```
 1  using Microsoft.EntityFrameworkCore;
 2  using PDLERP.Models;
 3  using PDLERP.ServiceInterfaces.BaseInterfaces;
 4  using System;
 5  using System.Collections.Generic;
 6  using System.Linq;
 7  using System.Threading.Tasks;
 8
 9  namespace PDLERP.ServiceInfrastructures.BaseInfrastructures
10  {
11      public class BaseRepository<TEntity> : IBaseService<TEntity> where
          TEntity : class
12      {
13          protected readonly PDLERPContext _pDLERPContext;
14
15          public BaseRepository(PDLERPContext pDLERPContext)
16          {
17              _pDLERPContext = pDLERPContext;
18          }
19          public async Task<IQueryable<TEntity>> All()
20          {
21              try
22              {
23                  var result = await _pDLERPContext.Set<TEntity>().AsNoTracking
                      ().ToListAsync();
24                  return result.AsQueryable();
25              }
26              catch (Exception e)
27              {
28                  Console.WriteLine(e);
29                  throw;
30              }
31          }
32
33          public async Task<bool> Delete(TEntity entity)
34          {
35              try
36              {
37                  _pDLERPContext.Set<TEntity>().Remove(entity);
38                  await SaveChanges();
39                  return true;
40              }
41              catch (Exception e)
42              {
43                  Console.WriteLine(e);
44                  throw;
45              }
46          }
47
```

```
48          public async Task<bool> DeleteRange(IEnumerable<TEntity> entities)
49          {
50              try
51              {
52                  _pDLERPContext.Set<TEntity>().RemoveRange(entities);
53                  await SaveChanges();
54                  return true;
55              }
56              catch (Exception e)
57              {
58                  Console.WriteLine(e);
59                  throw;
60              }
61          }
62
63          public async Task<TEntity> FindByIdAsync(int id)
64          {
65              try
66              {
67                  _pDLERPContext.ChangeTracker.QueryTrackingBehavior =
                      QueryTrackingBehavior.NoTracking;
68                  return await _pDLERPContext.Set<TEntity>().FindAsync(id);
69              }
70              catch (Exception ex)
71              {
72                  Console.Write(ex.Message);
73                  return null;
74              }
75          }
76
77          public async Task<IEnumerable<TEntity>> GetAll()
78          {
79              try
80              {
81                  return await _pDLERPContext.Set<TEntity>().ToListAsync();
82              }
83              catch (Exception e)
84              {
85                  Console.WriteLine(e);
86                  throw;
87              }
88          }
89
90          public async Task<TEntity> GetInsertedObjByAsync(TEntity entity)
91          {
92              try
93              {
94                  await _pDLERPContext.Set<TEntity>().AddAsync(entity);
95                  await SaveChanges();
```

```
96                  return entity;
97              }
98              catch (Exception ex)
99              {
100                 Console.Write(ex.Message);
101                 return null;
102             }
103         }
104
105         public async Task<bool> InsertByAsync(TEntity entity)
106         {
107             try
108             {
109                 await _pDLERPContext.Set<TEntity>().AddAsync(entity);
110                 await SaveChanges();
111                 return true;
112             }catch(Exception ex)
113             {
114                 Console.Write(ex.Message);
115                 return false;
116             }
117         }
118
119         public async Task<bool> InsertRangeByAsync(IEnumerable<TEntity>
              entities)
120         {
121             try
122             {
123                 await _pDLERPContext.Set<TEntity>().AddRangeAsync(entities);
124                 await SaveChanges();
125                 return true;
126             }catch(Exception ex)
127             {
128                 Console.Write(ex.Message);
129                 return false;
130             }
131         }
132
133         public async Task<bool> Update(TEntity entity)
134         {
135             try
136             {
137                 var result = _pDLERPContext.Set<TEntity>().Attach(entity);
138                 result.State = EntityState.Modified;
139                 await SaveChanges();
140                 return true;
141             }catch(Exception ex)
142             {
143                 Console.Write(ex.Message);
```

```
144                     return false;
145                 }
146             }
147
148             // UNDER DEVELOPMENT
149             public async Task<bool> UpdateRangeByAsync(IEnumerable<TEntity>
                  entities)
150             {
151                 _pDLERPContext.Set<TEntity>().UpdateRange(entities);
152                 await SaveChanges();
153                 return true;
154             }
155
156             public async Task<int> SaveChanges()
157             {
158                 return await _pDLERPContext.SaveChangesAsync();
159             }
160
161             public void Dispose()
162             {
163                 _pDLERPContext.Dispose();
164             }
165         }
166     }
167
```