Assignment Documentation

Github Link: <https://github.com/aman8198/MaxMobilityAssignment>

1. Created a ASP.NET MVC Project in visual studio
2. Installed packages to be used in this project via Nuget Package Manger below is the details

<ItemGroup>

<PackageReference Include="EPPlus" Version="7.2.1" />

<PackageReference Include="OfficeOpenXml.Core.ExcelPackage" Version="1.0.0" />

<PackageReference Include="System.Data.SqlClient" Version="4.8.6" />

</ItemGroup>

1. After Package installation Opened my MSSQL server created a Database for this assignment and in that database I created the table to hold data from excel upload, created stored procedure which I will be using in the project and create a user defined table type. So the query is listed below

CREATE TABLE UploadedData (

Id INT IDENTITY(1,1) PRIMARY KEY,

SLNO INT,

Name NVARCHAR(100),

Email NVARCHAR(100),

PhoneNo NVARCHAR(50),

Address NVARCHAR(200),

Status NVARCHAR(50)

);

CREATE TYPE dbo.ExampleTableType AS TABLE

(

SLNO INT,

Name NVARCHAR(100),

Email NVARCHAR(100),

PhoneNo NVARCHAR(50),

Address NVARCHAR(200),

Status NVARCHAR(50)

);

CREATE PROCEDURE usp\_SaveExcelData

@ExcelData dbo.ExampleTableType READONLY

AS

BEGIN

INSERT INTO UploadedData (SLNO, Name, Email, PhoneNo, Address, Status)

SELECT SLNO, Name, Email, PhoneNo, Address, Status

FROM @ExcelData

END;

1. Opened the project and in appsettings.json file added my connection string there

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=Maxmobilityassignment;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}

1. For handling the front end logic I opened the Index.cshtml file located in the Views/Home location of project and written my code to implement front end logic
2. @{
3. ViewBag.Title = "File Upload";
4. }
5. <h2>Upload Excel File</h2>
6. <form asp-action="Upload" asp-controller="Home" method="post" enctype="multipart/form-data">
7. <div class="form-group">
8. <label for="file">Upload Excel:</label>
9. <input type="file" name="file" id="file" class="form-control" />
10. </div>
11. <button type="submit" class="btn btn-primary">Submit</button>
12. </form>
13. @if (ViewBag.Message != null)
14. {
15. <div class="alert alert-info">@ViewBag.Message</div>
16. }
17. @if (ViewBag.DownloadLink != null)
18. {
19. <a href="@ViewBag.DownloadLink" class="btn btn-success">Download Uploaded File</a>
20. }

6. Then I opened my homecontroller to write logic for Uploading excelfile, reading data from excelfile, validating data coming from excel file, Saving the correct data in Database. The code is written below

using System.Data;

using Microsoft.AspNetCore.Mvc;

using OfficeOpenXml;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using System.Globalization;

namespace YourNamespace.Controllers

{

public class HomeController : Controller

{

private readonly string \_connectionString;

public HomeController(IConfiguration configuration)

{

\_connectionString = configuration.GetConnectionString("DefaultConnection");

}

public IActionResult Index()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Upload(IFormFile file)

{

if (file != null && file.Length > 0 &&

(Path.GetExtension(file.FileName).Equals(".xls") || Path.GetExtension(file.FileName).Equals(".xlsx")))

{

string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/Uploads");

if (!Directory.Exists(uploadsFolder))

{

Directory.CreateDirectory(uploadsFolder);

}

string filePath = Path.Combine(uploadsFolder, Path.GetFileName(file.FileName));

using (var stream = new FileStream(filePath, FileMode.Create))

{

await file.CopyToAsync(stream);

}

DataTable dataTable = ReadExcelFile(filePath);

if (ValidateDataTable(dataTable))

{

SaveToDatabase(dataTable);

ViewBag.Message = "Upload Successful";

ViewBag.DownloadLink = Url.Content("~/Uploads/" + Path.GetFileName(file.FileName));

}

else

{

ViewBag.Message = "Validation Failed";

}

}

else

{

ViewBag.Message = "Invalid File Format";

}

return View("Index");

}

private DataTable ReadExcelFile(string filePath)

{

// Set the license context before using EPPlus

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

DataTable dataTable = new DataTable();

using (var package = new ExcelPackage(new FileInfo(filePath)))

{

ExcelWorksheet worksheet = package.Workbook.Worksheets.First();

bool hasHeader = true; // adjust it accordingly

// Add columns to DataTable

foreach (var firstRowCell in worksheet.Cells[1, 1, 1, worksheet.Dimension.End.Column])

{

dataTable.Columns.Add(hasHeader ? firstRowCell.Text : $"Column {firstRowCell.Start.Column}");

}

// Add rows to DataTable

var startRow = hasHeader ? 2 : 1;

for (var rowNum = startRow; rowNum <= worksheet.Dimension.End.Row; rowNum++)

{

var wsRow = worksheet.Cells[rowNum, 1, rowNum, worksheet.Dimension.End.Column];

// Check if all cells in the row are empty (blank)

if (wsRow.All(cell => string.IsNullOrWhiteSpace(cell.Text)))

{

continue; // Skip blank rows

}

DataRow row = dataTable.NewRow();

foreach (var cell in wsRow)

{

row[cell.Start.Column - 1] = cell.Text;

}

dataTable.Rows.Add(row);

}

}

return dataTable;

}

private bool ValidateDataTable(DataTable dataTable)

{

bool isValid = true;

foreach (DataRow row in dataTable.Rows)

{

bool isRowValid = true;

// Check each field in the row for validity

if (string.IsNullOrEmpty(row["Name"].ToString()))

{

isRowValid = false;

}

if (string.IsNullOrEmpty(row["Email"].ToString()) || !IsValidEmail(row["Email"].ToString()))

{

isRowValid = false;

}

if (string.IsNullOrEmpty(row["Phone No"].ToString()))

{

isRowValid = false;

}

if (string.IsNullOrEmpty(row["Address"].ToString()))

{

isRowValid = false;

}

// If any field in the row is invalid, set isValid to false

if (!isRowValid)

{

isValid = false;

// Optionally, you can log or accumulate validation errors here

}

}

return isValid;

}

private bool IsValidEmail(string email)

{

try

{

// Normalize the domain part of the email address

email = Regex.Replace(email, @"(@)(.+)$", DomainMapper, RegexOptions.None, TimeSpan.FromMilliseconds(300));

// Return true if MailAddress accepts the email address

return new System.Net.Mail.MailAddress(email).Address == email;

}

catch (Exception)

{

return false;

}

}

private static string DomainMapper(Match match)

{

// Use IdnMapping class to convert Unicode domain names.

var idn = new IdnMapping();

// Pull out and process domain name (throws ArgumentException on invalid)

string domainName = idn.GetAscii(match.Groups[2].Value);

return match.Groups[1].Value + domainName;

}

private void SaveToDatabase(DataTable dataTable)

{

using (SqlConnection conn = new SqlConnection(\_connectionString))

{

using (SqlCommand cmd = new SqlCommand("usp\_SaveExcelData", conn))

{

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter tvpParam = cmd.Parameters.AddWithValue("@ExcelData", dataTable);

tvpParam.SqlDbType = SqlDbType.Structured;

conn.Open();

cmd.ExecuteNonQuery();

}

}

}

}

}

That’s it was able to achieve the assignment requirements.