HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

MÔN: PHÂN TÍCH THIẾT KẾ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

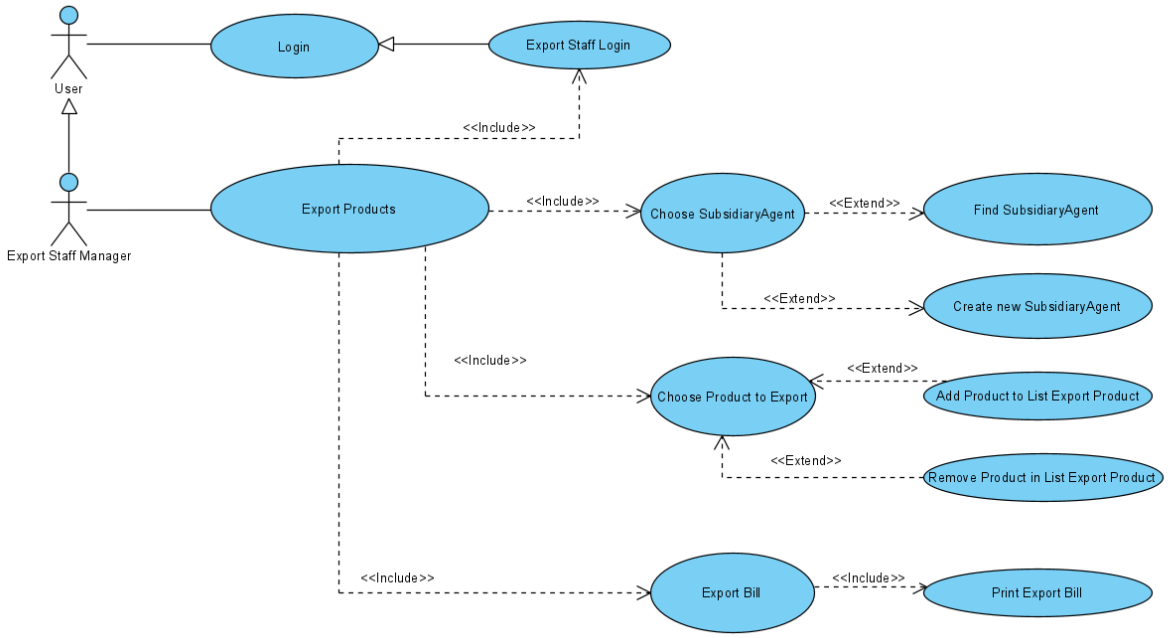
Hệ thống xuất nhập hàng tại một đại lí trung gian

Module: 4.3 Quản lý xuất hàng

Lớp: E16CN
Họ tên: Nguyễn Văn Huy
Mã SV: B16DCDT112

I. Vẽ biểu đồ usecase, mô tả use cho module.

+ Use Case: Product Export Manage



+ Mô tả:

- Đăng nhập → UC đăng nhập
- Export Products → Chọn chức năng xuất hàng
- Create new SubsidiaryAgent → Tạo mới đại lí con (TH chưa có).
- Find SubsidiaryAgent → Tìm kiếm đại lí con
- Find Product to Export → Tìm hàng để xuất
- Add Product to List Export Products → Thêm hàng vào danh sách muốn xuất
- Submit Result → Submit
- Success and Print Export Bill → báo thành công và in hóa đơn xuất.

II. Kịch bản (Scenario)

Use case	Export Products
Actor	Quản lý xuất kho
Tiền điều kiện	Quản lý xuất kho đăng nhập thành công.
Hậu điều kiện	Quản lý xuất hàng thành công và hóa đơn được in.
Kịch bản chính	1. Nhân viên login thành công vào hệ thống 2. Giao diện quản lý xuất hàng xuất hiện. 3. Nhân viên chọn menu xuất hàng

4. Giao diện danh sách ĐLC hiện ra với ô tìm kiếm đại lí con (ĐLC) :

Đại lí con: …….. **Tìm**

Nút: **Thêm mới ĐLC**

STT	***Tên Đại lí con***	***Người đứng tên***	***SĐT***	***Địa chỉ***	***Chọn***
1	Hung PC	Nguyen Van Hung	0456456454	Ha Noi	Chọn
2	Hưng Laptop	Nguyen Van Hưng	0445454777	Hai Duong	Chọn
3	Ratio PC	Nguyen Van Huy	0994764564	Ha Noi	Chọn
4	Ratio Phone	Nguyen Van Huy	0994586999	Hai Duong	Chọn
5	Ratio Store	Nguyen Van Huy	0977470992	Ha Noi	Chọn

5. NV nhập tên ĐL và click tìm : "Ratio".
6. Hệ thống hiện lên danh sách các ĐL có tên chứa tên vừa nhập :

STT	***Tên Đại lí con***	***Người đứng tên***	***SĐT***	***Địa chỉ***	***Chọn***
1	Ratio PC	Nguyen Van Huy	0994764564	Ha Noi	Chọn
2	Ratio Phone	Nguyen Van Huy	0994586999	Hai Duong	Chọn
3	Ratio Store	Nguyen Van Huy	0977470992	Ha Noi	Chọn

7. NV click chọn ĐLC: Ratio Store
8. Hệ thống hiện lên giao diện tìm hàng xuất của ĐLC Ratio Store

ĐẠI LÍ : RATIO STORE

STT	***Tên sản phẩm***	***Loại sản phẩm***	***Nhà cung cấp***	***Số lượng***	***Giá***	***Chọn***
1	Samsung Note 20	Điện thoại	Samsung Viet Nam cơ sở Hà Nội	20	20.000.000	Chọn
2	Tivi LG 50 inch	Tivi	LG Việt Nam	10	10.000.000	Chọn

3	Đồng hồ Daniel	Đồng hồ	Daniel Wellington Việt Nam	50	5.000.000	Chọn

9. NV nhập tên hàng: samsung note 20 và click tìm
10. Hệ thống hiện lên danh sách các MH có tên chứa từ khóa vừa nhập

STT	***Tên sản phẩm***	***Loại sản phẩm***	***Nhà cung cấp***	***Số lượng***	***Giá***	***Chọn***
1	Samsung Note 20	Điện thoại	Samsung Viet Nam cơ sở Hà Nội	20	20.000.000	Chọn

11. Nhân viên chọn hàng trong danh sách. : Samsung Note 20
12. Giao diện them vào MH xuất xuất hiện :
    Tên hàng: Samsung Note 20
    Loại SP: điện thoại
    NCC: Samsung Việt Nam cơ sở Hà Nội
    Số lượng: ……..
    Giá : …… vnd
    Nút: Add
13. Nhân viên quản lý điền thông tin:
    Tên hàng: Samsung Note 20
    Loại SP: điện thoại
    NCC: Samsung Việt Nam cơ sở Hà Nội
    Số lượng: 10.
    Giá : 29.000.000 vnd

    Click Nút: Add

14. MH xuất hiện vào danh sách MH xuất trong hóa đơn :

ĐẠI LÍ CON: RATIO STORE

STT	***Tên sản phẩm***	***Loại sản phẩm***	***Nhà cung cấp***	***Số lượng***	***Giá***	***Chọn***
1	Samsung Note 20	Điện thoại	Samsung Viet Nam cơ sở Hà Nội	20	29.000.000	Delete

Tổng tiền: 580.000.000 vnd

	Tổng sp: 20 sp Nút **Export Products** 15. NV Lặp lại các bước 8-> 13 để tìm hàng cần xuất. Sau khi tìm đủ click Export Products. 16. Hệ thống báo xuất thành công và in ra hóa đơn xuất như đã mô tả.
Ngoại lệ	7. Đại lí con chưa tồn tại 7. Nhân viên chọn nhầm đại lí con 10. Danh sách mặt hàng xuất trống.(do mặt hàng chưa tồn tại). 13. Nhân viên quản lý xuất điền nhầm số lượng/giá của sản phẩm.

III. Trích lớp thực thể

1. Trích danh từ

- **B1: Mô tả hệ thống**

Mỗi hàng hóa (Mã hàng, tên, mô tả) có thể được nhập nhiều lần khác nhau. Mỗi hàng hóa có thể xuất đi nhiều lần khác nhau, mỗi lần cho các đại lí con (mã ĐL, tên ĐL, địa chỉ, số ĐT) khác nhau, với số lượng khác nhau và giá xuất khác nhau. Mỗi lần xuất có thể xuất nhiều hàng khác nhau, miễn sao số lượng xuất không vượt quá số lượng hàng còn trong kho. Mỗi lần xuất có một phiếu xuất ghi thông tin đại lí con, tiếp theo là danh sách các mặt hàng xuất đi, mỗi mặt hàng có đầy đủ thông tin: mã hàng, tên hàng, số lượng, đơn giá, thành tiền (tự động tính) và dòng cuối cùng là tổng tiền của hóa đơn xuất.

- **B2: Trích danh từ**

Danh từ chỉ người: quản lí xuất kho, đại lí con, người, nhà cung cấp.

Danh từ chỉ vật : hang hóa, phiếu xuất, danh sách hang xuất, hóa đơn xuất, kho

Danh từ liên quan đến thông tin: mã hang, tên, mô tả, mã ĐL, tên ĐL, địa chỉ, số ĐT, số lượng, hang trong kho, mã hang, tên hang, số lượng, đơn giá, thành tiền, tổng tiền

- **B3: Đánh giá lựa chọn**

Danh từ chỉ người:

+ người-> people : tên, số đt, địa chỉ, email, tên đăng nhập, mật khẩu, vai trò, ghi chú

+ quản lí xuất kho-> kế thừa people, them thuộc tính salary,

+ đại lí con -> kế thừa people, them thuộc tính tên đại lí, tổng sản phẩm

+ nhà cung cấp -> kế thừa people, them thuộc tính mã NCC, tên NCC

Danh từ chỉ vật:

+ hang hóa -> lớp product : mã hang, tên hang, số lượng, đơn giá, ảnh, nhà cung cấp, ngày nhập, loại sản phẩm

+ phiếu xuất -> lớp phiếu xuất:mã xuất, danh sách hang xuất, tổng tiền, số lượng, ngày xuất, đại lí con

+ hóa đơn xuất -> lớp Bill Export : danh sách hang xuất, tổng tiền,

+ kho -> lớp Stock : tên kho, hang hóa(MaSP, ảnh, tên sp, mô tả, nhà cung cấp, số lượng, giá), tổng sp.

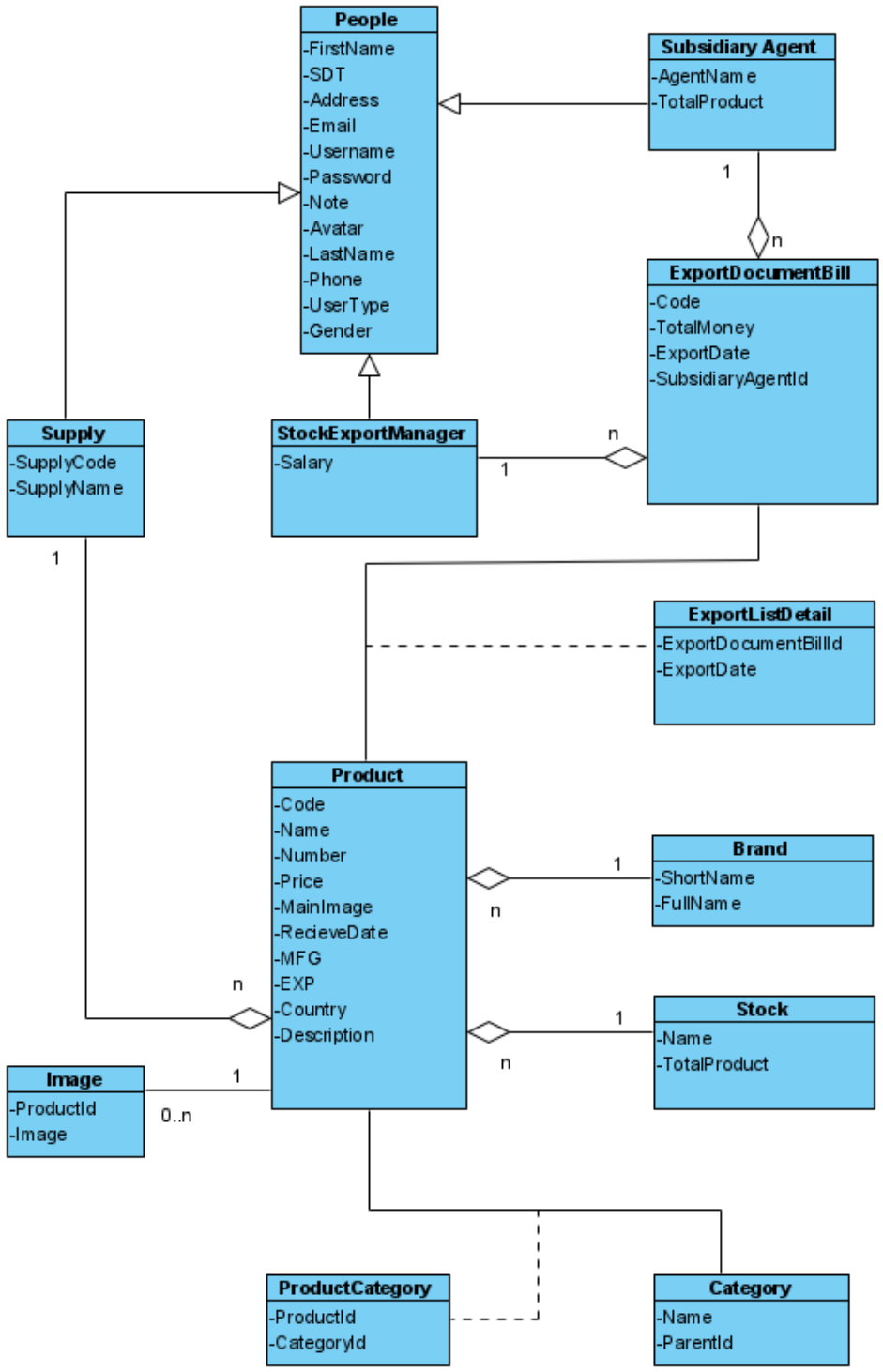
B4: Xác định quan hệ số lượng giữa các thực thể:

- 1 hàng thuộc nhiều loại hàng, 1 loại hàng có nhiều mặt hàng -> Product – Category: n-n
- 1 sản phẩm thuộc 1 kho, 1 kho có nhiều sản phẩm -> Stock - Product: 1 - n
- 1 hãng có nhiều sản phẩm : Brand – Product: 1-n
- 1 sản phẩm có nhiều ảnh: Product – Image : 1-n
- 1 phiếu xuất có nhiều sản phẩm, 1 sản phẩm có thể có trong nhiều phiếu xuất khác nhau -> ExportProduct - Product: n-n;
- 1 phiếu xuất có 1 hóa đơn xuất : ExportProduct – Bill: 1-1.

B5: Xác định quan hệ đối tượng giữa các thực thể

- Product và Category liên kiết tạo ra ProductCategory
- ExportProduct và Product liên kết tạo ra ExportListDetail
- Product là thành phần của Stock
- Product là thành phần của Brand
- Image là thành phần của Product
- Bill là thành phần của ExportProduct

2. Vẽ biểu đồ lớp thực thể pha phân tích



IV. TRÍCH LỚP BIÊN VÀ ĐIỀU KHIỂN

- **Biểu đồ lớp**

- First Step**: Page Login (Export Staff Manager):**
  Input: Username, password
  Button: Login, Register
  + **Page Register Account**
  Input: Username, Password, FirstName, LastName, Age, Gender, UserType(Supply, SubsidiaryAgent, StockExportManager)

Button : Submit, Back.

- Step 2: Export Products

  -> **Page Export Products** :

  + Find SubsidiagyAgent, Find Product to Export

  ➔ Input: 2 field search for Find SubsidiagyAgent, Find Product to Export

  ➔ Table Product of SubsidiaryAgent

  Situation: SubsidiaryAgent not exist ->

  + Create new SubsidiaryAgent

  -> **Page SubsidiaryAgent**

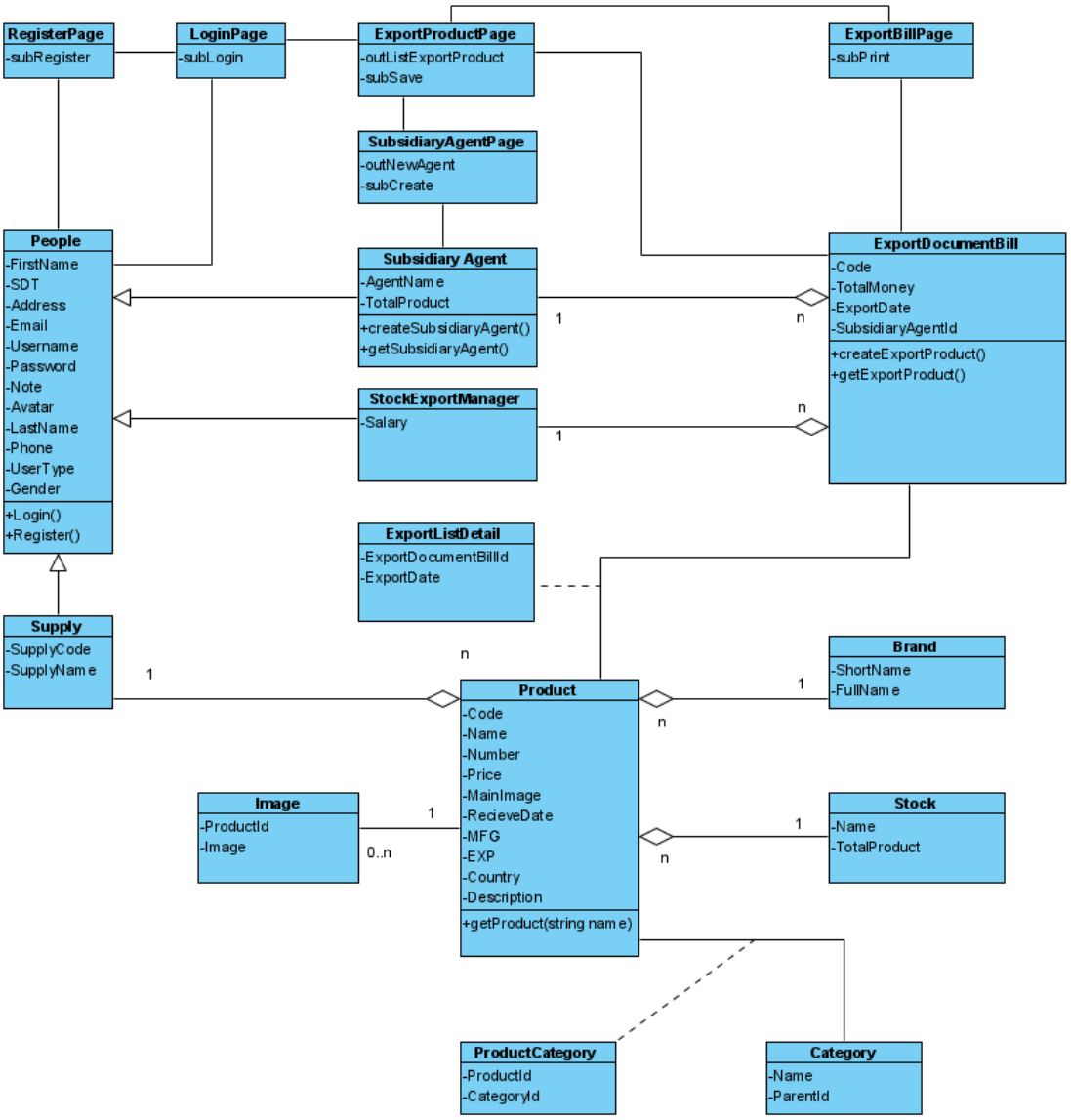
  List SubsidiaryAgent.

  Input: SubsidiaryAgent name, Button: Add SubsidiaryAgent.

- Step 3: Print Export Bill

  -> **page Bill Export.**

  Include: List product to export, subsidiaryAgent

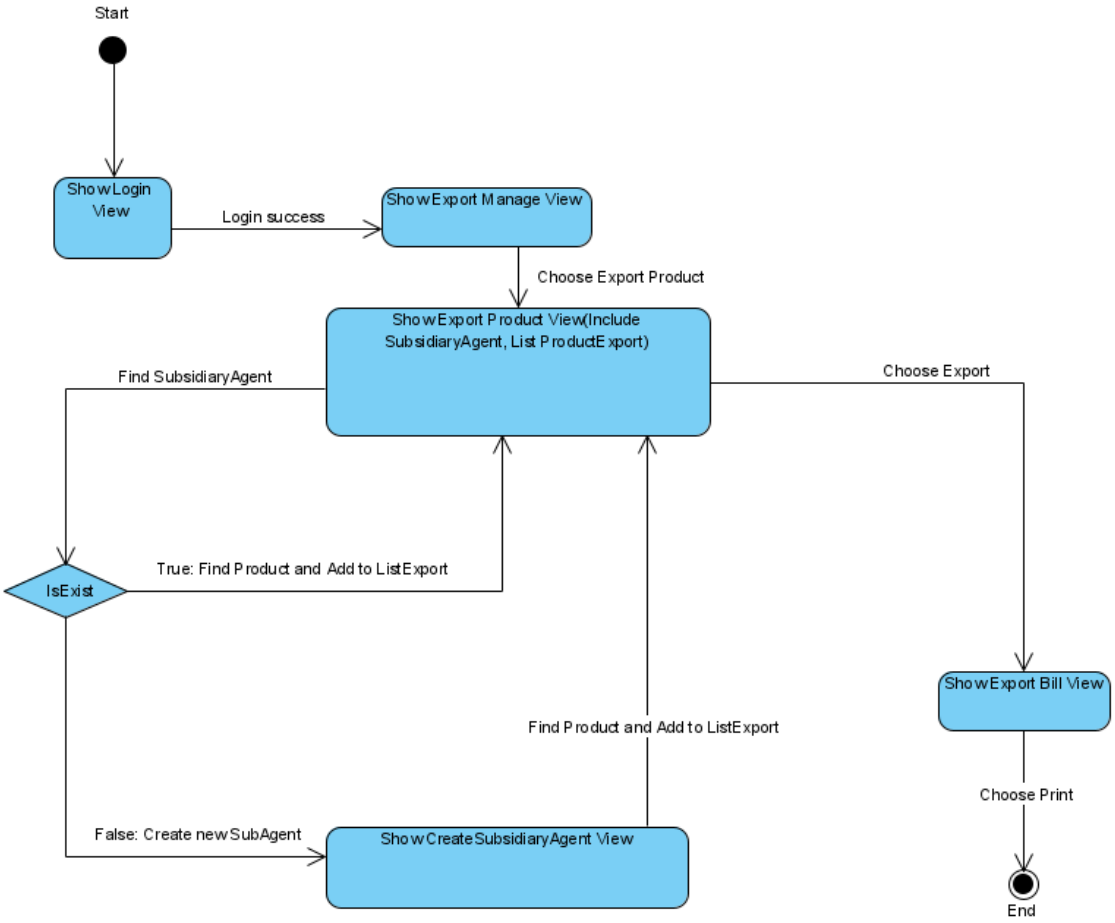


Biểu đồ lớp

V. PHÂN TÍCH HOẠT ĐỘNG

- **Biểu đồ trạng thái:**

- Tại giao diện login, khi login thành công -> hệ thống chuyển giao diện Export Manage.
- Tại giao diện Export Manage -> user chọn export product trên navigation
- Tại giao diện Export Product -> User tìm SubsidiaryAgent
  + Nếu chưa tồn tại -> User chọn Thêm mới SubsidiaryAgent
  + User chọn SubsidiaryAgent
- Tại giao diện Export Product -> User tìm sản phẩm, add sản phẩm Export vào danh sách
- Tại giao diện Export Product -> User chọn Export
- Tại giao diện Export Bill -> User chọn Print.



Biểu đồ trạng thái

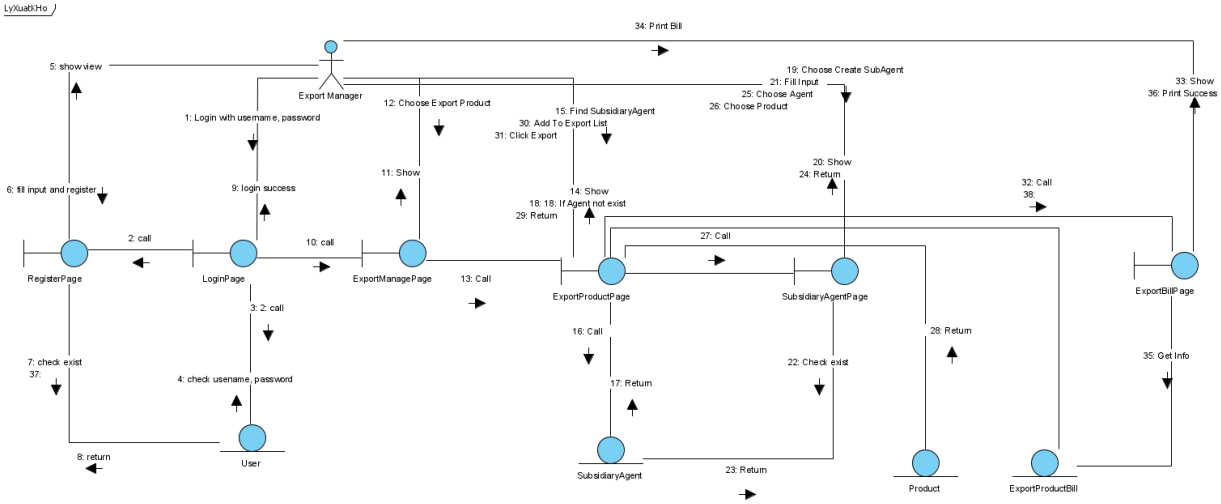
- **Biểu đồ giao tiếp :**

- **Scenario:**

1. Nhân viên login điền username, password và click login
2. Login Page call User
3. UserClass check username, password and return
4. Login Page show success to User
   4.1. Login Page fail. Return login page.

4.2 Export manager click Register
4.3. RegsterPage show
4.4. User Fill Infor and Click Register
4.5 Register Page call User
4.6 User check username exist ? , validation infor and return
4.7 Register show success to user

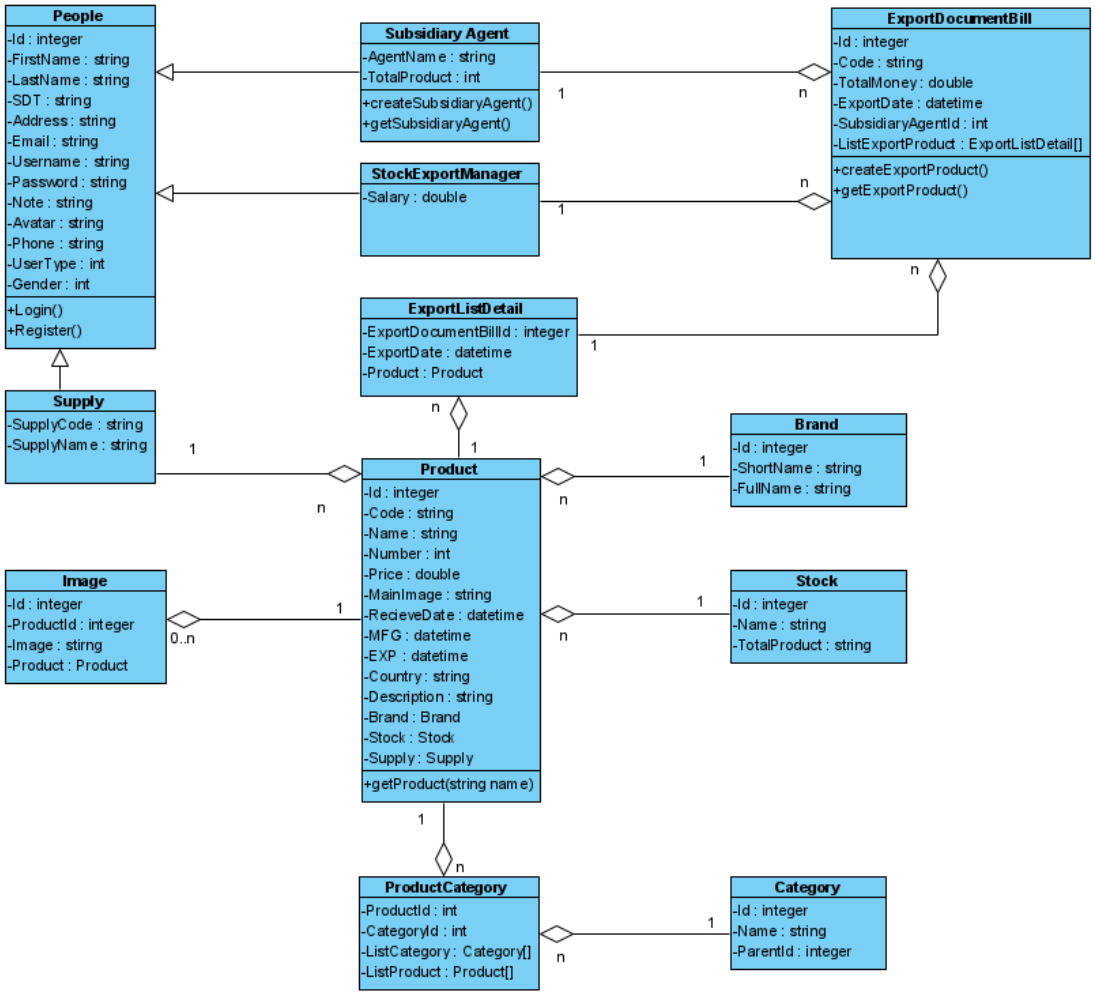
5. Login Page call Export Manage Page
6. Export Manager click Export Product
7. Export Product Page show
8. Export Manager Find SubsidiaryAgent
9. Export Product Page call SubsidiaryAgent
10. SubsidiaryAgent Check exist and return true
    10.1 If SubsidiaryAgent not exist , ExportProductPage return message to Manager
    10.2 ExportManager click create subsidiaryAgent
    10.3 ExportProductPage call subsidiaryAgent
    10.4 SubsidiaryAgent show to manager
    10.5 Manager fill infor new SubAgent and click create
    10.6 SubsidiaryAgent Page call SubsidiaryAgent class
    10.7 SubsidiaryAgentClass check not exist and return true
    10.8 ExportManager choose new SubsidiaryAgent .
11. Export Product Page show SubsidiaryAgent has been choose.
12. ExportManager Find Product
13. Export Product Page call product
14. Product return infor
15. Export Manager add Product to export (Repeat action util add all product want to export)
16. Export Manager Click Export
17. Export Product Page call ExportProductBill to create transaction.
18. Export Product Bill return
19. Export Product Page show success to manager
20. Export Manager click PrintBill
21. Export Product Page call Export Bill Page
22. Export Bill Page call ExportProductBill class to get transaction information
23. Export Product Bill return info
24. Export Bill Page Print Bill and return Print success message to Manager.



Biểu đồ giao tiếp

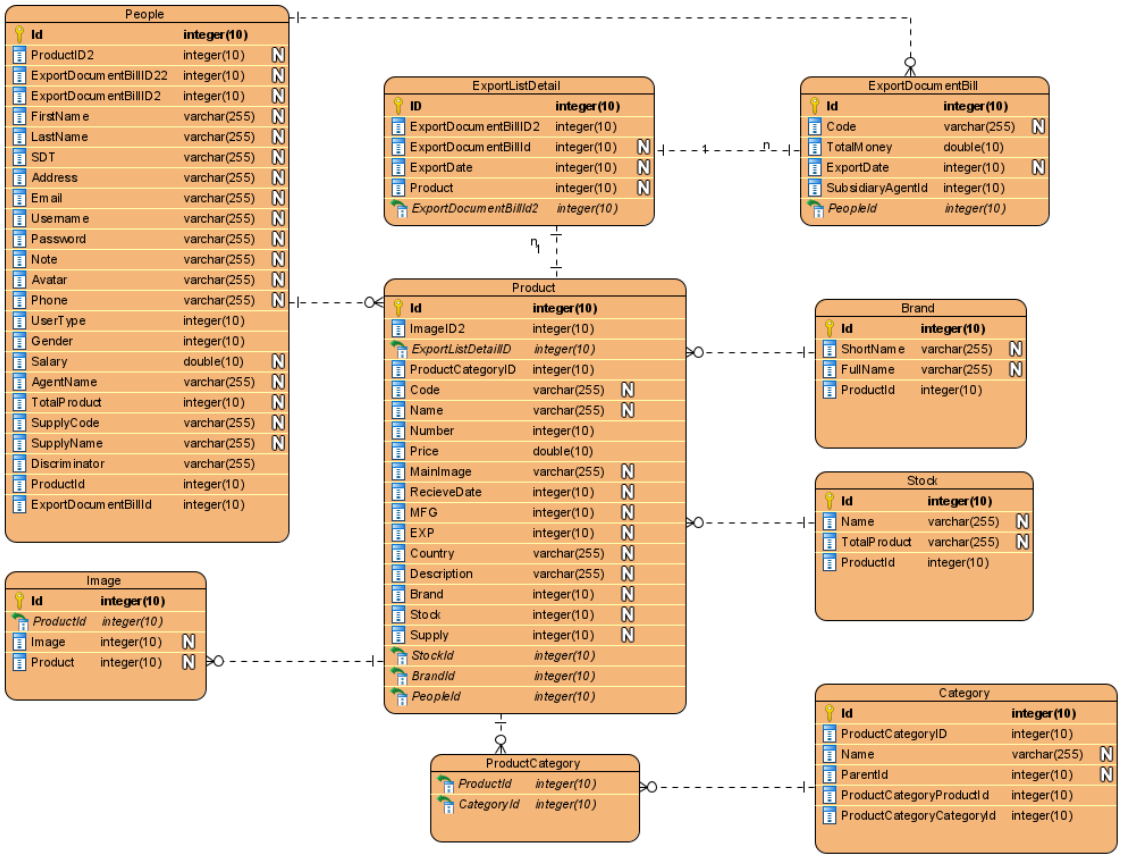
VI. THIẾT KẾ LỚP THỰC THỂ

- Sơ đồ lớp thực thể :



VII. THIẾT KẾ CSDL

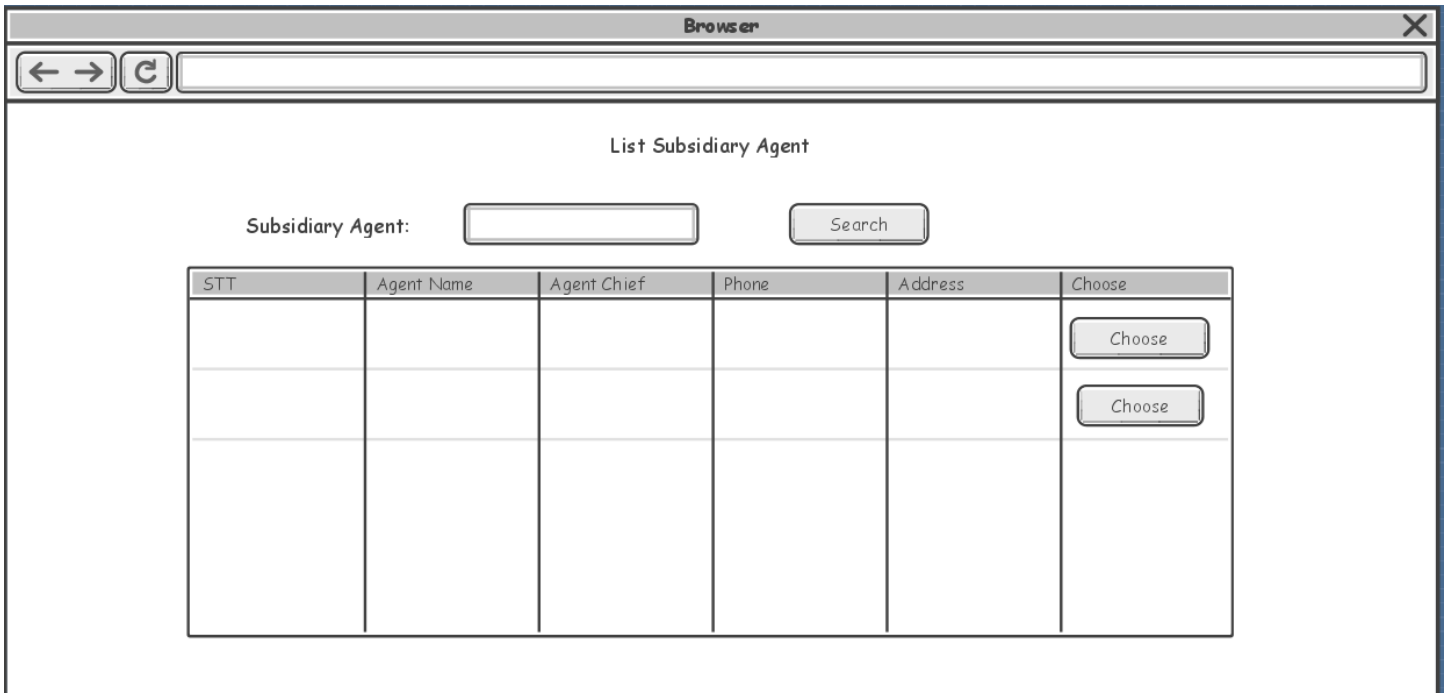
- Database:

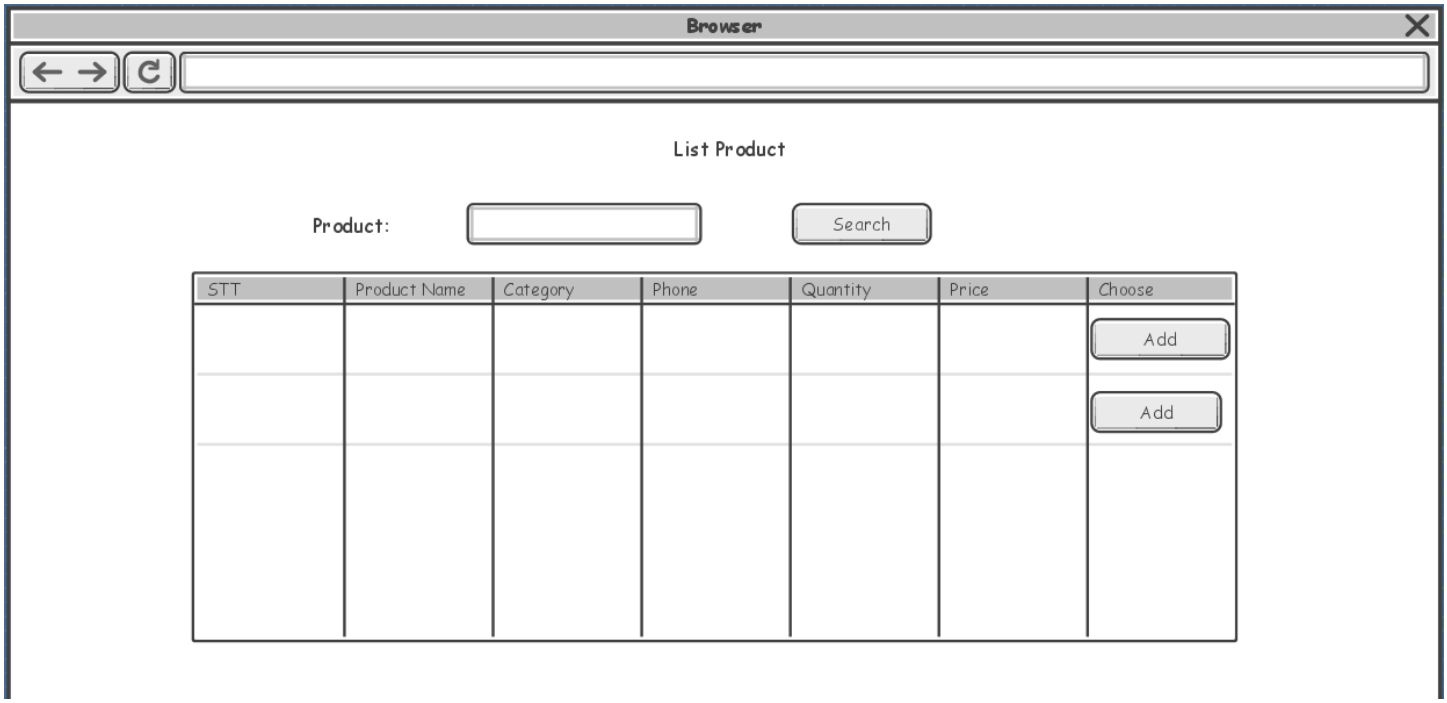


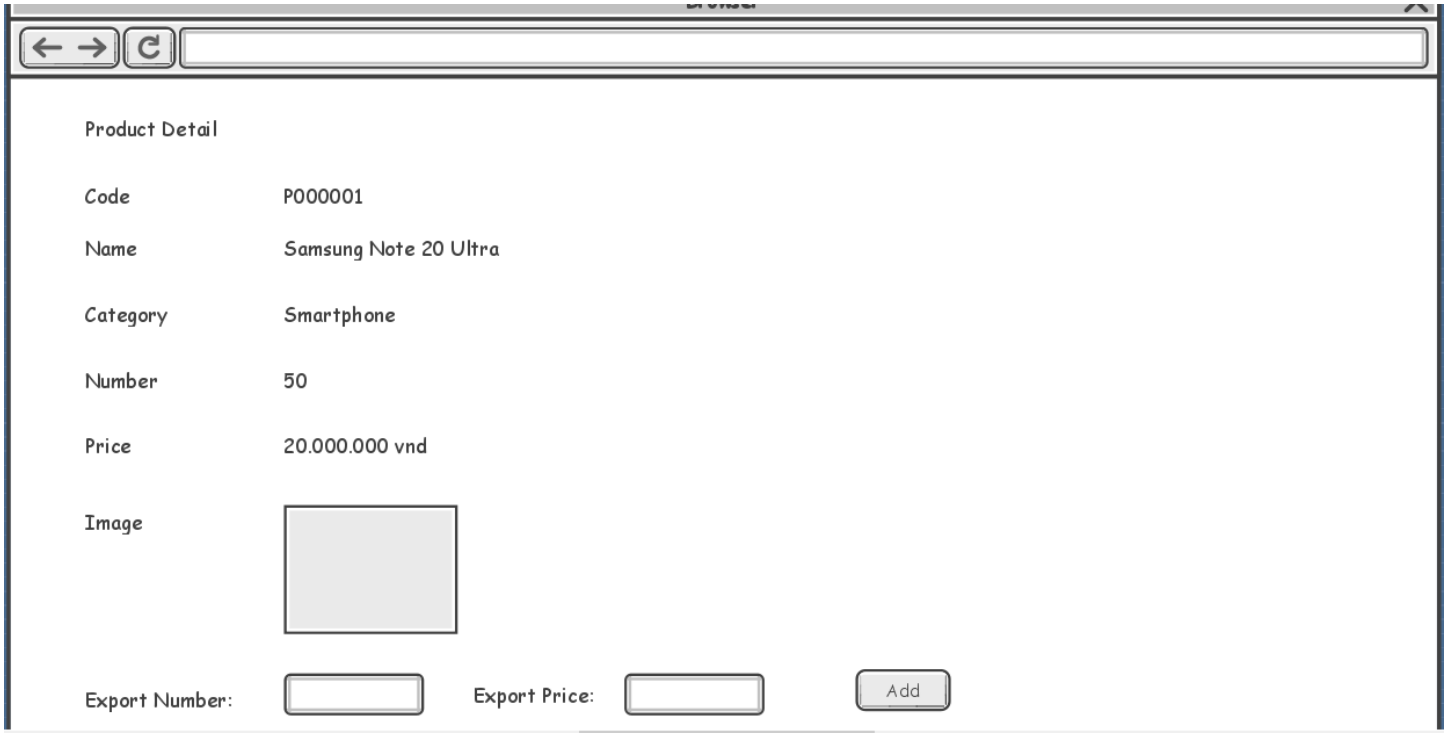
VIII. THIẾT KẾ CHI TIẾT CHO MODULE

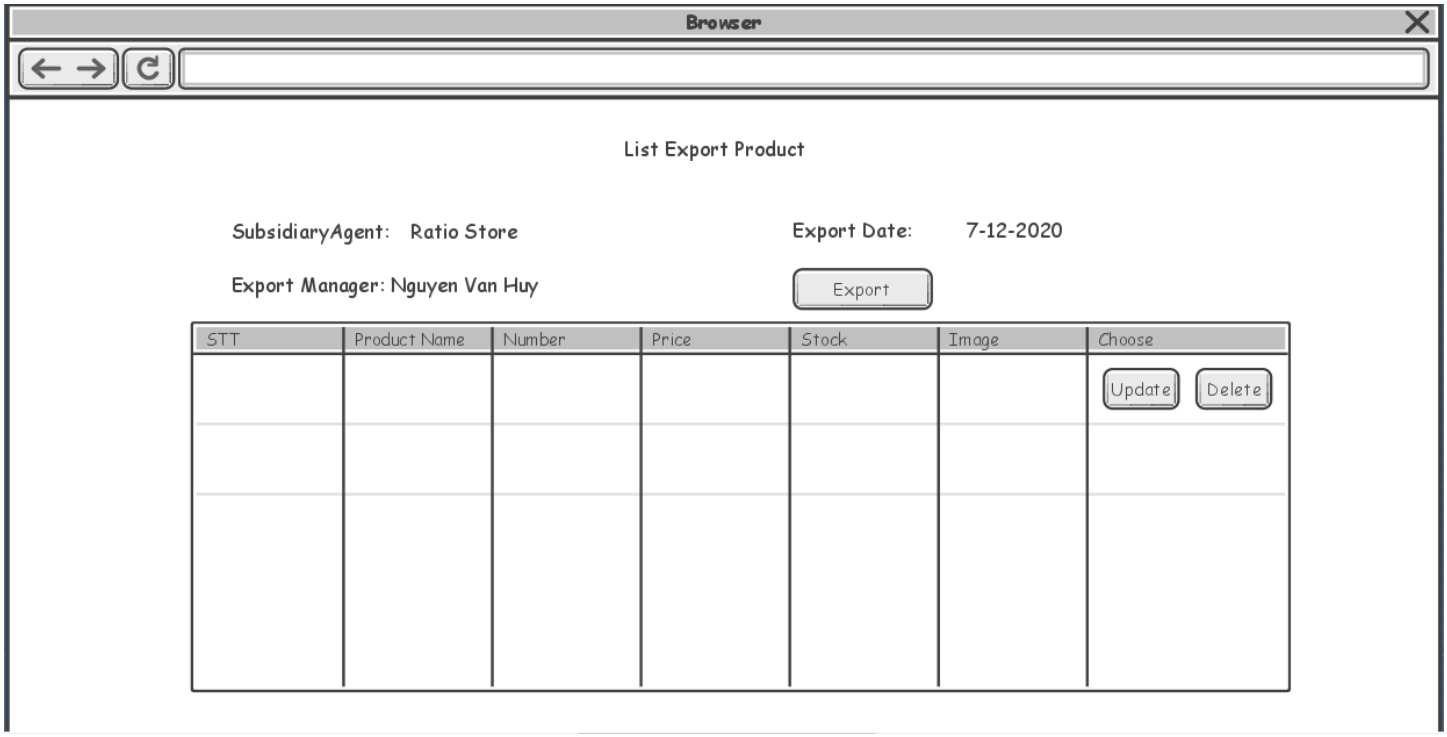
- **Thiết kế giao diện:**

-

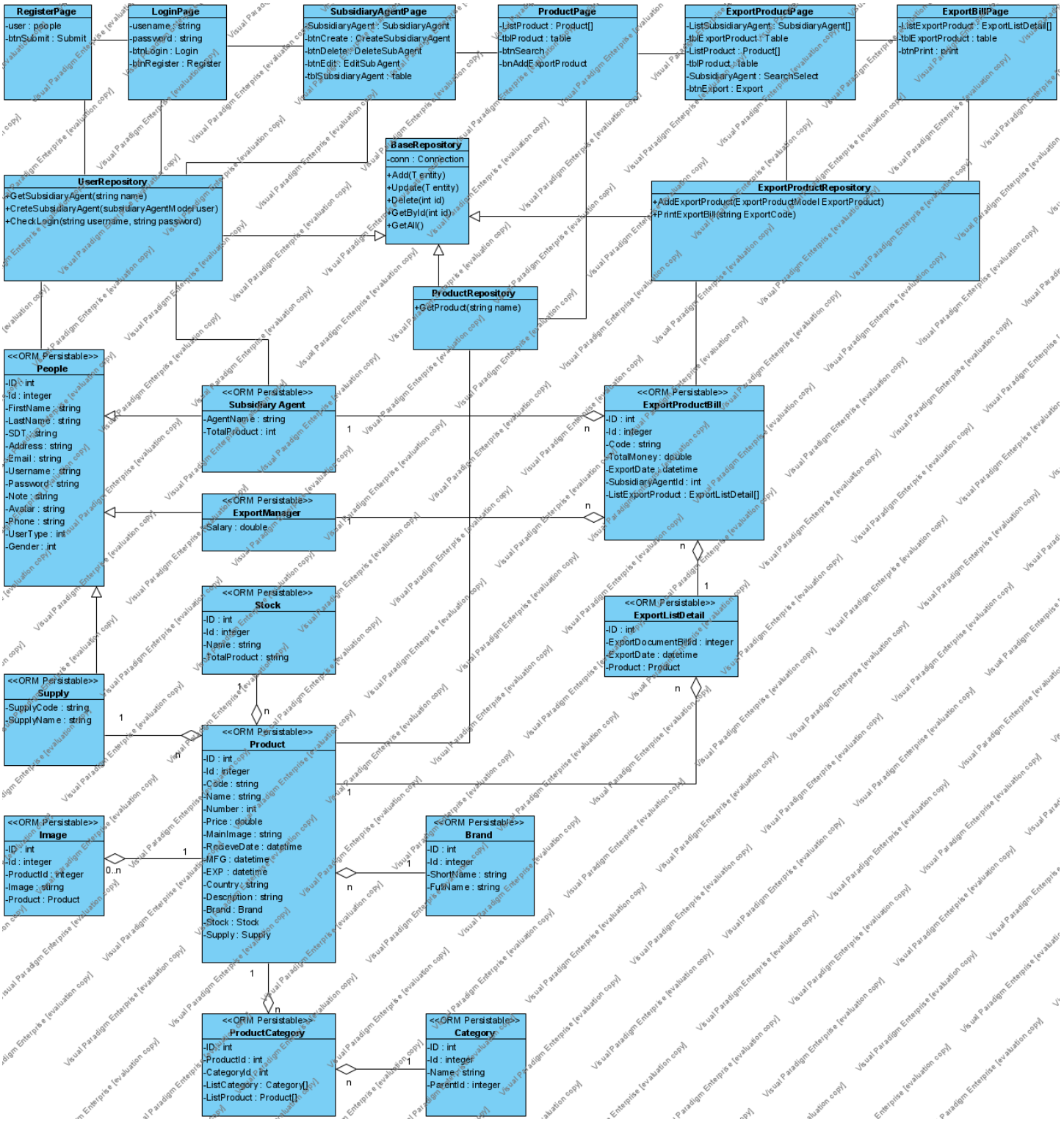

Browser
List Subsidiary Agent
Subsidiary Agent:
Search
STT
Agent Name
Agent Chief
Phone
Address
Choose
Choose
Choose


Browser
List Product
Product:
Search
STT
Product Name
Category
Phone
Quantity
Price
Choose
Add
Add

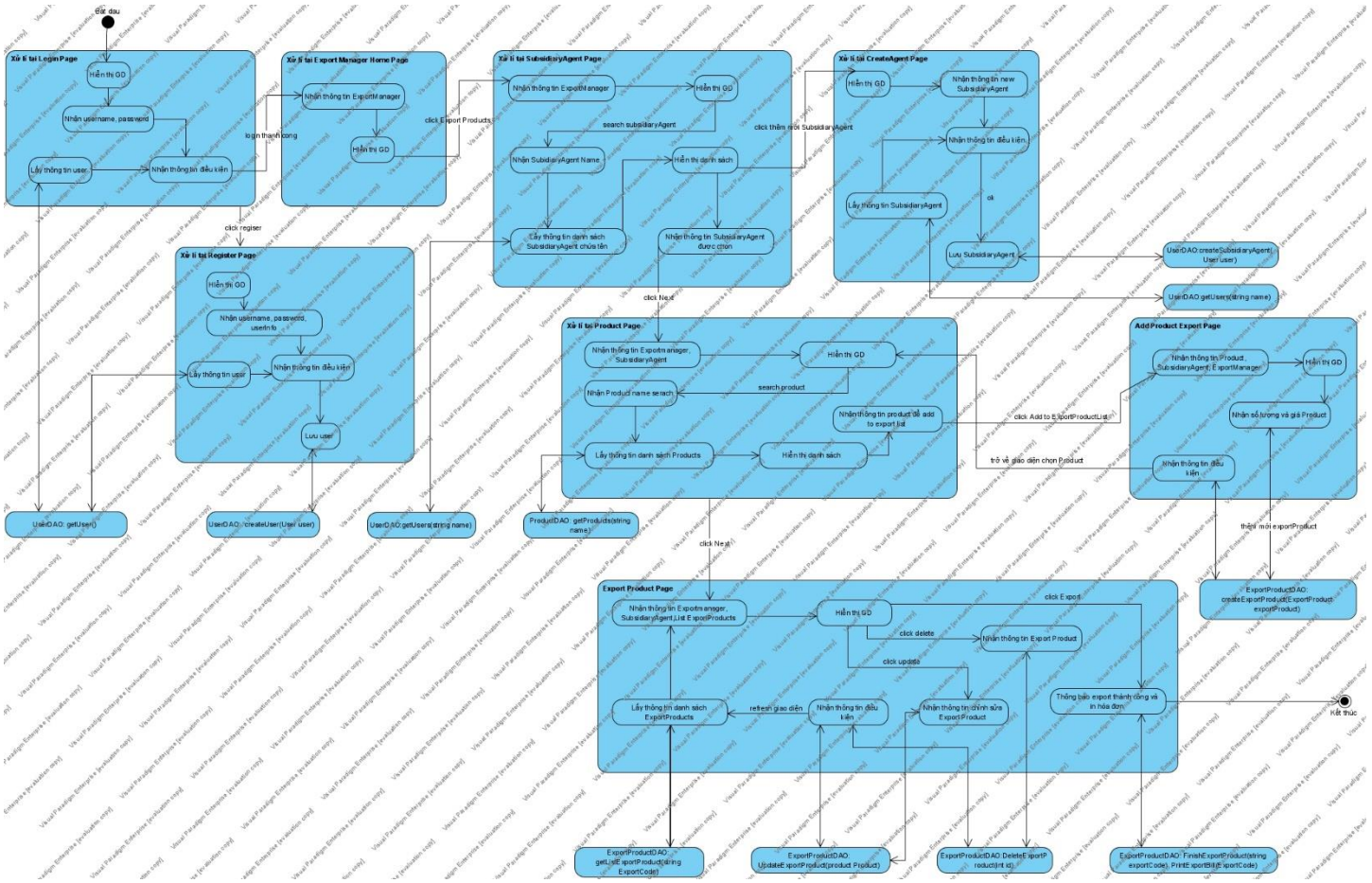




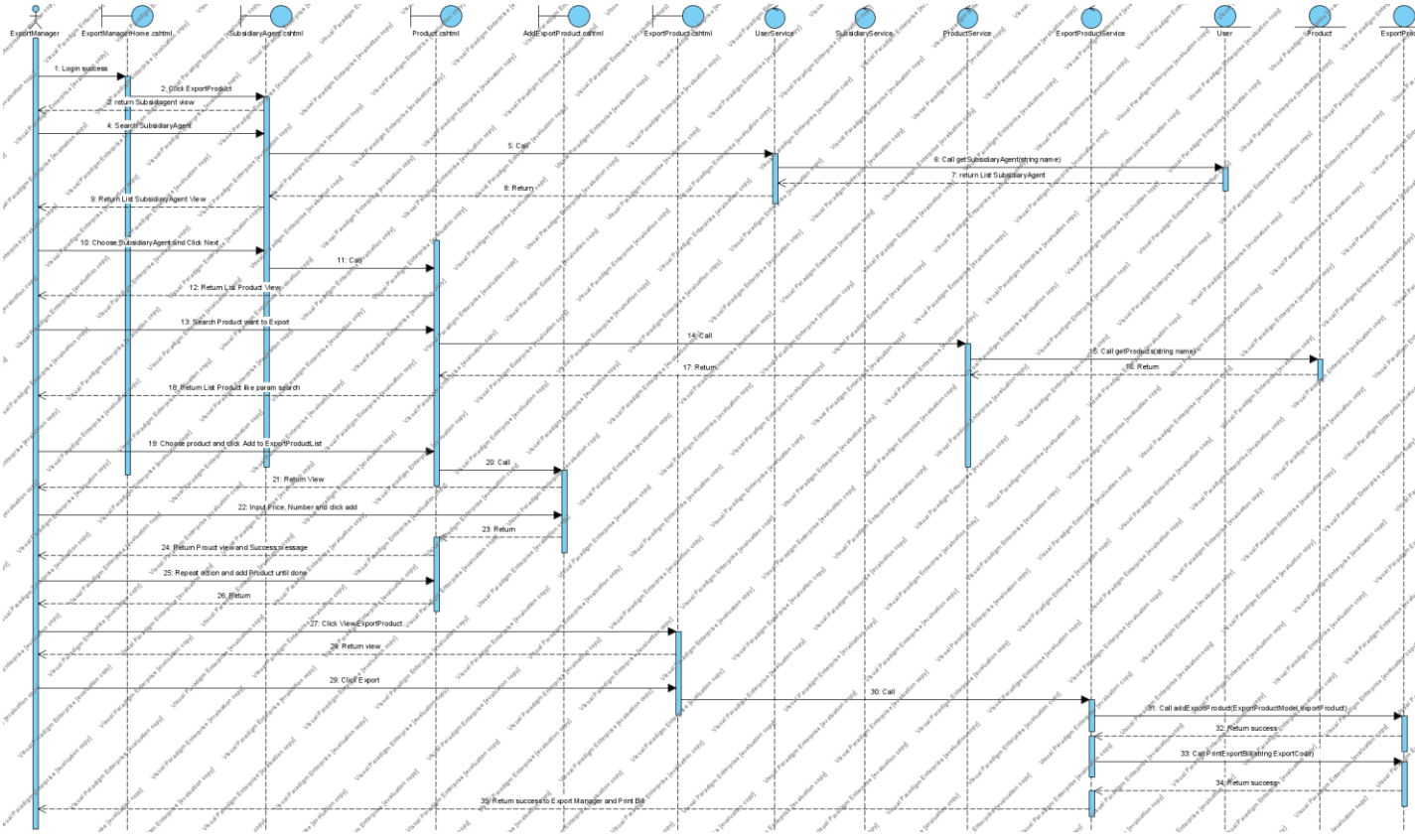
- **Biểu đồ lớp chi tiết:**



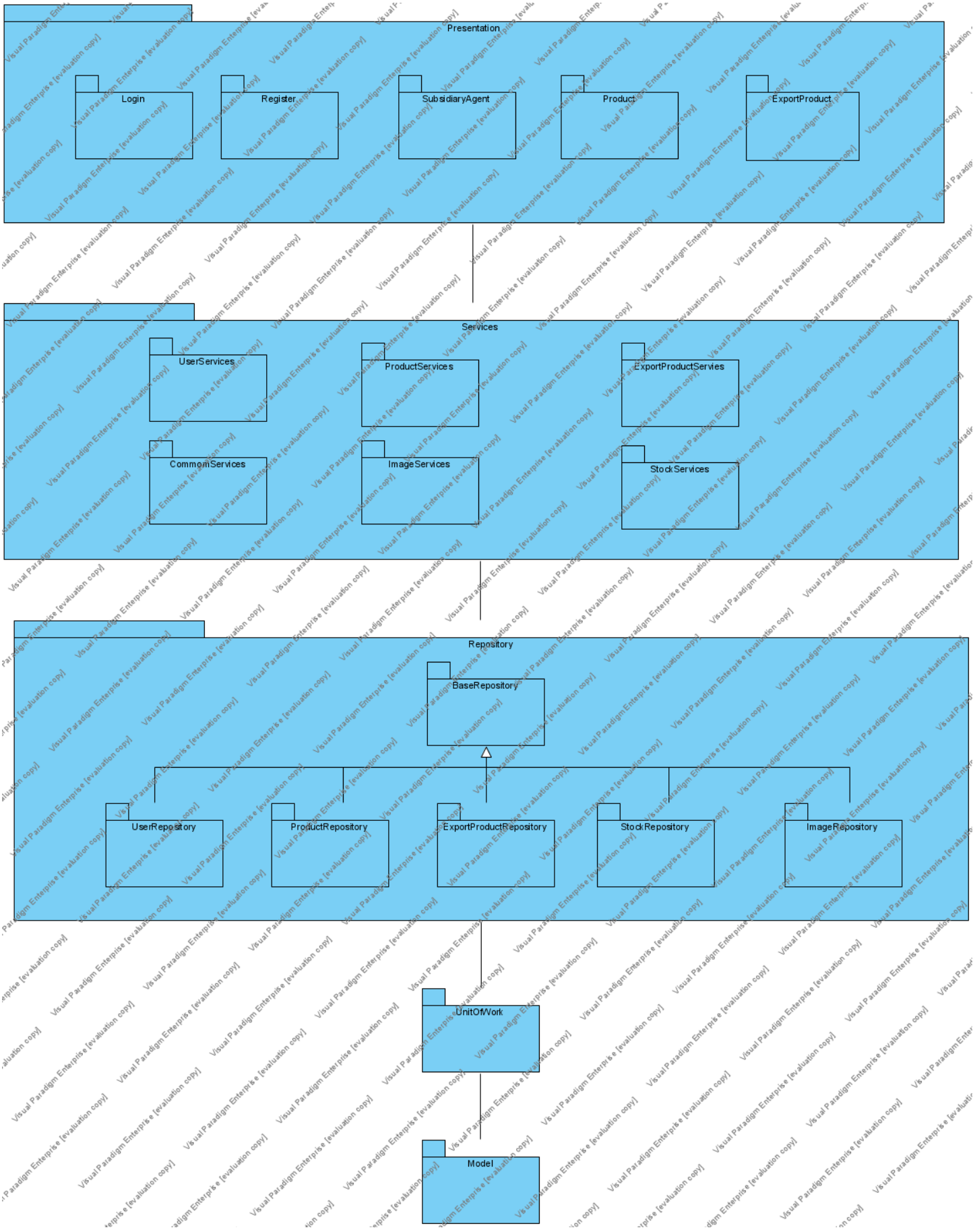
- **Biểu đồ hoạt động :**



- **Sequence Diagram:**

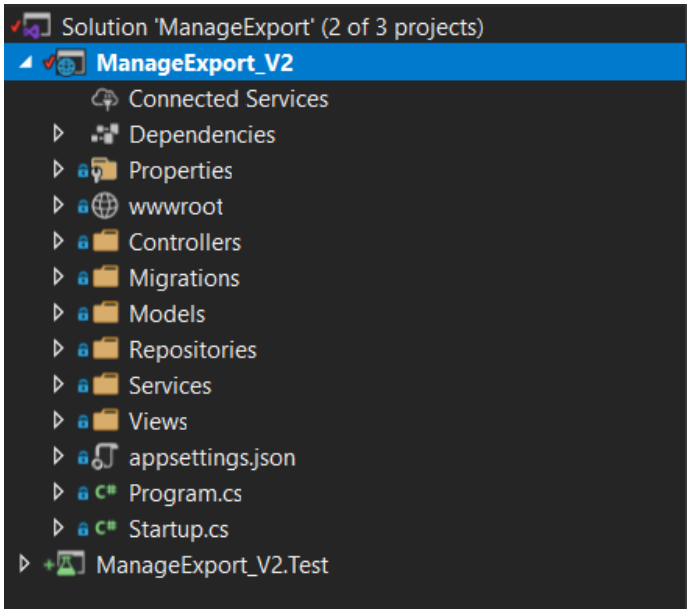


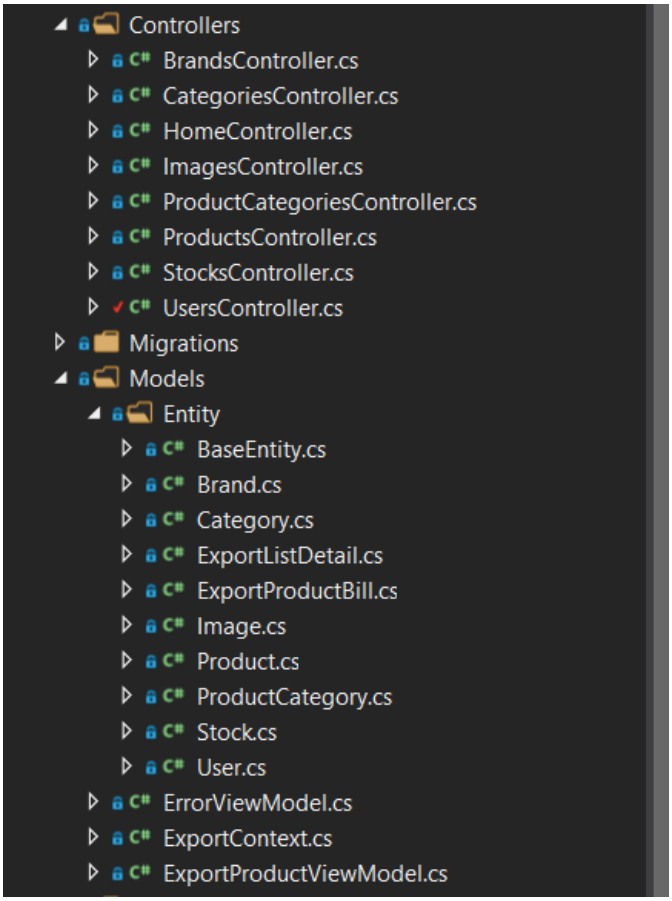
- **Package Diagram:**

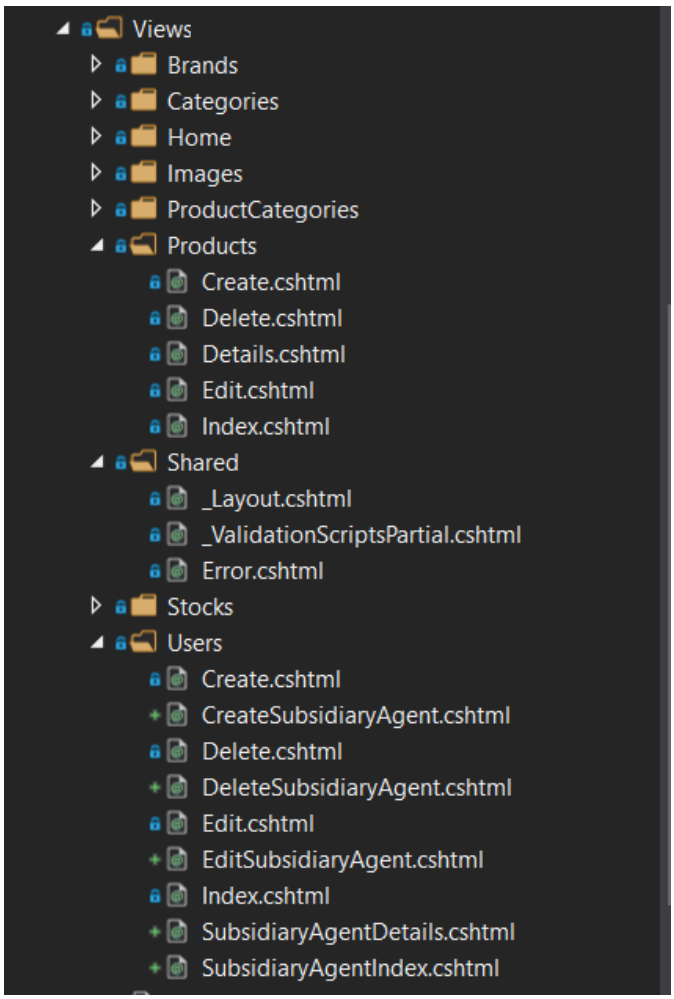


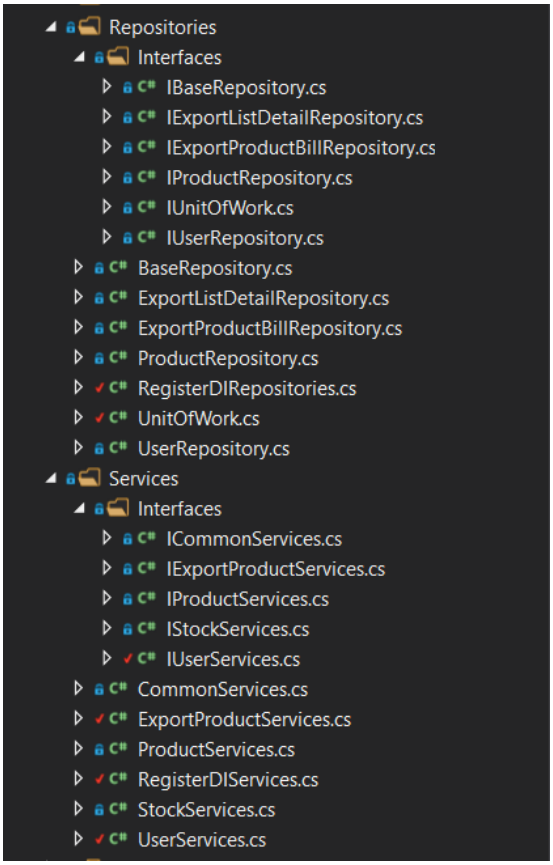
VII. Cài đặt hệ thống

a. Tổ chức dự án









b. Cài đặt Module

Nội dung phần này sẽ trình bày cài đặt chức năng: quản lí xuất hàng.

- Các interface repository, service đơn giản, nên bên dưới chỉ liệt kê code của class implement.

- Interface Base repository

```
public interface IBaseRepository<T> where T: class
  {
     void Add(T entity);

     Task Update(T entity);

     void Delete(T entity);

     void Delete(int id);

     Task DeleteMulti(Expression<Func<T, bool>> where);

     Task<T> GetSingleById(int id);

     Task<T> GetSingleByCondition(Expression<Func<T, bool>> expression, string[] includes = null);

     Task<IQueryable<T>> GetAll(string[] includes = null);

     Task<IQueryable<T>> GetMulti(Expression<Func<T, bool>> predicate, string[] includes = null);
```

```
    Task<IQueryable<T>> GetMultiPaging(Expression<Func<T, bool>> filter, int index = 0, int size = 50, string[] includes = null);


    Task<int> Count(Expression<Func<T, bool>> where);


    Task<bool> CheckContains(Expression<Func<T, bool>> predicate);
  }
```

- BaseRepository

```
public class BaseRepository<T> : IBaseRepository<T> where T: class
  {
    protected readonly ExportContext _context;
    private readonly DbSet<T> _dbSet;
    protected BaseRepository(ExportContext context)
    {
      _context = context;
      _dbSet = _context.Set<T>();
    }
    #region


    public virtual void Add(T entity)
    {
      _dbSet.Add(entity);
    }


    public virtual async Task Update(T entity)
    {
```

```
    _dbSet.Attach(entity);
    _context.Entry(entity).State = EntityState.Modified;
    await Task.CompletedTask;
}

public virtual void Delete(T entity)
{
    _dbSet.Remove(entity);
}

public virtual void Delete(int id)
{
    var entity = _dbSet.Find(id);
    _dbSet.Remove(entity);
}

public virtual async Task DeleteMulti(Expression<Func<T, bool>> where)
{
    IEnumerable<T> objects = _dbSet.Where<T>(where).AsEnumerable();
    foreach (T obj in objects)
        _dbSet.Remove(obj);
    await Task.CompletedTask;
}

public virtual async Task<T> GetSingleById(int id)
{
    return await Task.FromResult(_dbSet.Find(id));
```

```
    }

    public virtual async Task<IQueryable<T>> GetMany(Expression<Func<T, bool>> where, string includes)
    {
        return await Task.FromResult(_dbSet.Where(where).AsNoTracking());
    }

    public virtual async Task<int> Count(Expression<Func<T, bool>> where)
    {
        return await Task.FromResult(_dbSet.Count(where));
    }

    public async Task<IQueryable<T>> GetAll(string[] includes = null)
    {
        //HANDLE INCLUDES FOR ASSOCIATED OBJECTS IF APPLICABLE
        if (includes != null && includes.Any())
        {
            var query = _dbSet.Include(includes.First());
            foreach (var include in includes.Skip(1))
                query = query.Include(include);
            return await Task.FromResult(query.AsNoTracking());
        }

        return await Task.FromResult(_dbSet.AsNoTracking());
    }
```

```
    public async Task<T> GetSingleByCondition(Expression<Func<T, bool>> expression, string[] includes = null)
    {
        if (includes != null && includes.Any())
        {
            var query = _dbSet.Include(includes.First());
            foreach (var include in includes.Skip(1))
                query = query.Include(include);
            return await query.FirstOrDefaultAsync(expression);
        }
        return await _dbSet.FirstOrDefaultAsync(expression);
    }

    public virtual async Task<IQueryable<T>> GetMulti(Expression<Func<T, bool>> predicate, string[] includes = null)
    {
        if (includes != null && includes.Any())
        {
            var query = _dbSet.Include(includes.First());
            foreach (var include in includes.Skip(1))
                query = query.Include(include);
            return await Task.FromResult(query.Where<T>(predicate).AsNoTracking<T>());
        }

        return await Task.FromResult(_dbSet.Where<T>(predicate).AsNoTracking<T>());
    }
```

```
        public virtual async Task<IQueryable<T>> GetMultiPaging(Expression<Func<T, bool>> predicate, int index = 0, int size = 20, string[] includes = null)
        {
            int skipCount = index * size;
            IQueryable<T> resetSet;

            if (includes != null && includes.Any())
            {
                var query = _dbSet.Include(includes.First());
                foreach (var include in includes.Skip(1))
                    query = query.Include(include);
                resetSet = predicate != null ? query.Where<T>(predicate).AsQueryable() : query.AsQueryable();
            }
            else
            {
                resetSet = predicate != null ? _dbSet.Where<T>(predicate).AsQueryable() : _dbSet.AsQueryable();
            }

            resetSet = skipCount == 0 ? resetSet.Take(size) : resetSet.Skip(skipCount).Take(size);
            //total = resetSet.Count();
            return await Task.FromResult(resetSet.AsNoTracking());
        }

        public async Task<bool> CheckContains(Expression<Func<T, bool>> predicate)
        {
            return await Task.FromResult(_dbSet.Count<T>(predicate) > 0);
```

```
    }


    #endregion
}
```

- ExportProductRepository

```
public class ProductRepository : BaseRepository<Product>,IProductRepository
{
    public ProductRepository(ExportContext context): base(context)
    {


    }
}
```

- ExportListDetailRepository

```
public class ExportListDetailRepository : BaseRepository<ExportListDetail>,
IExportListDetailRepository
{
    public ExportListDetailRepository(ExportContext context) : base(context)
    {
    }
}
```

- ExportProductBillRepository

```
public class ExportProductBillRepository : BaseRepository<ExportProductBill>,
IExportProductBillRepository
{
```

```
    public ExportProductBillRepository(ExportContext context) : base(context)
    {
    }
}
```

- UnitOfWork

```
public class UnitOfWork : IUnitOfWork
{
    private ExportContext _context;
    private IUserRepository _userRepository;
    private IProductRepository _productRepository;
    private IExportListDetailRepository _exportListDetailRepository;
    private IExportProductBillRepository _exportProductBillRepository;
    public ExportContext ExportContext
    {
        get { return _context ?? (_context = new ExportContext()); }
    }

    public IUserRepository Users
    {
        get { return _userRepository ?? (_userRepository = new UserRepository(_context)); }
    }
    public IProductRepository Products
    {
        get { return _productRepository ?? (_productRepository = new ProductRepository(_context)); }
    }
    public IExportListDetailRepository ExportListDetailRepositorys
```

```
        {
            get { return _exportListDetailRepository ?? (_exportListDetailRepository = new
ExportListDetailRepository(_context)); }
        }
        public IExportProductBillRepository ExportProductBillRepositorys
        {
            get { return _exportProductBillRepository ?? (_exportProductBillRepository = new
ExportProductBillRepository(_context)); }
        }
        public async Task Commit()
        {
            await _context.SaveChangesAsync();
        }

        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }
        protected virtual void Dispose(bool disposing)
        {
            if (disposing)
            {
                _context.Dispose();
            }
        }
    }
```

- UserRepository

```
public class UserRepository : BaseRepository<User>, IUserRepository
  {
    public UserRepository(ExportContext context): base(context)
    {


    }


    public bool CheckLogin(string username, string password)
    {
      if (!String.IsNullOrEmpty(username) && !String.IsNullOrEmpty(password))
      {
        return _context.Users.Any(x => x.Username.Equals(username) &&
x.Password.Equals(password));
      }
      return false;
    }
  }
```

- CommonServices

```
public class CommonServices : ICommonServices
  {
    private readonly IWebHostEnvironment _hostEnvironment;
    public CommonServices(IWebHostEnvironment hostEnvironment)
    {
      _hostEnvironment = hostEnvironment;
    }
```

```
public async Task<string> CreateImage(IFormFile imageFile,string imageName,string saveFolder)
{
    try
    {
        string wwwRootPath = _hostEnvironment.WebRootPath;
        if (imageFile != null)
        {
            imageName = imageFile.FileName;
            //create save folder if not exist
            if (!Directory.Exists(wwwRootPath+saveFolder))
            {
                Directory.CreateDirectory(wwwRootPath+saveFolder);
            }
            string path = Path.Combine(wwwRootPath + saveFolder, imageName);
            using (var fileStream = new FileStream(path, FileMode.Create))
            {
                await imageFile.CopyToAsync(fileStream);
            }
        }
        return imageName;
    }
    catch(Exception e)
    {
        throw e;
    }

}
```

```
public async Task<string> EditImage(IFormFile imageFile,string imageName, string saveFolder)
{

    try
    {
        if (imageFile != null)
        {
            string wwwRootPath = _hostEnvironment.WebRootPath;
            var imagePath = Path.Combine(wwwRootPath+saveFolder, imageFile.FileName);
            if (!System.IO.File.Exists(imagePath))
            {
                // create image when path not exist
                imageName = imageFile.FileName;
                if (!Directory.Exists(wwwRootPath + saveFolder))
                {
                    Directory.CreateDirectory(wwwRootPath + saveFolder);
                }
                string path = Path.Combine(wwwRootPath + saveFolder, imageName);
                using (var fileStream = new FileStream(path, FileMode.Create))
                {
                    await imageFile.CopyToAsync(fileStream);
                }
            }
            else
            {
                imageName = imageFile.FileName;
            }
```

```
            }
            return imageName;
        }
        catch (Exception e)
        {
            throw e;
        }
    }
}
```

- ExportProductServices

```
public class ExportProductServices : IExportProductServices
  {
    private IUnitOfWork _unitOfWork;
    public ExportProductServices(IUnitOfWork unitOfWork)
    {
      _unitOfWork = unitOfWork;
    }

    public ExportProductViewModel ExportProduct()
    {
      return new ExportProductViewModel();
    }
    public bool AddExportProduct(ExportProductViewModel exportProductViewModel)
    {
      try
      {
```

```
    ExportProductBill exportProductBill = new ExportProductBill();
    // add exportProductBill
    exportProductBill.TotalMoney = exportProductViewModel.TotalMoney;
    exportProductBill.ExportDate = DateTime.UtcNow;
    exportProductBill.ExportManagerId = exportProductViewModel.ExportManager.Id;
    exportProductBill.UserId = exportProductViewModel.SubsidiaryAgent.Id;
    exportProductBill.Code = exportProductViewModel.Code;
    exportProductBill.CreatedDate = DateTime.UtcNow;
    exportProductBill.ModifiedDate = DateTime.UtcNow;
    _unitOfWork.ExportProductBillRepositorys.Add(exportProductBill);
    _unitOfWork.Commit();
    foreach (var item in exportProductViewModel.ExportProducts)
    {
        // add exportProductBillDetail
        ExportListDetail product = new ExportListDetail();
        product.ProductId = item.Id;
        product.ExportProductBillId = exportProductBill.Id;
        product.CreatedDate = DateTime.UtcNow;
        product.ModifiedDate = DateTime.UtcNow;
        product.ExportDate = DateTime.UtcNow;
        _unitOfWork.ExportListDetailRepositorys.Add(product);
    }
    _unitOfWork.Commit();
    return true;
}
catch (Exception e)
{
```

```
            return false;
        }
    }
}
```

- ProductServices

```
public class ProductServices :IProductServices
{
    private IUnitOfWork _unitOfWork;
    public ProductServices(IUnitOfWork unitOfWork)
    {
        _unitOfWork = unitOfWork;
    }
    public Task<IQueryable<Product>> GetProducts(string str)
    {
        try
        {
            return _unitOfWork.Products.GetMulti(x => x.DisplayName.Contains(str) ||
x.Brand.ShortName.Contains(str));
        }
        catch (Exception e)
        {
            throw e;
        }
    }
    public Task<Product> GetProductById(int id)
    {
```

```
        try
        {
            return _unitOfWork.Products.GetSingleById(id);
        }catch(Exception e)
        {
            throw e;
        }
    }

}
```

- UserServices

```
public class UserServices : IUserServices
 {
    private IUnitOfWork _unitOfWork;
    public UserServices(IUnitOfWork unitOfWork)
    {
       _unitOfWork = unitOfWork;
    }
    public Task<User> GetSubsidiaryAgentById(int id )
    {
       try
       {
          return _unitOfWork.Users.GetSingleById(id);
       }
       catch (Exception e)
       {
          throw e;
```

```
    }
}
public Task<IQueryable<User>> GetSubsidiaryAgent(string str)
{
    try
    {
        return _unitOfWork.Users.GetMulti(x => x.UserType.Equals(UserType.SubsidiaryAgent) && (x.Email.Contains(str) ||
                                        x.AgentName.Contains(str) ||
                                        x.Phone.Contains(str) || x.FirstName.Contains(str) ||
                                        x.LastName.Contains(str)));
    }
    catch(Exception e)
    {
        throw e;
    }
}
public async Task<IQueryable<User>> GetSubsidiaryAgents()
{
    return await _unitOfWork.Users.GetMulti(x => x.UserType.Equals(UserType.SubsidiaryAgent));
}

public  async Task CreateSubsidiaryAgent(User user)
{
    user.UserType = UserType.SubsidiaryAgent;
    _unitOfWork.Users.Add(user);
    await _unitOfWork.Commit();
}
public async Task UpdateSubsidiaryAgent(User user)
{
```

```
        await _unitOfWork.Users.Update(user);
        await _unitOfWork.Commit();
    }
    public async Task DeleteSubsidiaryAgent(int id)
    {
        _unitOfWork.Users.Delete(id);
        await _unitOfWork.Commit();
    }
    public bool CheckLogin(string username, string password)
    {
        return _unitOfWork.Users.CheckLogin(username, password);
    }
}
```

c. Kiểm thử đơn vị

- **Xây dựng bộ test case:**

TT	Chức năng/ use case	Lớp điều khiển	Phương thức	Trường hợp test
			getDKcuaSV()	SV đã đăng kí 1 LHP
				SV đã đăng kí >1 LHP
				Kì học tồn tại, SV không tồn tại
				Kì học không tồn tại, SV tồn tại
				Kì học không tồn tại, SV không tồn tại.

				SV có đăng kí ở kì học khác, không có đăng kí ở kì học này
				ĐK 1 LHP, chưa có ĐK cũ
				ĐK >1 LHP, chưa có ĐK cũ
				ĐK 1 LHP, cùng môn học với ĐK cũ, ĐK cũ cũng chỉ có 1 LHP
1	Đăng kí học	DangkihocDAO	luuDKcuaSV()	ĐK 1 LHP, khác môn học với ĐK cũ, ĐK cũ cũng chỉ có 1 LHP
				ĐK >1 LHP, ĐK cũ có ít LHP hơn
				ĐK >1 LHP, ĐK cũ có nhiều LHP hơn
				ĐK mới trùng hoàn toàn với ĐK cũ
				ĐK mới không có LHP nào
		KihocDAO	...	...
		MonhocKihocDAO	...	...
		LophocphanDAO	...	...

		LichhocDAO	...	...
2	Nhập điểm	...	...	...
3	Thống kê học	...	...	...

- **Cài đặt kiểm thử đơn vị:**