EFCRUD: S1:- Install packages: Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore.Tools, Microsoft.VisualStudio.Web.CodeGeneration

S2:- In appsettings.json, add code: "ConnectionStrings": { "DefaultConnection": " Data Source = . ; Initial Catalog = EfCoreCrud; Integrated Security = true" }

S3: Create a model as public class Student { [Column("StudentId")] [Required] [Key] [DatabaseGenerated(DatabaseGeneratedOption.Identity)] public int Id { get; set; }

[Column("Name")] [Required] [Display(Name = "Student Name")] [StringLength(200)] public string Name { get; set; }} and appDbContext.cs as public class AppDbContext : DbContext

{public AppDbContext( DbContextOptions<AppDbContext> options) : base(options) {} public DbSet<Student> Students { get; set; }}

S4: In Startup.cs, add code inside ConfigureServices: services.AddDbContext<AppDbContext>(o => o.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

In Configure Method add parameter (AppDbContext db) and in code add: db.Database.EnsureCreated(); S5:- Generate a controller as : public class StudentsController : Controller{

private readonly AppDbContext \_context; public StudentsController(AppDbContext context) {context = context; } public async Task<IActionResult> Index(){return View(await **\_context.Students.ToListAsync()**); } public async Task<IActionResult> Details(int? id) { var student = await **\_context.Students.FirstOrDefaultAsync(m => m.Id == id);** return View(student); } public IActionResult Create(){ return View();} [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Create([Bind("**Id,Name** ")] Student student) { if (ModelState.IsValid) {**\_context.Add(student)**; **await \_context.SaveChangesAsync()**;return RedirectToAction(nameof(Index)); }return View(student); }public async Task<IActionResult> Edit(int? id) { var student = await \_context.Students.FindAsync(id); return View(student); } [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Edit(int id, [Bind("Id,Name ")] Student student) {if (ModelState.IsValid) {try{context.Update(student); await \_context.SaveChangesAsync();}catch (DbUpdateConcurrencyException) {if (!StudentExists(student.Id)) {return NotFound();}else{throw; }}return RedirectToAction(nameof(Index)); }return View(student); } public async Task<IActionResult> Delete(int? id) { var student = await **\_context.Students.FirstOrDefaultAsync(m => m.Id == id);** return View(student); } [HttpPost, ActionName("Delete")] [ValidateAntiForgeryToken] public async Task<IActionResult> DeleteConfirmed(int id) {var student = **await \_context.Students.FindAsync(id);** **\_context.Students.Remove(student);** await \_context.SaveChangesAsync();return RedirectToAction(nameof(Index)); } private bool StudentExists(int id) {return \_context.Students.Any(e => e.Id == id); }}

Ado.Net: class Program{ public static void CreateConnection(SqlConnection conn) { conn.Open(); Console.WriteLine("Database Connected Successfully");conn.Close();}

public static void CreateTable(SqlConnection conn) { try{ SqlCommand cmd = new SqlCommand("CREATE TABLE student(name nvarchar(100) NOT NULL,email nvarchar(100) NOT NULL, )", conn); conn.Open(); cmd.ExecuteNonQuery();}catch(Exception e) {Console.WriteLine(e.ToString());}finally{conn.Close();}}public static void InsertIntoTable(SqlConnection conn) {

try{Console.Write("Enter your Name: ");string name = Console.ReadLine();Console.Write("Enter your Email: ");string email = Console.ReadLine();SqlCommand cmd = new SqlCommand($"INSERT INTO student (name,email) VALUES ( '{name}', '{email}' )", conn); conn.Open();cmd.ExecuteNonQuery();}catch(Exception e) {Console.WriteLine(e.ToString());}

finally{conn.Close();}}public static void Retrieve(SqlConnection conn) {try{ SqlCommand cmd = new SqlCommand("SELECT \* FROM student", conn); conn.Open();SqlDataReader sdr = cmd.ExecuteReader();while (sdr.Read()){Console.WriteLine("Name: "+sdr["name"] + " Email: " + sdr["email"]);}}catch(Exception e) {Console.WriteLine(e.ToString());}finally{conn.Close();}}

static void Main(string[] args) {string cs = "data source=.;database=AdoNetConsoleCrud;Integrated Security = true";SqlConnection conn = new SqlConnection(cs//CreateConnection(conn);

//CreateTable(conn); //InsertIntoTable(conn); Console.WriteLine("Retriving data from Database:"); Retrieve(conn); Console.ReadLine();}}

JS/jQuery validation: S1: <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script> S2: <form><div class="form-group"><label for="email">Email: </label>

<input type="email" name="email" id="email" required class="form-control"> <small id="emailvalid" class="form-text text-muted invalid-feedback"> Your email must be a valid email

</small> </div> <div class="form-group"> <label for="password"> Password: </label> <input type="password" name="pass" id="password" class="form-control"> <h6 id="passcheck" style="color: red;"> \*Please Fill the password </h6> </div><input type="submit" id="submitbtn" value="Submit" class="btn btn-primary"> </form> </div> </div>

S3:

$(document).ready(function () { const email = document.getElementById('email'); email.addEventListener('blur', ()=>{ let regex = /^([\_\-\.0-9a-zA-Z]+)@([\_\-\.0-9a-zA-Z]+)\.([a-zA-Z]){2,7}$/;

let s = email.value; if(regex.test(s)){ email.classList.remove('is-invalid'); emailError = true; }else{email.classList.add('is-invalid'); emailError = false; }});

// Validate Password$('#passcheck').hide();let passwordError = true; $('#password').keyup(function () {validatePassword();});function validatePassword() {let passwrdValue =$('#password').val();if (passwrdValue.length == '') {$('#passcheck').show();passwordError = false; return false; }if ((passwrdValue.length < 3)|| (passwrdValue.length > 10)) {$('#passcheck').show();$('#passcheck').html ("\*length of your password must be between 3 and 10");$('#passcheck').css("color", "red");passwordError = false; return false; } else {

$('#passcheck').hide();}} $('#submitbtn').click(function () { validateEmail();validatePassword();if ((emailError == true)&& (passwordError == true)) {return true; } else {return false; }});});

**Add Authentication**: services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true).AddEntityFrameworkStores<ApplicationDbContext>();

services.AddRazorPages(); and app.UseAuthentication(); app.UseAuthorization();

**SQLI**:- SqlCommand cmd = new SqlCommand (“select \* from tbl\_users where uname = ‘ ”+uname+” ‘ and password = ‘ ”+pwd+” ‘ ”, conn); is changed to SqlCommand cmd = new SqlCommand(“ select \* from tbl\_users where uname = @uname and password = @pwd”,conn); cmd.Parameters.AddWithValue(“@uname”, uname); cmd.Parameters.AddWithValue(“@pwd”, pwd);

**OpenRedirect**: <http://abc.com/login?returnUrl=/home/profile> is changed to <http://abc.com/login?returnUrl=abc1.com/login>

**Session**: **S1:** services.AddSession() and app.UseSession(); then S2: Set as HttpContext.Session.SetString(“Name”,”My name”); and get as: String myName = HttpContext.Session.GetString(“Name”);

**CooKie**: Read as string cookie1 = httpContextAccessor.HttpContext.Request.Cookies[“Id”]; or string cooke2 = Request.Cookies[“Name”]; Write cookie as Response.Append(key,value,opt);