DateAssigned)
25         </th>
26         <th>
27             @Html.DisplayNameFor(model => model.DateClosed)
28         </th>
29         <th>
30             @Html.DisplayNameFor(model => model.VulnerabilityTicketNotes)
31         </th>
32         <th>
33             @Html.DisplayNameFor(model => model.TicketStatus)
34         </th>
35     </th></th>
36 </tr>
37
38     @foreach (var item in Model)
39     {
40     <tr>
41         <td>
42             @Html.DisplayFor(modelItem => item.VulnerabilityAssignmentID)
43         </td>
44         <td>
45             @Html.DisplayFor(modelItem => item.VulnerabilityID)
46         </td>
47         <td>
48             @Html.DisplayFor(modelItem => item.TicketUser)
49         </td>
50         <td>
51             @Html.DisplayFor(modelItem => item.DateAssigned)
52         </td>
53         <td>
54             @Html.DisplayFor(modelItem => item.DateClosed)
55         </td>
56         <td>
57             @Html.DisplayFor(modelItem => item.VulnerabilityTicketNotes)
58         </td>
59         <td>
60             @Html.DisplayFor(modelItem => item.TicketStatus)
61         </td>
62     </tr>
63     }
64 </table>
65
66
```

Layout.cshtml

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>@ViewBag.Title - Vulnerability Management</title>
7 @Styles.Render("~/Content/css")
8 @Scripts.Render("~/bundles/modernizr")
9 </head>
10 <body>
11 <div class="navbar navbar-inverse navbar-fixed-top">
12 <div class="container">
13 <div class="navbar-header">
14 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
15 <span class="icon-bar"></span>
16 <span class="icon-bar"></span>
17 <span class="icon-bar"></span>
18 </button>
19 @Html.ActionLink("Vulnerability Management", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
20 </div>
21 <div class="navbar-collapse collapse">
22 <ul class="nav navbar-nav">
23 @if (Session["userRole"] != null)
24 {
25     String Role;
26     Role = Session["userRole"].ToString();
27     if (Role == "Administrator")
28     {
29         <li>@Html.ActionLink("Home", "Index", "Home")</li>
30         <li>@Html.ActionLink("Assign Test", "Index", "TestAssignment")</li>
31         <li>@Html.ActionLink("Tests", "Index", "Tests")</li>
32         <li>@Html.ActionLink("Assign Vulnerability", "Index", "VulnerabilityAssignment")</li>
33         <li>@Html.ActionLink("Vulnerabilities", "Index", "Vulnerabilities")</li>
34         <li>@Html.ActionLink("Assets", "Index", "Assets")</li>
35         if (Session["userRole"] != null)
36         {
37             <li> <a href="#"> @Session["username"]</a></li>
38         }
39         else
40         {
41             <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
42         }
43         <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>
44     }
45     else if (Role == "PenetrationTester")
46     {
47         <li>@Html.ActionLink("Home", "Index", "Home")</li>
48         <li>@Html.ActionLink("Tasks", "TestAssignmentsPenTester", "VulnerabilityAssignment", new { session = Session["Username"] }, null)</li>
49         <li>@Html.ActionLink("Ready For Re-Test", "VulnerabilityAssignmentsPenTester", "VulnerabilityAssignment")</li>
50         <li>@Html.ActionLink("Assign Test", "Index", "TestAssignment")</li>
51         <li>@Html.ActionLink("Tests", "Index", "Tests")</li>
52         <li>@Html.ActionLink("Assign Vulnerability", "Index", "VulnerabilityAssignment")</li>
53         <li>@Html.ActionLink("Vulnerabilities", "Index", "Vulnerabilities")</li>
54         <li>@Html.ActionLink("Assets", "Index", "Assets")</li>
55         if (Session["userRole"] != null)
56         {
57             <li> <a href="#"> @Session["username"]</a></li>
58         }
59         else
60         {
61             <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
62         }
63         <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>
64     }
65     else if (Role == "SecurityEngineer")
66     {
67         <li>@Html.ActionLink("Home", "Index", "Home")</li>
68         <li>@Html.ActionLink("Tasks", "VulnerabilityAssignmentsEngineer", "VulnerabilityAssignment", new { session = Session["Username"] }, null)</li>
69         <li>@Html.ActionLink("Assign Vulnerability", "Index", "VulnerabilityAssignment")</li>
70         <li>@Html.ActionLink("Vulnerabilities", "Index", "Vulnerabilities")</li>
71         if (Session["userRole"] != null)
72         {
73             <li> <a href="#"> @Session["username"]</a></li>
74         }
75         else
76         {
77             <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
78         }
79         <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>
80     }
81 }
82 </ul>
83 </div>
84 </div>
85 </body>
86 </html>
```



```

78     }
79     {
80         <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
81     }
82     <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>
83     }
84     }
85     else
86     {
87         <li>@Html.ActionLink("Home", "Index", "Home")</li>
88         <li>@Html.ActionLink("Login Tab", "Index", "Login")</li>
89         <li>@Html.ActionLink("Logout Tab", "Logout", "Login")</li>
90     }
91     }
92     }
93     }
94     </ul>
95     </div>
96     </div>
97     </div>
98     </div>
99     <div class="container body-content">
100     @RenderBody()
101     <hr />
102     <footer>
103         <p>&copy; @DateTime.Now.Year - Vulnerability Management</p>
104     </footer>
105     </div>
106     @Scripts.Render("~/bundles/jquery")
107     @Scripts.Render("~/bundles/bootstrap")
108     @RenderSection("scripts", required: false)
109 </body>
110 </html>
111
112

```

Error.cshtml

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta name="viewport" content="width=device-width" />
5      <title>Error</title>
6  </head>
7  <body>
8      <hgroup>
9          <h1>Error.</h1>
10         <h2>An error occurred while processing your request.</h2>
11     </hgroup>
12 </body>
13 </html>
14

```

VulnerabilityTasksController.cshtml

```
1 using Project.Data;
2 using Project.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Web;
7 using System.Web.Mvc;
8
9 namespace Project.Controllers
10 {
11     public class VulnerabilityTasksController : Controller
12     {
13         [CustomAuthorization]
14         // GET: VulnerabilityTasks
15         public ActionResult Index()
16         {
17             List<VulnerabilityTasksModel> vulnerabilityTasksModel = new List<VulnerabilityTasksModel>();
18             //Session["userName"] = vulnerabilityTasksModel.UserName;
19             VulnerabilityTasksDAO vulnerabilityTasksDAO = new VulnerabilityTasksDAO();
20             // vulnerabilityTasksModel = vulnerabilityTasksDAO.GetTasks();
21             return View("VulnerabilityTasks", vulnerabilityTasksModel);
22         }
23     }
24 }
```

VulnerabilityTasksDAO.cs

```
1 using Project.Models;
2 using System.Data.SqlClient;
3
4 namespace Project.Data
5 {
6     internal class VulnerabilityTasksDAO
7     {
8         string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProjectDB;Integrated Security=True;Connect
9         Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
10
11         public VulnerabilityTasksModel GetTasks()
12         {
13             using (SqlConnection connection = new SqlConnection(connectionString))
14             {
15                 //SQL Statement: Get all vulnerability tasks for user logged in
16                 string sqlQuery = "SELECT * FROM dbo.VulnerabilityAssignment WHERE TicketUser = @UserName AND TicketStatus = 'Open'";
17                 VulnerabilityTasksModel vulnerabilityTasksModel = new VulnerabilityTasksModel();
18                 SqlCommand command = new SqlCommand(sqlQuery, connection);
19                 command.Parameters.Add("@UserName", System.Data.SqlDbType.VarChar, 50).Value = vulnerabilityTasksModel.UserName;
20                 connection.Open();
21                 SqlDataReader reader = command.ExecuteReader();
22
23                 if (reader.HasRows)
24                 {
25                     while (reader.Read())
26                     {
27                         vulnerabilityTasksModel.VulnerabilityAssignmentID = reader.GetInt32(0);
28                         vulnerabilityTasksModel.VulnerabilityID = reader.GetInt32(1);
29                         vulnerabilityTasksModel.TicketUser = reader.GetString(2);
30                         vulnerabilityTasksModel.DateAssigned = reader.GetDateTime(3);
31                         vulnerabilityTasksModel.DateClosed = reader.GetDateTime(4);
32                         vulnerabilityTasksModel.TicketStatus = reader.GetString(5);
33                         vulnerabilityTasksModel.VulnerabilityTicketNotes = reader.GetString(6);
34                     }
35                 }
36                 reader.Close();
37                 connection.Close();
38
39                 return vulnerabilityTasksModel;
40             }
41         }
42     }
43 }
44 }
```

VulnerabilityTaskModel.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 {
9     public class VulnerabilityTasksModel
10    {
11        //public List<SelectListItem> vulnerabilityTasksModel { get; set; }
12        [Display(Name = "Vulnerability Assignment ID")]
13        public int VulnerabilityAssignmentID { get; set; }
14        [Display(Name = "Vulnerability ID")]
15        public int VulnerabilityID { get; set; }
16        [Display(Name = "Assignee")]
17        public string TicketUser { get; set; }
18        [Display(Name = "Date Assigned")]
19        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM//yyyy}")]
20        public DateTime DateAssigned { get; set; }
21        [Display(Name = "Date Closed")]
22        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM//yyyy}")]
23        public DateTime DateClosed { get; set; }
24        [Display(Name = "Ticket Status")]
25        public string TicketStatus { get; set; }
26        [Display(Name = "Ticket Notes")]
27        public string VulnerabilityTicketNotes { get; set; }
28        [Display(Name = "User")]
29        public string UsersName { get; set; }
30
31
32        0 references
33        public VulnerabilityTasksModel(int vulnerabilityAssignmentID, int vulnerabilityID, string ticketUser, DateTime dateAssigned, DateTime dateClosed, string ticketStatus, string
34        vulnerabilityTicketNotes)
35        {
36            VulnerabilityAssignmentID = vulnerabilityAssignmentID;
37            VulnerabilityID = vulnerabilityID;
38            TicketUser = ticketUser;
39            DateAssigned = dateAssigned;
40            DateClosed = dateClosed;
41            TicketStatus = ticketStatus;
42            VulnerabilityTicketNotes = vulnerabilityTicketNotes;
43        }
44
45        1 reference
46        public VulnerabilityTasksModel()
47        {
48        }
49    }
50 }
```