Sequence diagram

POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY

INFORMATION TECHNOLOGY DEPARTMENT

PROJECT INTRODUCTION TO SOFTWARE ENGINEERING

Course Group: 01

Group Topic: Store Management

Project Group: 08
Member List:

1. Vu Mai Linh - B20DCCN023
2. **Nguyen Hong Son - B20DCVT311**
3. Nguyen Dang Tien - B20DCCN594

Ha Noi, March 2022

Sequence diagram

Use case for module: Import product

Function's description:

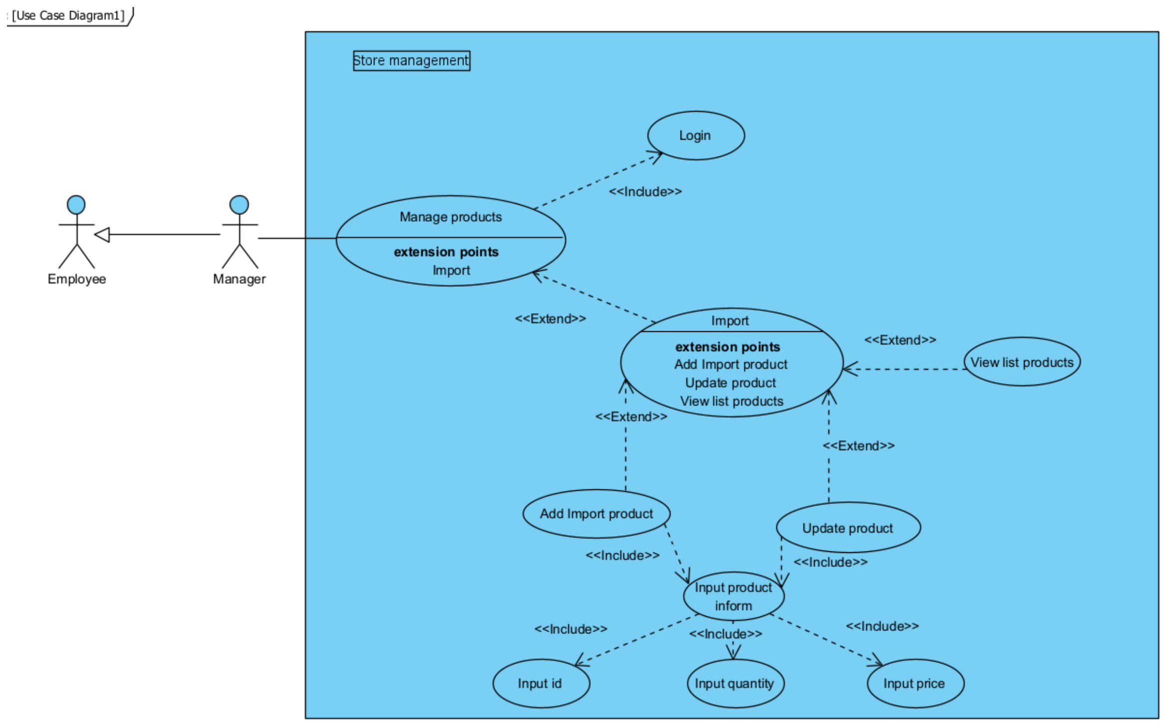
1. A manager logs in into the system.
2. The manager's UI appears, it has the following options: manage the store information, manage the storage, and view the statistical reports.
3. The manager selects to manage storage.
4. The storage management UI appears with three options: import products, export products.
5. The manager clicks on the import function.
6. The import products UI appears with an input text to enter the id, quantity, unit price and a submit button.
7. The manager enters an id, quantity, unit price of the product to add and then, clicks on the save button.
8. The list of products will show all products are saved -> The manager clicks on the submit button.
9. The bill UI appears with the list of import products, total quantity and total price, and a create button.
10. The manager can click to each product to edit or remove, then click to create a button to complete importing products.
11. The system announces a success importing and then returns to the manager storage UI.

The interfaces with the related interactions:

1. Login.
2. Import has three sub functions: add import product, view and update import product.
3. Import and update has the same one action: Input product information

So we have the use case as follow:

Sequence diagram



Use case description:

- Import: this use case enables the manager to add a new product into the store
- View list products: this use case enables the manager can see the list of imported products, total quantity and total price
- Add import product: this use case enables the manager add new import product
- Update product: this use case enables the manager change imported product
- Input product information: this use case enables the manager input the information of import product

Sequence diagram

Standard scenario of use case: Import products

Scenario	Import products
Actor	Manager
Pre-condition	The managers have successfully logged in, get all the import products
Post-Condition	Products in storage is added
Main event	1. The login interface appears with: an input text for username as “manager01”, an input text for password as “******”, a button to login. 2. The manager enters username, password and clicks on the login button. 3. The main manager UI appears with two options: Manage store, manage products. 4. The manager chooses to manage products. 5. The storage management UI appears with the three options: import, export products. 6. The manager chooses to import products. 7. The import products UI appears with: three input texts for the id, unit price and quantity of product, a button to save, a button to clear list, a button to confirm list. 8. The manager entered product id as “RICE01”, unit price as “100,000 VND”, quantity as “100” and clicked the save button. 9. The list of products will appear as follows:

number	id	unit price	quantity	price
1	RICE01	100,000 VND	100	10,000,000 VND

	10. The manager click on submit button, complete import product 11. The UI alert for successful import appears and the system returns to the storage manager UI.
Exception	4. The system alerts that the username/password is incorrect. 10.1. This products is imported 10.2. Input data is incorrect format

Sequence diagram

Class Entity Extraction

Step 1: describe the module within a paragraph

The manager logs into the store system. After login successfully, the manager chooses the storage management section. The system gives the manager a form to add new products. The manager chooses to add a new product and fill it with information. The system updates a screen including: a list of products that the manager will import to storage. The manager clicks the submit button. On importing successfully, the system automatics save detail of all products, importing time and creating a bill.

Step 2+3: Extract nouns and classify them

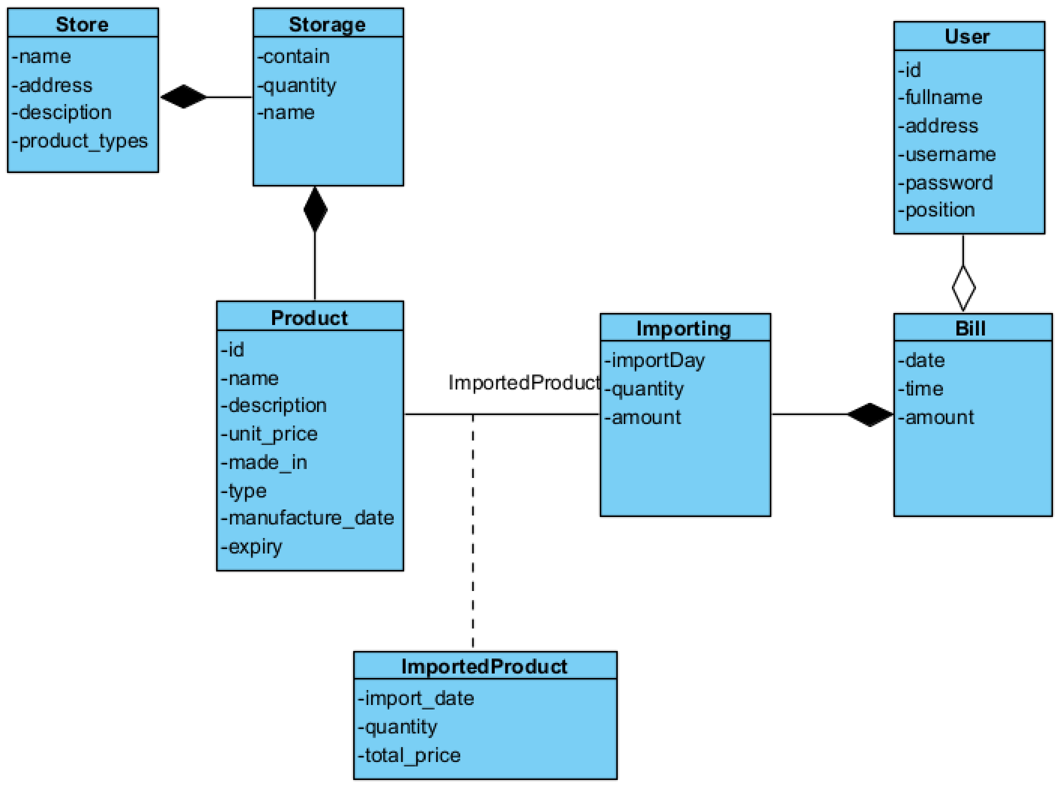
- System: an abstract noun –> reject.
- Information: a common noun –> reject.
- Manager: this is a kind of member in the system → class: User
- Store: need to be managed -> class: Store
- Storage: need to be managed -> class Storage
- Form: need to be managed -> class: Form
- Product: need to be managed -> class: Product
- Bill: need to be managed -> class: Bill

So, we obtain the initial classes: User, Store, Form, Product, Bill

Step 4+5: quantity and object relationships among classes:

- The store may have many Storage, a Storage belongs only to the store. So, Store-Storage is 1-n.
- The User can import many Products, a Product may be imported at different order but it will show in the list is just total quantity. So, the User-Product is 1-n.
- The User can have many Importings in different times. A Product may be imported in many Importings. In the Importing, the User may import many different Products. So, Importing-Product is n-n. We need more class between them: Form. The Importing and Product determine an unique Form. This association relationship also determines some information: import date, quantity, total price.
- The receptionist may create many Bills for many Importings. So, User-Bill is 1-n.

Sequence diagram



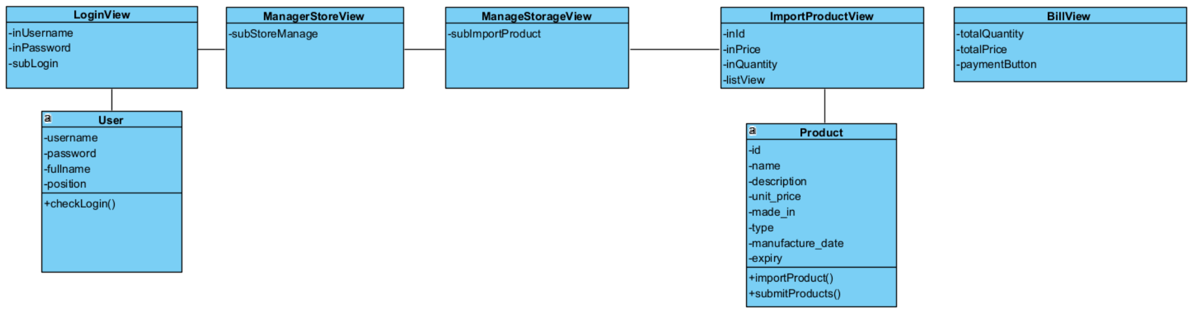
Sequence diagram

Classes diagram (analysis)

Analysis module “Import products”

- Enter the system → The login interface is appeared → need a class: LoginView
    - Input for username → inUsername
    - Input for password → inPassword
    - A submit to login → subLogin
- Enter the username/password → The system must check if the login is correct → need a method:
    - Name: checkLogin()
    - Input: username, password (of the class User)
    - Output: Boolean
    - Assign to the entity class: User
- Once login is successful → the main interface of the manager is appeared → need a class: ManagerStoreView which has at least:
    - An option to choose to manage storage → subStoreManage
- Choose the option subStorageManage → the return menu interface is appeared → need a class: ManageStorageView which has at least:
    - An option to choose to import products → subImportProducts
- Choose the option insubImportProducts → The import product interface is appeared -> need a class: ImportProductView:
    - three input text to enter id, unitPrice, quantity -> inId, inPrice, inQuantity
    - a save button -> subSave
- Enter id, unitPrice, quantity and click save -> the system has to save products into the storage:
    - Name: importProducts()
    - Input: id, unitPrice, quantity (of the class Product)
    - Output: Boolean
    - Assign to the entity class: Product
- The list of imported product will be listed in the ImportProductView -> listView
- Add some products and click submit -> the system has to update all in to the DB -> need a method:
    - name: submitProducts()
    - input: list object of Product
    - output: none or boolean
    - assign to the entity class: Product
- On submit successful -> the interface of the bill is appeared -> need a class: BillView include:
    - total quantity: totalQuantity
    - total price: totalPrice
    - a button payment: paymentButton

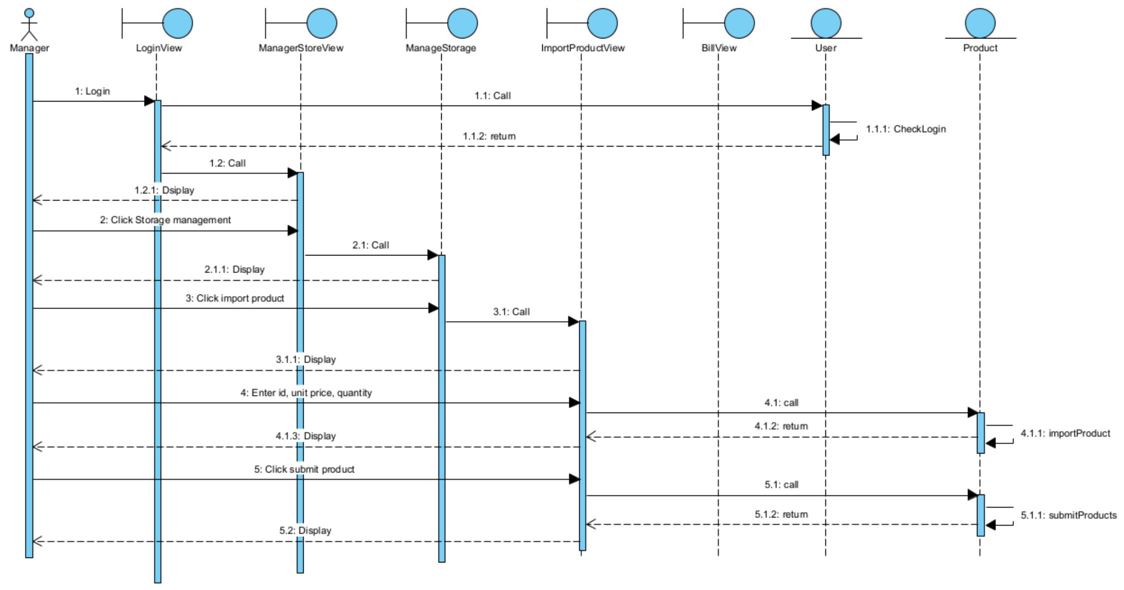
Sequence diagram



Sequence diagram

Scenario version 2

1. The manager enters username/password and then clicks on the Login button.
2. The class LoginView calls the class User to process.
3. The class User calls the method checkLogin(). The login is successful.
4. The class User returns the results to the class LoginView
5. The class LoginView calls the class MangerStoreView.
6. The class ManagerStoreView displays itself to the manager.
7. The manager chooses the option to see ManageStorageView.
8. The class ManagerStoreView calls the class ManageStorageView.
9. The class ManageStorageView displays itself to the manager.
10. The manager chose the option to import products.
11. The class ImportProductView displays itself to the manager.
12. The manager clicks into the button and adds a new product.
13. The ImportProductView displays input text to enter id, unit price and quantity.
14. The manager enters the id, unit price and quantity of the product and clicks the save button.
15. The class ImportProductView calls the class Product to process
16. The class Product calls the method importProduct()
17. The class ImportProductView displays a successful message to the manager.
18. The manager clicks the submit button.
19. The class Product calls the method submitProducts().
20. The class ImportProductView displays a successful message to the manager.


Manager
LoginView
ManagerStoreView
ManageStorage
ImportProductView
BillView
User
Product
1: Login
1.1: Call
1.1.1: CheckLogin
1.1.2: return
1.2: Call
1.2.1: Dsiplay
2: Click Storage management
2.1: Call
2.1.1: Display
3: Click import product
3.1: Call
3.1.1: Display
4: Enter id, unit price, quantity
4.1: call
4.1.1: importProduct
4.1.2: return
4.1.3: Display
5: Click submit product
5.1: call
5.1.1: submitProducts
5.1.2: return
5.2: Display