**MINISTRY OF EDUCATION**

**AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

Distribution Management System in pharmaceutical

Report #4 – Software Requirement Specification

|  |  |
| --- | --- |
| Distribustion Management System in Pharmaceutial | |
| Group Members | Trương Võ Thiên Vũ – Team Leader – SE60626  Tạ Đức Tùng – Team Member – 60416  Nguyễn Trần Hoàng Quý – Team Member – 60356  Nguyễn Trọng Việt – Team Member – SE02406 |
| Supervisor | Nguyễn Văn Sang |
| Capstone Project code | DMS |

HaNoi, 11/2015

Table of Contents

[1 ARCHITECTUREDESIGN 4](#_Toc437618871)

[1.1 Choice of Architecture Design 4](#_Toc437618872)

[1.1.1 MVC Model overview 4](#_Toc437618873)

[1.1.2 Advantages and disadvantages of Layer Model 4](#_Toc437618874)

[1.2 Component Diagram 5](#_Toc437618875)

[2 Detailed Description of Components 6](#_Toc437618876)

[2.1 Entities Classes 6](#_Toc437618877)

[2.2 Generic Repository Class 7](#_Toc437618878)

[2.3 Screen Design 10](#_Toc437618879)

[2.3.1 Register 10](#_Toc437618880)

[2.3.2 Login 11](#_Toc437618881)

[2.3.3 Create Order 11](#_Toc437618882)

[2.3.4 View Order History 12](#_Toc437618883)

[2.3.5 Approve Order 12](#_Toc437618884)

[2.3.6 Assign Salesman 13](#_Toc437618885)

[2.3.7 Create delivery schedule 13](#_Toc437618886)

[2.3.8 Add Drug 13](#_Toc437618887)

[2.4 Sequence Diagram 14](#_Toc437618888)

[3 Database Design 21](#_Toc437618889)

[3.1 Physical database design 21](#_Toc437618890)

[3.1.1 Account 21](#_Toc437618891)

[3.1.2 AccountProfile 22](#_Toc437618892)

[3.1.3 DeliveryMan 22](#_Toc437618893)

[3.1.4 DeliverySchedule 23](#_Toc437618894)

[3.1.5 DeliveryScheduleDetails 23](#_Toc437618895)

[3.1.6 DiscountRate 24](#_Toc437618896)

[3.1.7 Drugorder 24](#_Toc437618897)

[3.1.8 DrugOrderDetails 24](#_Toc437618898)

[3.1.9 Drugstore 25](#_Toc437618899)

[3.1.10 DrugstoreType 26](#_Toc437618900)

[3.1.11 Drugtype 26](#_Toc437618901)

[3.1.12 Payment 26](#_Toc437618902)

[3.1.13 UnitPrice 27](#_Toc437618903)

[3.1.14 Role 27](#_Toc437618904)

[3.1.15 Unit 27](#_Toc437618905)

[3.1.16 City 28](#_Toc437618906)

[3.1.17 District 28](#_Toc437618907)

[4 Algorithms 28](#_Toc437618908)

[4.1.1 Neareast neighbor alrorithms: 28](#_Toc437618909)

# ARCHITECTUREDESIGN

## Choice of Architecture Design

### MVC Model overview

**CONTROLLER**

* Intercepts user input;
* Coordinates the view and model;
* Handles communication between the model anddata layer.

**VIEW**

* Binds to the model;
* Renders the UIs (HTML, CSS, JavaScript);
* Allows navigating between controllers.

**MODEL**

* Communicates with data source;
* Exposes functionalities, business logics, and data validation.

Database   
Server

Application Server

1. Request

8. Response

5. Select views

7.Changes made

2. Invoke methods

6. Query states

Method invocations

Change notifications

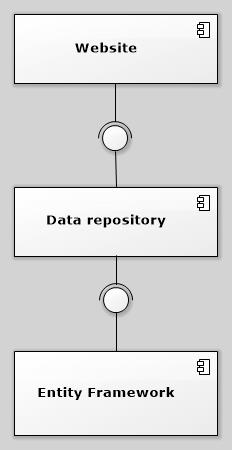
The components in MVC Model :

* **Model** is the part of the application that handles the logic for the application data. Often model objects retrieve data (and store data) from a database.
* **View** is the parts of the application that handles the display of the data. Most often the views are created from the model data.
* **Controller** is the part of the application that handles user interaction. Typically controllers read data from a view, control user input, and send input data to the model.

### Advantages and disadvantages of Layer Model

* Advantages :
* Easy to maintain the code and future improvements.
* Each object in mvc have distinct responsibilities. So change in one class doesn't need alternation in other classes.
* Clear separation between presentation logic and business logic.
* Disadvantages :
* For parallel development there is a needed multiple programmers.
* Too complex to implement and is not suitable for smaller application.

## Component Diagram



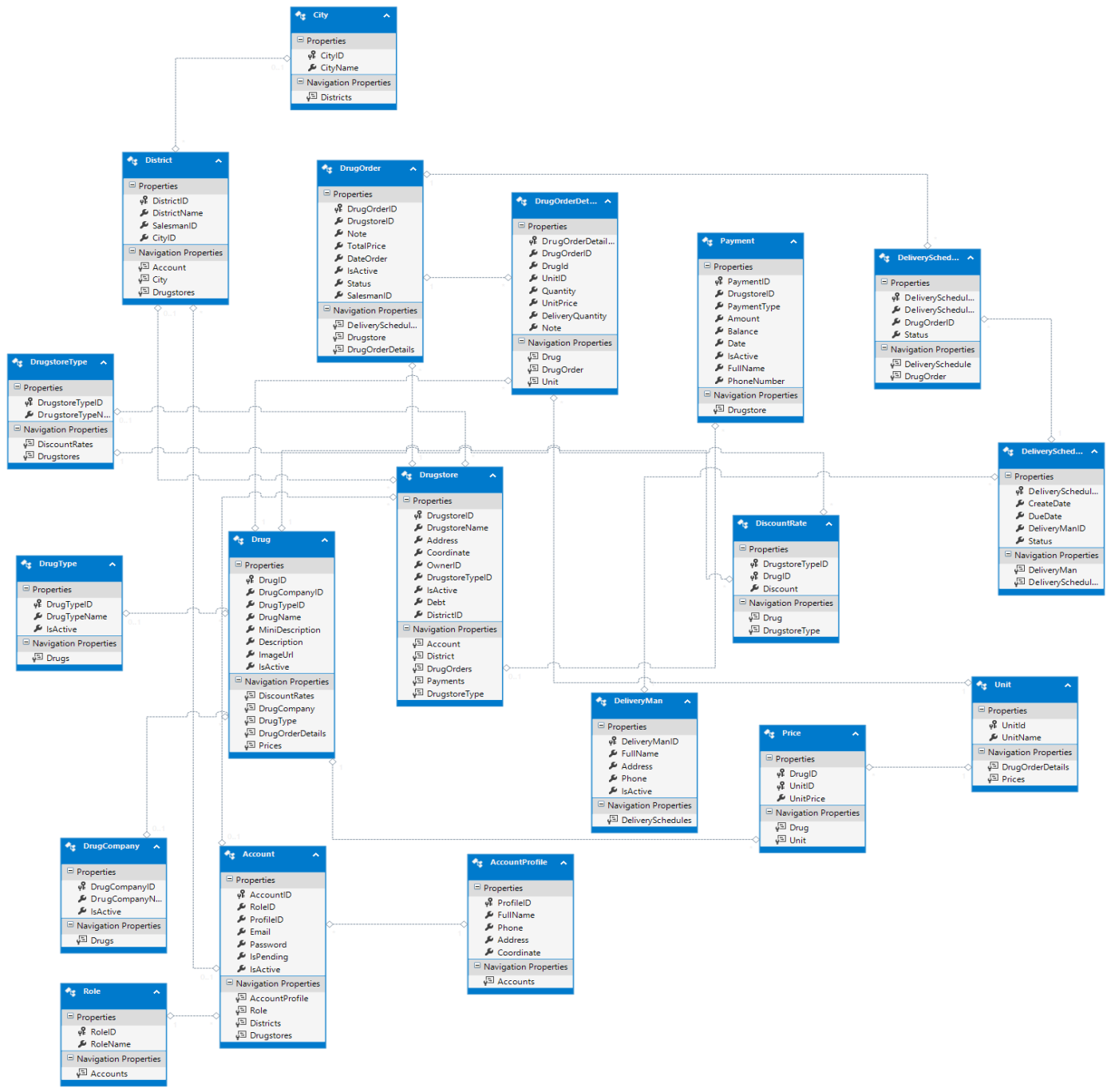
The DMS system includes the following major components:

* **Entity Framework:** An object-relational mapper that enables working with relational data persistence using domain-specific objects;
* **Data Repository:** A data store that provides data access functionality, using the Entity Framework to communicates with the database;
* **Website:** A web-application that helps exposes all the functionalities to end-users.

From the point of view of the MVC-III pattern, the components Entity Framework 5.0, Data Repository all belong to the Model part. The Controller and the View parts are wrapped in the component Website.

# Detailed Description of Components

## Entities Classes



## Generic Repository Class



Diagram 6. Data Repository Class Diagram

Properties:

|  |  |  |  |
| --- | --- | --- | --- |
| **Property name** | **Type** | **Visibility** | **Description** |
| Context | DMPEntities | internal | Database context |
| dbSet | DbSet<TEntity> | internal | The entity set that the repository is instantiated for. |

Operations:

|  |  |  |
| --- | --- | --- |
| Signature: voidDelete(objectid)  Description:Delete an entity by given id. | | |
| Parameter | **Type** | **Description** |
| *id* | object | ID of the entity to be deleted |

|  |  |  |
| --- | --- | --- |
| Signature: voidDelete(TEntity entityToDelete)  Description:Delete entity which was given | | |
| Parameter | **Type** | **Description** |
| entityToDelete | TEntity | Entity instance of the entity to be deleted |

|  |  |  |
| --- | --- | --- |
| Signature: GenericRepository(DMPEntities context)  Description:The constructor accepts a database context instance and initializes the entity set variable | | |
| Parameter | **Type** | **Description** |
| context | **DMPEntities** | Database context |

|  |  |  |
| --- | --- | --- |
| Signature: IEnumerable<TEntity> Get(  Expression<Func<TEntity, bool>> filter = null,  Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>> orderBy = null,  string includeProperties = "")  Description:This method uses lambda expressions to allow the calling code to specify a filter condition and a column to order the results by, and a string parameter lets the caller provide a comma-delimited list of navigation properties for eager loading. | | |
| Parameter | **Type** | **Description** |
| filter | Expression<Func<TEntity, bool>> | The caller will provide a lambda expression based on the TEntity type, and this expression will return a Boolean value. |
| orderBy | Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>> | The input to the expression is an IQueryable object for the TEntitytype. The expression will return an ordered version of that IQueryable object. |
| includeProperties | string |  |

|  |  |  |
| --- | --- | --- |
| Signature: IEnumerable<TEntity>GetAll()  Description:Get all entity. | | |
| Parameter | **Type** | **Description** |

|  |  |  |
| --- | --- | --- |
| Signature: voidGetByID(object id)  Description:Get an entity by given id. | | |
| Parameter | **Type** | **Description** |
| *id* | object | ID of the entity to be selected |

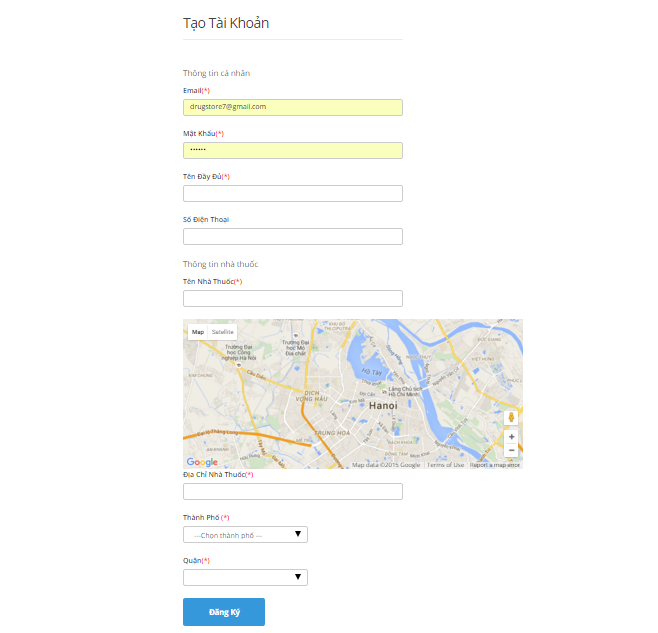
|  |  |  |
| --- | --- | --- |
| Signature: boolInsert(TEntityentity)  Description:Insert an entity by given id. | | |
| Parameter | **Type** | **Description** |
| entity | TEntity | Entity instance of the entity to be inserted |

|  |  |  |
| --- | --- | --- |
| Signature: voidSaveChanges()  Description:Save entity changes. | | |
| Parameter | **Type** | **Description** |

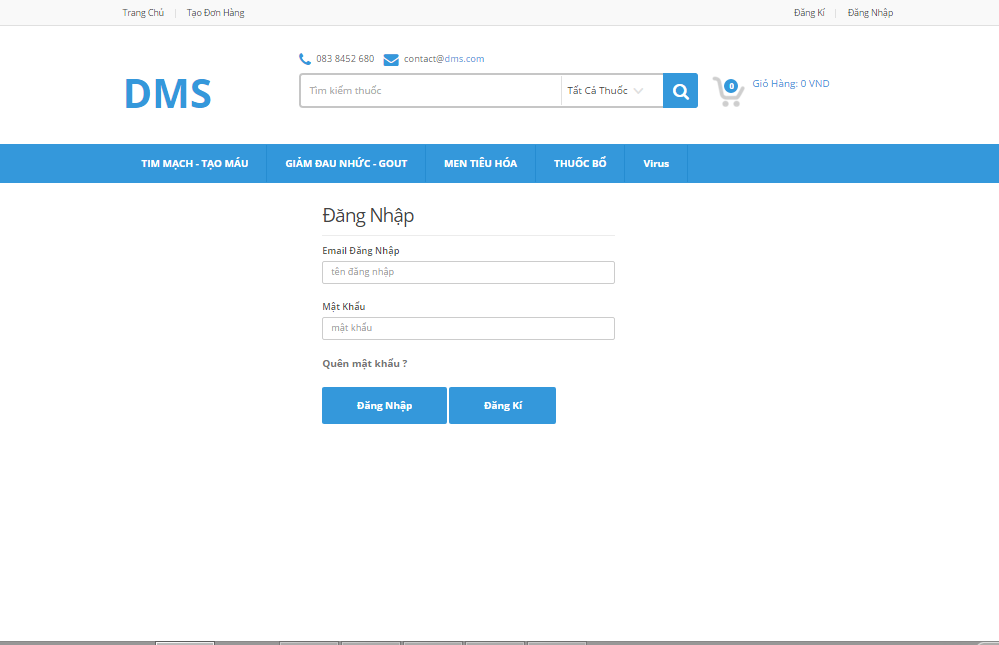
|  |  |  |
| --- | --- | --- |
| Signature: voidUpdate(object id)  Description: Delete an entity by given id. | | |
| Parameter | **Type** | **Description** |

## Screen Design

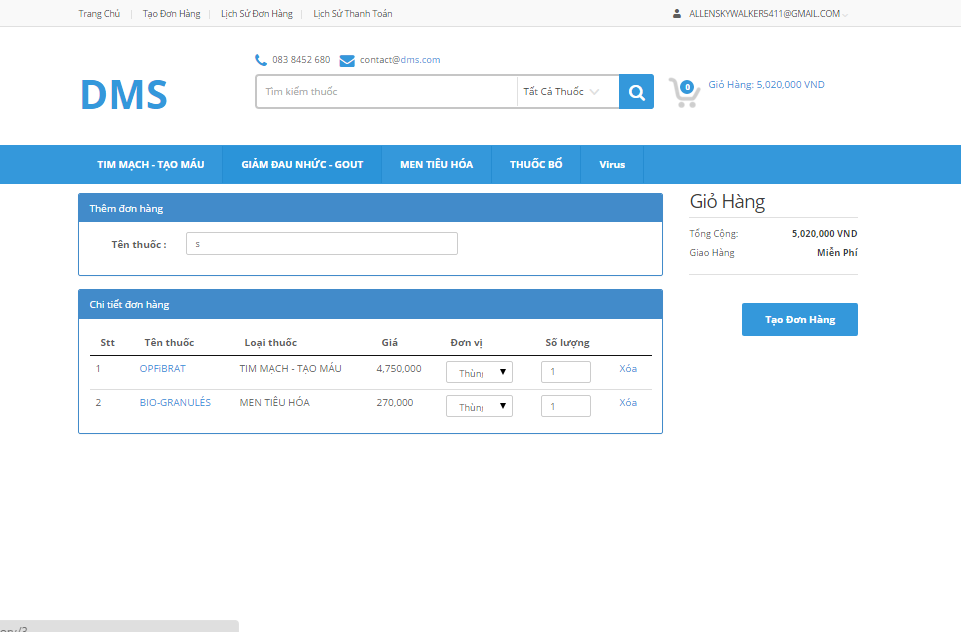
### Register



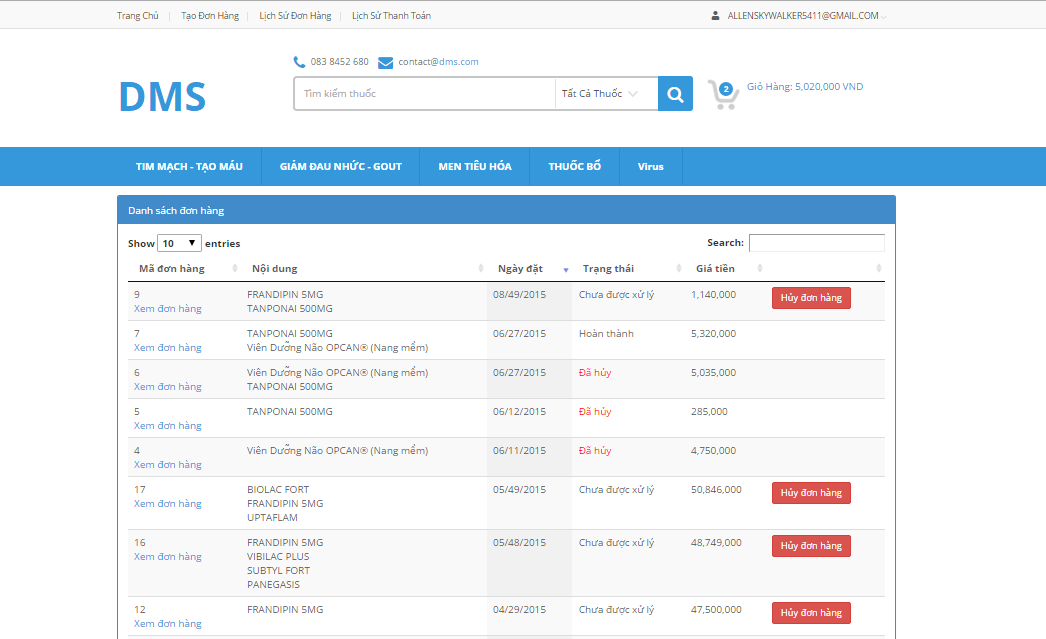
### Login



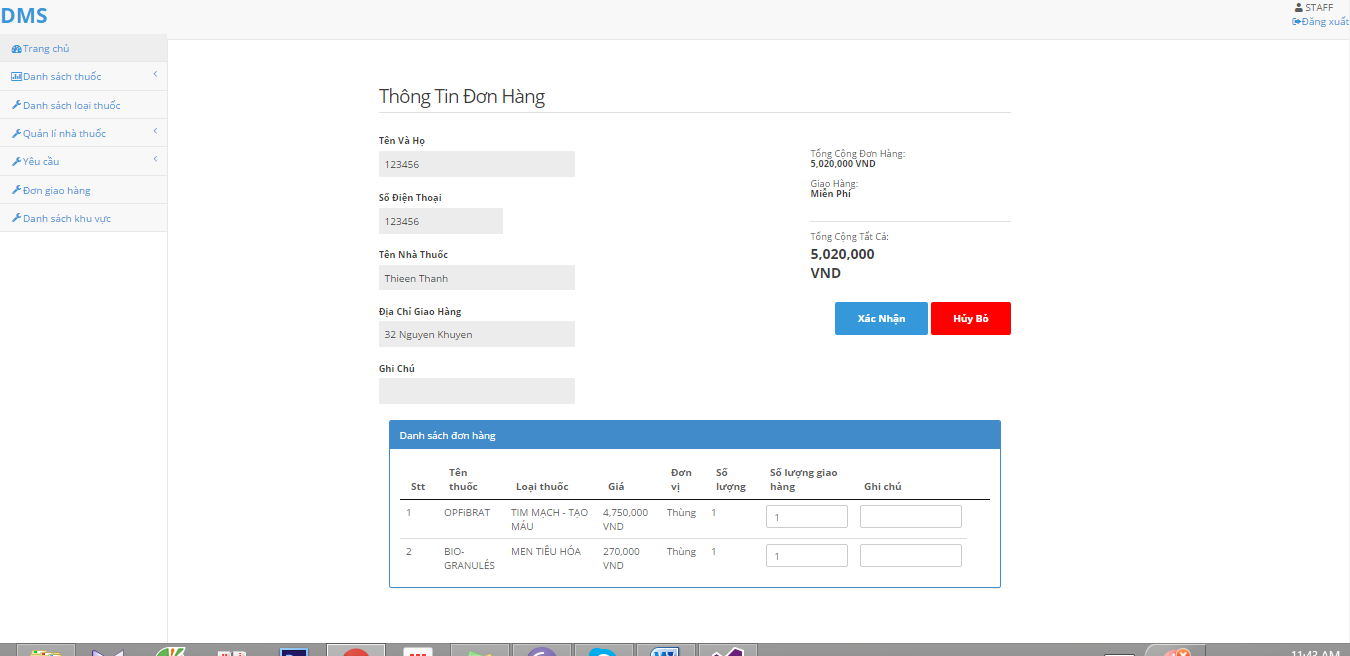
### Create Order



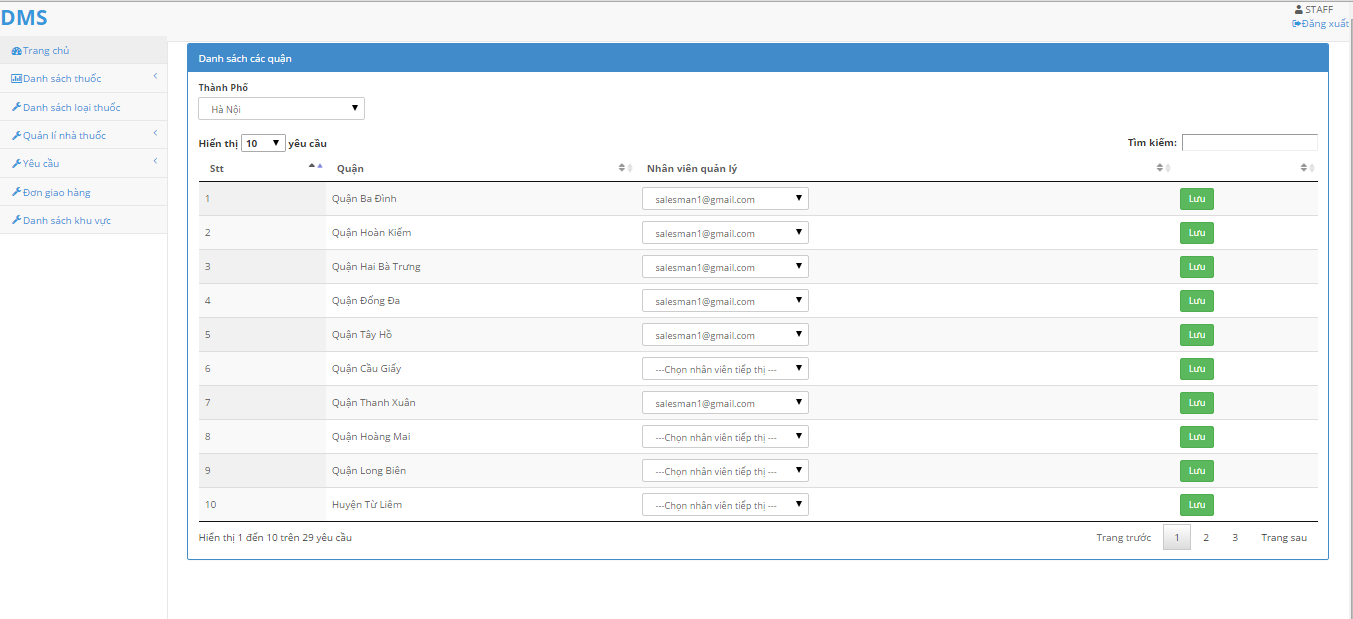
### View Order History



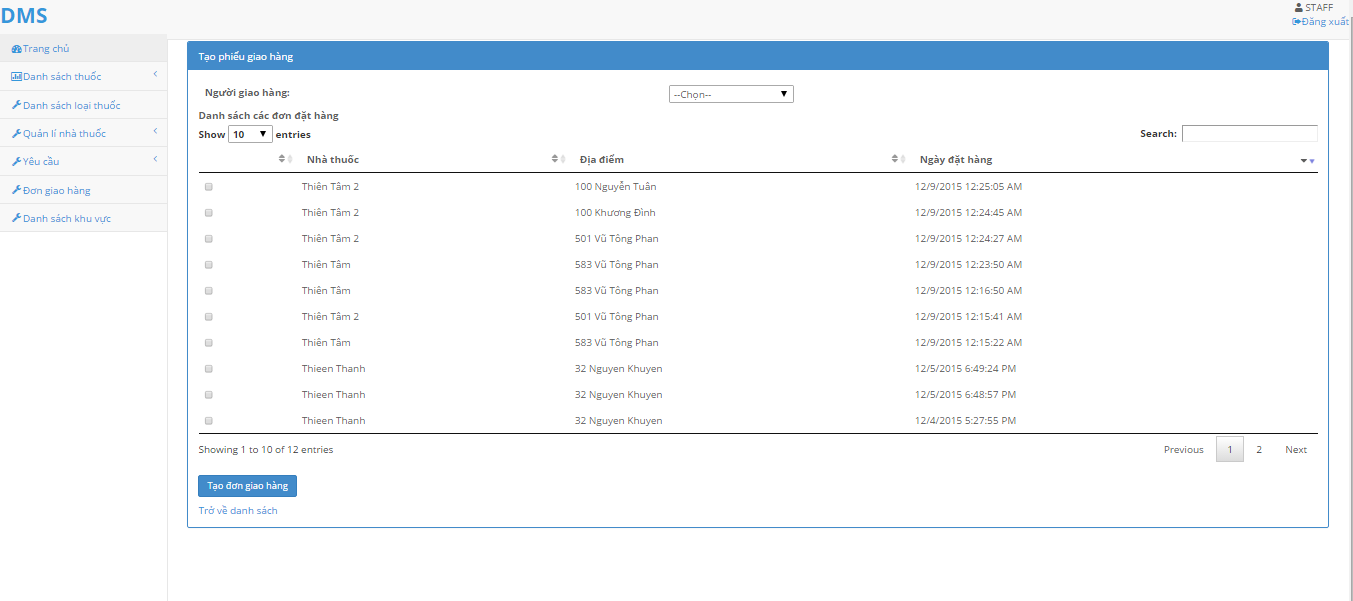
### Approve Order



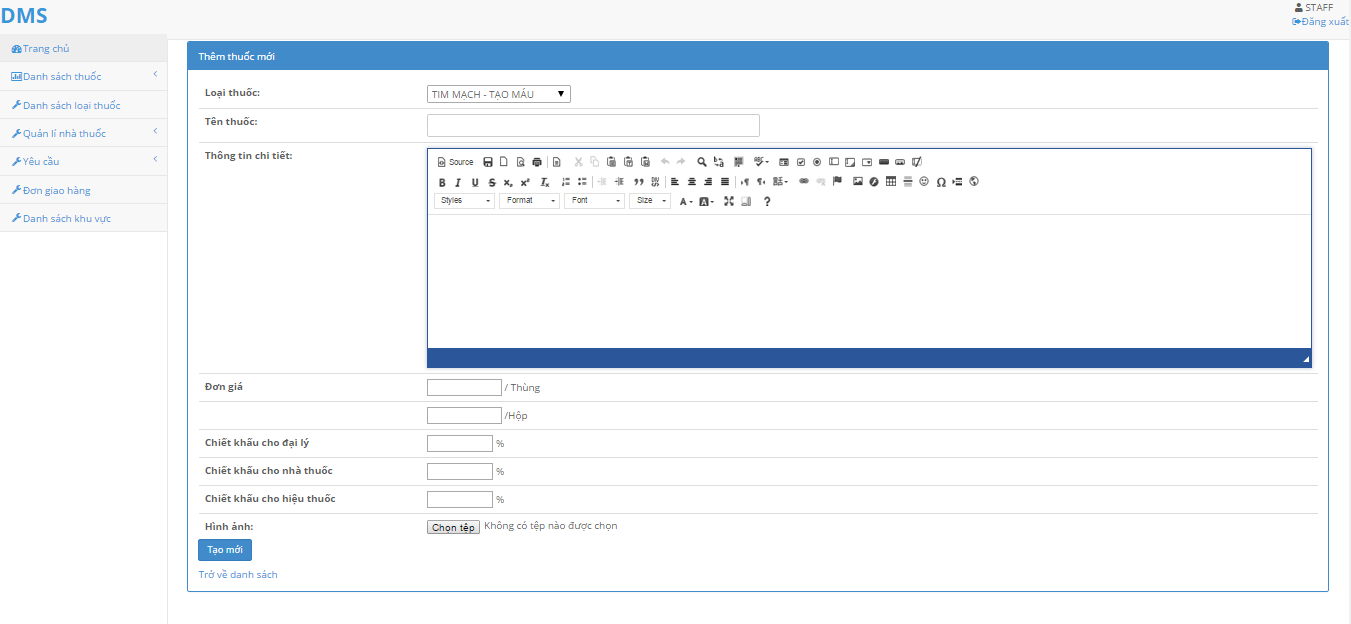
### Assign Salesman



### Create delivery schedule

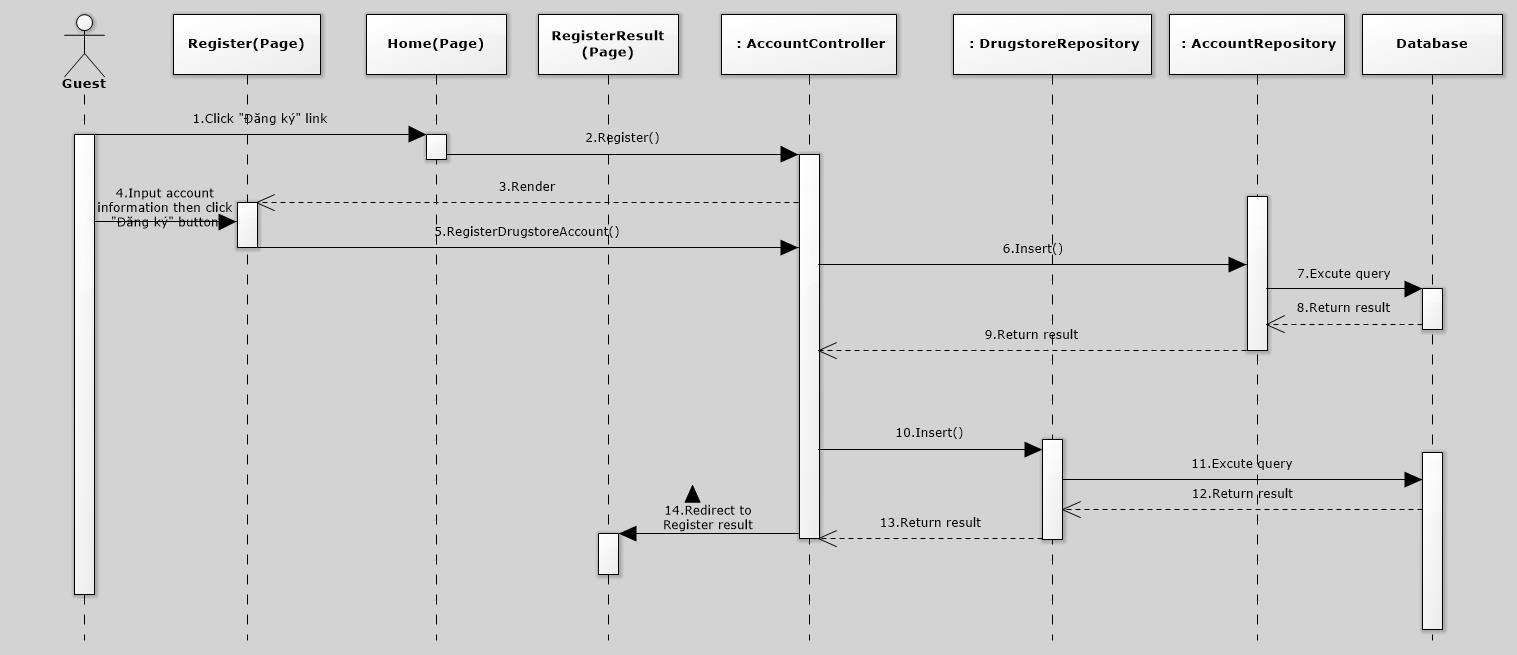


### Add Drug

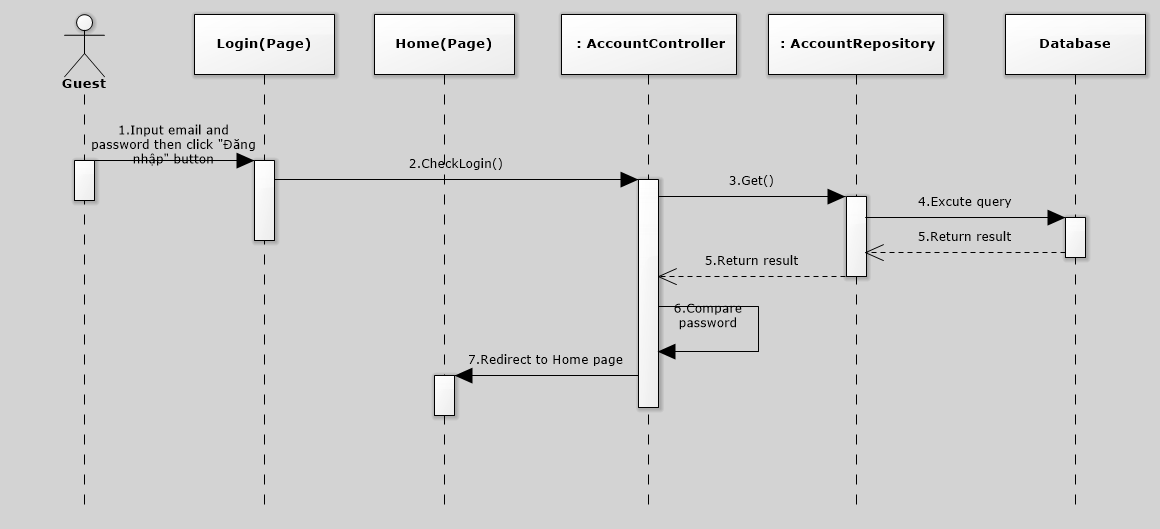


## Sequence Diagram

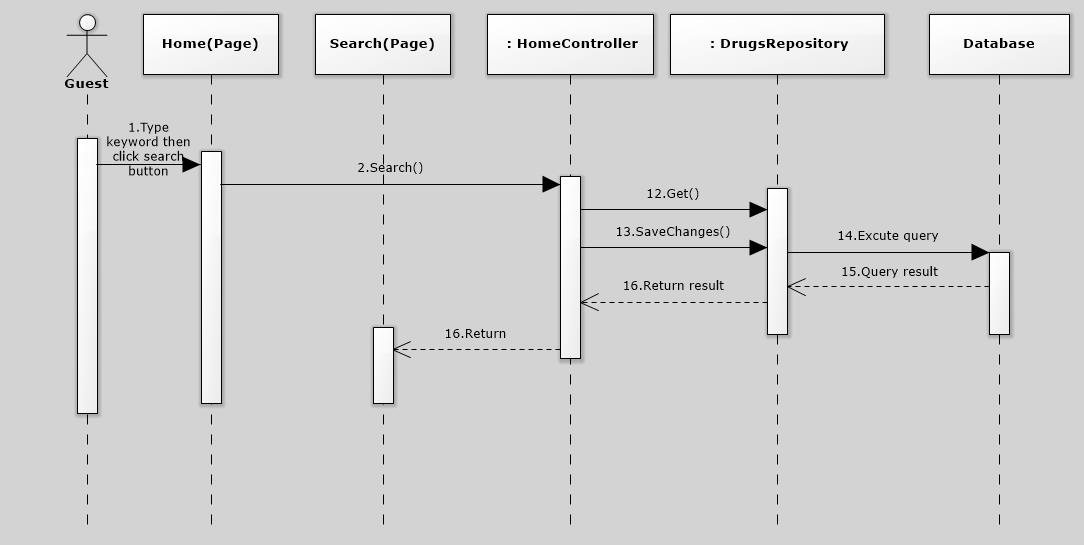
### <Guest> Register



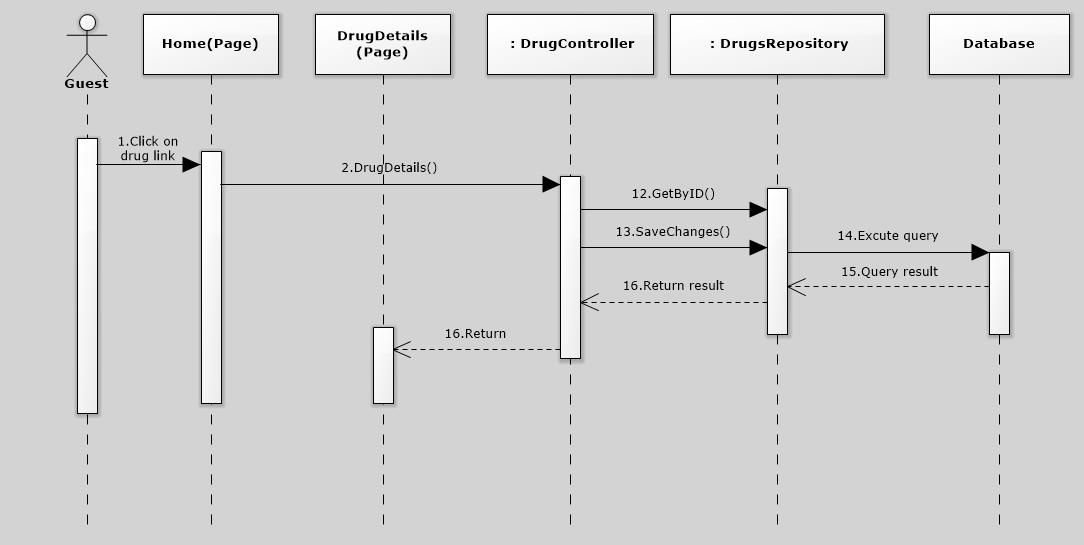
### <Guest> Login



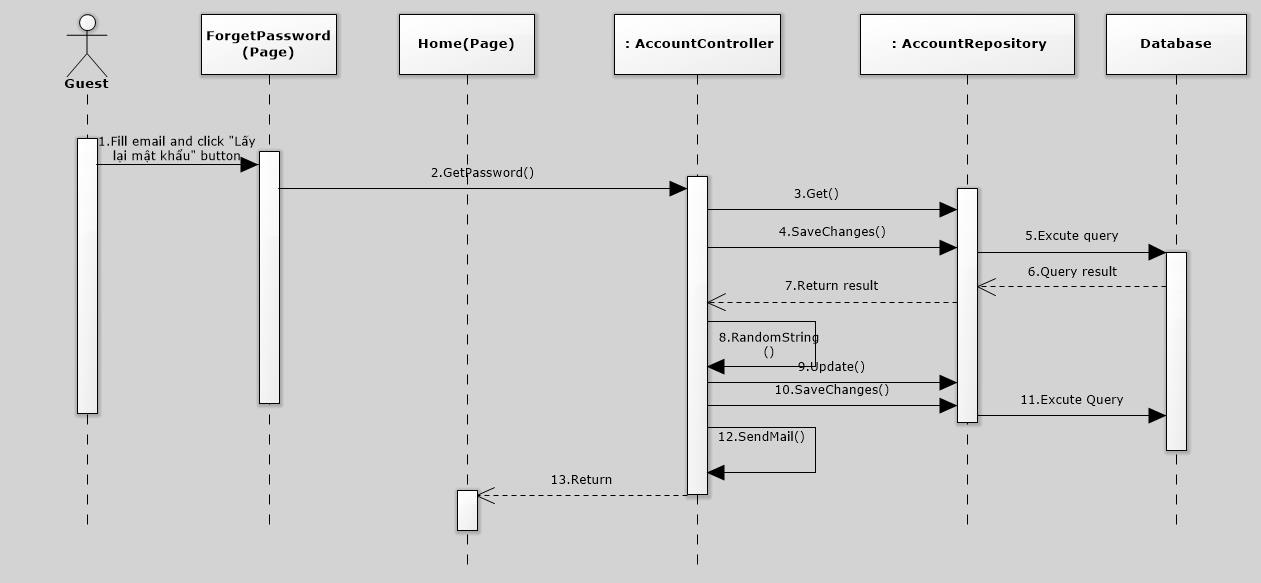
### <Guest> Search



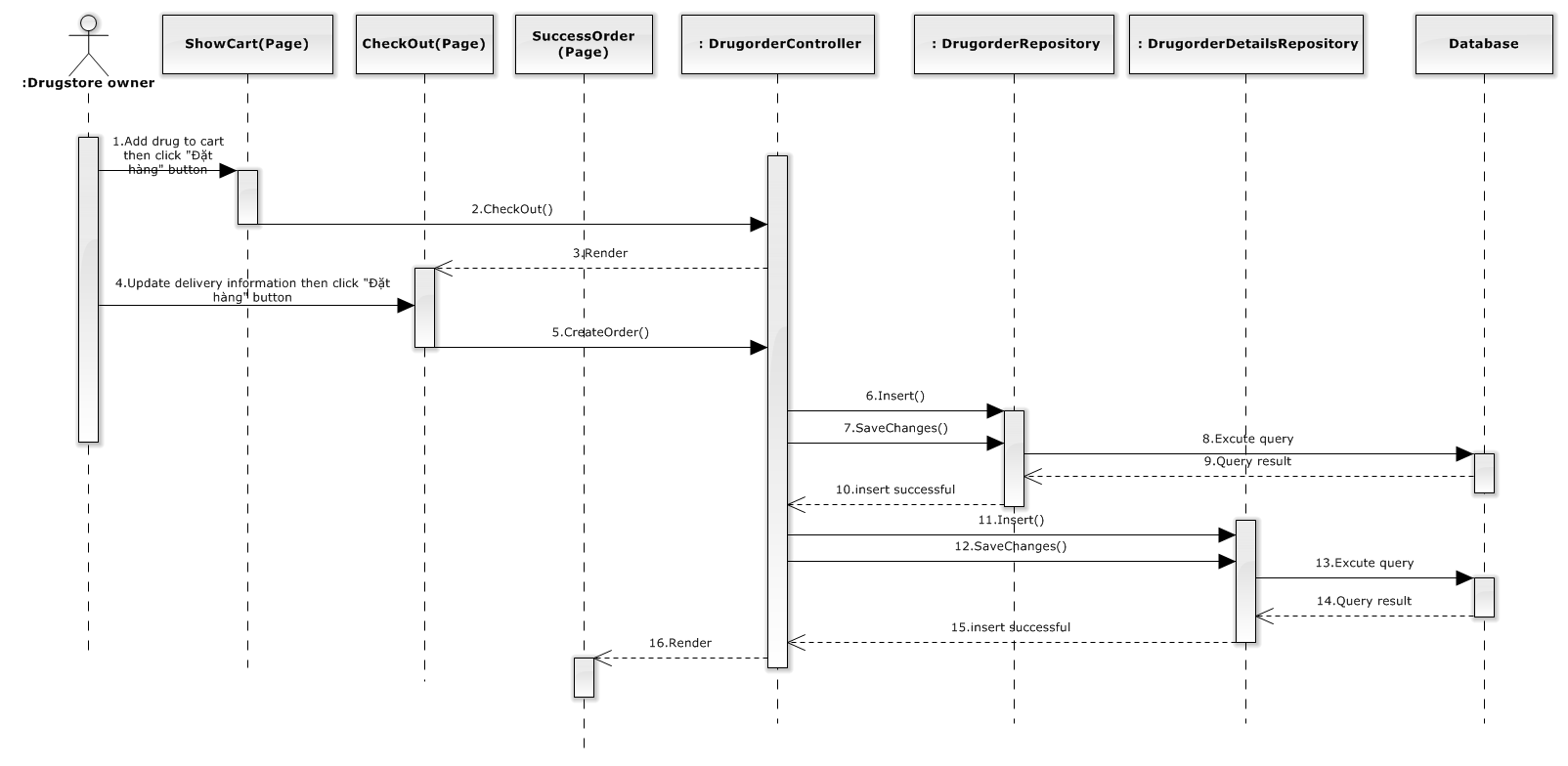
### <Guest> DrugDetails



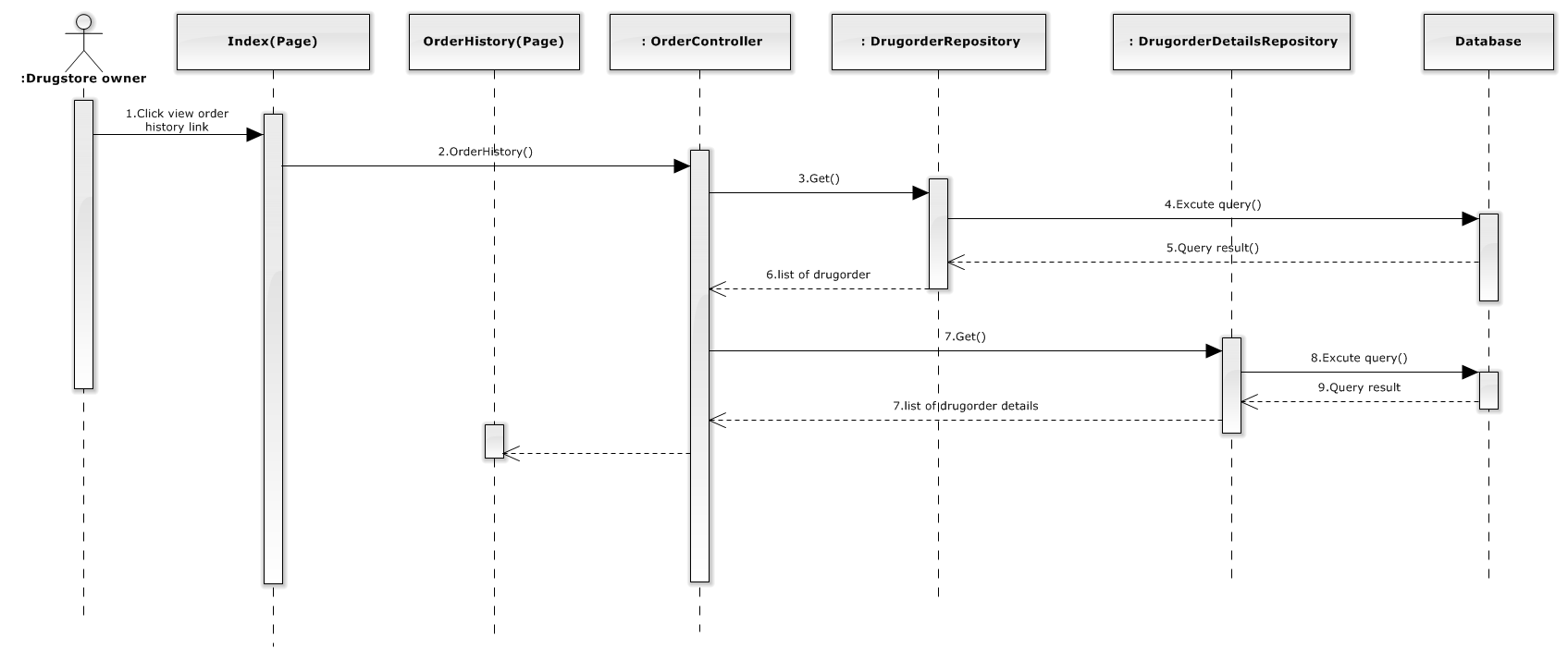
### <Guest> ForgotPassword



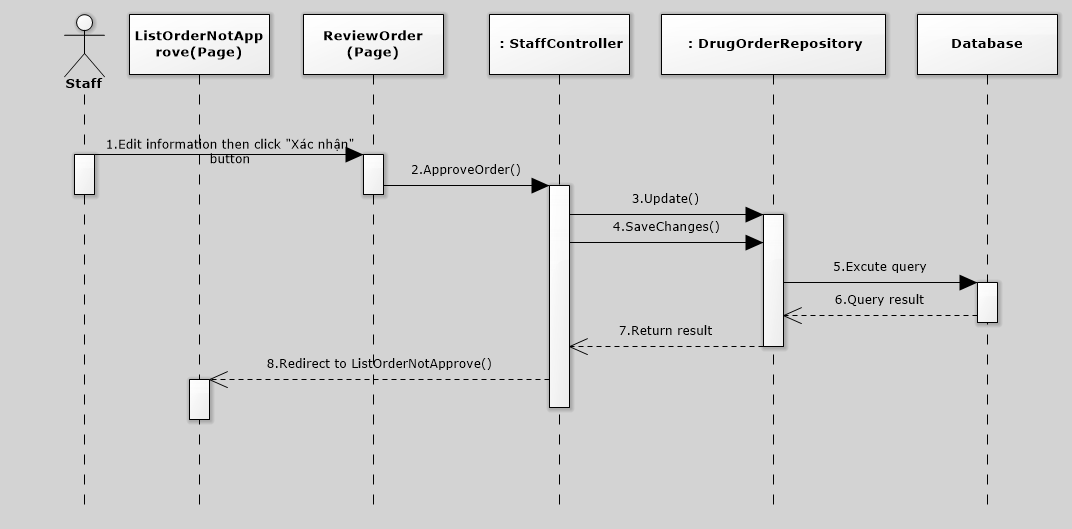
### <User> Create Order



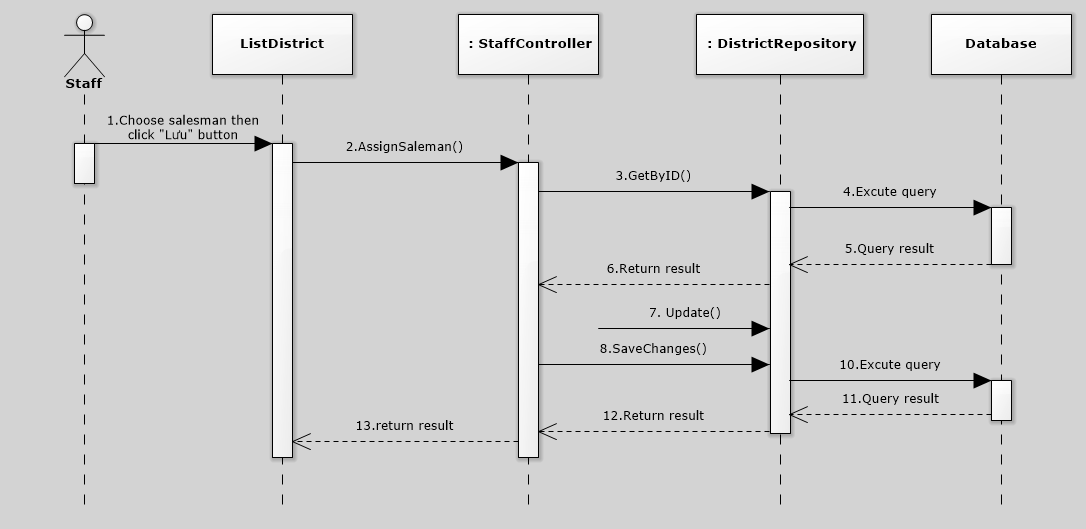
### <User> View order history



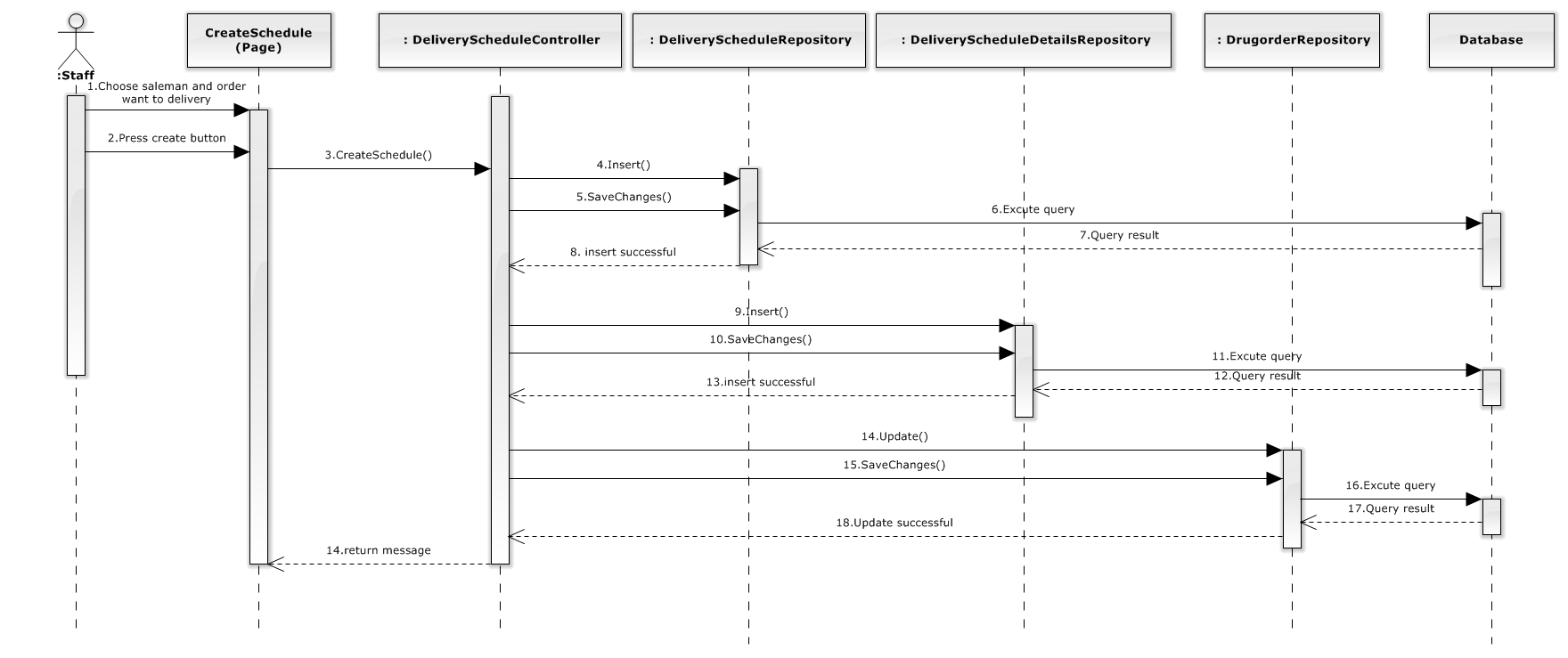
### <Staff> Approve order



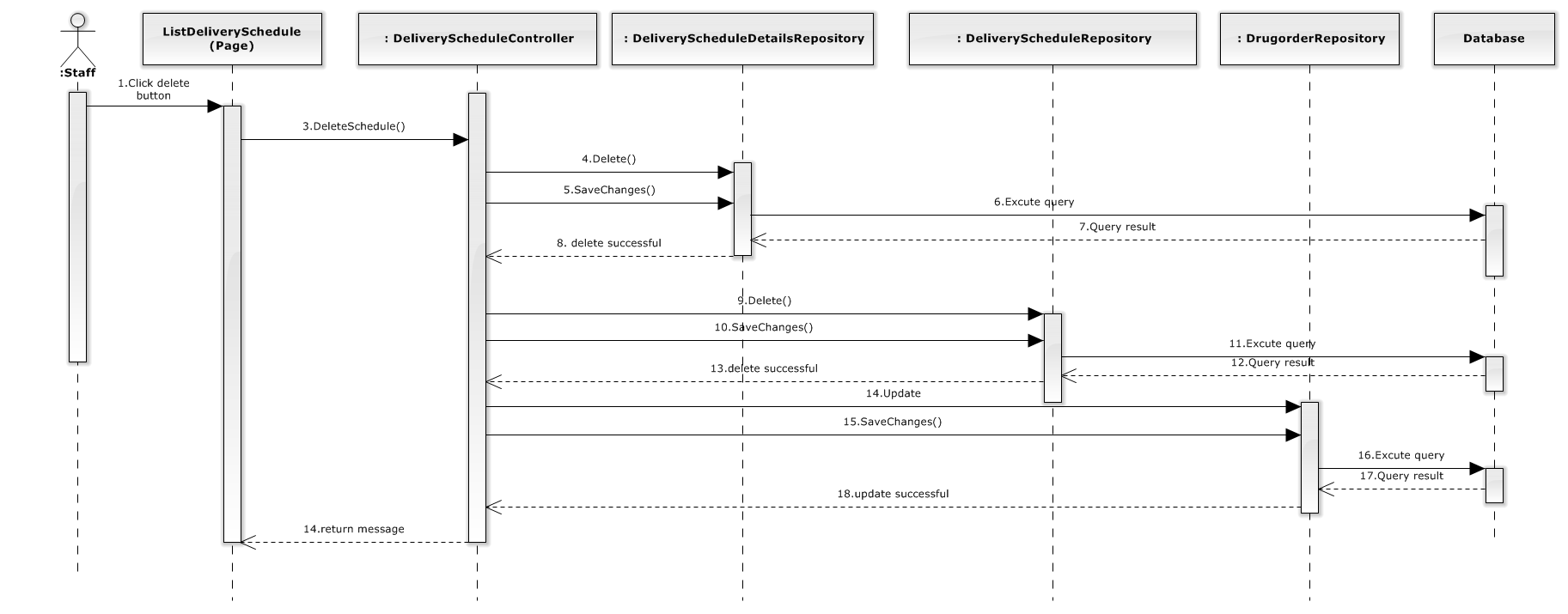
### <Staff> Assign salesman



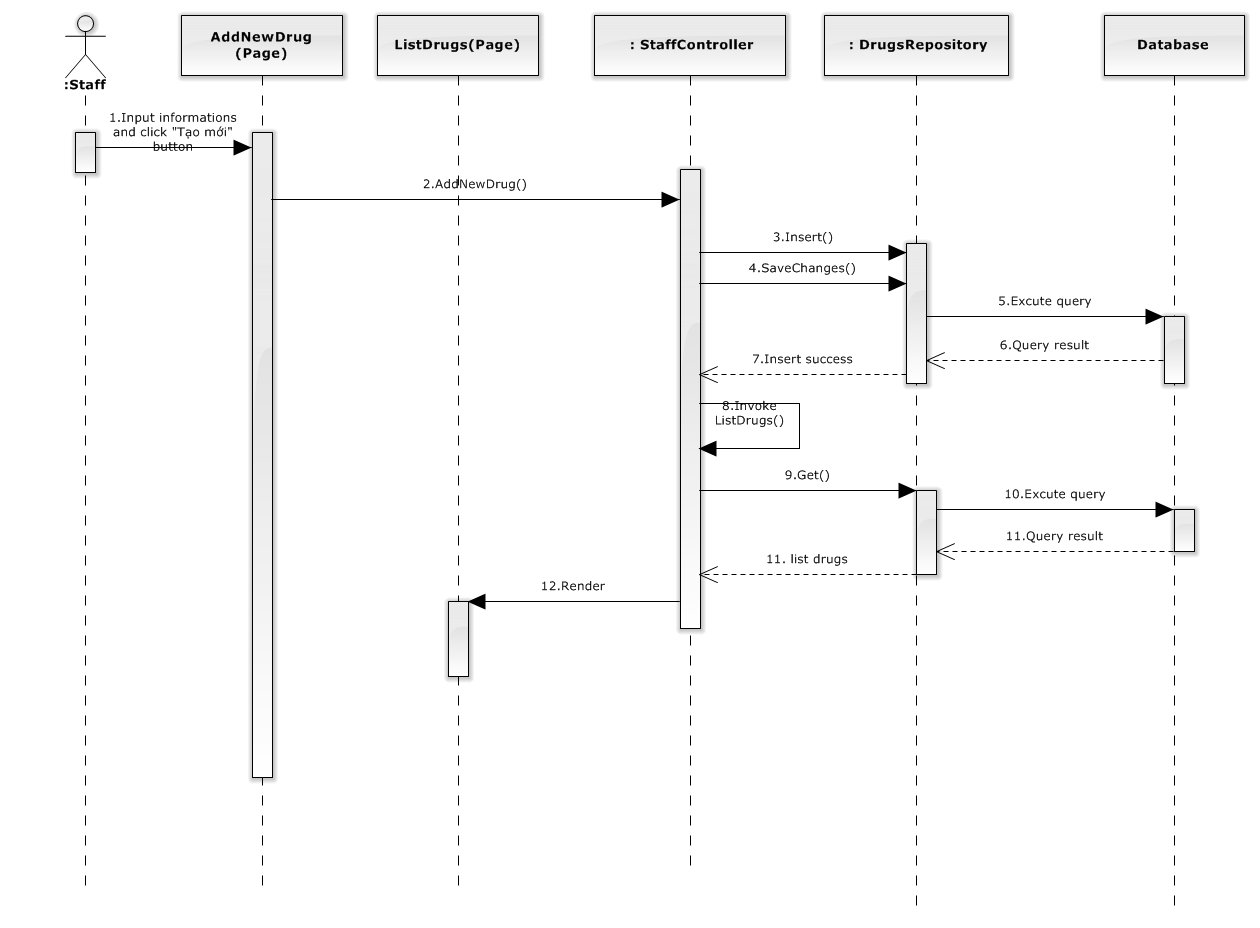
### <Staff> Create delivery schedule



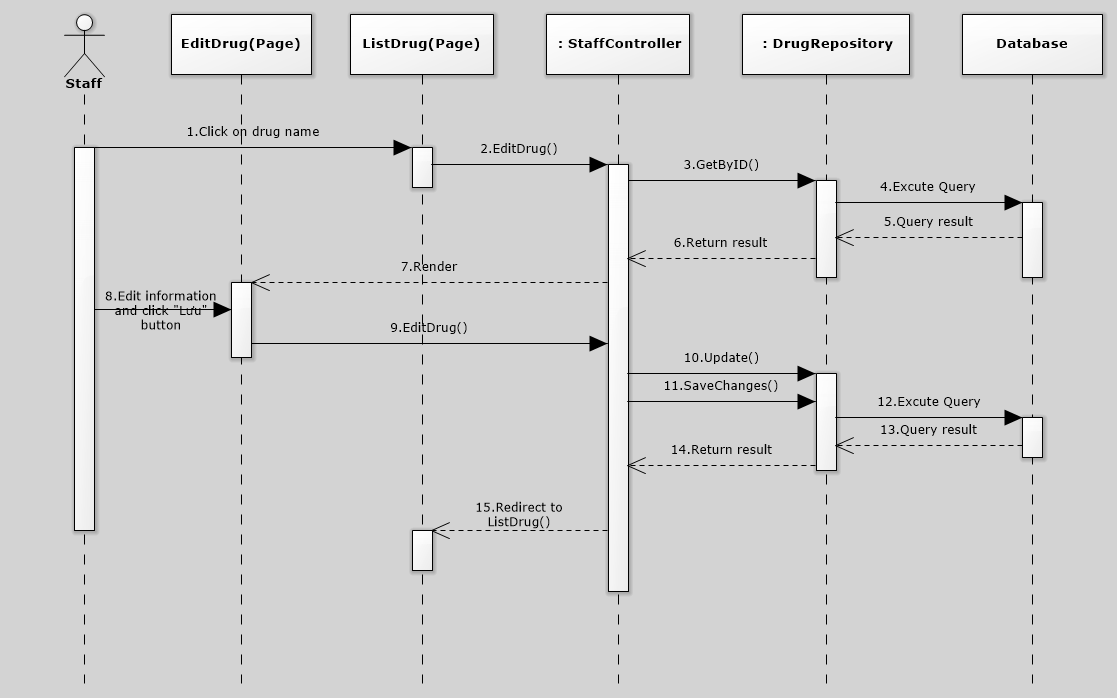
### <Staff> Delete delivery schedule



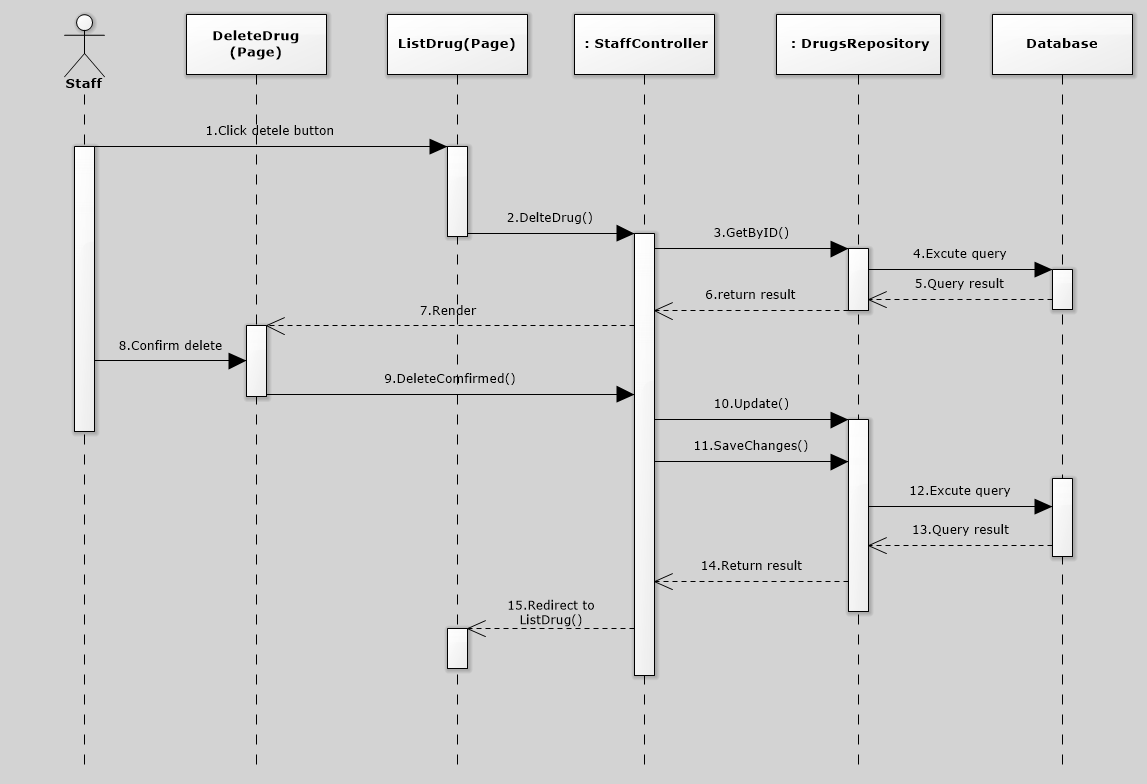
### <Staff> Add drug



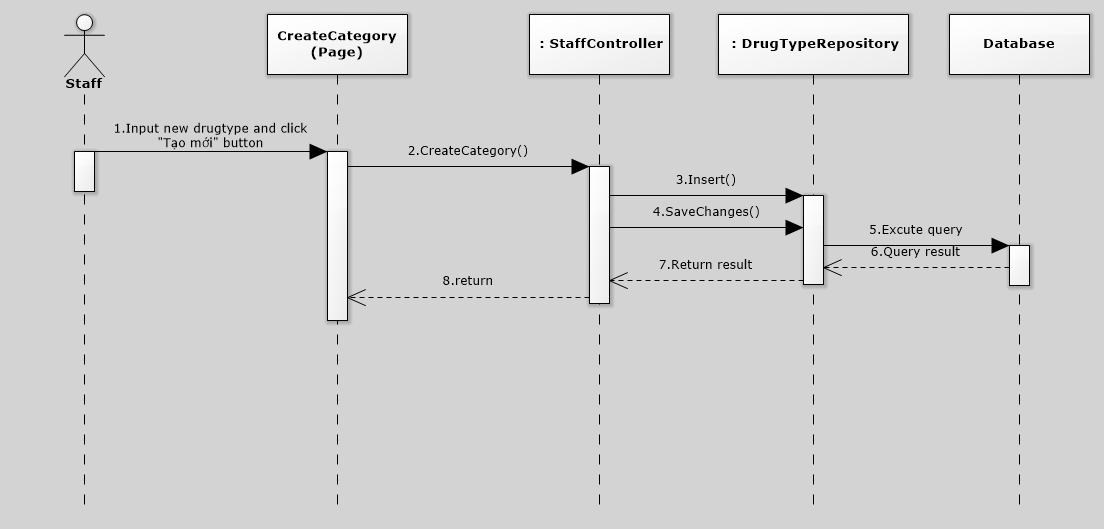
### <Staff> Edit drug



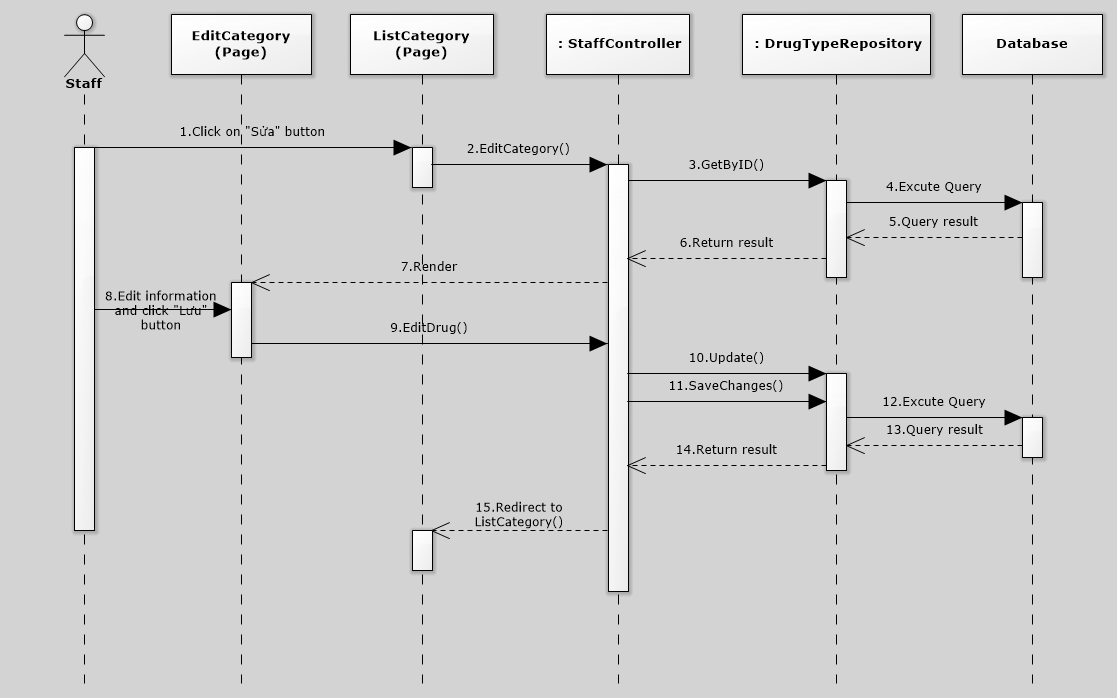
### < Staff > Delete drug



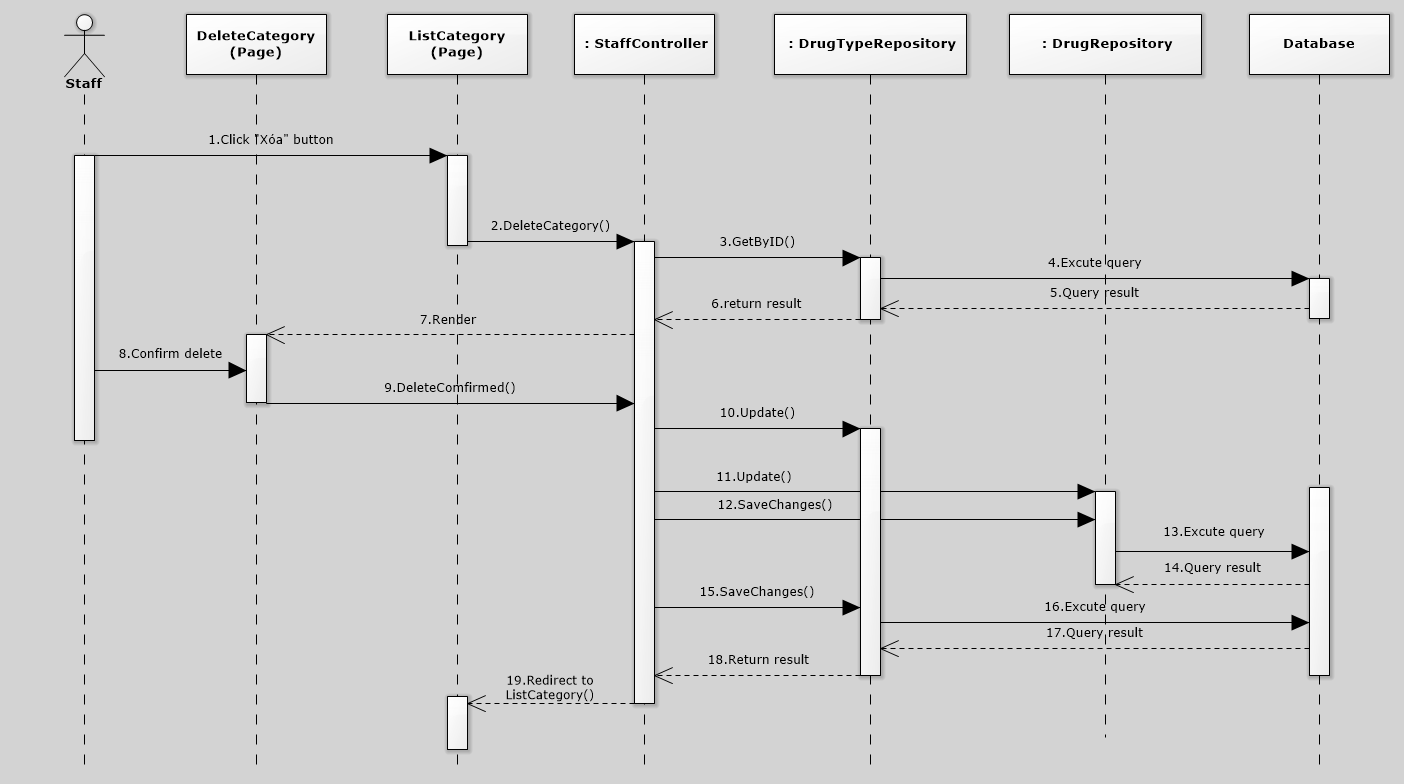
### <Staff> Add drug type



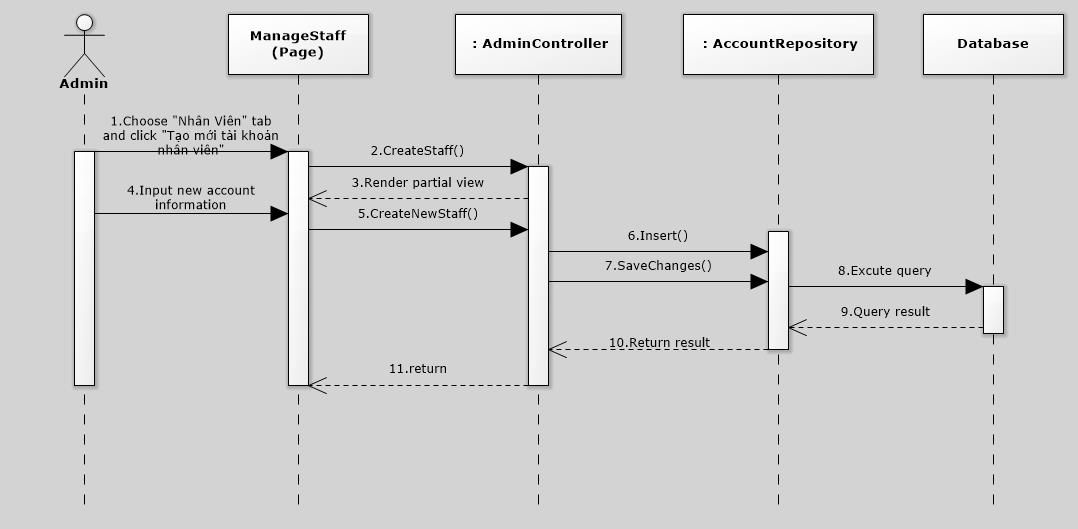
### <Staff> Edit drug type



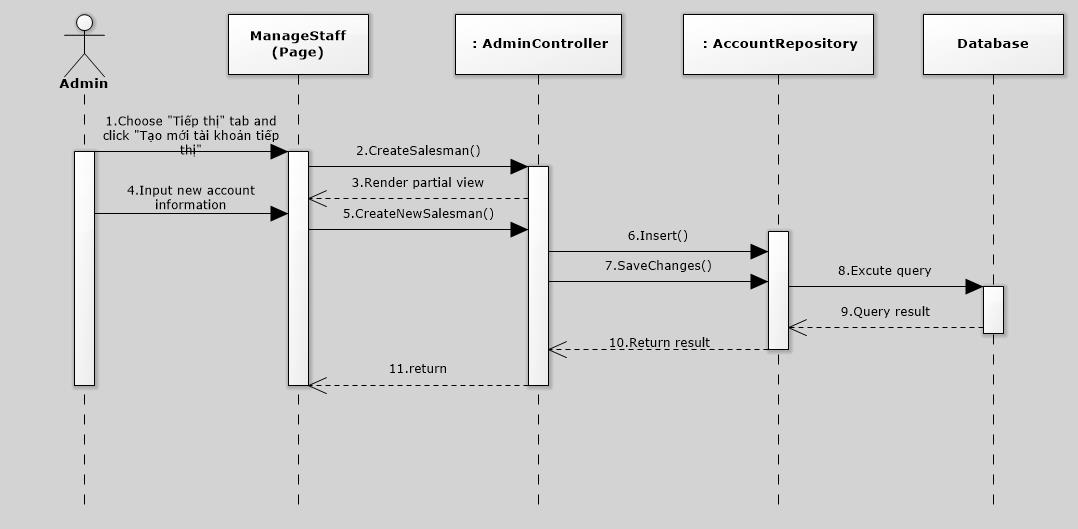
### <Staff> Delete drug type



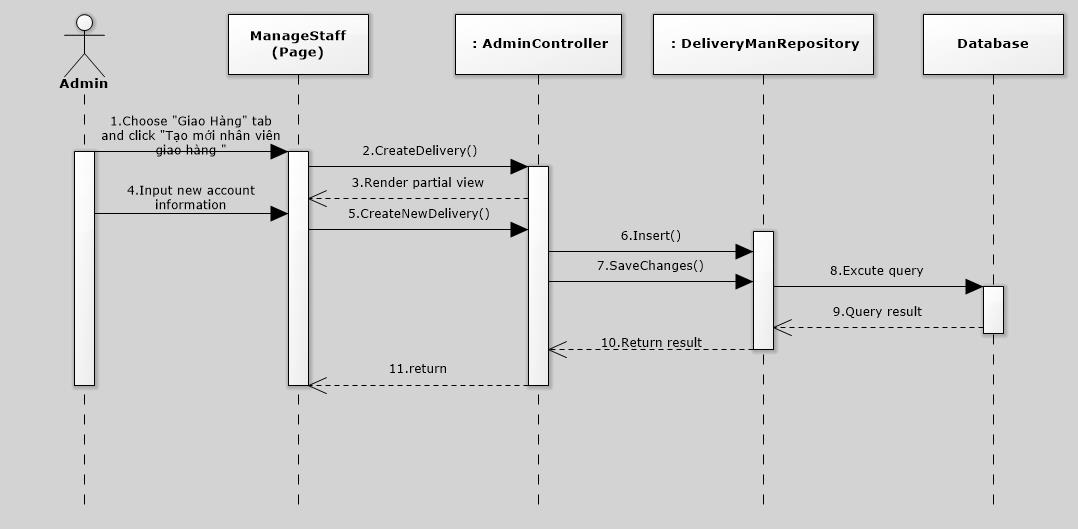
### <Admin> Create staff



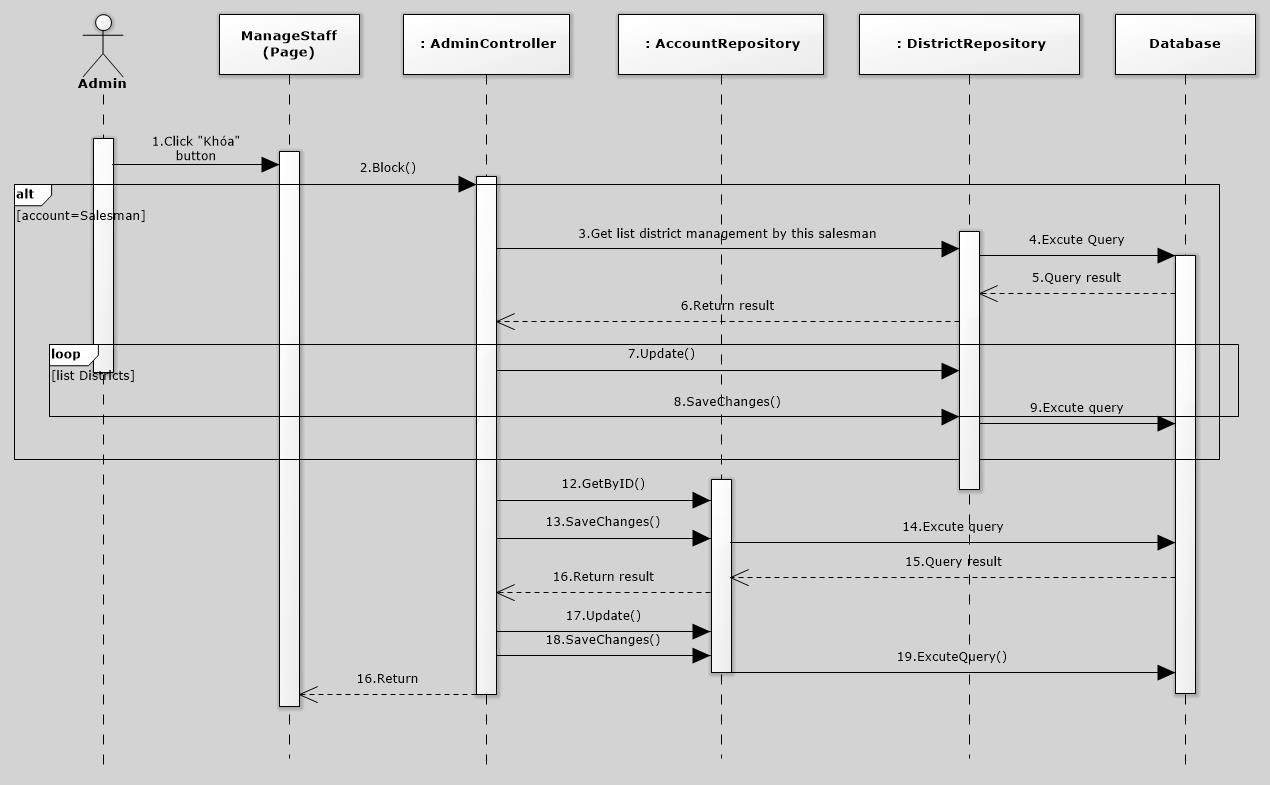
### <Admin> Create salesman



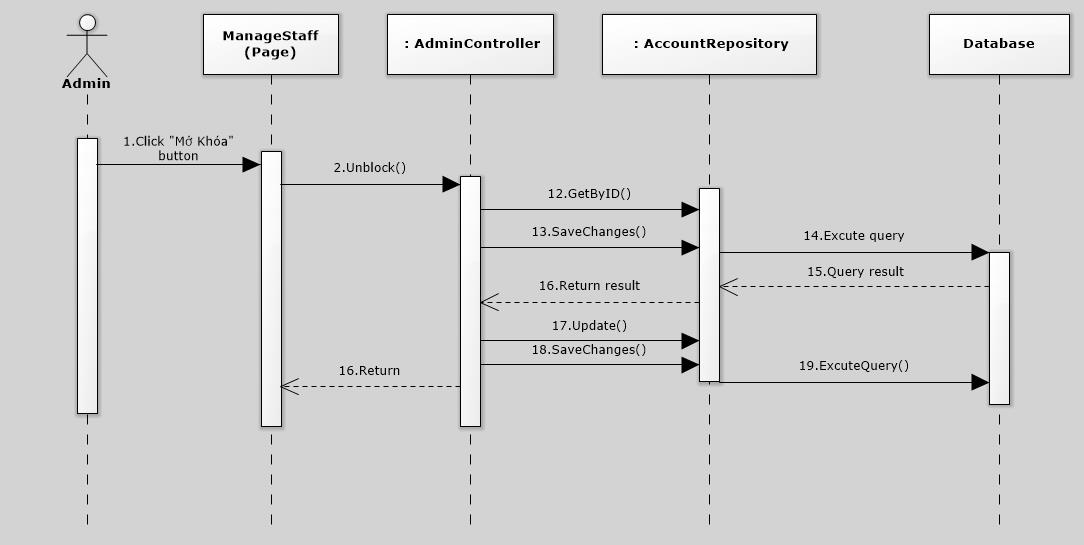
### <Admin> Create deliveryman



### <Admin> Deactivate account



### <Admin> Activate account



# Database Design

## Physical database design



The details of each relation are specified below

### Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| AccountID | int |  | Auto generated | PK | Uniquely identifies of the account |
| Password | nvarchar(50) |  | - |  | Password of user |
| Username | nvarchar(50) |  | - |  | Username of user |
| RoleID | int |  | - | FK | Role of user in the system:customer, admin, salesman or staff |
| ProfileID | int |  | - | FK | Profile Id of user |
| IsPending | bit |  | - |  | Status of user |
| IsActive | bit |  | - |  | User is banned or not |

Unique: AccountID

Foreign key: RoleID, ProfileID

### AccountProfile

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| ProfileID | int |  | Auto generated | PK | Uniquely identifies of the profile |
| Fullname | nvarchar(50) |  | - |  | Fullname of user registered to the system |
| Phone | nvarchar(50) |  | - |  | Phone of user registered to the system |
| Email | nvarchar(50) |  | - |  | Email of user registered to the system |
| Address | nvarchar(250) |  | - |  | Address of user registered to the system |
| Coordinate | nvarchar(500) |  | - |  | Coordiante of address. |

### DeliveryMan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DeliveryManID | int |  | Auto generated | PK | Uniquely identifies of the DeliveryMan |
| Fullname | nvarchar(50) |  | - |  | Fullname of delivery man |
| Phone | nvarchar(50) |  | - |  | Phone of delivery man |
| Email | nvarchar(50) |  | - |  | Email delivery man |

Unique: DeliveryManID

Foreign key: N/A

### DeliverySchedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DeliveryScheduleID | int |  | Auto generated | PK | Uniquely identifies of the schedule |
| CreateDate | date |  | - |  | Date of create schedule |
| DueDate | date |  |  |  | Date of delivery product |
| DeliveryManID | int |  |  | FK | Deliveryman is assigned to delivery schedule |
| Status | int |  |  |  | Status of schedule |

Unique: DeliveryScheduleID.

Foreign key: DeliveryManID.

### DeliveryScheduleDetails

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DeliveryScheduleDetailsID | int |  | Auto generated | PK | Uniquely identifies a schedule details |
| DeliveryScheduleID | int |  | - | FK | Delivery details belong to delivery schedule |
| DrugOrderID | int |  |  | FK | Orders belong to delivery schedule. |

Unique: DeliveryScheduleDetailsID

Foreign key: DeliveryScheduleID, DrugOrderID

### DiscountRate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugstoreTypeID | int |  | - | FK | Type of drugstore has different discount rate |
| DrugID | int |  | - | FK | Drug has price discount |
| Discount | float |  | - |  | Price discount of drug |

Unique: N/A.

Foreign key: DrugID, DrugstoreTypeID

### Drugorder

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugOrderID | int |  | Auto generated | PK | Uniquely identifies the drug order. |
| DrugstoreID | int |  |  | FK | Drugstore order delivery |
| Note | nvarchar(50) |  |  |  | Some more information for order |
| TotalPrice | float |  |  |  | Total price of order |
| DateOrder | datetime |  |  |  | Date that user order |
| IsActive | bit |  |  |  |  |
| Status | int |  |  |  | Status of order |

Unique: DrugOrderID

Foreign key: DrugstoreID

### DrugOrderDetails

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugOrderDetailsID | int |  | Auto generated | PK | Uniquely identifies the Drug order details |
| DrugOrderID | int |  |  | FK | Details is belong drug order |
| DrugID | int |  |  | FK | All of drug that user order |
| UnitID | int |  |  | FK | Type of drug( box, package) |
| Quantity | int |  |  |  | Quantity of drug |

Unique: DrugOrderDetailsID.

Foreign key: DrugOrderID, DrugID, UnitID

### Drugstore

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugstoreID | int |  | Auto generated | PK | Uniquely identifies an drugstore |
| DrugstoreName | nvarchar(100) |  |  |  | Name of drugstore |
| Address | nvarchar(100) |  |  |  | Address of drugstore |
| Coordinate | nvarchar(100) |  |  |  | Address location of drugstore |
| OwnerID | int | - |  | FK | The account owner of drugstore |
| DistrictID | int |  |  | FK | District which drugstore belong to |
| DrugstoreTypeID | int |  |  | FK | Type of drugstore |
| IsActive | bit |  |  |  | Check active drugstore |
| Debt | float |  |  |  | Debt of drugstore |

Unique: DrugstoreID.

Foreign key: OwnerID, DistrictID, DrugstoreTypeID

### DrugstoreType

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugstoreTypeID | int |  | Auto generated | PK | Uniquely identifies the drugstore type |
| DrugstoreTypeName | nvarchar(50) |  | - | FK | Name of drugstore type |

Unique: DrugstoreTypeID.

Foreign key: N/A.

### Drugtype

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugTypeID | int |  | Auto generated | PK | Uniquely identifies the drug type |
| DrugTypeName | nvarchar(50) |  |  | FK | Name of drug type |

Unique: DrugTypeID

Foreign key: N/A

### Payment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| PaymentID | int |  | Auto generated | PK | Uniquely identifies the payment |
| DrugstoreID | int |  |  | FK | Payment belong to this drugstore |
| PaymentType | bit |  |  |  | Check drugstore paid or loan |
| Amount | float |  |  |  | Amount of payment |
| Balance | float |  |  |  | Price after paid or loan |
| Date | datetime |  |  |  | Date of payment |

Unique: PaymentID.

Foreign key: DrugstoreID.

### UnitPrice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DrugID | int |  | - | PK | The ID of drug |
| UnitID | int |  | - | The ID of Unit |
| UnitPrice | float |  | - |  | Price of unit |

Unique: DrugID, UnitID

Foreign key: N/A

### Role

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| RoleID | int |  | Auto generated | PK | Uniquely identifies the role |
| RoleName | int |  | - |  | The name of role(admin, drugstoreuser, staff, salesman) |

Unique: N/A.

Foreign key: N/A.

### Unit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| UnitID | int |  | Auto generated | PK | Uniquely identifies the unit |
| UnitName | Int |  | - |  | Name of unit |

Unique: UnitID.

Foreign key: N/A

### City

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| CityID | int |  | Auto generated | PK | Uniquely identifies the unit |
| CityName | string |  | - |  | Name of city |

Unique: CityID.

Foreign key: N/A

### District

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Null** | **Default Value** | **P/F Key** | **Description** |
| DistrictID | int |  | Auto generated | PK | Uniquely identifies the unit |
| DisctricName | string |  | - |  | Name of disctrict |
| CityID | Int |  | - |  | ID of city belong to |
| SalesmanID | int |  | - |  | ID of account managed |
|  |  |  |  |  |  |

Unique: DistrictID.

Foreign key: CityID, SalesmanID

# Algorithms

### Neareast neighbor alrorithms:

#### Definition

The **nearest neighbour algorithm** was one of the first [algorithms](http://en.wikipedia.org/wiki/Algorithm) used to determine a solution to the [travelling salesman problem](http://en.wikipedia.org/wiki/Travelling_salesman_problem). In it, the salesman starts at a random city and repeatedly visits the nearest city until all have been visited. It quickly yields a short tour, but usually not the optimal one.

Reference: <http://en.wikipedia.org/wiki/Nearest_neighbour_algorithm>

#### Flowchart

