spécification

# Entrevue du Client

Objectif : Comprendre le problème à résoudre.

Le client exposera ce qu’il veut faire exactement et vous poserez des questions vous aidant à bien comprendre le problème.

# Prototype / Mock-up

Utiliser un outil de Prototype tel que Balsamiq ou autre pour évaluer votre compréhension du problème et proposer une solution basée sur la demande du client.

# Générer un document

Ce document va expliquer au client ce que vous avez compris après ses explications, attentes, etc.

Ce document sera un narratif accompagné, de diagramme montrant l’enchainement des écrans du logiciel et des screenshots provenant de l’outil de prototype.

Ce document sera fourni au client sous forme de PDF pour révision.

Le coût au client sera un paragraphe de ce document.

# Contrat

Le document généré à l’étape III sera modifié pour prendre en compte les changements ou corrections du client.

Le draft final constituera un contrat entre votre équipe et le client. Les 2 parties doivent signer ce document pour verrouiller cette partie du processus.

Tout changement proposé par le client après la signature de ce document doit faire l’objet d’un autre contrat ou un amendement du contrat courant modifiant les conditions de paiement.

# Partie Technique

La partie technique du document peut être sous la forme suivante.

| **Req. #** | **System** | **Description** |
| --- | --- | --- |
| 1.0 |  | **Intégration Base de Données** |
| **1.1** | **Backend** | Quelle base de données ? |
| 1.5 |  | Description des tables et leurs liens |
|  |  |  |
| **2.0** | **FrontEnd** | Web Application ou Desktop ou Mobile ou une combinaison d’eux. Expliquez ! |
|  |  |  |
| **3.0** | **Implémentation des écrans** | Mockup |
| 3.1 | Ecrans | Détails |

**Exemples**

Structure des Tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Personnels** | | | | |
| Champs | Type | Description | Clé Primaire | Présence dans autre table |
| Prenom |  |  |  |  |
| Prenom |  |  |  |  |
| PersonnelID |  |  |  |  |
| Code |  |  |  |  |
| Remarque |  |  |  |  |

USE [MaBaseDeDonnees]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Personnels] Script Date: 11/10/2017 12:21:29 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[Personnels](

[PersonnelID] [int] IDENTITY(1,1) NOT NULL,

[Prenom] [varchar](50) NOT NULL,

[Nom] [varchar](50) NOT NULL,

[Remarque] [varchar](150) NULL,

[Photo] [varbinary](max) NULL,

[Code] [nvarchar](4) NOT NULL,

CONSTRAINT [PK\_Personnels] PRIMARY KEY CLUSTERED

(

[PersonnelID] ASC

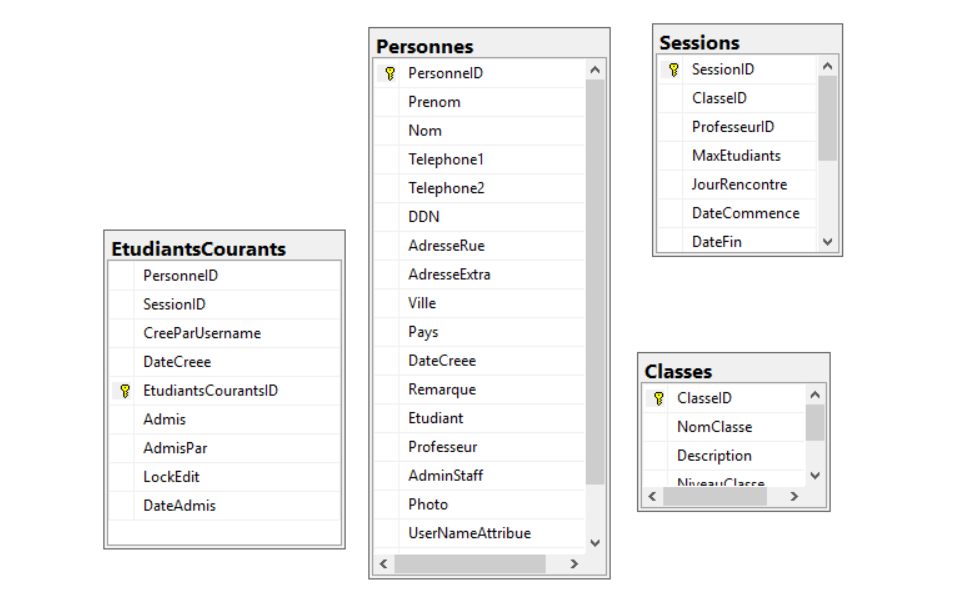
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

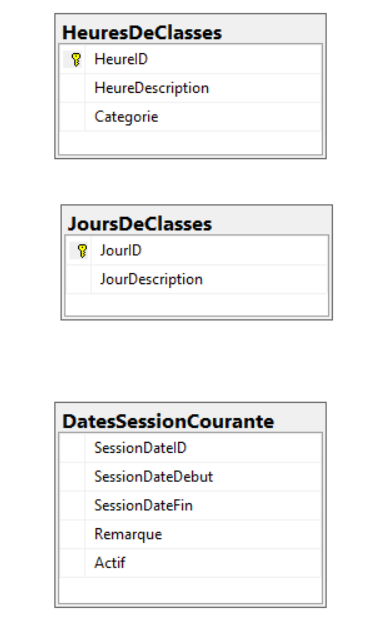
) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

**Existing Data Structure**

**Microsoft SQL Server 2016**





USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Classes] Script Date: 2/7/2019 4:06:39 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Classes](

[ClasseID] [int] IDENTITY(1,1) NOT NULL,

[NomClasse] [nvarchar](50) NOT NULL,

[Description] [nvarchar](50) NULL,

[NiveauClasse] [int] NOT NULL,

[Categorie] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Classes] PRIMARY KEY CLUSTERED ([ClasseID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Classes] ADD CONSTRAINT [DF\_Classes\_NiveauClasse] DEFAULT ((0)) FOR [NiveauClasse]

GO

USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Personnes] Script Date: 2/7/2019 4:08:39 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Personnes](

[PersonneID] [int] IDENTITY(1,1) NOT NULL,

[Prenom] [nvarchar](50) NOT NULL,

[Nom] [nvarchar](50) NOT NULL,

[Telephone1] [nvarchar](50) NULL,

[Telephone2] [nvarchar](50) NULL,

[DDN] [date] NULL,

[AdresseRue] [nvarchar](50) NULL,

[AdresseExtra] [nvarchar](50) NULL,

[Ville] [nvarchar](50) NULL,

[Pays] [nvarchar](50) NULL,

[DateCreee] [date] NOT NULL,

[Remarque] [nvarchar](150) NULL,

[Etudiant] [int] NOT NULL,

[Professeur] [int] NOT NULL,

[AdminStaff] [int] NOT NULL,

[Photo] [varbinary](max) NULL,

[UserNameAttribue] [nvarchar](50) NULL,

[CreeParUsername] [nvarchar](50) NOT NULL,

[NumeroRecu] [nvarchar](50) NULL,

CONSTRAINT [PK\_Personnes] PRIMARY KEY CLUSTERED ([PersonneID] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Personnes] ADD CONSTRAINT [DF\_Personnes\_Pays] DEFAULT (N'Haiti') FOR [Pays]

GO

ALTER TABLE [dbo].[Personnes] ADD CONSTRAINT [DF\_Personnes\_DateCree] DEFAULT (getdate()) FOR [DateCreee]

GO

ALTER TABLE [dbo].[Personnes] ADD CONSTRAINT [DF\_Personnes\_Etudiant] DEFAULT ((0)) FOR [Etudiant]

GO

ALTER TABLE [dbo].[Personnes] ADD CONSTRAINT [DF\_Personnes\_Professeur] DEFAULT ((0)) FOR [Professeur]

GO

ALTER TABLE [dbo].[Personnes] ADD CONSTRAINT [DF\_Personnes\_AdminStaff] DEFAULT ((0)) FOR [AdminStaff]

GO

USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Sessions] Script Date: 2/7/2019 4:14:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Sessions](

[SessionID] [int] IDENTITY(1,1) NOT NULL,

[ClasseID] [int] NOT NULL,

[ProfesseurID] [int] NOT NULL,

[MaxEtudiants] [int] NOT NULL,

[JourRencontre] [nvarchar](50) NOT NULL,

[DateCommence] [date] NOT NULL,

[DateFin] [date] NOT NULL,

[Heures] [nvarchar](50) NOT NULL,

[MontantParticipation] [money] NOT NULL,

[byusername] [nvarchar](50) NOT NULL,

[actif] [int] NOT NULL,

CONSTRAINT [PK\_Sessions] PRIMARY KEY CLUSTERED ( [SessionID] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Sessions] ADD CONSTRAINT [DF\_Sessions\_actif] DEFAULT ((1)) FOR [actif]

GO

USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[EtudiantsCourants] Script Date: 2/7/2019 4:14:28 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[EtudiantsCourants](

[PersonneID] [int] NOT NULL,

[SessionID] [int] NOT NULL,

[CreeParUsername] [nvarchar](50) NOT NULL,

[DateCreee] [date] NOT NULL,

[EtudiantsCourantsID] [int] IDENTITY(1,1) NOT NULL,

[Admis] [int] NOT NULL,

[AdmisPar] [nvarchar](50) NULL,

[LockEdit] [int] NOT NULL,

[DateAdmis] [date] NOT NULL,

CONSTRAINT [PK\_EtudiantsCourants] PRIMARY KEY CLUSTERED ([EtudiantsCourantsID] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[EtudiantsCourants] ADD CONSTRAINT [DF\_EtudiantsCourants\_DateCreee] DEFAULT (getdate()) FOR [DateCreee]

GO

ALTER TABLE [dbo].[EtudiantsCourants] ADD CONSTRAINT [DF\_EtudiantsCourants\_Admis] DEFAULT ((0)) FOR [Admis]

GO

ALTER TABLE [dbo].[EtudiantsCourants] ADD CONSTRAINT [DF\_EtudiantsCourants\_LockEdit] DEFAULT ((0)) FOR [LockEdit]

GO

ALTER TABLE [dbo].[EtudiantsCourants] ADD CONSTRAINT [DF\_EtudiantsCourants\_DateAdmis] DEFAULT (getdate()) FOR [DateAdmis]

GO

USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[DatesSessionCourante] Script Date: 2/7/2019 4:15:00 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DatesSessionCourante](

[SessionDateID] [int] IDENTITY(1,1) NOT NULL,

[SessionDateDebut] [date] NOT NULL,

[SessionDateFin] [date] NOT NULL,

[Remarque] [nvarchar](50) NULL,

[Actif] [int] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[DatesSessionCourante] ADD CONSTRAINT [DF\_DatesSessionCourante\_Actif] DEFAULT ((1)) FOR [Actif]

GO

USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[HeuresDeClasses] Script Date: 2/7/2019 4:15:34 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[HeuresDeClasses](

[HeureID] [int] IDENTITY(1,1) NOT NULL,

[HeureDescription] [nvarchar](50) NOT NULL,

[Categorie] [nvarchar](50) NULL,

CONSTRAINT [PK\_HeuresDeClasses] PRIMARY KEY CLUSTERED ([HeureID] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

USE [CCPAP\_Web\_Edu]

GO

/\*\*\*\*\*\* Object: Table [dbo].[JoursDeClasses] Script Date: 2/7/2019 4:15:44 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[JoursDeClasses](

[JourID] [int] IDENTITY(1,1) NOT NULL,

[JourDescription] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_JoursDeClasses] PRIMARY KEY CLUSTERED

([JourID] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

**Manuel d’Opérations Existant**

**Database Layer (Couche Base de Données) – Exemple.**

public static class ConnectionString

{

private static string cName = "Data Source=.;

Initial Catalog=StudentManagement;User ID=sa;Password=123";

public static string CName { get => cName; }

}

public class Student

{

public int Id { set; get; }

[Required]

public string FirstName { set; get; }

[Required]

public string LastName { set; get; }

[Required]

public string Email { set; get; }

[Required]

public string Mobile { set; get; }

public string Address { set; get; }

}

public class StudentDataAccessLayer

{

string connectionString = ConnectionString.CName;

public IEnumerable<Student> GetAllStudent()

{

List<Student> lstStudent = new List<Student>();

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spGetAllStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Student student = new Student();

student.Id = Convert.ToInt32(rdr["Id"]);

student.FirstName = rdr["FirstName"].ToString();

student.LastName = rdr["LastName"].ToString();

student.Email = rdr["Email"].ToString();

student.Mobile = rdr["Mobile"].ToString();

student.Address = rdr["Address"].ToString();

lstStudent.Add(student);

}

con.Close();

}

return lstStudent;

}

public void AddStudent(Student student)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spAddStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@FirstName", student.FirstName);

cmd.Parameters.AddWithValue("@LastName", student.LastName);

cmd.Parameters.AddWithValue("@Email", student.Email);

cmd.Parameters.AddWithValue("@Mobile", student.Mobile);

cmd.Parameters.AddWithValue("@Address", student.Address);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

public void UpdateStudent(Student student)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spUpdateStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Id", student.Id);

cmd.Parameters.AddWithValue("@FirstName", student.FirstName);

cmd.Parameters.AddWithValue("@LastName", student.LastName);

cmd.Parameters.AddWithValue("@Email", student.Email);

cmd.Parameters.AddWithValue("@Mobile", student.Mobile);

cmd.Parameters.AddWithValue("@Address", student.Address);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

public Student GetStudentData(int? id)

{

Student student = new Student();

using (SqlConnection con = new SqlConnection(connectionString))

{

string sqlQuery = "SELECT \* FROM Student WHERE Id= " + id;

SqlCommand cmd = new SqlCommand(sqlQuery, con);

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

student.Id = Convert.ToInt32(rdr["Id"]);

student.FirstName = rdr["FirstName"].ToString();

student.LastName = rdr["LastName"].ToString();

student.Email = rdr["Email"].ToString();

student.Mobile = rdr["Mobile"].ToString();

student.Address = rdr["Address"].ToString();

}

}

return student;

}

public void DeleteStudent(int? id)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spDeleteStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Id", id);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

}

Exemple Complet :

https://www.codeproject.com/Articles/1276169/CRUD-operations-using-ASP-NET-Core-and-ADO-NET

# CRUD operations using ASP.NET Core and ADO.NET



[Shamim Uddin](https://www.codeproject.com/script/Membership/View.aspx?mid=10230195), 3 Feb 2019

|  |  |
| --- | --- |
| https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-fill-md.png  https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-empty-md.png | 4.78 (5 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate: | [vote 1vote 2vote 3vote 4vote 5](https://www.codeproject.com/Articles/1276169/CRUD-operations-using-ASP-NET-Core-and-ADO-NET) |  | |

In this article, we are going to create a web application using ASP.NET Core MVC and ADO.NET. We will be creating a simple student record management system and performing CRUD operations on it.

* [Download project](https://github.com/shamimuddin1419/CRUD-operations-using-ASP.NET-Core-and-ADO.NET.git)
* [Download database](https://github.com/shamimuddin1419/DB-For-CRUD-operations-using-ASP.NET-Core-and-ADO.NET.git)

## Introduction

In this article, we are going to create a web application using ASP.NET Core MVC and ADO.NET. We will be creating a simple student record management system and performing CRUD operations on it.

## Prerequisites

* Install .NET Core 2.0.0 or above SDK from [here](https://www.microsoft.com/net/core#windowscmd)
* Download and install Visual Studio Code from [here](https://visualstudio.microsoft.com/downloads/)
* SQL Server 2008 or above

### Creating the Table and Stored Procedures

We will be using a DB table to store all the records of the student. First of all, we will create a database named StudentManagement.

Hide   Copy Code

CREATE DATABASE StudentManagement

Then we will create a table named “Student”.

Hide   Copy Code

Create table Student(

Id int IDENTITY(1,1) NOT NULL,

FirstName varchar(50) NOT NULL,

LastName varchar(50) NOT NULL,

Email varchar(30) NOT NULL,

Mobile varchar(20) NOT NULL,

Address varchar(220) NULL,

)

Now, we will create stored procedures to add, delete, update, and get student data.

#### To Insert a Student Record

Hide   Copy Code

Create procedure spAddStudent

(

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@Email VARCHAR(30),

@Mobile VARCHAR(20),

@Address VARCHAR(220)

)

as

Begin

Insert into Student (FirstName,LastName,Email, Mobile,Address)

Values (@FirstName,@LastName,@Email, @Mobile,@Address)

End

#### To Update a Student Record

Hide   Copy Code

Create procedure spUpdateStudent

(

@Id INTEGER ,

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@Email VARCHAR(30),

@Mobile VARCHAR(20),

@Address VARCHAR(220)

)

as

begin

Update Student

set FirstName=@FirstName,

LastName=@LastName,

Email=@Email,

Mobile=@Mobile,

Address=@Address

where Id=@Id

End

#### To Delete a Student Record

Hide   Copy Code

Create procedure spDeleteStudent

(

@Id int

)

as

begin

Delete from Student where Id=@Id

End

#### To View all Student Records

Hide   Copy Code

Create procedure spGetAllStudent

as

Begin

select \*

from Student

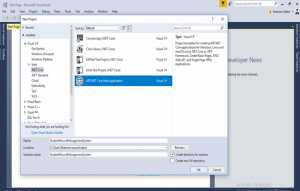
order by Id

End

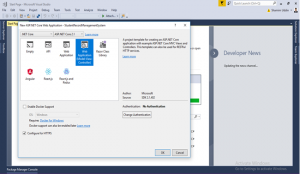
Our database part has been completed.

### Create the ASP.NET MVC Web Application

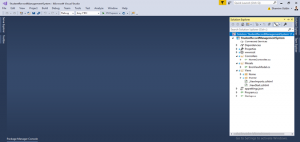
Now we are going to create an ASP.NET MVC Web Application. The project name is “StudentRecordManagementSystem”.



Click Ok button. Then, the below window will appear on screen.



Click ok button. Solution creation has done with loading all needed files. Given below is the picture of solution structure.



## What is MVC

1. **Model**: Classes that represent the data of the solution and it enforces business.
2. **View**: Simple word view means UI (User Interface) which dynamically generates HTML responses.
3. **Controller**: A Controller is the link between User and System. It handles incoming browser requests and after process using model data or specific task, returns a response to the browser.

Create a folder named “Utility” in the project. Now we will create a class named “ConnectionString” within Utility folder.

Hide   Copy Code

public static class ConnectionString

{

private static string cName = "Data Source=.;

Initial Catalog=StudentManagement;User ID=sa;Password=123";

public static string CName { get => cName; }

}

After that, we will create a class named “Student” within model folder:

Hide   Copy Code

public class Student

{

public int Id { set; get; }

[Required]

public string FirstName { set; get; }

[Required]

public string LastName { set; get; }

[Required]

public string Email { set; get; }

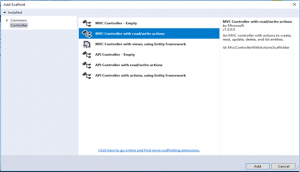
[Required]

public string Mobile { set; get; }

public string Address { set; get; }

}

We will rebuild our solution and create a “StudentController” within Controller folder. Right-->Add-->Controller -->Select MVC Controller with read/write actions and click add.



Another window will appear on screen:

https://www.codeproject.com/KB/aspnet/1276169/19703fe1-a2f5-45bd-9961-66131286f6c5.Png

Our StudentController has been created.

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

public class StudentController : Controller

{

// GET: Student

public ActionResult Index()

{

return View();

}

// GET: Student/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: Student/Create

public ActionResult Create()

{

return View();

}

// POST: Student/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(IFormCollection collection)

{

try

{

// TODO: Add insert logic here

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

// GET: Student/Edit/5

public ActionResult Edit(int id)

{

return View();

}

// POST: Student/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(int id, IFormCollection collection)

{

try

{

// TODO: Add update logic here

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

// GET: Student/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: Student/Delete/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, IFormCollection collection)

{

try

{

// TODO: Add delete logic here

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

}

We have to work with database, so we will create a data access layer class within model folder named “StudentDataAccessLayer”.

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

public class StudentDataAccessLayer

{

string connectionString = ConnectionString.CName;

public IEnumerable<Student> GetAllStudent()

{

List<Student> lstStudent = new List<Student>();

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spGetAllStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Student student = new Student();

student.Id = Convert.ToInt32(rdr["Id"]);

student.FirstName = rdr["FirstName"].ToString();

student.LastName = rdr["LastName"].ToString();

student.Email = rdr["Email"].ToString();

student.Mobile = rdr["Mobile"].ToString();

student.Address = rdr["Address"].ToString();

lstStudent.Add(student);

}

con.Close();

}

return lstStudent;

}

public void AddStudent(Student student)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spAddStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@FirstName", student.FirstName);

cmd.Parameters.AddWithValue("@LastName", student.LastName);

cmd.Parameters.AddWithValue("@Email", student.Email);

cmd.Parameters.AddWithValue("@Mobile", student.Mobile);

cmd.Parameters.AddWithValue("@Address", student.Address);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

public void UpdateStudent(Student student)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spUpdateStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Id", student.Id);

cmd.Parameters.AddWithValue("@FirstName", student.FirstName);

cmd.Parameters.AddWithValue("@LastName", student.LastName);

cmd.Parameters.AddWithValue("@Email", student.Email);

cmd.Parameters.AddWithValue("@Mobile", student.Mobile);

cmd.Parameters.AddWithValue("@Address", student.Address);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

public Student GetStudentData(int? id)

{

Student student = new Student();

using (SqlConnection con = new SqlConnection(connectionString))

{

string sqlQuery = "SELECT \* FROM Student WHERE Id= " + id;

SqlCommand cmd = new SqlCommand(sqlQuery, con);

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

student.Id = Convert.ToInt32(rdr["Id"]);

student.FirstName = rdr["FirstName"].ToString();

student.LastName = rdr["LastName"].ToString();

student.Email = rdr["Email"].ToString();

student.Mobile = rdr["Mobile"].ToString();

student.Address = rdr["Address"].ToString();

}

}

return student;

}

public void DeleteStudent(int? id)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("spDeleteStudent", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Id", id);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

}

#### Create Action

Now, we will work Create Action within Student Controller. There are two Create Actions, one is GET and another is POST. Now we will create a view for create action. Before creating a view, we will create a constructor.

Hide   Copy Code

StudentDataAccessLayer studentDataAccessLayer = null;

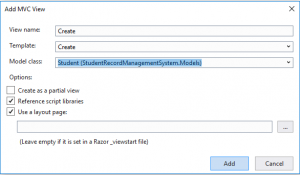
public StudentController()

{

studentDataAccessLayer = new StudentDataAccessLayer();

}

Right click on create (GET) action, then click add view below window will appear on screen.



Click add:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

@model StudentRecordManagementSystem.Models.Student

@{

ViewData["Title"] = "Create";

}

<h2>Create</h2>

<h4>Student</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Create">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="Id" class="control-label"></label>

<input asp-for="Id" class="form-control" />

<span asp-validation-for="Id" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="FirstName" class="control-label"></label>

<input asp-for="FirstName" class="form-control" />

<span asp-validation-for="FirstName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="LastName" class="control-label"></label>

<input asp-for="LastName" class="form-control" />

<span asp-validation-for="LastName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Email" class="control-label"></label>

<input asp-for="Email" class="form-control" />

<span asp-validation-for="Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Mobile" class="control-label"></label>

<input asp-for="Mobile" class="form-control" />

<span asp-validation-for="Mobile" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Address" class="control-label"></label>

<input asp-for="Address" class="form-control" />

<span asp-validation-for="Address" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Back to List</a>

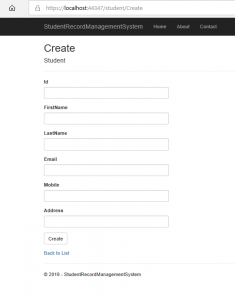
</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Now run our application:



We will remove Id field from view. We have done auto increment of Id field in database. Now we will work with Create (POST).

Hide   Copy Code

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Student student)

{

try

{

// TODO: Add insert logic here

studentDataAccessLayer.AddStudent(student);

return RedirectToAction(nameof(Index));

}

catch(Exception ex)

{

return View();

}

}

#### Index Action

We have to call GetAllStudent method from StudentDataAccessLayer class for getting all student in Index action:

Hide   Copy Code

public ActionResult Index()

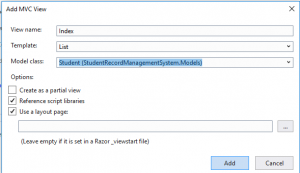
{

IEnumerable<Student> students = studentDataAccessLayer.GetAllStudent();

return View(students);

}

Right click on Index action, then click add view below window will appear on screen.



Click add.

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

@model IEnumerable<StudentRecordManagementSystem.Models.Student>

@{

ViewData["Title"] = "Index";

}

<h2>Index</h2>

<p>

<a asp-action="Create">Create New</a>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Id)

</th>

<th>

@Html.DisplayNameFor(model => model.FirstName)

</th>

<th>

@Html.DisplayNameFor(model => model.LastName)

</th>

<th>

@Html.DisplayNameFor(model => model.Email)

</th>

<th>

@Html.DisplayNameFor(model => model.Mobile)

</th>

<th>

@Html.DisplayNameFor(model => model.Address)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Id)

</td>

<td>

@Html.DisplayFor(modelItem => item.FirstName)

</td>

<td>

@Html.DisplayFor(modelItem => item.LastName)

</td>

<td>

@Html.DisplayFor(modelItem => item.Email)

</td>

<td>

@Html.DisplayFor(modelItem => item.Mobile)

</td>

<td>

@Html.DisplayFor(modelItem => item.Address)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

</tr>

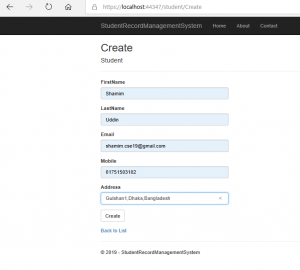
}

</tbody>

</table>

Student save and show list has been done. Now we will save a student and see student in list.

Save a student:



Show student list:



#### Edit Action

Now we will work with Edit Action within Student Controller. There are two Edit Action, one is GET and another is POST. Now we will create a view for create action. We have to call GetStudentData method from StudentDataAccessLayer class for getting student by Id.

Hide   Copy Code

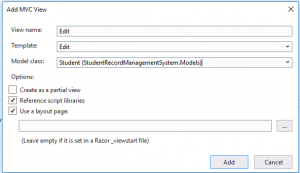
public ActionResult Edit(int id)

{

Student student = studentDataAccessLayer.GetStudentData(id);

return View(student);

}

Right click on Edit (GET) action, then click add view below window will appear on screen. 

Click Add:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

@model StudentRecordManagementSystem.Models.Student

@{

ViewData["Title"] = "Edit";

}

<h2>Edit</h2>

<h4>Student</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Edit">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="Id" class="control-label"></label>

<input asp-for="Id" class="form-control" readonly/>

<span asp-validation-for="Id" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="FirstName" class="control-label"></label>

<input asp-for="FirstName" class="form-control" />

<span asp-validation-for="FirstName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="LastName" class="control-label"></label>

<input asp-for="LastName" class="form-control" />

<span asp-validation-for="LastName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Email" class="control-label"></label>

<input asp-for="Email" class="form-control" />

<span asp-validation-for="Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Mobile" class="control-label"></label>

<input asp-for="Mobile" class="form-control" />

<span asp-validation-for="Mobile" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Address" class="control-label"></label>

<input asp-for="Address" class="form-control" />

<span asp-validation-for="Address" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Update" class="btn btn-default" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Back to List</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Now we will work with Edit (POST):

Hide   Copy Code

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(Student student)

{

try

{

// TODO: Add update logic here

studentDataAccessLayer.UpdateStudent(student);

return RedirectToAction(nameof(Index));

}

catch

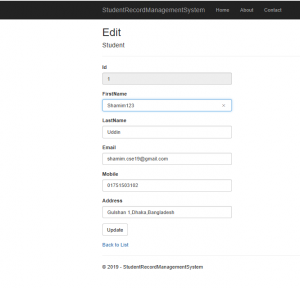
{

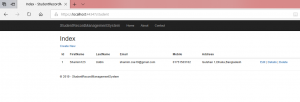
return View();

}

}

Edit option has been done. Now we will test if it works or not.





It works fine. Now we will work with Delete action.

#### Delete Action

Now we will work with Delete Action within Student Controller. There are two Delete Action, one is GET and another is POST. Now we will create a view for delete action. We have to call GetStudentData method from StudentDataAccessLayer class for getting student by Id.

Hide   Copy Code

public ActionResult Delete(int id)

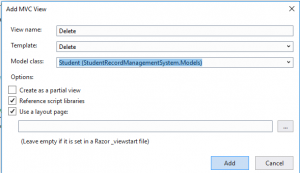
{

Student student = studentDataAccessLayer.GetStudentData(id);

return View(student);

}

Right click on Delete (GET) action, then click add view below window will appear on screen.



Click Add:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

@model StudentRecordManagementSystem.Models.Student

@{

ViewData["Title"] = "Delete";

}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Student</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Id)

</dt>

<dd>

@Html.DisplayFor(model => model.Id)

</dd>

<dt>

@Html.DisplayNameFor(model => model.FirstName)

</dt>

<dd>

@Html.DisplayFor(model => model.FirstName)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LastName)

</dt>

<dd>

@Html.DisplayFor(model => model.LastName)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Email)

</dt>

<dd>

@Html.DisplayFor(model => model.Email)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Mobile)

</dt>

<dd>

@Html.DisplayFor(model => model.Mobile)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd>

@Html.DisplayFor(model => model.Address)

</dd>

</dl>

<form asp-action="Delete">

<input type="submit" value="Delete" class="btn btn-default" /> |

<a asp-action="Index">Back to List</a>

</form>

</div>

Now we will work with Delete (POST):

Hide   Copy Code

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(Student student)

{

try

{

// TODO: Add delete logic here

studentDataAccessLayer.DeleteStudent(student.Id);

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

#### Details Action

We have to call GetStudentData method from StudentDataAccessLayer class for getting student by Id in Index action:

Hide   Copy Code

public ActionResult Details(int id)

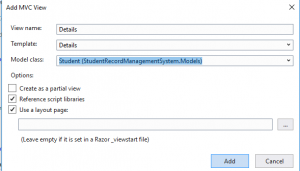
{

Student student = studentDataAccessLayer.GetStudentData(id);

return View(student);

}

Right click on Details action, then click add view, below window will appear on screen.



Click add:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png  Copy Code

@model StudentRecordManagementSystem.Models.Student

@{

ViewData["Title"] = "Details";

}

<h2>Details</h2>

<div>

<h4>Student</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Id)

</dt>

<dd>

@Html.DisplayFor(model => model.Id)

</dd>

<dt>

@Html.DisplayNameFor(model => model.FirstName)

</dt>

<dd>

@Html.DisplayFor(model => model.FirstName)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LastName)

</dt>

<dd>

@Html.DisplayFor(model => model.LastName)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Email)

</dt>

<dd>

@Html.DisplayFor(model => model.Email)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Mobile)

</dt>

<dd>

@Html.DisplayFor(model => model.Mobile)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd>

@Html.DisplayFor(model => model.Address)

</dd>

</dl>

</div>

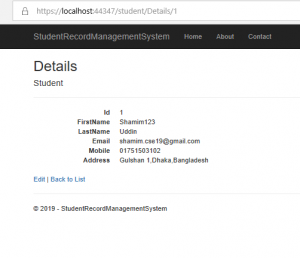
<div>

@Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |

<a asp-action="Index">Back to List</a>

</div>

Now we will run our application:



Hope it will be helpful!



## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License (CPOL)](http://www.codeproject.com/info/cpol10.aspx)

## Share

## About the Author



[**Shamim Uddin**](https://www.codeproject.com/Members/shamimuddin)

|  |  |  |
| --- | --- | --- |
|  | Software Developer Amber Software Solution Ltd. | Follow  this Member |
| Bangladesh Bangladesh |

Hi, I am Shamim Uddin.Working with Microsoft Technologies.