**Angular**

**Base Service File (base-http.service.ts)**

constructor(private httpClient: HttpClient, protected apiConfig: ConfigDataProvider, ) { }

  //baseUrl: string = 'http://192.168.254.35:8087/';

  baseUrl: string = 'http://localhost:58853/';

  /\*\*

   \* Function used for HTTP get

   \* @param relativeUrl API endpoint

   \* @param options Optional parameters

   \*/

  public get<T, OT = any>(relativeUrl: string, options?: OT) {

    if (options) {

      return this.httpClient.get<T>(`${this.baseUrl}${relativeUrl}`, options );

    } else {

      return this.httpClient.get<T>(`${this.baseUrl}${relativeUrl}` );

    }

  }

  /\*\*

   \* Function used for HTTP get post

   \* @param relativeUrl API endpoint

   \* @param data Data object

   \* @param options Optional parameters

   \*/

  public post<T, OT = any>(relativeUrl: string, data: T, options?: OT) {

    if (options ) {

      return this.httpClient.post<T>(`${this.baseUrl}${relativeUrl}`, data, options);

    } else {

      return this.httpClient.post<T>(`${this.baseUrl}${relativeUrl}`, data);

    }

  }

  /\*\*

   \* Function used for HTTP put

   \* @param relativeUrl API endpoint

   \* @param data Data object

   \* @param options Optional parameters

   \*/

  public put<T, OT = any>(relativeUrl: string, data?: T, options?: OT) {

    if (options ) {

      return this.httpClient.put<T>(`${this.baseUrl}${relativeUrl}`, data, options);

    } else {

      return this.httpClient.put<T>(`${this.baseUrl}${relativeUrl}`, data);

    }

  }

  /\*\*

   \* Function used for HTTP delete

   \* @param relativeUrl: API endpoint

   \* @param options: Optional parameters

   \*/

  public delete<T, OT = any>(relativeUrl: string, options?: OT) {

    if (options) {

      return this.httpClient.delete<T>(`${this.baseUrl}${relativeUrl}`, options );

    } else {

      return this.httpClient.delete<T>(`${this.baseUrl}${relativeUrl}` );

    }

  }

**Login Screen (**login.component.ts)

  /\*\*

  \* Method to called login user api

  \* @param requestData

  \*/

  callLoginUserApi(requestData: any) {

    this.isDataLoading = true;

    this.authService.loginExistingUser(requestData)

      .pipe(takeUntil(this.onDestroy$))

      .subscribe({

        next: (retData: any) => {

          if (retData.status) {

            this.parseResponse(retData,requestData);

          } else {

            this.toastService.errorMessage(retData.message);

          }

          this.isDataLoading = false;

        },

        error: (err: any) => {

          console.log(err);

          this.isDataLoading = false;

        },

        complete: () => {

          console.log('complete');

          this.isDataLoading = false;

        }

      });

  }

**Registration Screen (**registration.component.ts)

 /\*\*

   \* Method to called register user api

   \* @param respData

   \*/

  callRegisterUserApi(respData: any) {

    this.isDataLoading=true;

    this.authService.registerNewUser(respData)

      .pipe(takeUntil(this.onDestroy$))

      .subscribe({

        next: (retData: any) => {

          this.isDataLoading = false;

          if (retData.status) {

            this.toastService.successMessage(Messages.RegisterUserSuccess);

            this.router.navigate(['login']);

          } else {

            this.toastService.errorMessage(retData.message);

          }

        },

        error: (err: any) => {

          console.log(err);

          this.isDataLoading = false;

        },

        complete: () => {

          console.log('complete');

          this.isDataLoading = false;

        }

      });

  }

}

**ASP .NET**

**HospitalManagementController.ts**

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using WebApplication4.UtilityClasses;

using System.Web.Http.Cors;

namespace WebApplication4.Controllers

{

    [EnableCors(origins: "\*", headers: "\*", methods: "\*")]

    public class HospitalManagementController : ApiController

    {

        static string nodatafound = @"{""Message"": ""No data found""}";

        static string noaction = @"{""Message"": ""No action performed, please try again""}";

        #region API Health

        [HttpGet]

        [Route("api/Health")]

        public async Task<string> Get()

        {

            await Task.Delay(1);

            return "Service Running! Healthy!";

        }

        private void BadRequestValidation<T>(T RequestObject) where T : class

        {

            var modelStateJson = ModelState.SelectMany(x => x.Value.Errors.Select(e => (string.IsNullOrEmpty(e.ErrorMessage) ? e.Exception.Message : e.ErrorMessage))).Aggregate((a, b) => a + b);

            var message = "Model Validation faiiled, Required fields are not provided. More details : " + modelStateJson;

            int id = 0;

            if (typeof(T) == typeof(Doctor)) { Doctor newT1 = (Doctor)(object)RequestObject; id = newT1.DoctorId; }

            if (typeof(T) == typeof(Nurse)) { Nurse newT1 = (Nurse)(object)RequestObject; id = newT1.NurseId; }

            if (typeof(T) == typeof(HospitalBranch)) { HospitalBranch newT1 = (HospitalBranch)(object)RequestObject; id = newT1.HospitalId; }

            if (typeof(T) == typeof(Patient)) { Patient newT1 = (Patient)(object)RequestObject; id = newT1.PatientId; }

            if (typeof(T) == typeof(Medicine)) { Medicine newT1 = (Medicine)(object)RequestObject; id = newT1.MedicineId; }

            if (typeof(T) == typeof(Service)) { Service newT1 = (Service)(object)RequestObject; id = newT1.ServiceRoomId; }

            if (typeof(T) == typeof(OperativeRoom)) { OperativeRoom newT1 = (OperativeRoom)(object)RequestObject; id = newT1.OperativeRoomId; }

            if (typeof(T) == typeof(Medication)) { Medication newT1 = (Medication)(object)RequestObject; id = newT1.MedicationId; }

            if (typeof(T) == typeof(TreatmentNurse)) { TreatmentNurse newT1 = (TreatmentNurse)(object)RequestObject; id = newT1.TreatmentNurseId; }

            if (typeof(T) == typeof(Treatment)) { Treatment newT1 = (Treatment)(object)RequestObject; id = newT1.TreatmentId; }

            if (typeof(T) == typeof(Admission)) { Admission newT1 = (Admission)(object)RequestObject; id = newT1.AdmissionId; }

            if (typeof(T) == typeof(Payment)) { Payment newT1 = (Payment)(object)RequestObject; id = newT1.PaymentId; }

            var response = new HttpResponseMessage(HttpStatusCode.BadRequest);

            var responseModel = new Respone { ID = id, Message = message };

            response.Content = new StringContent(JsonConvert.SerializeObject(responseModel), System.Text.Encoding.UTF8, "application/json");

            throw new HttpResponseException(response);

        }

        private HttpResponseMessage PrepareDMLOutput(ResponseMessage action)

        {

            string Json = "";

            Json = JsonConvert.SerializeObject(action);

            return Request.CreateResponse(HttpStatusCode.OK, Json);

        }

        #endregion

        #region Login

        [HttpPost]

        [Route("api/Login/GetLoginAccess")]

        public async Task<HttpResponseMessage> GetLoginAccess(Users users)

        {

            ResponseMessage objres = new ResponseMessage();

            try

            {

                await Task.Yield();

                string Json = "";

                var response = new HttpResponseMessage();

                string command = "select UserRole from Users where UserName='" + users.Email + "' and Password='" + users.Password + "'";

                objres = Dal.GetDataforcommand(command);

                Json = JsonConvert.SerializeObject(objres);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Login/CreateLogin")]

        public async Task<HttpResponseMessage> CreateLogin(Users users)

        {

            await Task.Yield();

            ResponseMessage objres = new ResponseMessage();

            string Json = "";

            objres = Dal.InsertRecord(users);

            Json = JsonConvert.SerializeObject(objres);

            return Request.CreateResponse(HttpStatusCode.OK, Json);

        }

        #endregion

        #region Specializations

        [HttpPost]

        [Route("api/Specializations/CreateSpecializations")]

        public async Task<HttpResponseMessage> CreateSpecializations(Specializations specializations)

        {

            await Task.Yield();

            ModelState.Remove("specializations.SpecializationsId");

            if (specializations == null)

                BadRequestValidation(specializations);

            return PrepareDMLOutput(Dal.InsertRecord(specializations));

        }

        [HttpPut]

        [Route("api/Specializations/Editspecializations")]

        public async Task<HttpResponseMessage> Editspecializations(Specializations specializations)

        {

            await Task.Yield();

            if (specializations == null)

                BadRequestValidation(specializations);

            return PrepareDMLOutput(Dal.UpdateRecord(specializations));

        }

        [HttpDelete]

        [Route("api/Specializations/DeleteSpecializations/{SpecializationsId:int?}")]

        public async Task<HttpResponseMessage> DeleteSpecializations(int SpecializationsId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("Specializations", SpecializationsId));

        }

        [HttpPost]

        [Route("api/Specializations/GetSpecializations/{SpecializationsId:int?}")]

        public async Task<HttpResponseMessage> GetSpecializations(int SpecializationsId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("Specializations", SpecializationsId, "SpecializationID");

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Specializations/GetSpecializations")]

        public async Task<HttpResponseMessage> GetSpecializations()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("Specializations", 0, string.Empty);

                Json = JsonConvert.SerializeObject(objres,Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region HospitalBranch

        [HttpPost]

        [Route("api/HospitalBranch/CreateHospitalBranch")]

        public async Task<HttpResponseMessage> CreateHospitalBranch(HospitalBranch hospitalbranch)

        {

            await Task.Yield();

            if (hospitalbranch == null)

                BadRequestValidation(hospitalbranch);

            return PrepareDMLOutput(Dal.InsertRecord(hospitalbranch));

        }

        [HttpPut]

        [Route("api/HospitalBranch/EditHospitalBranch")]

        public async Task<HttpResponseMessage> EditHospitalBranch(HospitalBranch hospitalbranch)

        {

            await Task.Yield();

            if (hospitalbranch == null)

                BadRequestValidation(hospitalbranch);

            return PrepareDMLOutput(Dal.UpdateRecord(hospitalbranch));

        }

        [HttpDelete]

        [Route("api/HospitalBranch/DeleteHospitalBranch/{HospitalBranchId:int?}")]

        public async Task<HttpResponseMessage> DeleteHospitalBranch(int HospitalBranchId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("hospitals", HospitalBranchId));

        }

        [HttpPost]

        [Route("api/HospitalBranch/GetHospitalBranch/{HospitalBranchId:int?}")]

        public async Task<HttpResponseMessage> GetHospitalBranch(int HospitalBranchId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("hospitals", HospitalBranchId, "HospitalId");

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/HospitalBranch/GetHospitalBranch")]

        public async Task<HttpResponseMessage> GetHospitalBranch()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres =Dal.GetData("hospitals", 0, string.Empty);

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Doctors

        [HttpPost]

        [Route("api/Doctors/CreateDoctor")]

        public async Task<HttpResponseMessage> CreateDoctor(Doctor doctor)

        {

            await Task.Yield();

            ModelState.Remove("doctor.DoctorCode");

            if (doctor == null)

                BadRequestValidation(doctor);

            return PrepareDMLOutput(Dal.InsertRecord(doctor));

        }

        [HttpPut]

        [Route("api/Doctors/UpdateDoctor")]

        public async Task<HttpResponseMessage> UpdateDoctor(Doctor doctor)

        {

            await Task.Yield();

            if (doctor == null)

                BadRequestValidation(doctor);

            return PrepareDMLOutput(Dal.UpdateRecord(doctor));

        }

        [HttpDelete]

        [Route("api/Doctors/DeleteDoctor/{DoctorId:int?}")]

        public async Task<HttpResponseMessage> DeleteDoctor(int DoctorId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("Doctor", DoctorId));

        }

        [HttpPost]

        [Route("api/Doctors/GetDoctors/{DoctorId:int?}")]

        public async Task<HttpResponseMessage> GetDoctors(int DoctorId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("doctors", DoctorId, "DoctorId");

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Doctors/GetDoctors")]

        public async Task<HttpResponseMessage> GetDoctors()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("doctors", 0, string.Empty);

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Patients

        [HttpPost]

        [Route("api/Patients/CreatePatient")]

        public async Task<HttpResponseMessage> CreatePatient(Patient Patient)

        {

            await Task.Yield();

            ModelState.Remove("patient.PatientCode");

            if (Patient == null)

                BadRequestValidation(Patient);

            return PrepareDMLOutput(Dal.InsertRecord(Patient));

        }

        [HttpPut]

        [Route("api/Patients/UpdatePatient")]

        public async Task<HttpResponseMessage> UpdatePatient(Patient Patient)

        {

            await Task.Yield();

            if (Patient == null)

                BadRequestValidation(Patient);

            return PrepareDMLOutput(Dal.UpdateRecord(Patient));

        }

        [HttpDelete]

        [Route("api/Patients/DeletePatient/{PatientId:int?}")]

        public async Task<HttpResponseMessage> DeletePatient(int PatientId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("patients", PatientId));

        }

        [HttpPost]

        [Route("api/Patients/GetPatients/{PatientId:int?}")]

        public async Task<HttpResponseMessage> GetPatients(int PatientId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres =Dal.GetData("patients", PatientId, string.Empty);

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region PaymentMode

        [HttpPost]

        [Route("api/PaymentMode/CreatePaymentMode")]

        public async Task<HttpResponseMessage> CreatePaymentMode(PaymentMode paymentmodes)

        {

            await Task.Yield();

            if (paymentmodes == null)

                BadRequestValidation(paymentmodes);

            return PrepareDMLOutput(Dal.InsertRecord(paymentmodes));

        }

        [HttpPut]

        [Route("api/PaymentMode/EditPaymentMode")]

        public async Task<HttpResponseMessage> EditPaymentMode(PaymentMode paymentmodes)

        {

            await Task.Yield();

            if (paymentmodes == null)

                BadRequestValidation(paymentmodes);

            return PrepareDMLOutput(Dal.UpdateRecord(paymentmodes));

        }

        [HttpDelete]

        [Route("api/PaymentMode/DeletePaymentMode/{PaymentModeId:int?}")]

        public async Task<HttpResponseMessage> DeletePaymentMode(int PaymentModeId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("paymentmodes", PaymentModeId));

        }

        [HttpPost]

        [Route("api/PaymentMode/GetPaymentMode/{PaymentModeId:int?}")]

        public async Task<HttpResponseMessage> GetPaymentMode(int PaymentModeId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("paymentmodes", PaymentModeId, "PaymentModeId");

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/PaymentMode/GetPaymentMode")]

        public async Task<HttpResponseMessage> GetPaymentMode()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("paymentmodes", 0, string.Empty);

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region RoomType

        [HttpPost]

        [Route("api/RoomType/CreateRoomType")]

        public async Task<HttpResponseMessage> CreateRoomType(RoomType roomtype)

        {

            await Task.Yield();

            if (roomtype == null)

                BadRequestValidation(roomtype);

            return PrepareDMLOutput(Dal.InsertRecord(roomtype));

        }

        [HttpPut]

        [Route("api/RoomType/EditRoomType")]

        public async Task<HttpResponseMessage> EditRoomType(RoomType roomtype)

        {

            await Task.Yield();

            if (roomtype == null)

                BadRequestValidation(roomtype);

            return PrepareDMLOutput(Dal.UpdateRecord(roomtype));

        }

        [HttpDelete]

        [Route("api/RoomType/DeleteRoomType/{RoomTypeId:int?}")]

        public async Task<HttpResponseMessage> DeleteRoomType(int RoomTypeId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("roomtypes", RoomTypeId));

        }

        [HttpPost]

        [Route("api/RoomType/GetRoomType/{RoomTypeId:int?}")]

        public async Task<HttpResponseMessage> GetRoomType(int RoomTypeId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage objres = Dal.GetData("roomtypes", RoomTypeId, "RoomTypeId");

                Json = JsonConvert.SerializeObject(objres, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/RoomType/GetRoomType")]

        public async Task<HttpResponseMessage> GetRoomType()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("roomtypes", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region OperativeRoom

        [HttpPost]

        [Route("api/OperativeRoom/CreateOperativeRoom")]

        public async Task<HttpResponseMessage> CreateOperativeRoom(OperativeRoom operativeroom)

        {

            await Task.Yield();

            if (operativeroom == null)

                BadRequestValidation(operativeroom);

            return PrepareDMLOutput(Dal.InsertRecord(operativeroom));

        }

        [HttpPut]

        [Route("api/OperativeRoom/EditOperativeRoom")]

        public async Task<HttpResponseMessage> EditOperativeRoom(OperativeRoom operativeroom)

        {

            await Task.Yield();

            if (operativeroom == null)

                BadRequestValidation(operativeroom);

            return PrepareDMLOutput(Dal.UpdateRecord(operativeroom));

        }

        [HttpDelete]

        [Route("api/OperativeRoom/DeleteOperativeRoom/{OperativeRoomId:int?}")]

        public async Task<HttpResponseMessage> DeleteOperativeRoom(int OperativeRoomId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("operativerooms", OperativeRoomId));

        }

        [HttpPost]

        [Route("api/OperativeRoom/GetOperativeRoom/{OperativeRoomId:int?}")]

        public async Task<HttpResponseMessage> GetOperativeRoom(int OperativeRoomId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("operativerooms", OperativeRoomId, "OperativeRoomId");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/OperativeRoom/GetOperativeRoom")]

        public async Task<HttpResponseMessage> GetOperativeRoom()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("operativerooms", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Nurse

        [HttpPost]

        [Route("api/Nurse/CreateNurse")]

        public async Task<HttpResponseMessage> CreateNurse(Nurse nurse)

        {

            await Task.Yield();

            if (nurse == null)

                BadRequestValidation(nurse);

            return PrepareDMLOutput(Dal.InsertRecord(nurse));

        }

        [HttpPut]

        [Route("api/Nurse/EditNurse")]

        public async Task<HttpResponseMessage> EditNurse(Nurse nurse)

        {

            await Task.Yield();

            if (nurse == null)

                BadRequestValidation(nurse);

            return PrepareDMLOutput(Dal.UpdateRecord(nurse));

        }

        [HttpDelete]

        [Route("api/Nurse/DeleteNurse/{NurseId:int?}")]

        public async Task<HttpResponseMessage> DeleteNurse(int NurseId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("nurse", NurseId));

        }

        [HttpPost]

        [Route("api/Nurse/GetNurse/{NurseId:int?}")]

        public async Task<HttpResponseMessage> GetNurse(int NurseId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("nurses", NurseId, "NurseId");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Nurse/GetNurse")]

        public async Task<HttpResponseMessage> GetNurse()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("nurses", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Medicines

        [HttpPost]

        [Route("api/Medicines/CreateMedicines")]

        public async Task<HttpResponseMessage> CreateMedicines(Medicine medicines)

        {

            await Task.Yield();

            if (medicines == null)

                BadRequestValidation(medicines);

            return PrepareDMLOutput(Dal.InsertRecord(medicines));

        }

        [HttpPut]

        [Route("api/Medicines/EditMedicines")]

        public async Task<HttpResponseMessage> EditMedicines(Medicine medicines)

        {

            await Task.Yield();

            if (medicines == null)

                BadRequestValidation(medicines);

            return PrepareDMLOutput(Dal.UpdateRecord(medicines));

        }

        [HttpDelete]

        [Route("api/Medicines/DeleteMedicines/{MedicinesId:int?}")]

        public async Task<HttpResponseMessage> DeleteMedicines(int MedicinesId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("Medicines", MedicinesId));

        }

        [HttpPost]

        [Route("api/Medicines/GetMedicines/{MedicinesId:int?}")]

        public async Task<HttpResponseMessage> GetMedicines(int MedicinesId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("medicines", MedicinesId, "MedicinesId");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Medicines/GetMedicines")]

        public async Task<HttpResponseMessage> GetMedicines()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("medicines", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Treatment

        [HttpPost]

        [Route("api/Treatment/CreateTreatment")]

        public async Task<HttpResponseMessage> CreateTreatment(Treatment treatment)

        {

            await Task.Yield();

            if (treatment == null)

                BadRequestValidation(treatment);

            return PrepareDMLOutput(Dal.InsertRecord(treatment));

        }

        [HttpPut]

        [Route("api/Treatment/EditTreatment")]

        public async Task<HttpResponseMessage> EditTreatment(Treatment treatment)

        {

            await Task.Yield();

            if (treatment == null)

                BadRequestValidation(treatment);

            return PrepareDMLOutput(Dal.UpdateRecord(treatment));

        }

        [HttpDelete]

        [Route("api/Treatment/DeleteTreatment/{TreatmentId:int?}")]

        public async Task<HttpResponseMessage> DeleteTreatment(int TreatmentId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("treatments", TreatmentId));

        }

        [HttpPost]

        [Route("api/Treatment/GetTreatment/{TreatmentId:int?}")]

        public async Task<HttpResponseMessage> GetTreatment(int TreatmentId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("treatments", TreatmentId, "treatmentid");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Treatment/GetTreatment")]

        public async Task<HttpResponseMessage> GetTreatment()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("treatments", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Payment

        [HttpPost]

        [Route("api/Payment/CreatePayment")]

        public async Task<HttpResponseMessage> CreatePayment(Payment Payment)

        {

            await Task.Yield();

            if (Payment == null)

                BadRequestValidation(Payment);

            return PrepareDMLOutput(Dal.InsertRecord(Payment));

        }

        [HttpPut]

        [Route("api/Payment/EditPayment")]

        public async Task<HttpResponseMessage> EditPayment(Payment Payment)

        {

            await Task.Yield();

            if (Payment == null)

                BadRequestValidation(Payment);

            return PrepareDMLOutput(Dal.UpdateRecord(Payment));

        }

        [HttpDelete]

        [Route("api/Payment/DeletePayment/{PaymentId:int?}")]

        public async Task<HttpResponseMessage> DeletePayment(int PaymentId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("payments", PaymentId));

        }

        [HttpPost]

        [Route("api/Payment/GetPayment/{PaymentId:int?}")]

        public async Task<HttpResponseMessage> GetPayment(int PaymentId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("payments", PaymentId, "paymentId");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Payment/GetPayment")]

        public async Task<HttpResponseMessage> GetPayment()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("payments", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Admission

        [HttpPost]

        [Route("api/Admission/CreateAdmission")]

        public async Task<HttpResponseMessage> CreateAdmission(Admission admission)

        {

            await Task.Yield();

            if (admission == null)

                BadRequestValidation(admission);

            return PrepareDMLOutput(Dal.InsertRecord(admission));

        }

        [HttpPut]

        [Route("api/Admission/EditAdmission")]

        public async Task<HttpResponseMessage> EditAdmission(Admission admission)

        {

            await Task.Yield();

            if (admission == null)

                BadRequestValidation(admission);

            return PrepareDMLOutput(Dal.UpdateRecord(admission));

        }

        [HttpDelete]

        [Route("api/Admission/DeleteAdmission/{AdmissionId:int?}")]

        public async Task<HttpResponseMessage> DeleteAdmission(int AdmissionId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("Admission", AdmissionId));

        }

        [HttpPost]

        [Route("api/Admission/GetAdmission/{AdmissionId:int?}")]

        public async Task<HttpResponseMessage> GetAdmission(int AdmissionId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("admissions", AdmissionId, "Admissionid");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Admission/GetAdmission")]

        public async Task<HttpResponseMessage> GetAdmission()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("admissions", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Appointment

        [HttpPost]

        [Route("api/Appointment/CreateAppointment")]

        public async Task<HttpResponseMessage> CreateAppointment(Appointment appointment)

        {

            await Task.Yield();

            if (appointment == null)

                BadRequestValidation(appointment);

            return PrepareDMLOutput(Dal.InsertRecord(appointment));

        }

        [HttpPut]

        [Route("api/Appointment/EditAppointment")]

        public async Task<HttpResponseMessage> EditAppointment(Appointment appointment)

        {

            await Task.Yield();

            if (appointment == null)

                BadRequestValidation(appointment);

            return PrepareDMLOutput(Dal.UpdateRecord(appointment));

        }

        [HttpDelete]

        [Route("api/Appointment/DeleteAppointment/{AppointmentId:int?}")]

        public async Task<HttpResponseMessage> DeleteAppointment(int AppointmentId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("appointment", AppointmentId));

        }

        [HttpPost]

        [Route("api/Appointment/GetAppointment/{AppointmentId:int?}")]

        public async Task<HttpResponseMessage> GetAppointment(int AppointmentId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("appointments", AppointmentId, "Appointmentid");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Appointment/GetAppointment")]

        public async Task<HttpResponseMessage> GetAppointment()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("appointments", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region Appointmentmedicine

        [HttpPost]

        [Route("api/Appointmentmedicine/CreateAppointmentmedicine")]

        public async Task<HttpResponseMessage> CreateAppointmentmedicine(Appointmentmedicine appointmentmedicine)

        {

            await Task.Yield();

            if (appointmentmedicine == null)

                BadRequestValidation(appointmentmedicine);

            return PrepareDMLOutput(Dal.InsertRecord(appointmentmedicine));

        }

        [HttpPut]

        [Route("api/Appointmentmedicine/EditAppointmentmedicine")]

        public async Task<HttpResponseMessage> EditAppointmentmedicine(Appointmentmedicine appointmentmedicine)

        {

            await Task.Yield();

            if (appointmentmedicine == null)

                BadRequestValidation(appointmentmedicine);

            return PrepareDMLOutput(Dal.UpdateRecord(appointmentmedicine));

        }

        [HttpDelete]

        [Route("api/Appointmentmedicine/DeleteAppointmentmedicine/{AppointmentmedicineId:int?}")]

        public async Task<HttpResponseMessage> DeleteAppointmentmedicine(int AppointmentmedicineId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("Trt\_Medication", AppointmentmedicineId));

        }

        [HttpPost]

        [Route("api/Appointmentmedicine/GetAppointmentmedicine/{AppointmentmedicineId:int?}")]

        public async Task<HttpResponseMessage> GetAppointmentmedicine(int AppointmentmedicineId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("appointmentmedicines", AppointmentmedicineId, "ApptMedicineId");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/Appointmentmedicine/GetAppointmentmedicine")]

        public async Task<HttpResponseMessage> GetAppointmentmedicine()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("appointmentmedicines", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

        #region TreatmentMedicine

        [HttpPost]

        [Route("api/TreatmentMedicine/CreateTreatmentMedicine")]

        public async Task<HttpResponseMessage> CreateTreatmentMedicine(TreatmentMedicine treatmentMedicine)

        {

            await Task.Yield();

            if (treatmentMedicine == null)

                BadRequestValidation(treatmentMedicine);

            return PrepareDMLOutput(Dal.InsertRecord(treatmentMedicine));

        }

        [HttpPut]

        [Route("api/TreatmentMedicine/EditTreatmentMedicine")]

        public async Task<HttpResponseMessage> EditTreatmentMedicine(TreatmentMedicine treatmentMedicine)

        {

            await Task.Yield();

            if (treatmentMedicine == null)

                BadRequestValidation(treatmentMedicine);

            return PrepareDMLOutput(Dal.UpdateRecord(treatmentMedicine));

        }

        [HttpDelete]

        [Route("api/TreatmentMedicine/DeleteTreatmentMedicine/{TreatmentMedicineId:int?}")]

        public async Task<HttpResponseMessage> DeleteTreatmentMedicine(int TreatmentMedicineId)

        {

            await Task.Yield();

            return PrepareDMLOutput(Dal.DeleteRecord("Trt\_Medication", TreatmentMedicineId));

        }

        [HttpPost]

        [Route("api/TreatmentMedicine/GetTreatmentMedicine/{TreatmentMedicineId:int?}")]

        public async Task<HttpResponseMessage> GetTreatmentMedicine(int TreatmentMedicineId = 0)

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("treamentmedicines", TreatmentMedicineId, "TreatmentMedicineId");

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        [HttpPost]

        [Route("api/TreatmentMedicine/GetTreatmentMedicine")]

        public async Task<HttpResponseMessage> GetTreatmentMedicine()

        {

            try

            {

                await Task.Yield();

                string Json = "";

                ResponseMessage data = Dal.GetData("treamentmedicines", 0, string.Empty);

                Json = JsonConvert.SerializeObject(data, Formatting.Indented);

                return Request.CreateResponse(HttpStatusCode.OK, Json);

            }

            catch (Exception ex)

            {

                return Request.CreateResponse(HttpStatusCode.ExpectationFailed, ex.InnerException.Message);

            }

        }

        #endregion

    }

}