Test Repo

```
using CrudwithGeneric.Interface;
using CrudwithGeneric.Models;
using System.Data;
using System.Data.SqlClient;
using System.Reflection;

namespace CrudwithGeneric.Repositories
{
    public class TestRepo : ITest
    {
        private readonly Crudbase _crudbase;
        private readonly IConfiguration _configuration;
        private readonly string conn;
        public TestRepo(Crudbase crudbase, IConfiguration _configuration)
        {
            this._configuration = _configuration;
            _crudbase = crudbase;
            conn = _configuration.GetConnectionString("DefaultConnection");

        }
        public List<Student> getdata()
        {
            Student student = new Student();
            student.Name = "Mird";
            student.Id = 1;
            student.Description = "latest";
            student.StudentId = 2;
            var prams = returnSppram(student);
            var data = _crudbase.ExecuteProc(conn, "AddStudent",prams);
            var students = ConverttoObject(data, typeof(Student));
            return students;
        }
        private List<SqlParameter> returnSppram(Object obj)
        {
            List<SqlParameter> parameters = new List<SqlParameter>();

            Type objectType = obj.GetType();
            PropertyInfo[] properties = objectType.GetProperties();

            foreach (PropertyInfo property in properties)
            {
                parameters.Add(new System.Data.SqlClient.SqlParameter("@" +
property.Name, property.GetValue(obj)));
            }
            return parameters;

        }
        private List<Object> ConverttoObject(DataTable data, Type objectType)
        {
            List<Object> objectList = new List<Object>();

            foreach (DataRow row in data.Rows)
            {
                object obj = Activator.CreateInstance(objectType);

                foreach (DataColumn column in data.Columns)
```

```
                {
                    string propertyName = column.ColumnName;
                    object propertyValue = row[column];

                    PropertyInfo property = objectType.GetProperty(propertyName);
                    if (property != null && property.CanWrite)
                    {
                        property.SetValue(obj, propertyValue);
                    }
                }

                objectList.Add(obj);
            }

            return objectList;
        }
    }
}
```

----------calling

```
using System.Data;
using System.Data.SqlClient;

namespace CrudwithGeneric.Repositories
{
    public class Crudbase
    {
        private readonly IConfiguration _configuration;
        public Crudbase(IConfiguration configuration)
        {

            _configuration = configuration;

        }
        public DataTable ExecuteProc(string ConnectionString, string pProcedureName,
List<SqlParameter> param)
        {
            DataTable data=new DataTable();
            using (SqlConnection mobjConnection = new
SqlConnection(ConnectionString))
            {
                using (SqlCommand mobjCommand = new SqlCommand(pProcedureName,
mobjConnection))
                {
                    mobjCommand.CommandType = CommandType.StoredProcedure;
                    //mobjCommand.CommandTimeout = CommandTimeOut;// 20;
                    if (param != null)
                    {
                        for (int i = 0; i < param.Count; i++)
                        {
                            if (param[i] != null)
                            {
```

```
                            mobjCommand.Parameters.Add(param[i]);
                        }
                    }
                }
                try
                {
                    mobjConnection.Open();
                    SqlDataAdapter mobjAdapter;
                    mobjAdapter = new SqlDataAdapter(mobjCommand);
                    DataSet mDataSet = new DataSet();
                    mobjAdapter.Fill(mDataSet);
                    data = mDataSet.Tables[0];


                }
                catch (System.Data.SqlClient.SqlException Ex)
                {

                }
                catch (Exception ex)
                {
                    //SMS800.Utilities.dataExceptionHandler objData = new
SMS800.Utilities.dataExceptionHandler();
                    //string strMessage = objData.dataException("data_1", ex);
                    //SMS800.Utilities.logger objLogger = new
SMS800.Utilities.logger();
                    //objLogger.Log(objConfig.getLogPath(), strMessage + ". ----
---- Exception Raised Is : " + ex.Message.ToString());
                }
                finally
                {
                    mobjCommand.Dispose();
                    mobjConnection.Close();
                    mobjConnection.Dispose();
                }
                return data;
            }
        }
    }
}
```