public async Task<List<ProductDetailsDTO>> GetListUnitCost()

{

try

{

List<ProductDetails> oProductDetails = await \_productRepository.GetListAsync();

List<Guid> gproductId = new List<Guid>();

gproductId.AddRange(oProductDetails.Select(s => s.Id).ToList());

var vHcpcCode = \_ProductHcpcRepository.Where(i => gproductId.Contains(i.productId)).ToList();

var vUserData = await \_userRepository.GetListAsync();

var productDetails = (from product in oProductDetails

//where (prodet.Hide == 0 || prodet.Hide == null) && (prodet.Inactive == 0 || prodet.Inactive == null)

select new ProductDetailsDTO

{

mainProductName = product.productDescription,

hcpcCode = string.Join(", ", vHcpcCode.Where(t => t.productId == product.Id && t.status == Status.Active).Select(a => a.hcpcCode)),

productCode = product.productCode != null && product.productCode != "" ? product.productCode : "-",

UnitCost = product.UnitCost != null ? Decimal.Round(product.UnitCost.Value, 2) : 0,

Id = product.Id,

CreationTime = product.CreationTime == DateTime.MinValue ? null : product.CreationTime,

LastModificationTime = product.LastModificationTime != null && product.LastModificationTime != default(DateTime) ? product.LastModificationTime : product.CreationTime,

createdBy = vUserData.Where(x => x.Id == product.CreatorId).Select(t => t.Name).SingleOrDefault() ?? "-",

modifiedBy = product.LastModifierId != null && product.LastModifierId != Guid.Empty ? vUserData.Where(x => x.Id == product.LastModifierId).Select(t => t.Name).SingleOrDefault() ?? "-" : vUserData.Where(x => x.Id == product.CreatorId).Select(t => t.Name).SingleOrDefault() ?? "-",

}).ToList();

return productDetails;

}

catch (Exception ex)

{

throw new UserFriendlyException(ex.Message);

}

}

public async Task<ProductDetailsDTO> UpdateUnitCost(Guid gProductId,string productCode,decimal unitCost)

{

try

{

if(gProductId==Guid.Empty || gProductId == null)

{

throw new UserFriendlyException("Product ID is Required");

}

if(unitCost== 0 || unitCost == null)

{

throw new UserFriendlyException("Unit cost must not be zero");

}

if(productCode == string.Empty)

{

throw new UserFriendlyException("product Code must not be empty");

}

var data = \_productRepository.Where(x => x.Id == gProductId && x.productCode==productCode ).FirstOrDefault();

data.UnitCost = unitCost;

await \_productRepository.UpdateAsync(data);

return ObjectMapper.Map<ProductDetails,ProductDetailsDTO>(data);

}

catch(Exception ex)

{

throw new UserFriendlyException(ex.Message);

}

}