HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communication Technology

Software Design Description

Version: final

eStoreManager
Course: ITSS Software Development

Group 1

Nguyễn Việt Anh

Nguyễn Sỹ An

Theo Mercurio

Hanoi, December 10th, 2018

Index

1 Introduction

1.1 Objective

This document presents the detailed description for eStoreManager – a store management system user group and their usable function at run time. This document also describes the objectives and features of the system, interfaces and constraints of the system in response to external action.

This document is for stakeholders and related software developers.

We are trying to implement a desktop application, which:

+ Have multiple language. Firstly we support English and French.

+ Have clean structure for ease in development and maintain.

+ Have a good UI design using EJS Templating.

1.2 Scope

In reality, any software needs to have the function of managing users or group of users, and dynamically granting permissions for usage of certain functions in the system.

This software will be built as a management tool for a small store, with only 1 branch.

The software's goal includes creating following components:

- Interface to make bills and sell products.
- Interface to manage suppliers, import products.
- Interface to manage employees.
- Interface to manage customer list.
- Interface to manage store finance.

Each function may be utilised by many roles of users in: Admin, cashier, warehouse manager.

1.3 Glossary

- ***Admin:*** *administrator in this software often is the store owner. This user can interact with the system in full permission.*

- ***Cashier:*** *This user is a role of shop keeper: make bills and sell products. Cashier also can have access to history of bills, remove a bill in the past (to modify some information in the bill).*

- ***Warehouse:*** *is a place where goods may be stored before their export or distribution for sale.*

- *Warehouse manager:* *is a user who can manage suppliers, add import bills and be responsible to manage goods in warehouse.*

- ***Supplier:*** *a vendor who provide product for the store at whole sale price. The store imports goods there to sell to to their customers.*

1.4 References

An online store management system: Kiot Viet: https://kiotviet.vn

2 Architecture Design

2.1 *Software Architecture*

2.1.1 Main design

For this project, we use passive Model-View-Controller (MVC) as our architecture pattern for server-side and multi-tier architecture for desktop application(See detail in 2.1.2 and 2.1.3).

In passive MVC on the server, the model is inactive. It is not in action. It does not notify View when it is changes by Controller. The notification task is done by Controller. That's the major difference to the classic MVC.

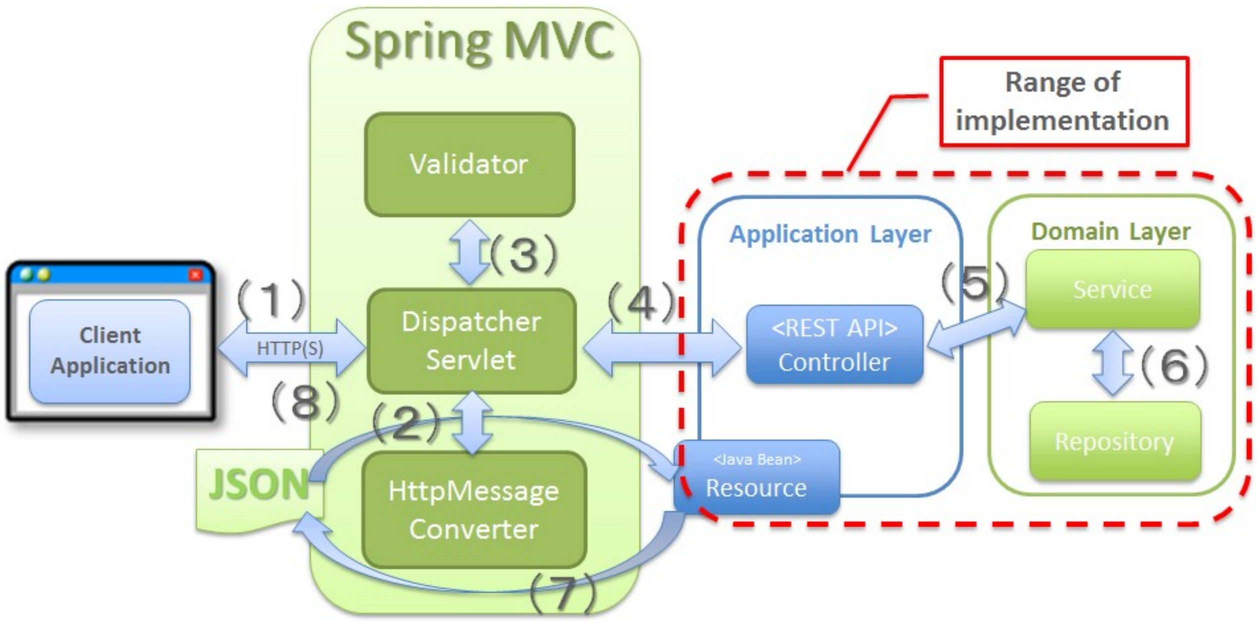
* View: View represents the user interface. The data it shows, gets form Model Component. View gets updated notification from Controller when it updates the Model.
* Controller: Controller acts as intermediate between View and Model. When user fires an event, then View receives it and handover the control to Controller. Then Controller updates the Model and notifies View about update.
* Model: Model contains data for the Application.

In the multi-tier architecture used in desktop application we design the application with multiple layers with 3 main groups: Controllers, Views and Services.

For communicating between cliend-side and server-side, we build a RESTful Web Service.

2.1.2 Server side

When RESTful Web Service is developed using Spring MVC, the application is configured as given below. Among these, implementation is necessary for the portion marked with red frame



(1) : Spring MVC receives a request from client and determines the REST API (handler method of Controller) to be called.

(2): Spring MVC converts the JSON format message specified in request BODY to Resource object by using HttpMessageConverter.

(3): Spring MVC performs input validation for the value stored in Resource object using Validator.

(4): Spring MVC calls REST API.

Here, the Resource that has been converted from JSON and for which input validation is carried out, is delivered to REST API.

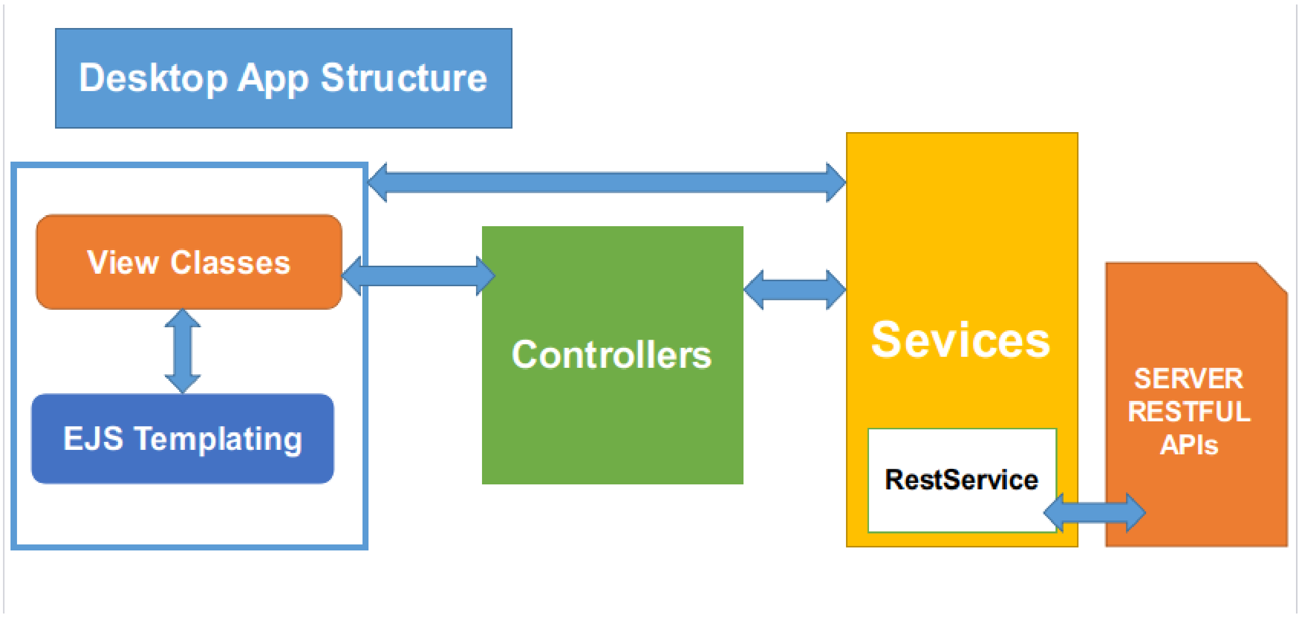
(5): REST API calls Service method and performs the process for DomainObject such as Entity etc.

(6): Service method calls the Repository method and performs CRUD process for the DomainObject such as Entity etc.

(7): Spring MVC converts the Resource object returned from REST API to JSON format message, by using HttpMessageConverter.

(8): Spring MVC sets JSON format message in response BODY and responds to client.

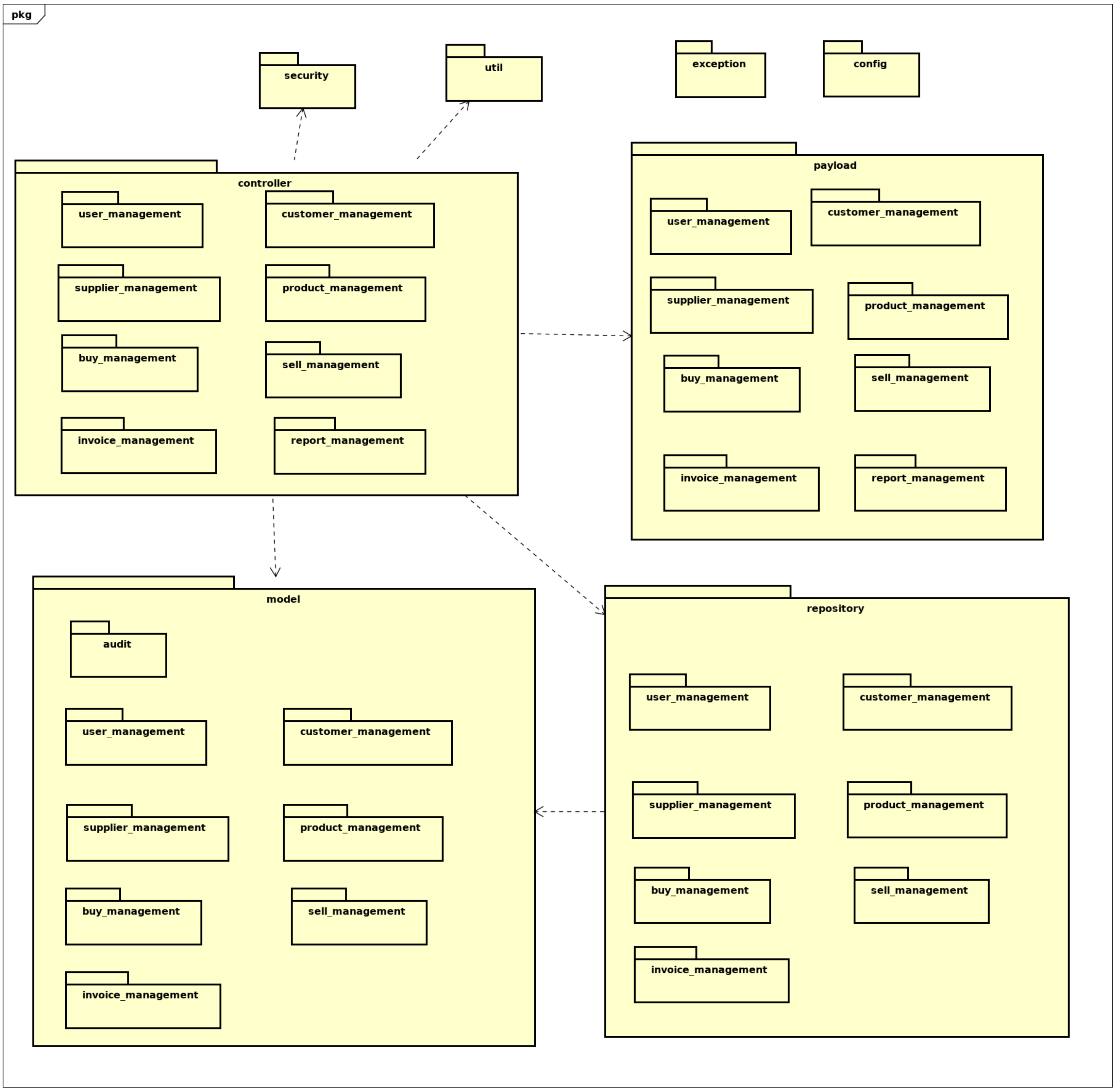
2.1.3 Client side



In the client side desktop application, we divide the architecture into 3 main groups: Views (View classes and EJS Templating), Controllers and Services. These components communicate with others like in the figure.

2.2 *Overall Design*

2.2.1 Overall Design Server-side



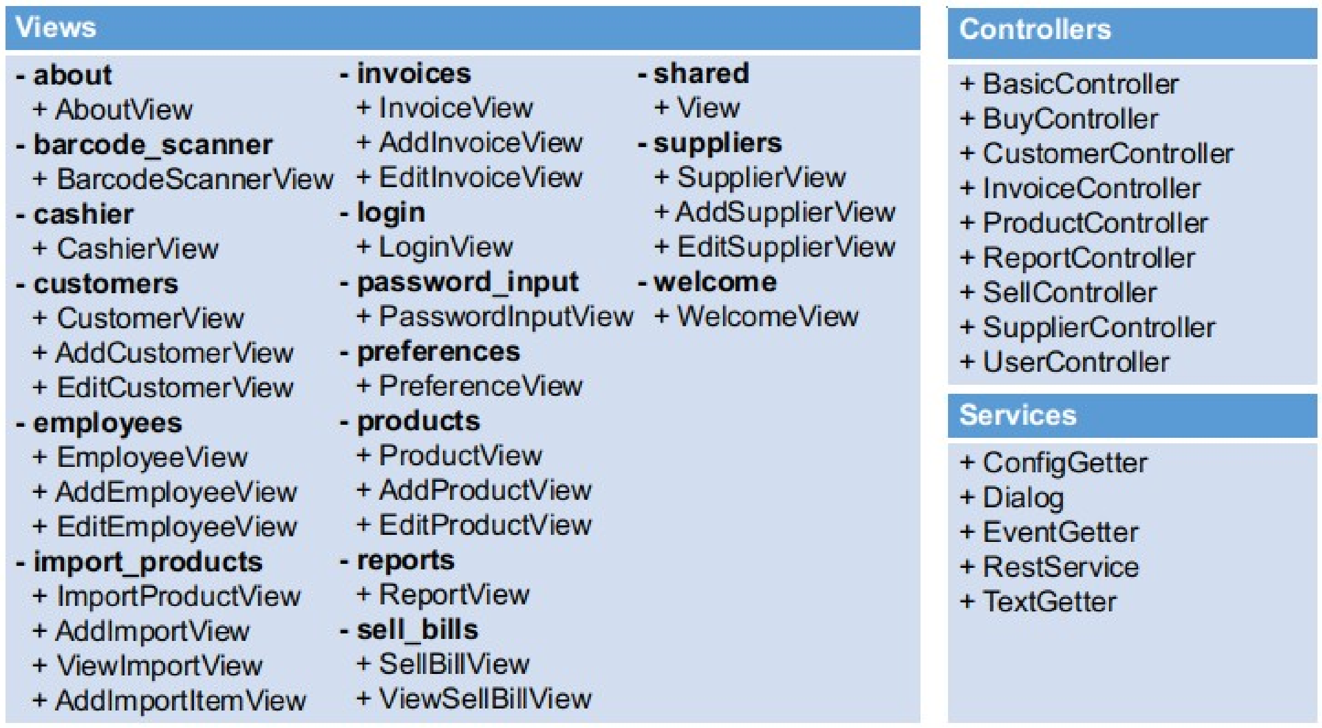
SERVER OVERALL PACKAGE DESIGN UML

There are four main packages:

+ Controller: Take request and calls the appropriate method, set up the model data based on the business logic and returns result.
+ Payload: Create objects hold data that are received and send via HTTP between client-side and server-side.
+ Model: Domain of specific data. Model objects retrieve and store model state.
+ Repository: Implement data access layer, it is close to DAO pattern where DAO classes are responsible or providing CRUD operations on database tables.

In each main package, we divide into many sub-package or sub-module based on its function:
+ User management
+ Customer management
+ Product management
+ Bill management
+ Invoice management

2.2.2 Overall Design Client-side



Views

- about
+ AboutView

- barcode_scanner
+ BarcodeScannerView

- cashier
+ CashierView

- customers
+ CustomerView
+ AddCustomerView
+ EditCustomerView

- employees
+ EmployeeView
+ AddEmployeeView
+ EditEmployeeView

- import_products
+ ImportProductView
+ AddImportView
+ ViewImportView
+ AddImportItemView

- invoices
+ InvoiceView
+ AddInvoiceView
+ EditInvoiceView

- login
+ LoginView

- password_input
+ PasswordInputView

- preferences
+ PreferenceView

- products
+ ProductView
+ AddProductView
+ EditProductView

- reports
+ ReportView

- sell_bills
+ SellBillView
+ ViewSellBillView

- shared
+ View

- suppliers
+ SupplierView
+ AddSupplierView
+ EditSupplierView

- welcome
+ WelcomeView

Controllers

+ BasicController
+ BuyController
+ CustomerController
+ InvoiceController
+ ProductController
+ ReportController
+ SellController
+ SupplierController
+ UserController

Services

+ ConfigGetter
+ Dialog
+ EventGetter
+ RestService
+ TextGetter

CLIENT OVERALL PACKAGE DESIGN UML

- In the **Views** package, we have 16 subpackages, each package contains at least 1 view.
- **shared/View** is the class of a common view. Other views extend this View and add more logic handlers.
- In the **Controllers** package, we have some **Controllers** controlling the logic of the whole application and communicate with server via **Services/RestService.**
- In the **Services** package we have:

+ **ConfigGetter** to save and provide some global configuration.

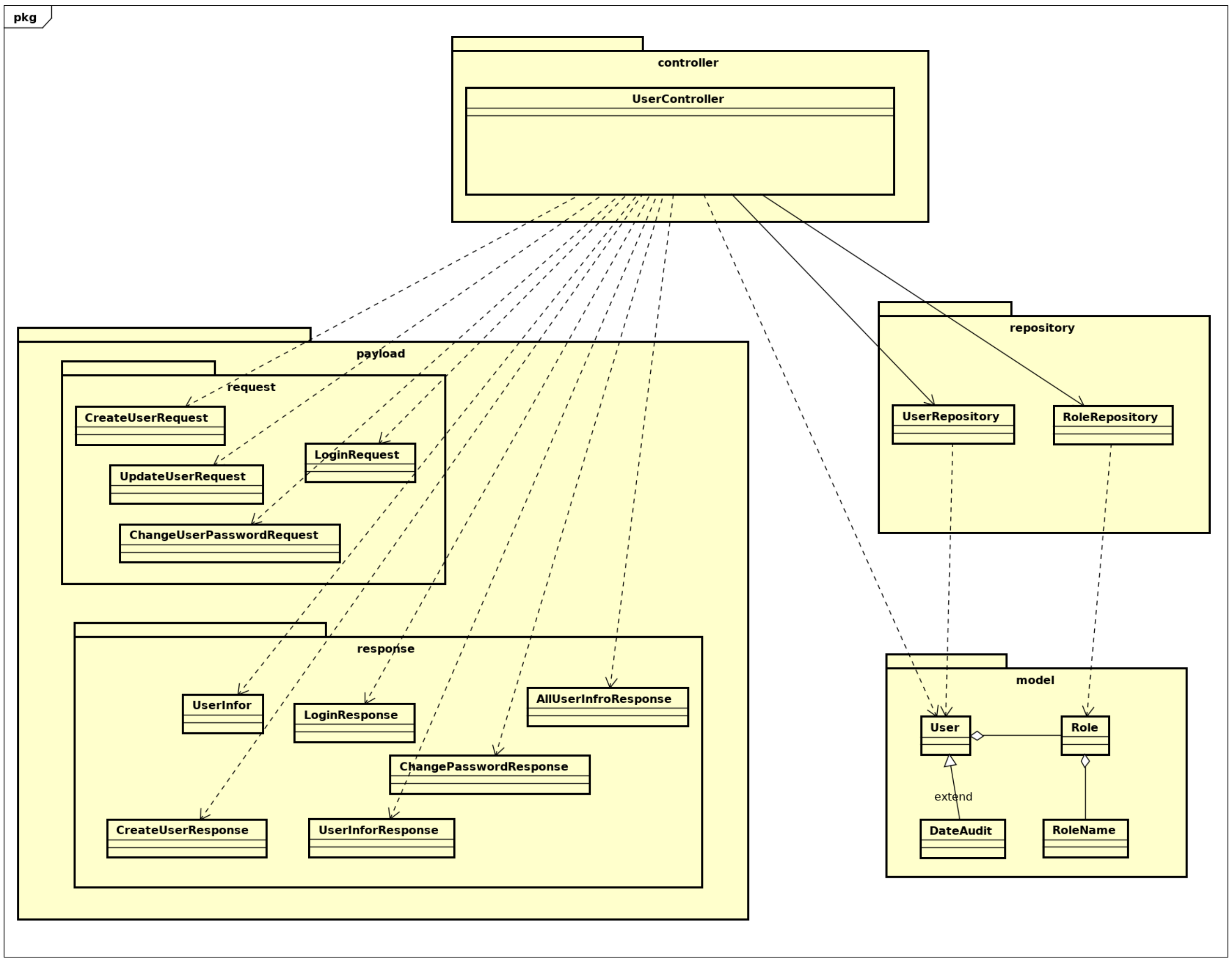
+ **Dialog**: provide Dialog UI service

+ **EventGetter**: provide the event by the event ID. Because this application is mainly based on events, this service plays an important role.

+ **RestService:** provide service to communicate with the server.

+ **TextGetter:** Provide the language service. This application uses this class to implement multiple language.

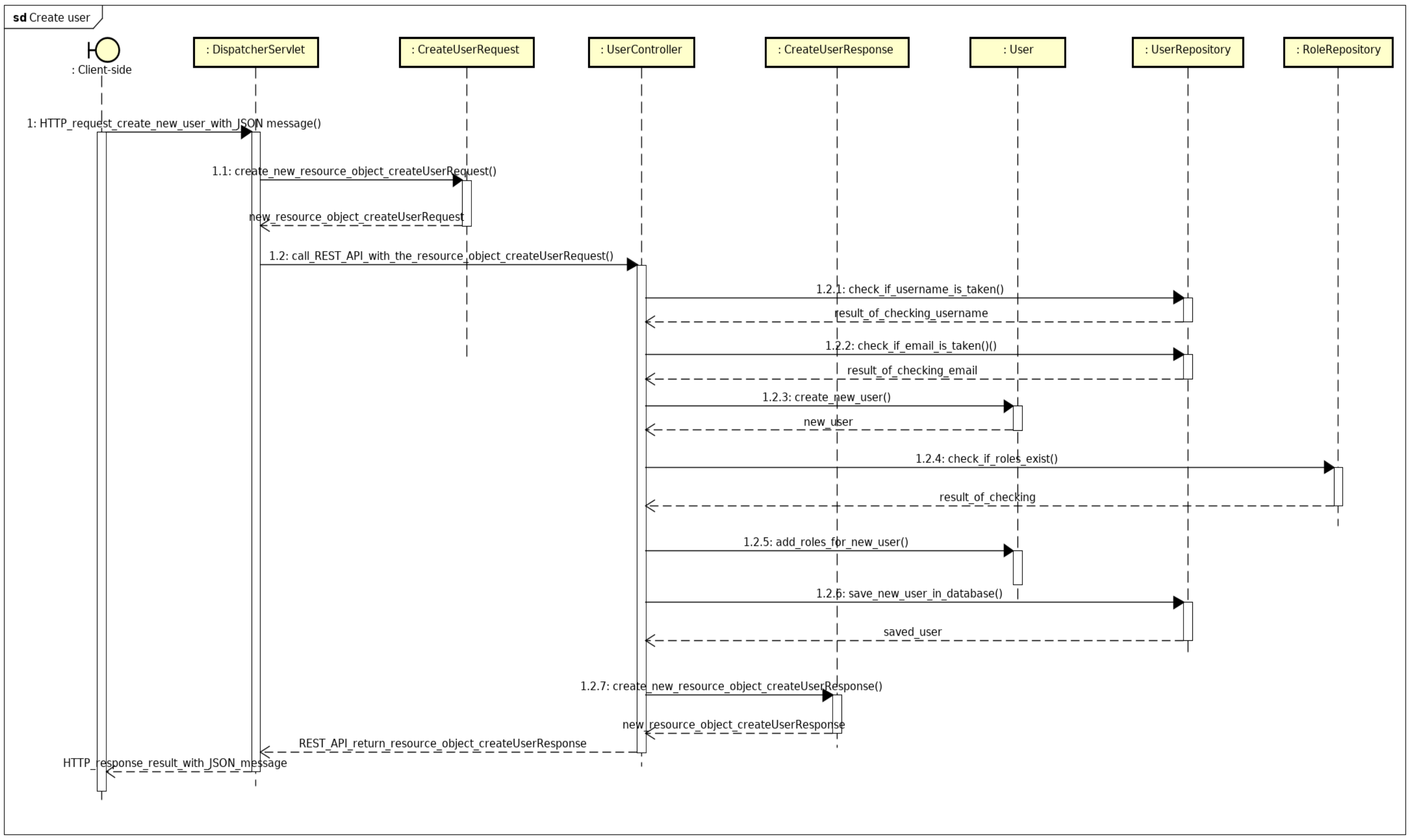
2.3 Package Detail Design



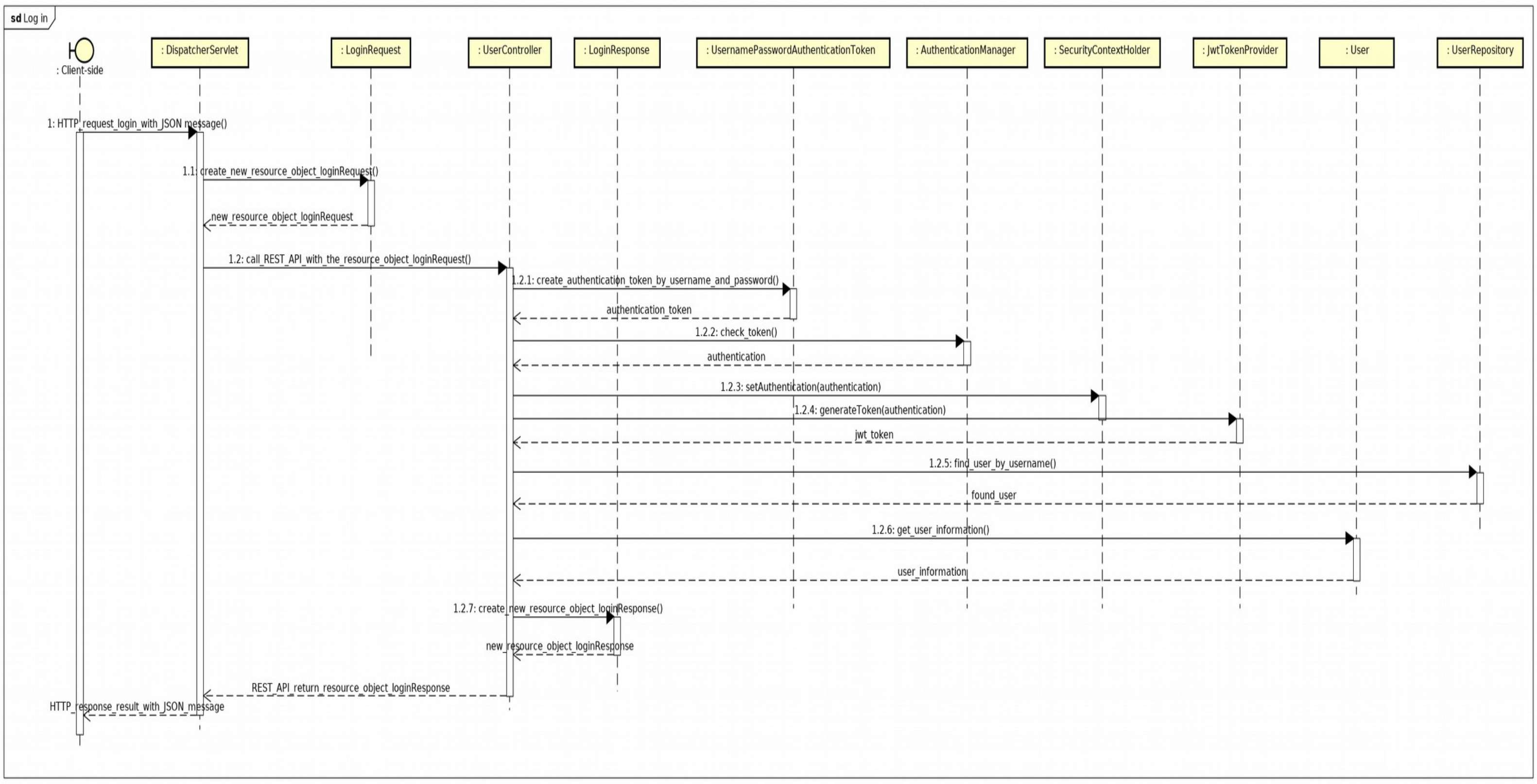
USER MANAGER PACKAGE DESIGN UML

2.4 *Interaction Diagram - SERVER*

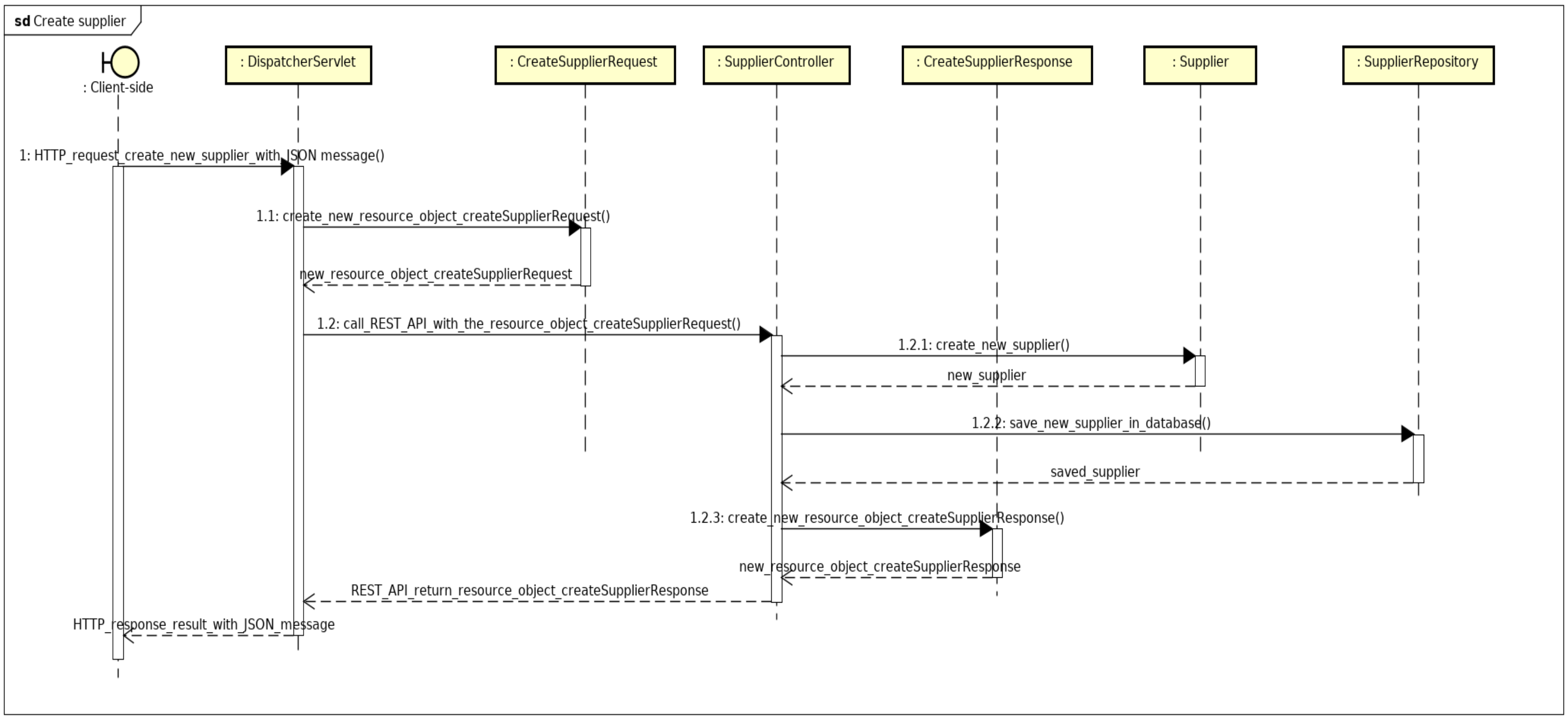
2.4.1 Interaction diagram – Create User – Sever side


sd Create user
: Client-side
: DispatcherServlet
: CreateUserRequest
: UserController
: CreateUserResponse
: User
: UserRepository
: RoleRepository
1: HTTP_request_create_new_user_with_JSON message()
1.1: create_new_resource_object_createUserRequest()
new_resource_object_createUserRequest
1.2: call_REST_API_with_the_resource_object_createUserRequest()
1.2.1: check_if_username_is_taken()
result_of_checking_username
1.2.2: check_if_email_is_taken()()
result_of_checking_email
1.2.3: create_new_user()
new_user
1.2.4: check_if_roles_exist()
result_of_checking
1.2.5: add_roles_for_new_user()
1.2.6: save_new_user_in_database()
saved_user
1.2.7: create_new_resource_object_createUserResponse()
new_resource_object_createUserResponse
REST_API_return_resource_object_createUserResponse
HTTP_response_result_with_JSON_message

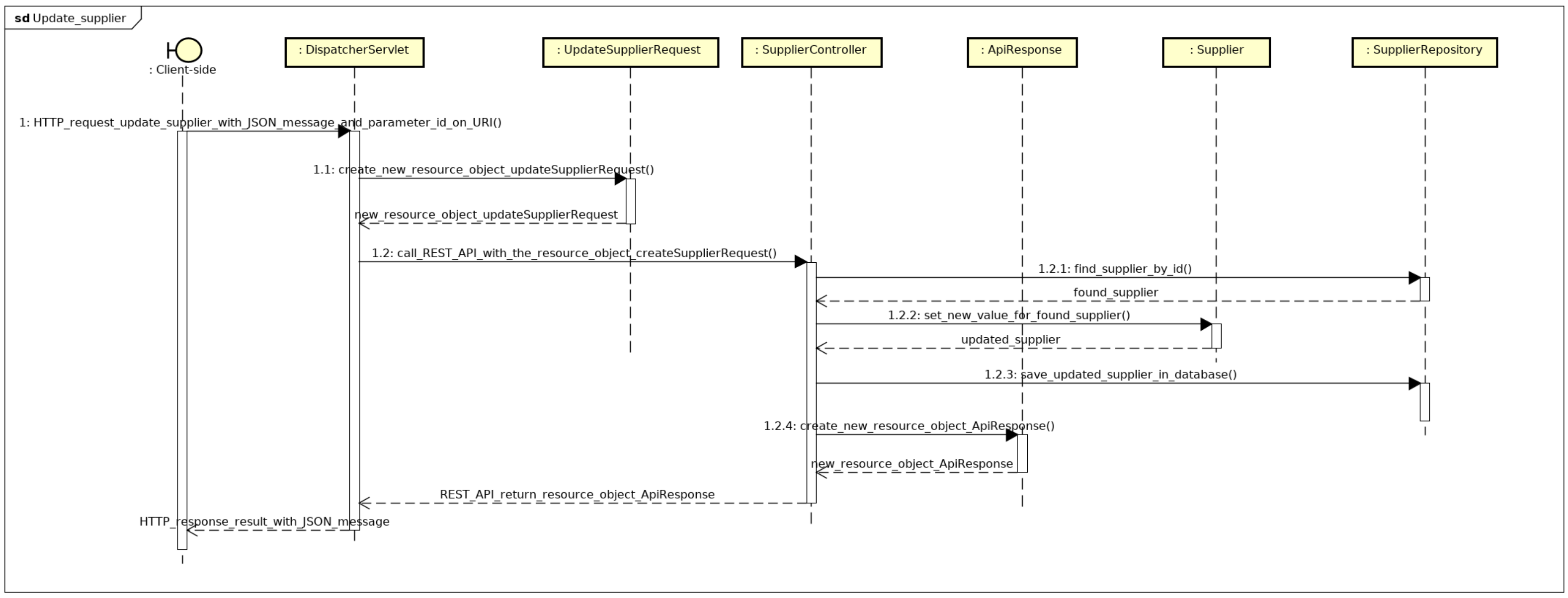
2.4.2 Interaction diagram – Login – Server side


sd Log in
: Client-side
: DispatcherServlet
: LoginRequest
: UserController
: LoginResponse
: UsernamePasswordAuthenticationToken
: AuthenticationManager
: SecurityContextHolder
: JwtTokenProvider
: User
: UserRepository
1: HTTP_request_login_with_JSON message()
1.1: create_new_resource_object_loginRequest()
new_resource_object_loginRequest
1.2: call_REST_API_with_the_resource_object_loginRequest()
1.2.1: create_authentication_token_by_username_and_password()
authentication_token
1.2.2: check_token()
authentication
1.2.3: setAuthentication(authentication)
1.2.4: generateToken(authentication)
jwt_token
1.2.5: find_user_by_username()
found_user
1.2.6: get_user_information()
user_information
1.2.7: create_new_resource_object_loginResponse()
new_resource_object_loginResponse
REST_API_return_resource_object_loginResponse
HTTP_response_result_with_JSON_message

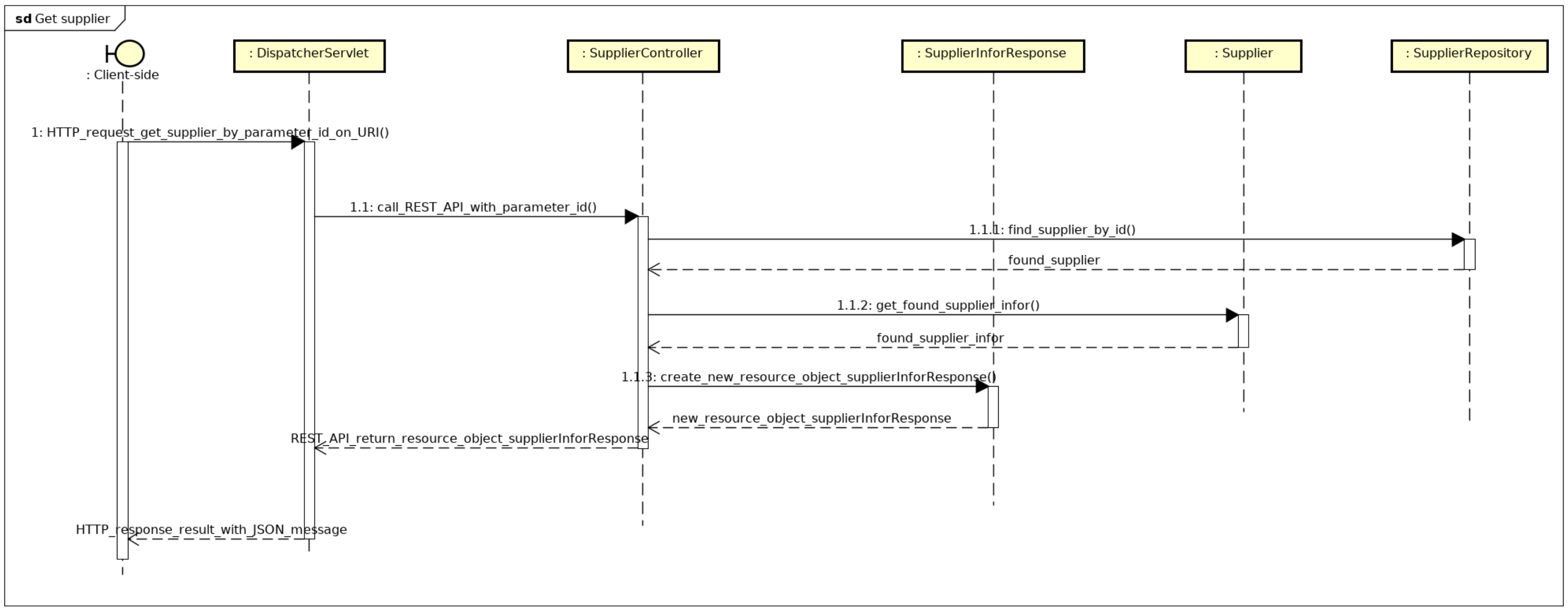
2.4.3 Interaction diagram – Create supplier – Server side


sd Create supplier
: Client-side
: DispatcherServlet
: CreateSupplierRequest
: SupplierController
: CreateSupplierResponse
: Supplier
: SupplierRepository
1: HTTP_request_create_new_supplier_with_JSON message()
1.1: create_new_resource_object_createSupplierRequest()
new_resource_object_createSupplierRequest
1.2: call_REST_API_with_the_resource_object_createSupplierRequest()
1.2.1: create_new_supplier()
new_supplier
1.2.2: save_new_supplier_in_database()
saved_supplier
1.2.3: create_new_resource_object_createSupplierResponse()
new_resource_object_createSupplierResponse
REST_API_return_resource_object_createSupplierResponse
HTTP_response_result_with_JSON_message

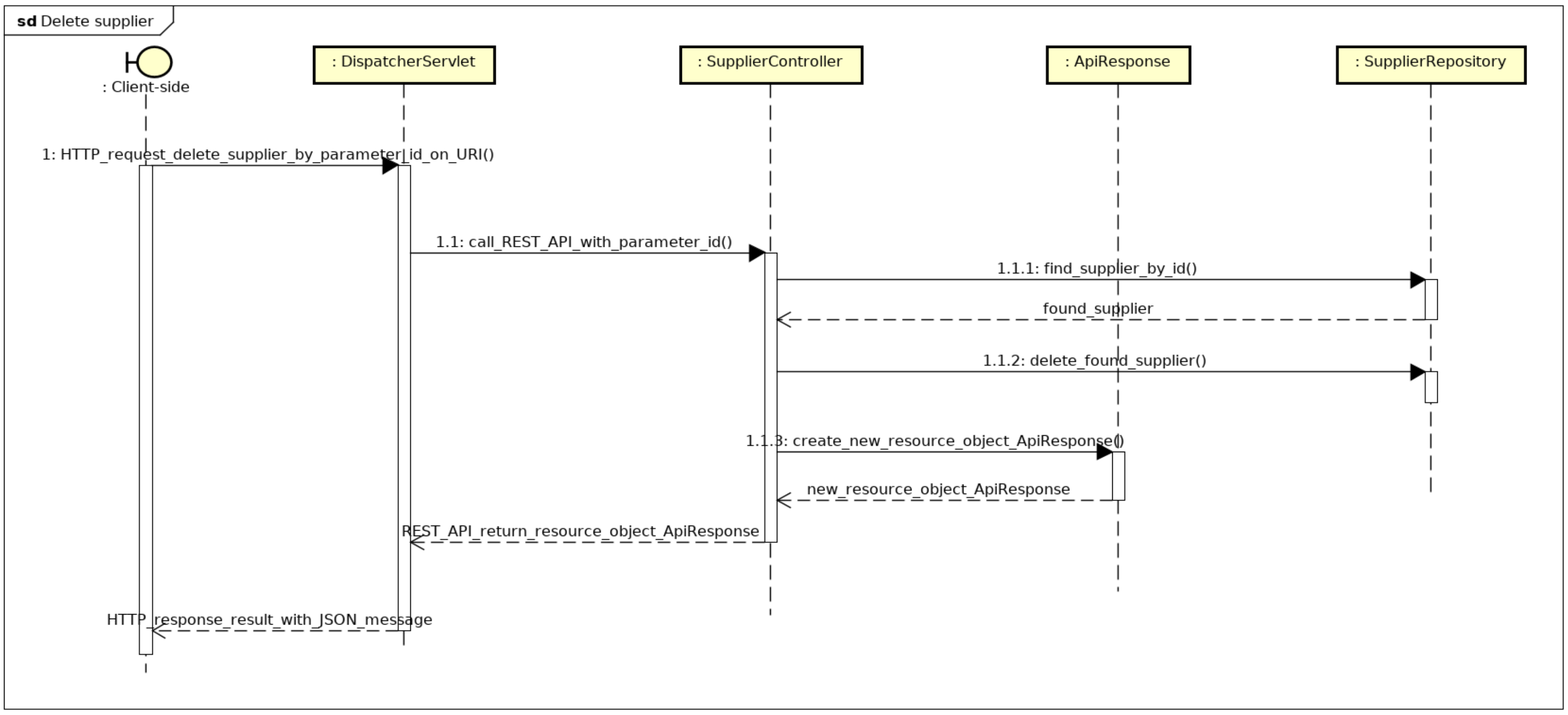
2.4.4 Interaction diagram – Update supplier – Server side


sd Update_supplier
: Client-side
: DispatcherServlet
: UpdateSupplierRequest
: SupplierController
: ApiResponse
: Supplier
: SupplierRepository
1: HTTP_request_update_supplier_with_JSON_message_and_parameter_id_on_URI()
1.1: create_new_resource_object_updateSupplierRequest()
new_resource_object_updateSupplierRequest
1.2: call_REST_API_with_the_resource_object_createSupplierRequest()
1.2.1: find_supplier_by_id()
found_supplier
1.2.2: set_new_value_for_found_supplier()
updated_supplier
1.2.3: save_updated_supplier_in_database()
1.2.4: create_new_resource_object_ApiResponse()
new_resource_object_ApiResponse
REST_API_return_resource_object_ApiResponse
HTTP_response_result_with_JSON_message

2.4.5 Interaction diagram – Get supplier – Server side

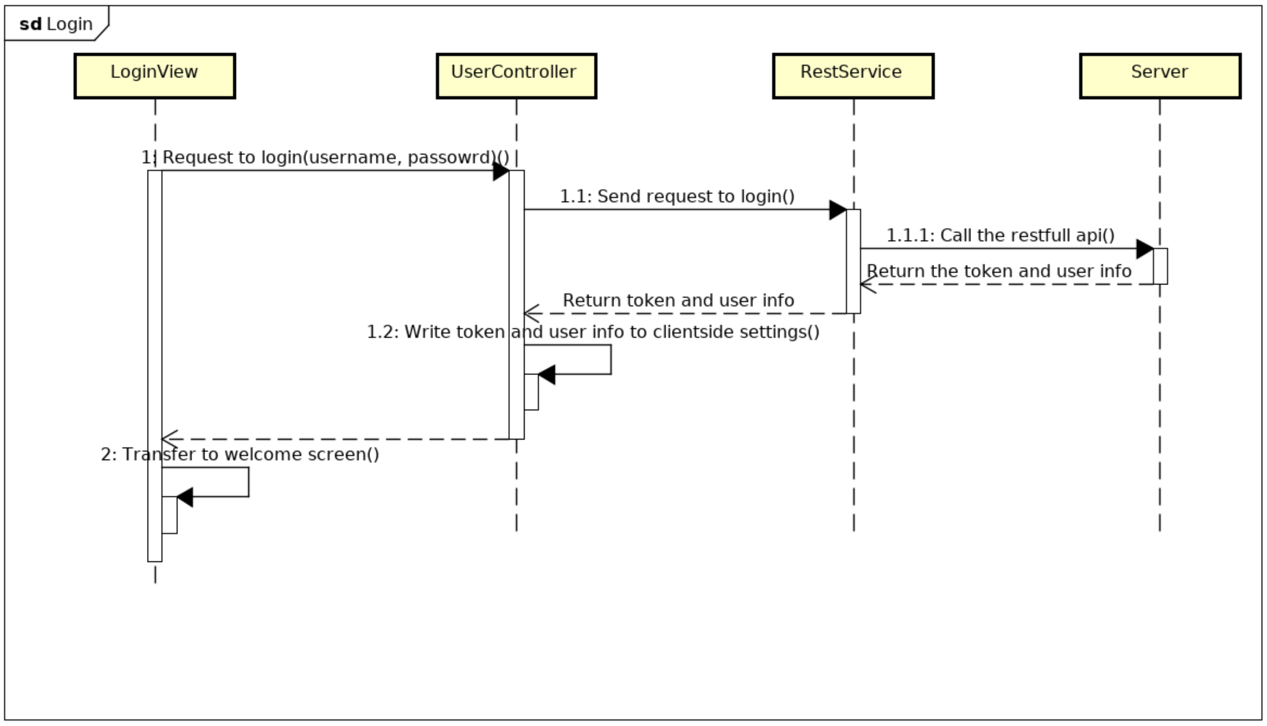

sd Get supplier
: Client-side
: DispatcherServlet
: SupplierController
: SupplierInforResponse
: Supplier
: SupplierRepository
1: HTTP_request_get_supplier_by_parameter_id_on_URI()
1.1: call_REST_API_with_parameter_id()
1.1.1: find_supplier_by_id()
found_supplier
1.1.2: get_found_supplier_infor()
found_supplier_infor
1.1.3: create_new_resource_object_supplierInforResponse()
new_resource_object_supplierInforResponse
REST_API_return_resource_object_supplierInforResponse
HTTP_response_result_with_JSON_message

2.4.6 Interaction diagram – Delete supplier – Server side


sd Delete supplier
: Client-side
: DispatcherServlet
: SupplierController
: ApiResponse
: SupplierRepository
1: HTTP_request_delete_supplier_by_parameter_id_on_URI()
1.1: call_REST_API_with_parameter_id()
1.1.1: find_supplier_by_id()
found_supplier
1.1.2: delete_found_supplier()
1.1.3: create_new_resource_object_ApiResponse()
new_resource_object_ApiResponse
REST_API_return_resource_object_ApiResponse
HTTP_response_result_with_JSON_message

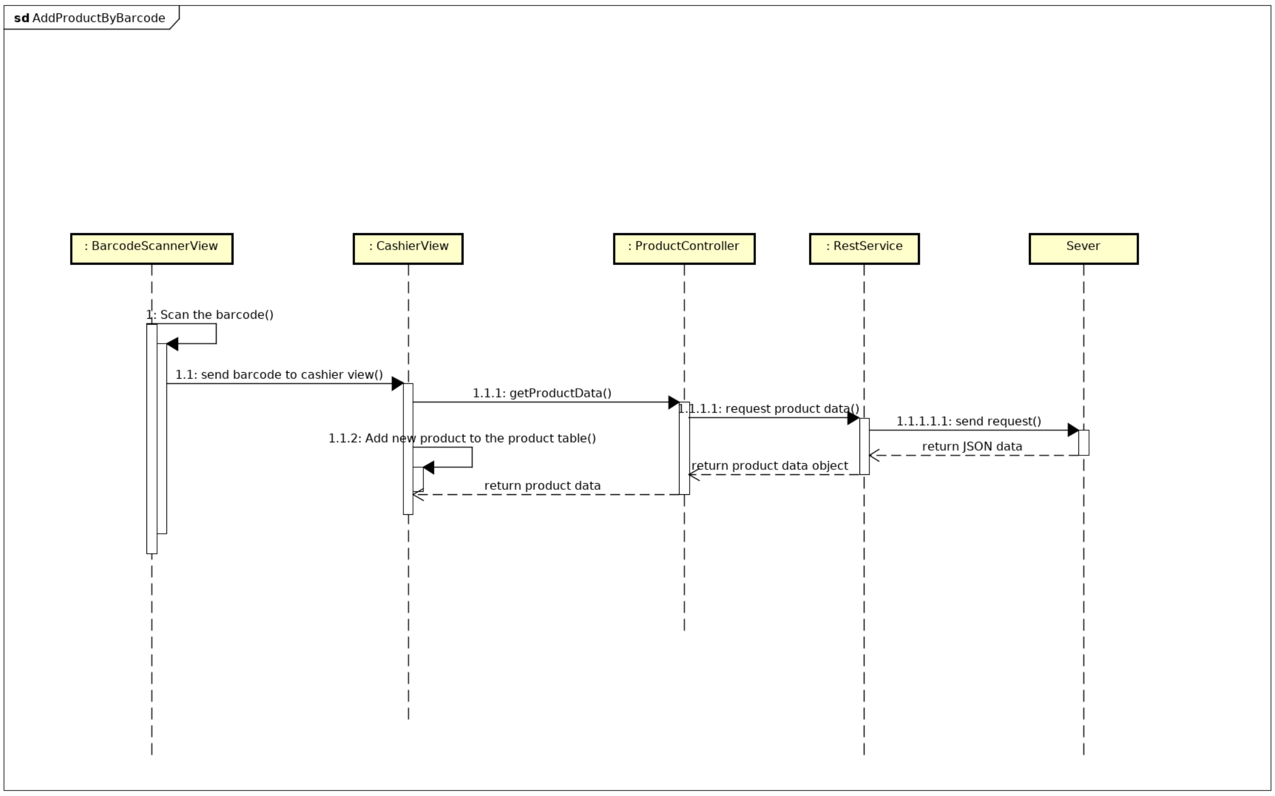
2.5 *Interaction Diagram – DESKTOP APPLICATION*

***** LOGIN**

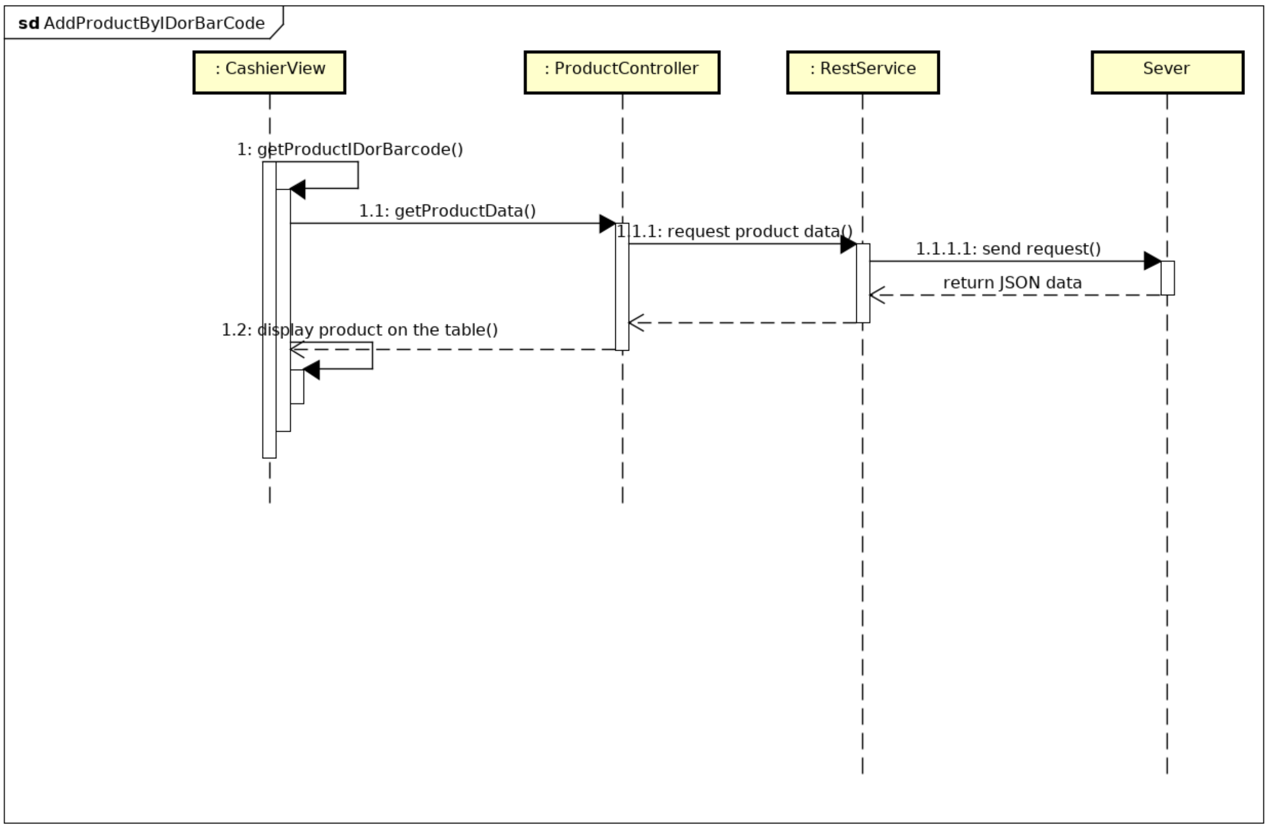

sd Login
LoginView
UserController
RestService
Server
1: Request to login(username, passowrd)()
1.1: Send request to login()
1.1.1: Call the restfull api()
Return the token and user info
Return token and user info
1.2: Write token and user info to clientside settings()
2: Transfer to welcome screen()

***** CASHIER SCREEN**

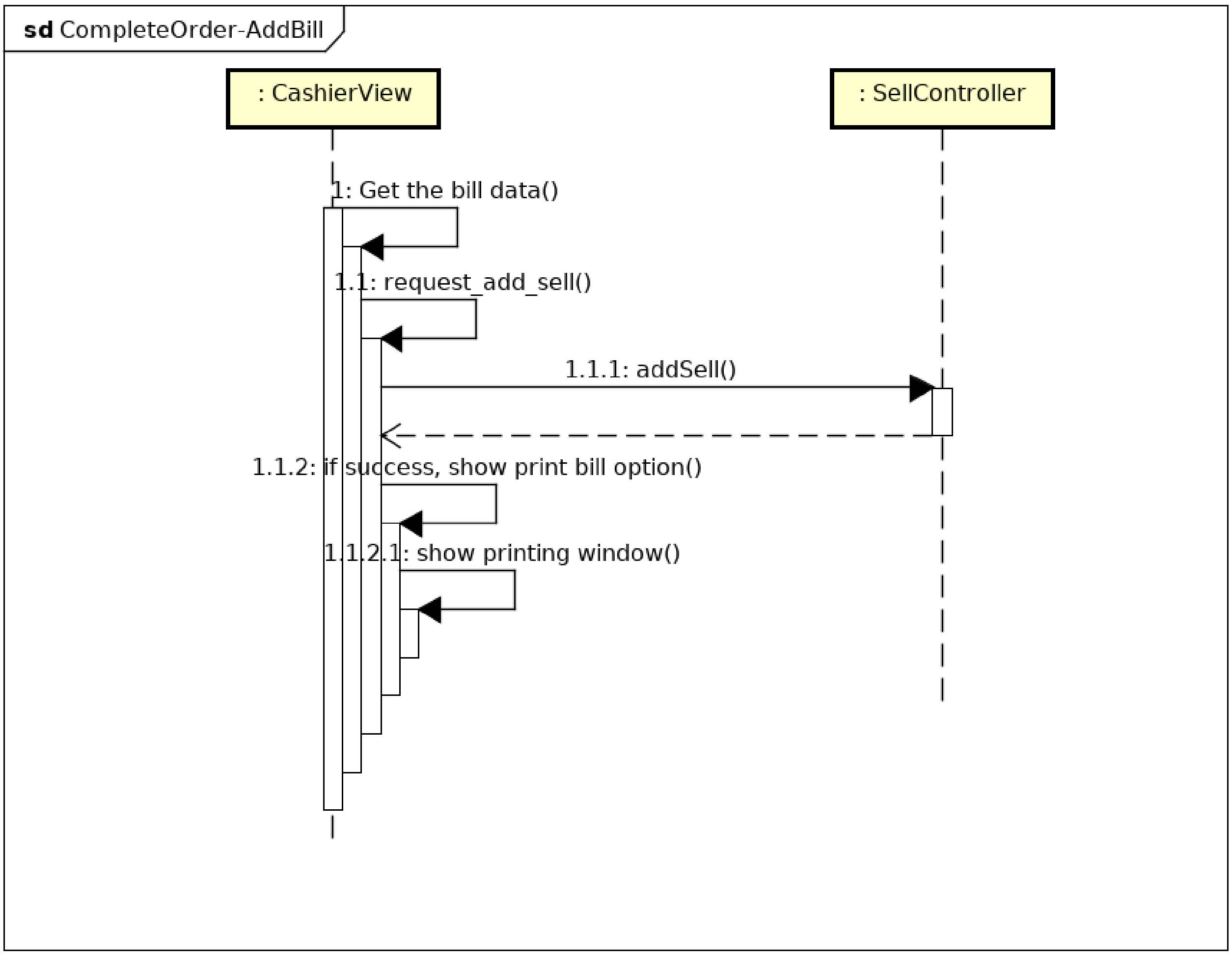
- Add product by barcode – from scanner



- Add product by barcode / id

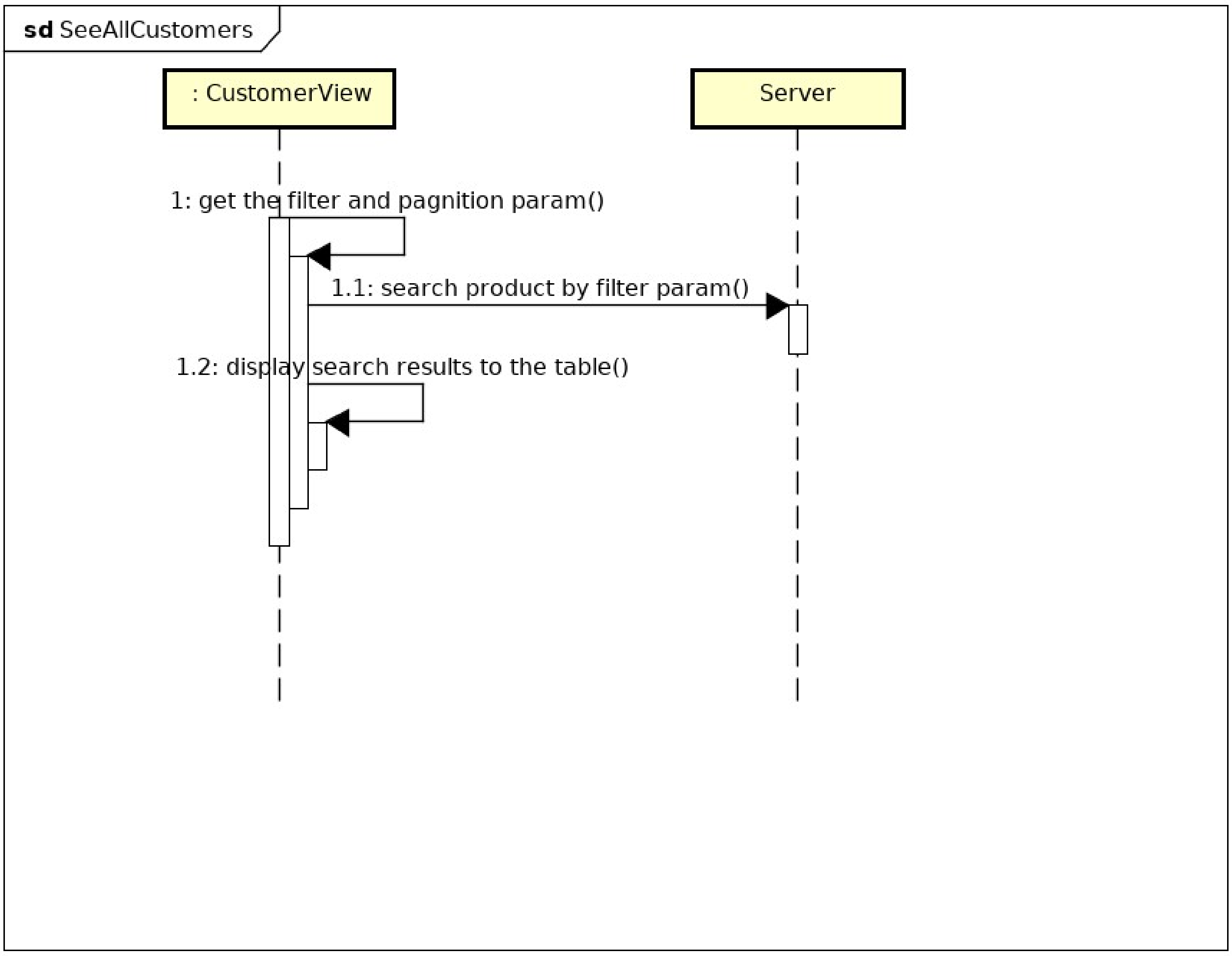

sd AddProductByIDorBarCode
: CashierView
: ProductController
: RestService
Sever
1: getProductIDorBarcode()
1.1: getProductData()
1.1.1: request product data()
1.1.1.1: send request()
return JSON data
1.2: display product on the table()

- Complete order


sd CompleteOrder-AddBill
: CashierView
: SellController
1: Get the bill data()
1.1: request_add_sell()
1.1.1: addSell()
1.1.2: if success, show print bill option()
1.1.2.1: show printing window()

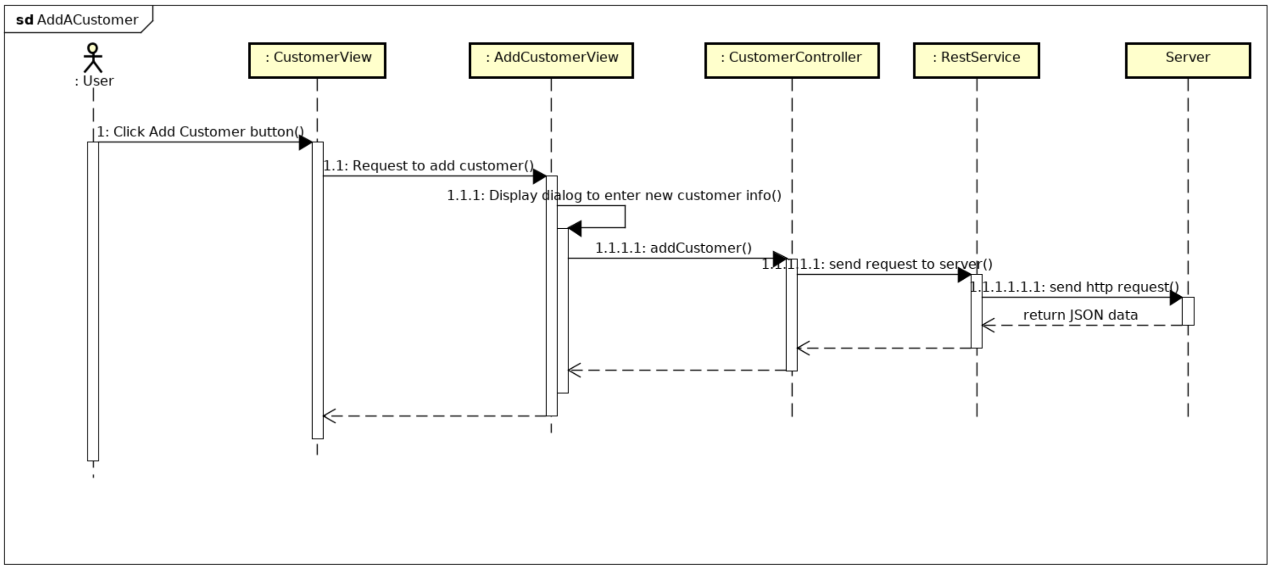
***** CUSTOMER MANAGEMENT**

- See all the customers



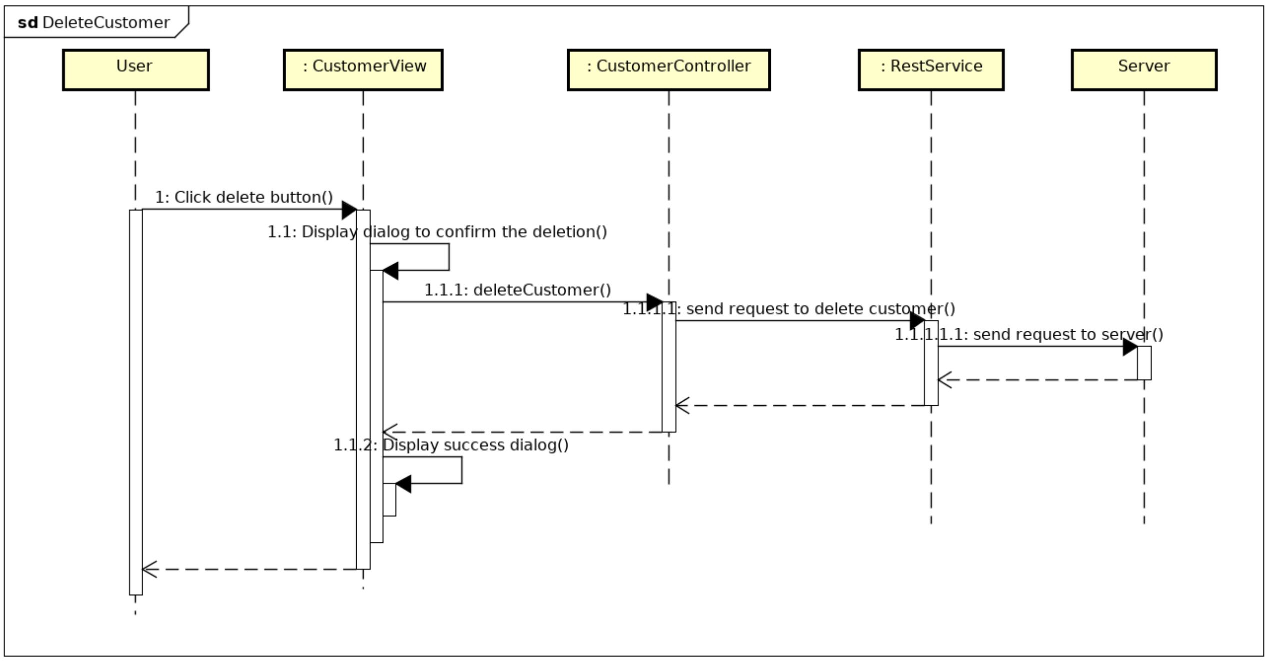
sd SeeAllCustomers
: CustomerView
Server
1: get the filter and pagnition param()
1.1: search product by filter param()
1.2: display search results to the table()

- Add customer

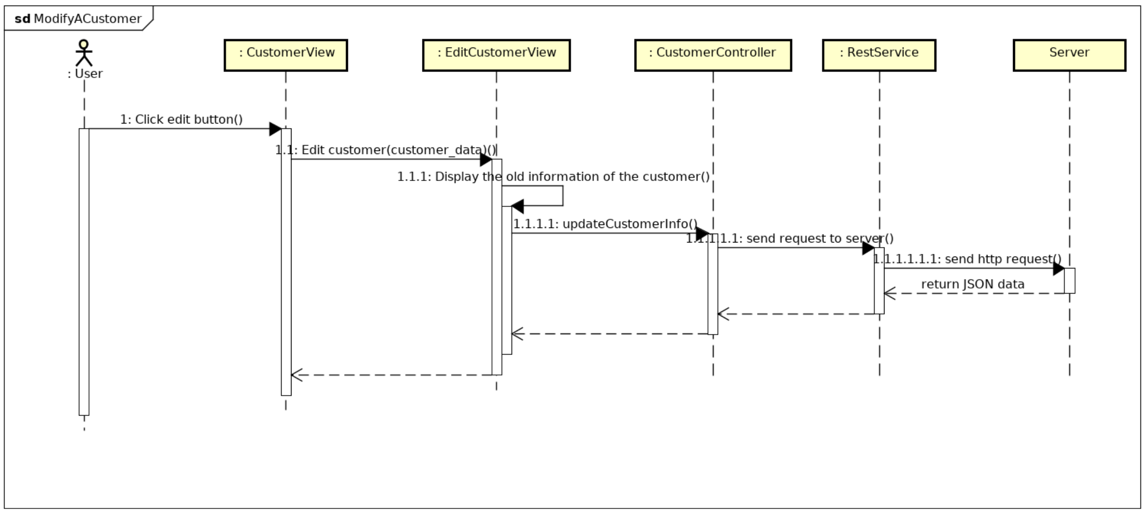


sd AddACustomer
: User
: CustomerView
: AddCustomerView
: CustomerController
: RestService
Server
1: Click Add Customer button()
1.1: Request to add customer()
1.1.1: Display dialog to enter new customer info()
1.1.1.1: addCustomer()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

- Delete Customer

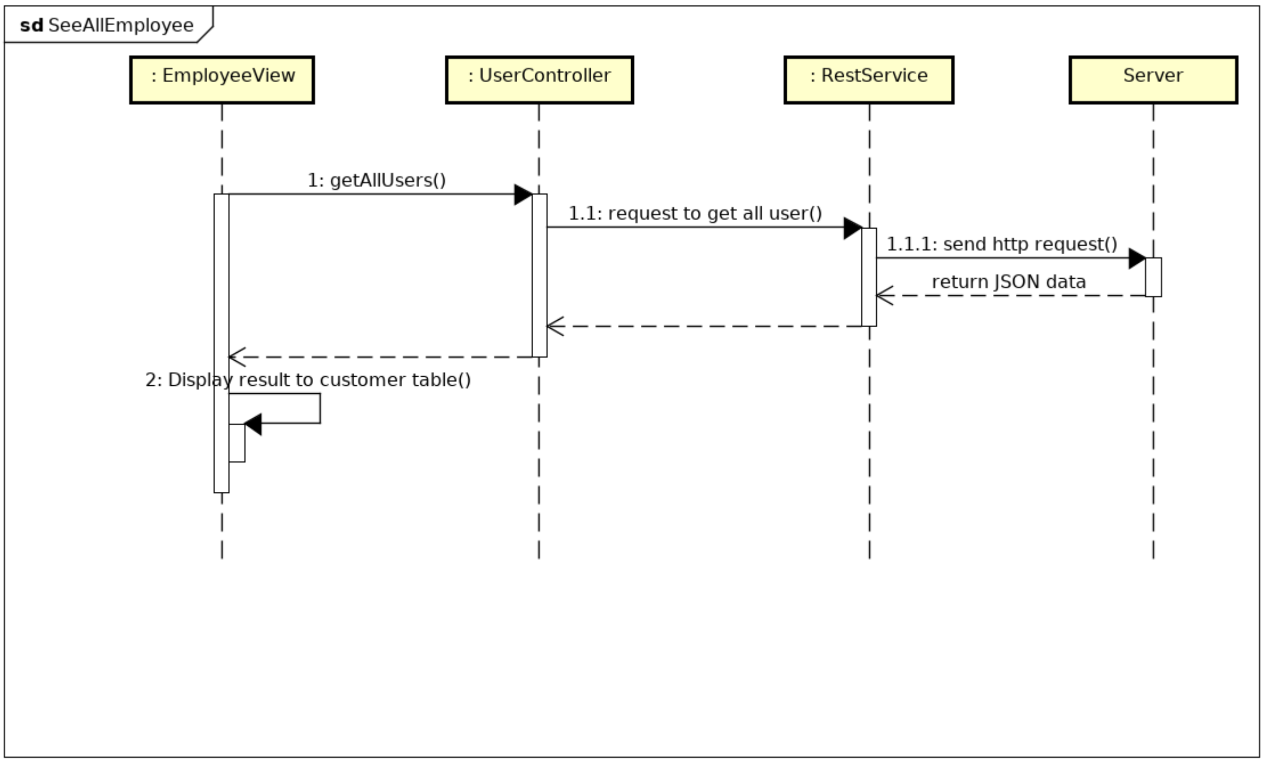

sd DeleteCustomer
User
: CustomerView
: CustomerController
: RestService
Server
1: Click delete button()
1.1: Display dialog to confirm the deletion()
1.1.1: deleteCustomer()
1.1.1.1: send request to delete customer()
1.1.1.1.1: send request to server()
1.1.2: Display success dialog()

- Modify customer

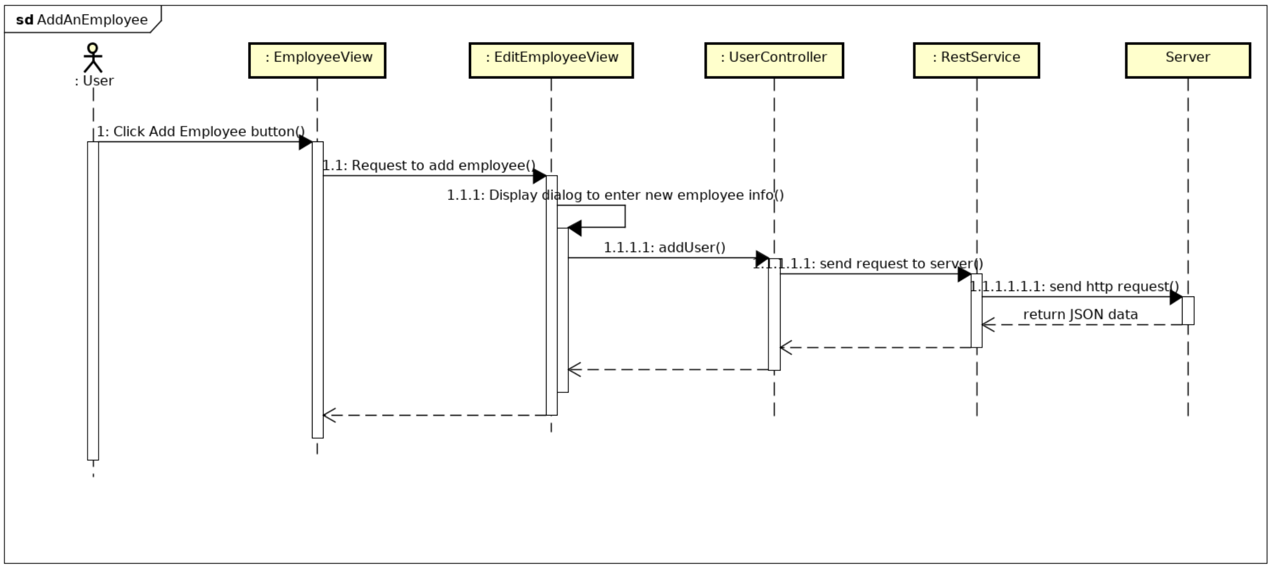

sd ModifyACustomer
: User
: CustomerView
: EditCustomerView
: CustomerController
: RestService
Server
1: Click edit button()
1.1: Edit customer(customer_data)()
1.1.1: Display the old information of the customer()
1.1.1.1: updateCustomerInfo()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

*** EMPLOYEE MANAGEMENT

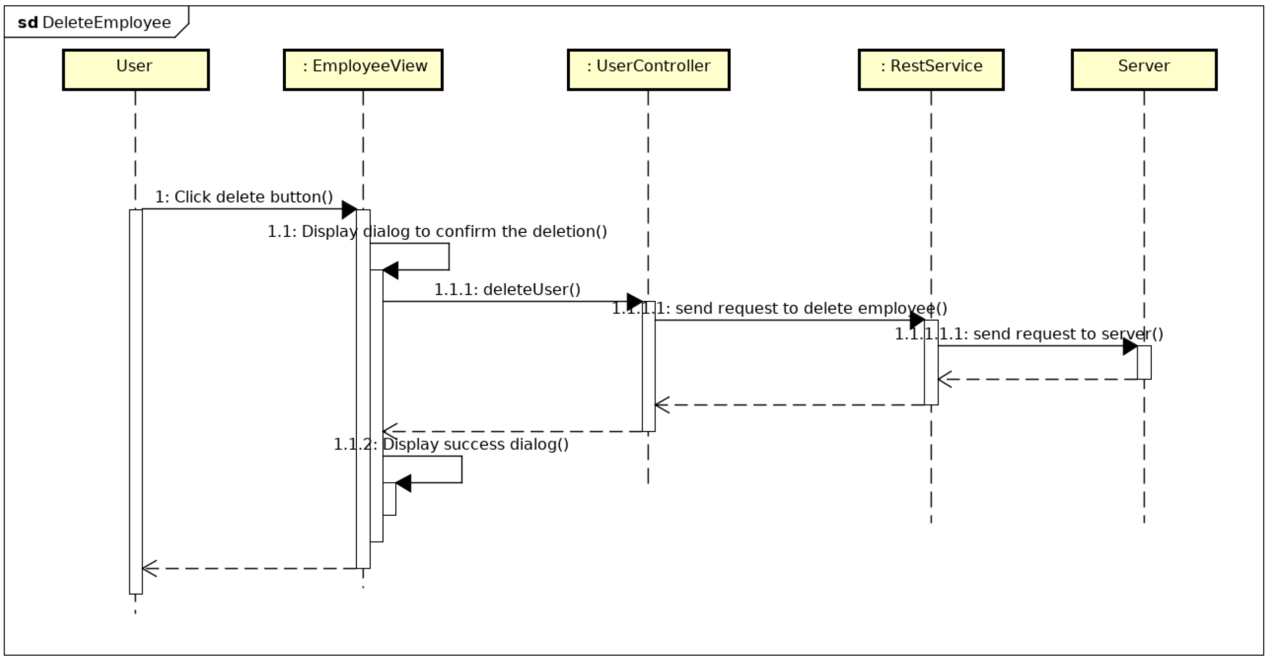
- See all employees



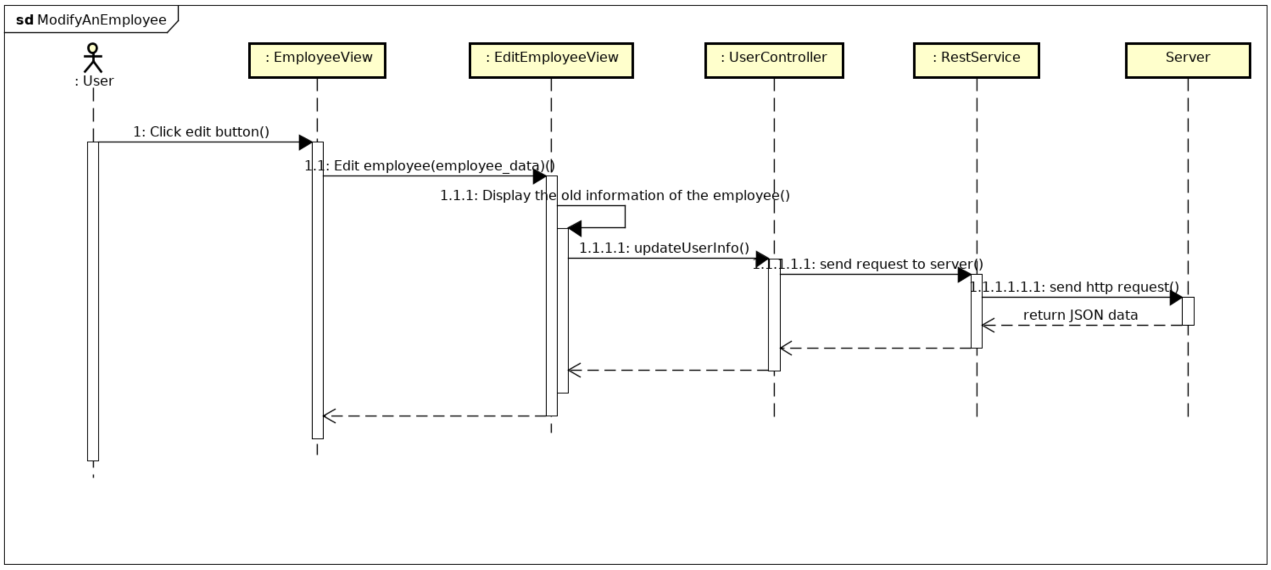
- Add Employee


sd AddAnEmployee
: User
: EmployeeView
: EditEmployeeView
: UserController
: RestService
Server
1: Click Add Employee button()
1.1: Request to add employee()
1.1.1: Display dialog to enter new employee info()
1.1.1.1: addUser()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

- Delete Employee

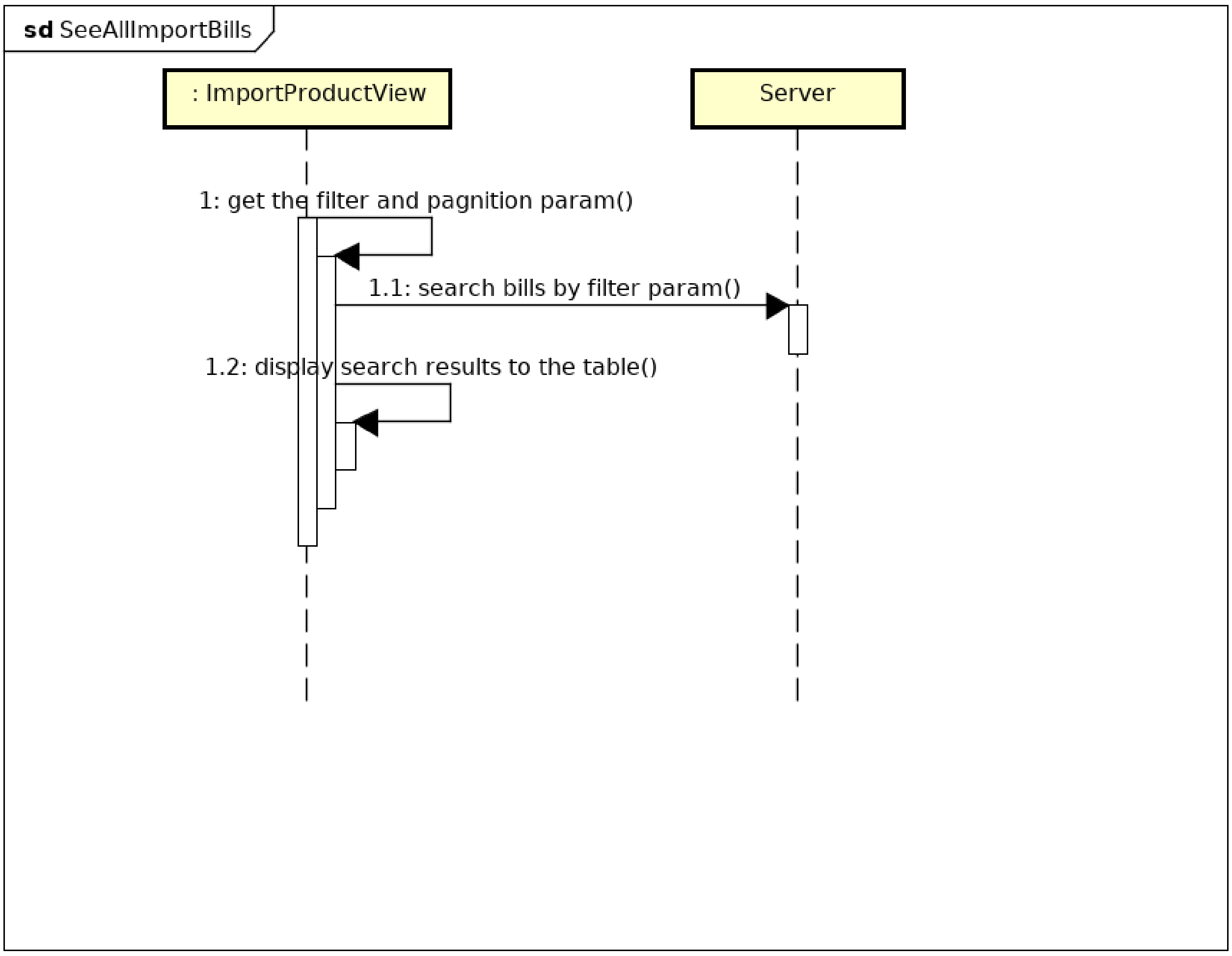

sd DeleteEmployee
User
: EmployeeView
: UserController
: RestService
Server
1: Click delete button()
1.1: Display dialog to confirm the deletion()
1.1.1: deleteUser()
1.1.1.1: send request to delete employee()
1.1.1.1.1: send request to server()
1.1.2: Display success dialog()

- Edit employee

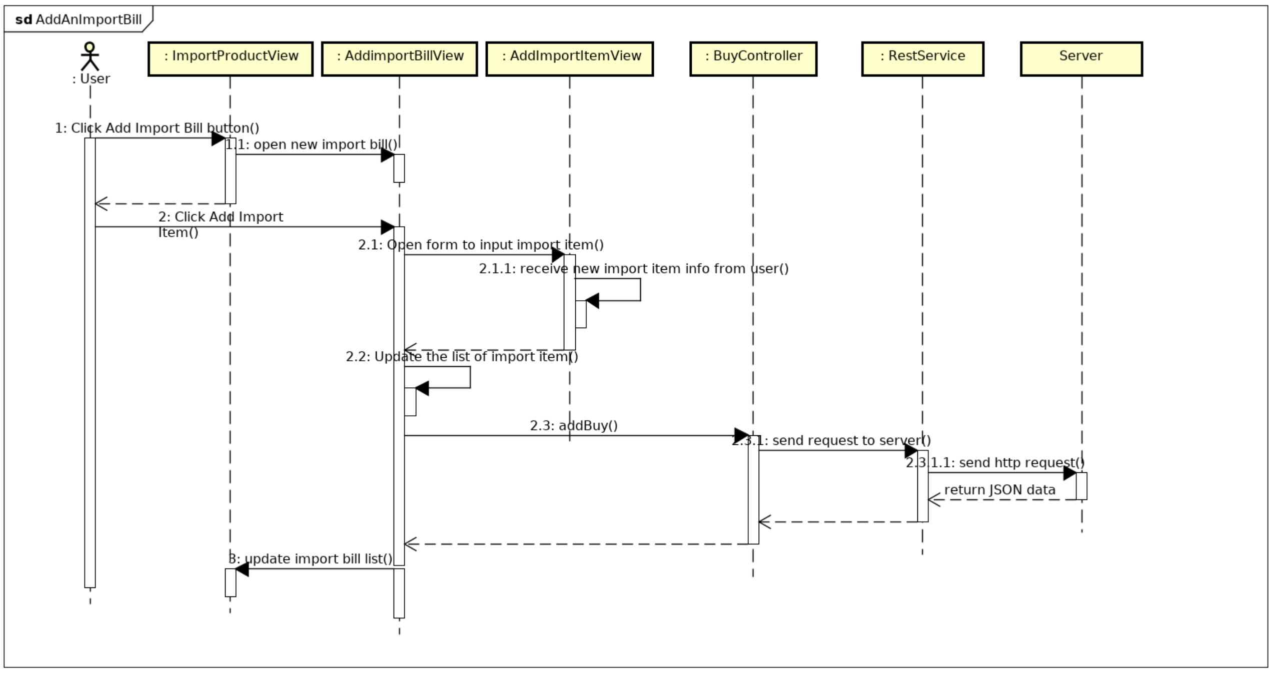

sd ModifyAnEmployee
: User
: EmployeeView
: EditEmployeeView
: UserController
: RestService
Server
1: Click edit button()
1.1: Edit employee(employee_data)()
1.1.1: Display the old information of the employee()
1.1.1.1: updateUserInfo()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

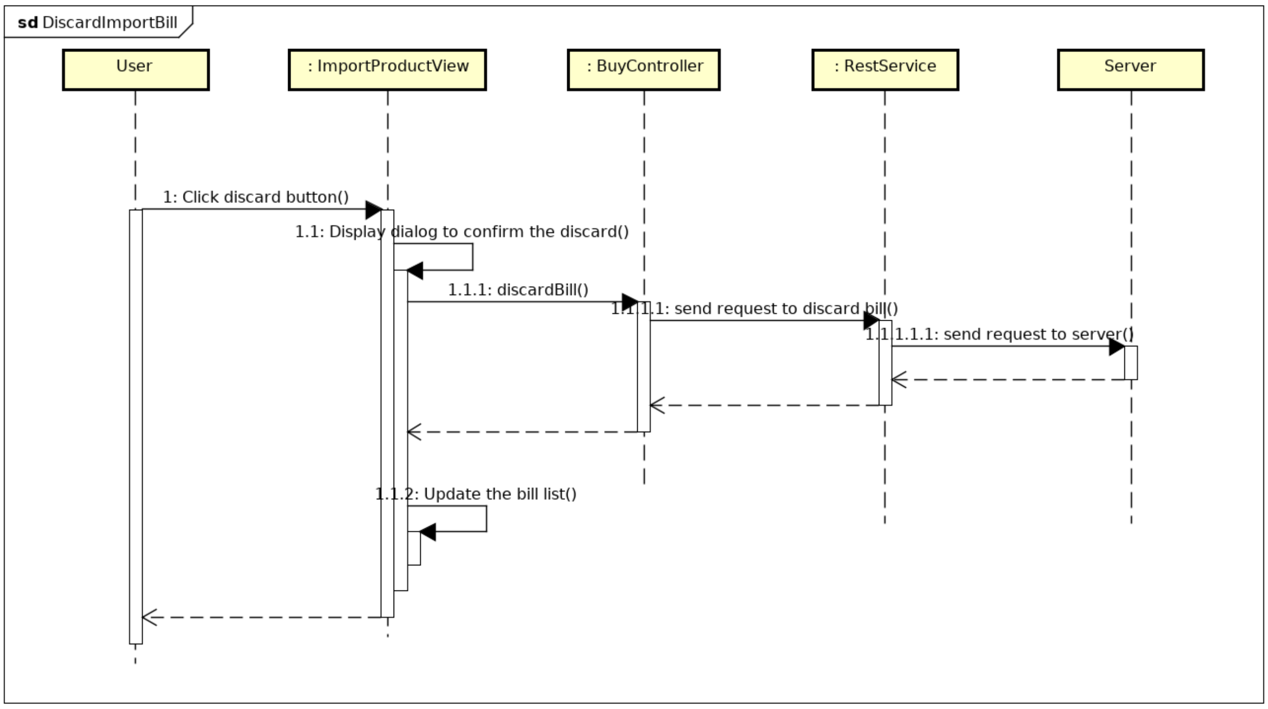
*** IMPORT PRODUCTS AND MANAGE IMPORT BILLS

- See all import bills


sd SeeAllImportBills
: ImportProductView
Server
1: get the filter and pagnition param()
1.1: search bills by filter param()
1.2: display search results to the table()

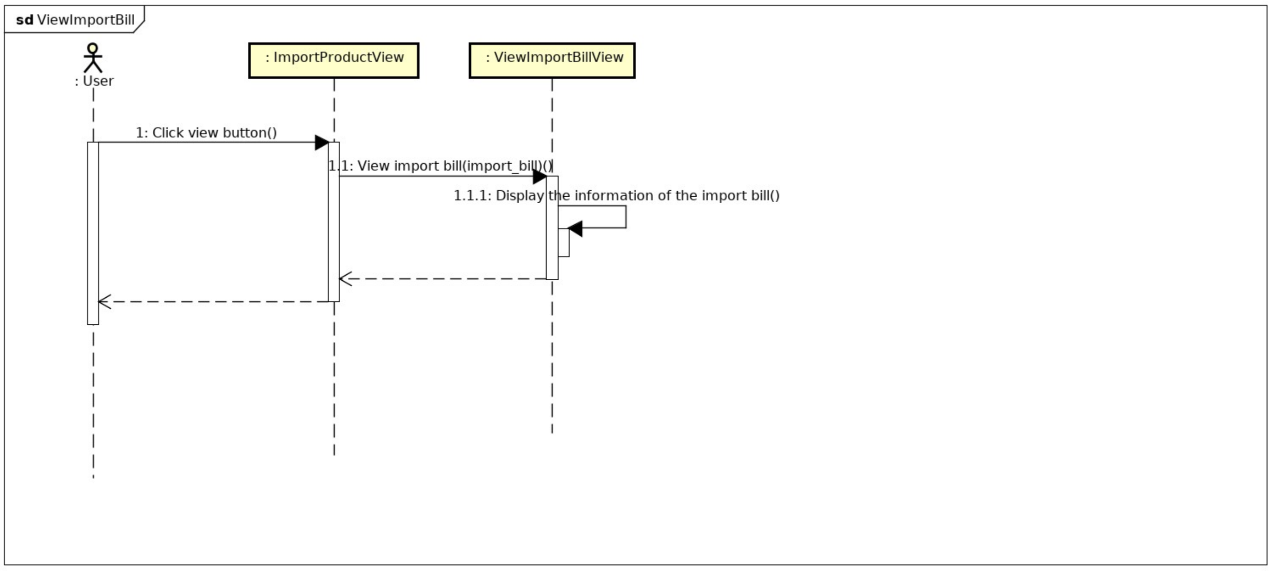
- **Add An Import Bill**


sd AddAnImportBill
: User
: ImportProductView
: AddimportBillView
: AddImportItemView
: BuyController
: RestService
Server
1: Click Add Import Bill button()
1.1: open new import bill()
2: Click Add Import Item()
2.1: Open form to input import item()
2.1.1: receive new import item info from user()
2.2: Update the list of import item()
2.3: addBuy()
2.3.1: send request to server()
2.3.1.1: send http request()
return JSON data
3: update import bill list()


sd DiscardImportBill
User
: ImportProductView
: BuyController
: RestService
Server
1: Click discard button()
1.1: Display dialog to confirm the discard()
1.1.1: discardBill()
1.1.1.1: send request to discard bill()
1.1.1.1.1: send request to server()
1.1.2: Update the bill list()

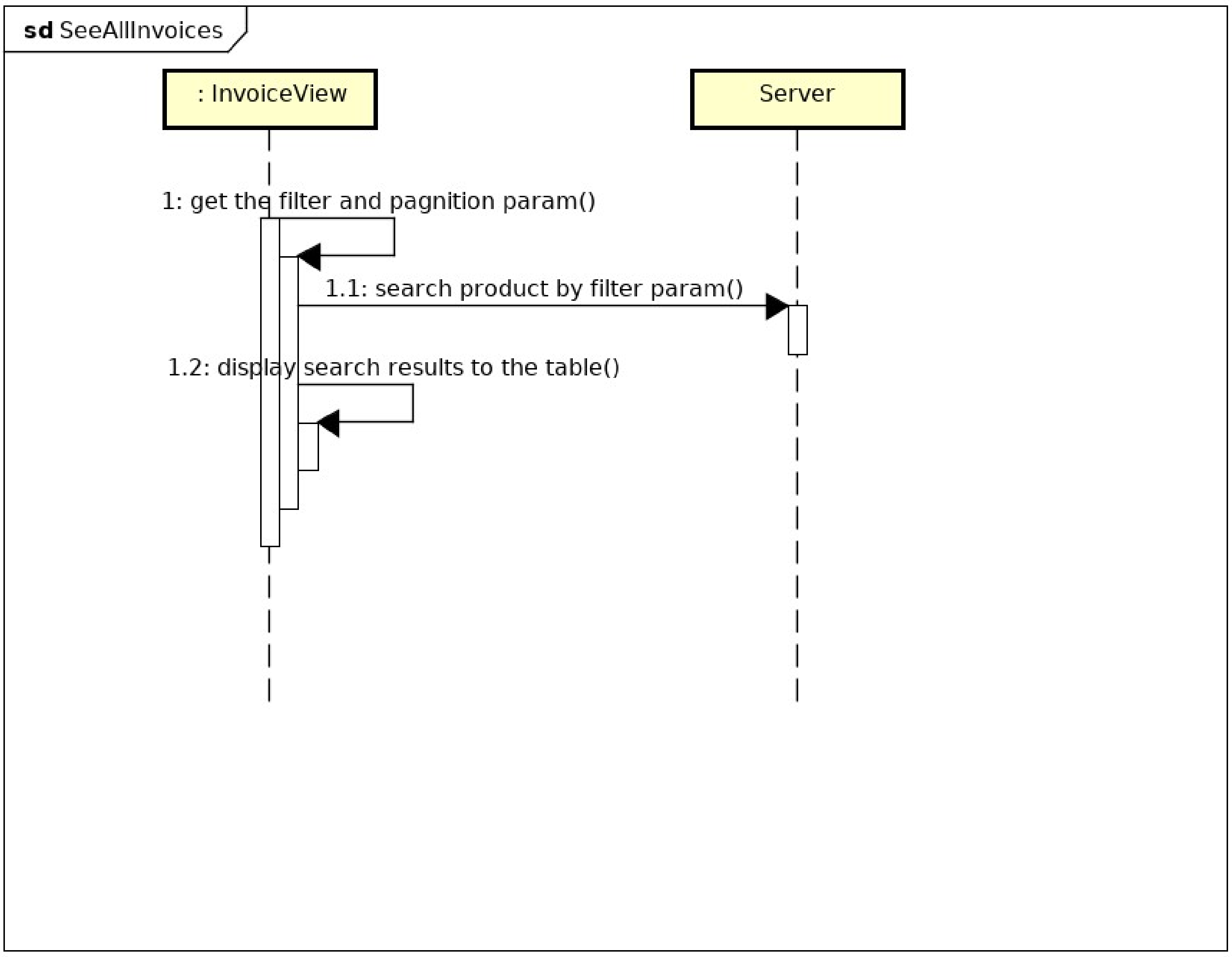
- Discard Import Bill

- View Import Bill Detail

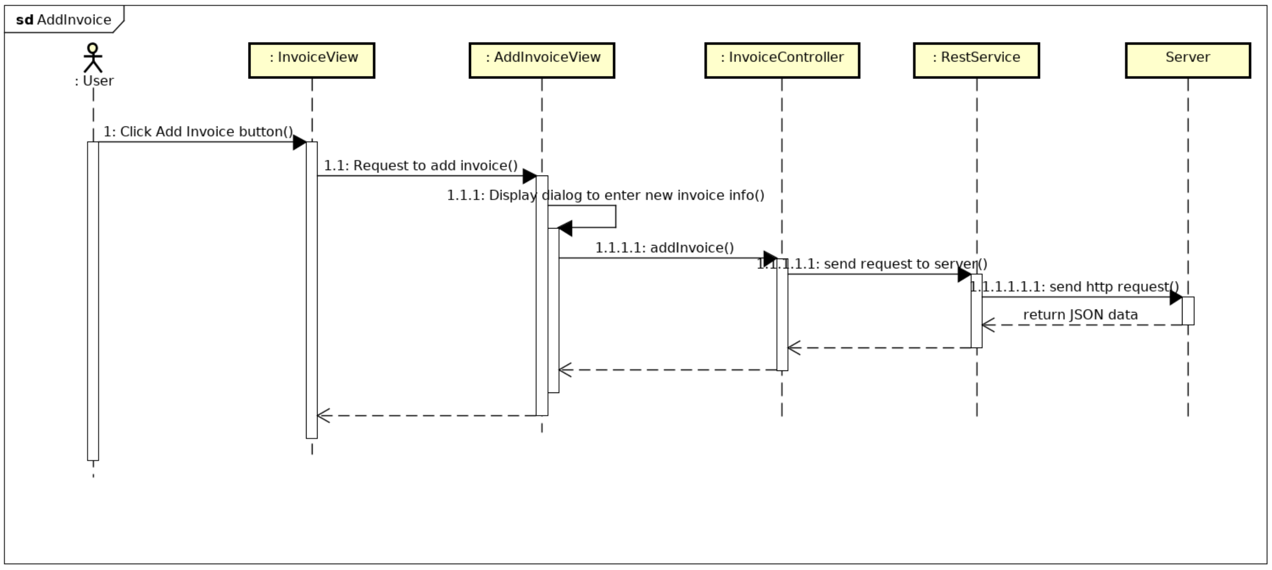

sd ViewImportBill
: User
: ImportProductView
: ViewImportBillView
1: Click view button()
1.1: View import bill(import_bill)()
1.1.1: Display the information of the import bill()

***** INVOICE MANAGEMENT – FINANCE**

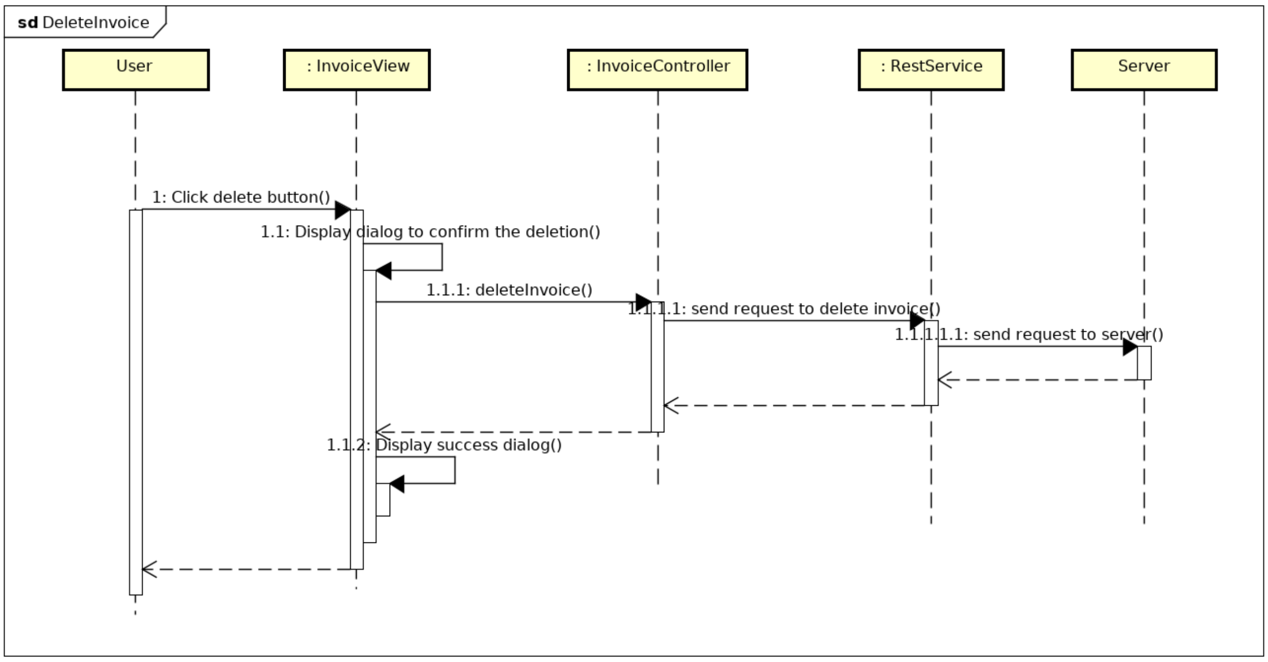
- See all invoices


sd SeeAllInvoices
: InvoiceView
Server
1: get the filter and pagnition param()
1.1: search product by filter param()
1.2: display search results to the table()

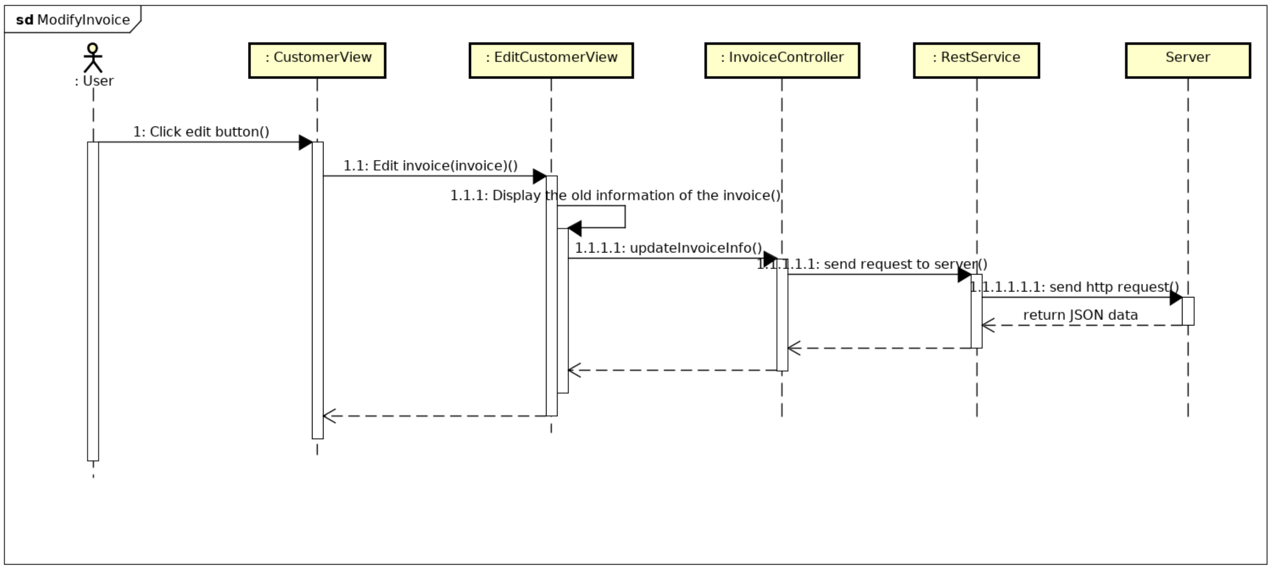
- Add invoice


sd AddInvoice
: User
: InvoiceView
: AddInvoiceView
: InvoiceController
: RestService
Server
1: Click Add Invoice button()
1.1: Request to add invoice()
1.1.1: Display dialog to enter new invoice info()
1.1.1.1: addInvoice()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

- Delete invoice

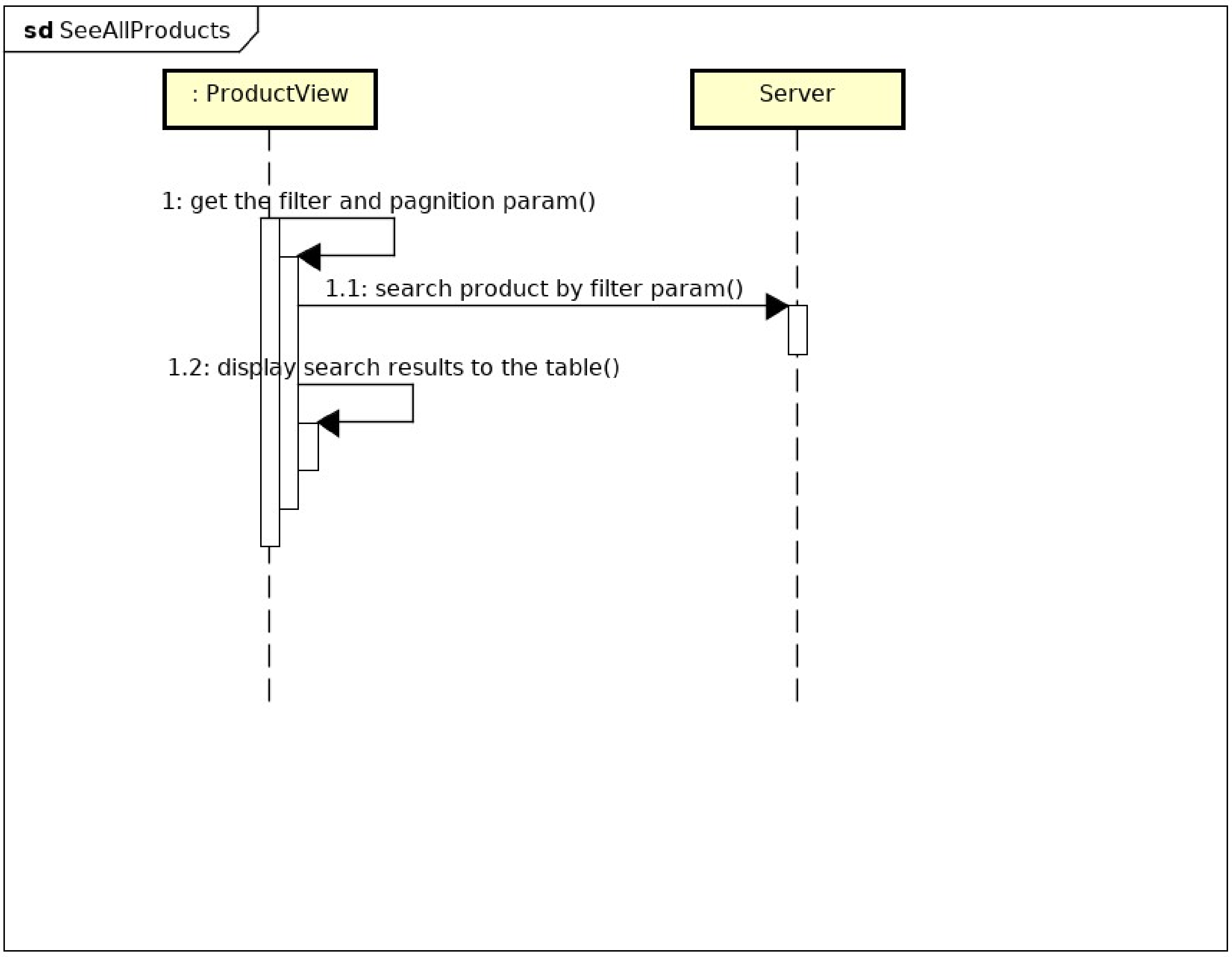

sd DeleteInvoice
User
: InvoiceView
: InvoiceController
: RestService
Server
1: Click delete button()
1.1: Display dialog to confirm the deletion()
1.1.1: deleteInvoice()
1.1.1.1: send request to delete invoice()
1.1.1.1.1: send request to server()
1.1.2: Display success dialog()

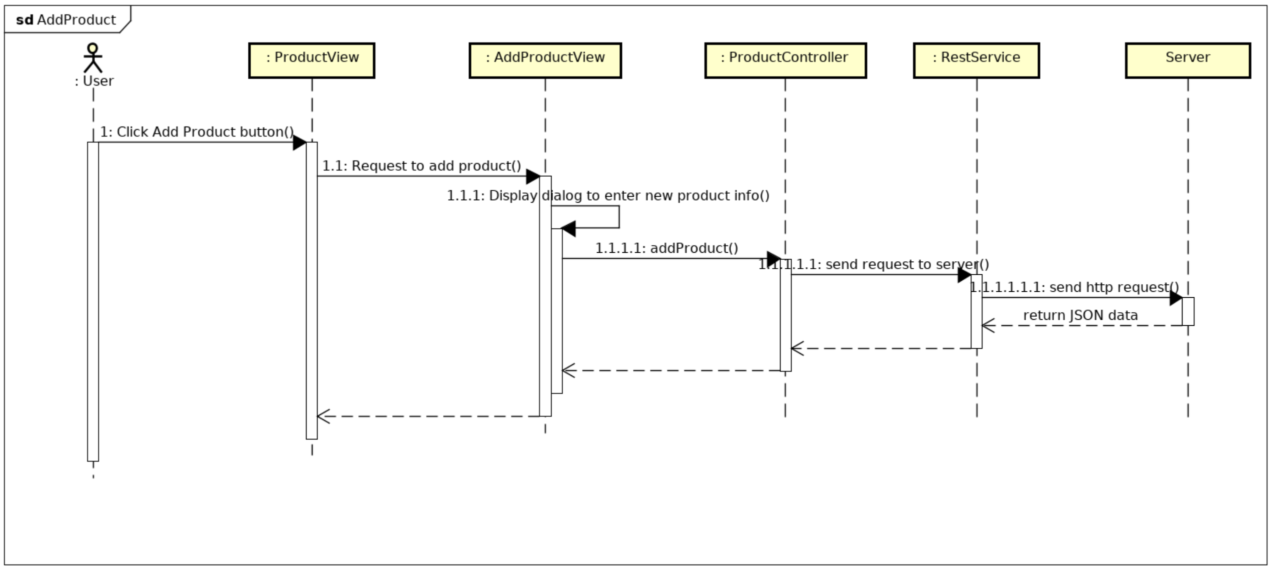
- Modify invoice


sd ModifyInvoice
: User
: CustomerView
: EditCustomerView
: InvoiceController
: RestService
Server
1: Click edit button()
1.1: Edit invoice(invoice)()
1.1.1: Display the old information of the invoice()
1.1.1.1: updateInvoiceInfo()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

*** PRODUCT MANAGEMENT

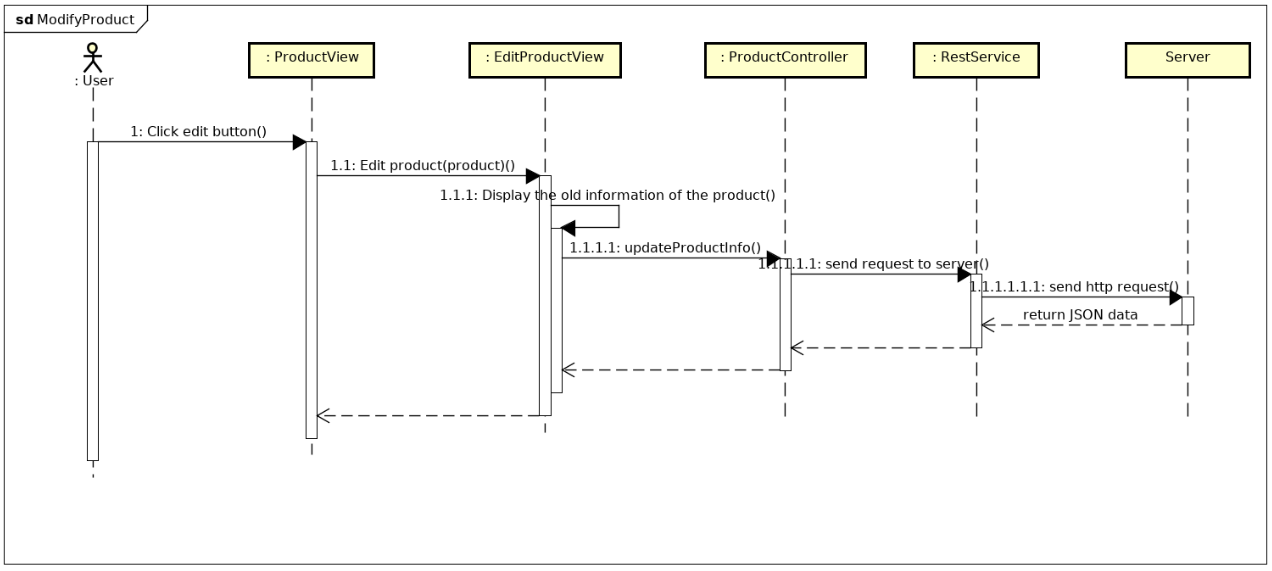
- See all products

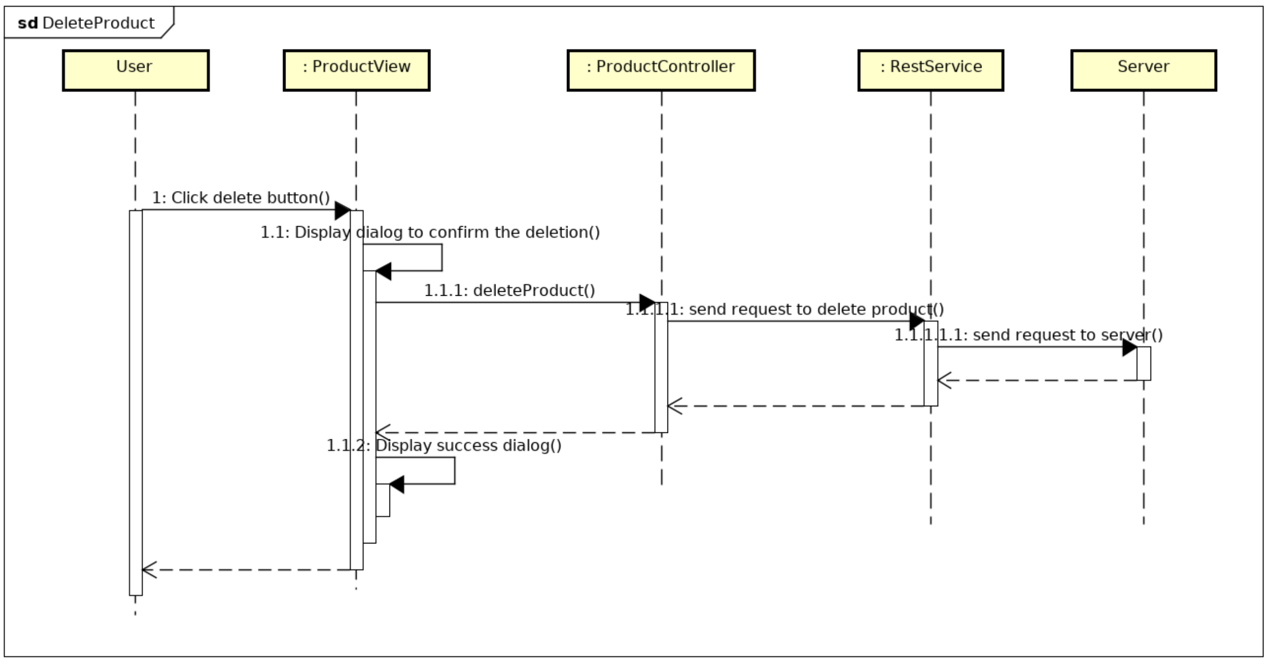

sd SeeAllProducts
: ProductView
Server
1: get the filter and pagnition param()
1.1: search product by filter param()
1.2: display search results to the table()


sd AddProduct
: User
: ProductView
: AddProductView
: ProductController
: RestService
Server
1: Click Add Product button()
1.1: Request to add product()
1.1.1: Display dialog to enter new product info()
1.1.1.1: addProduct()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

- Add product

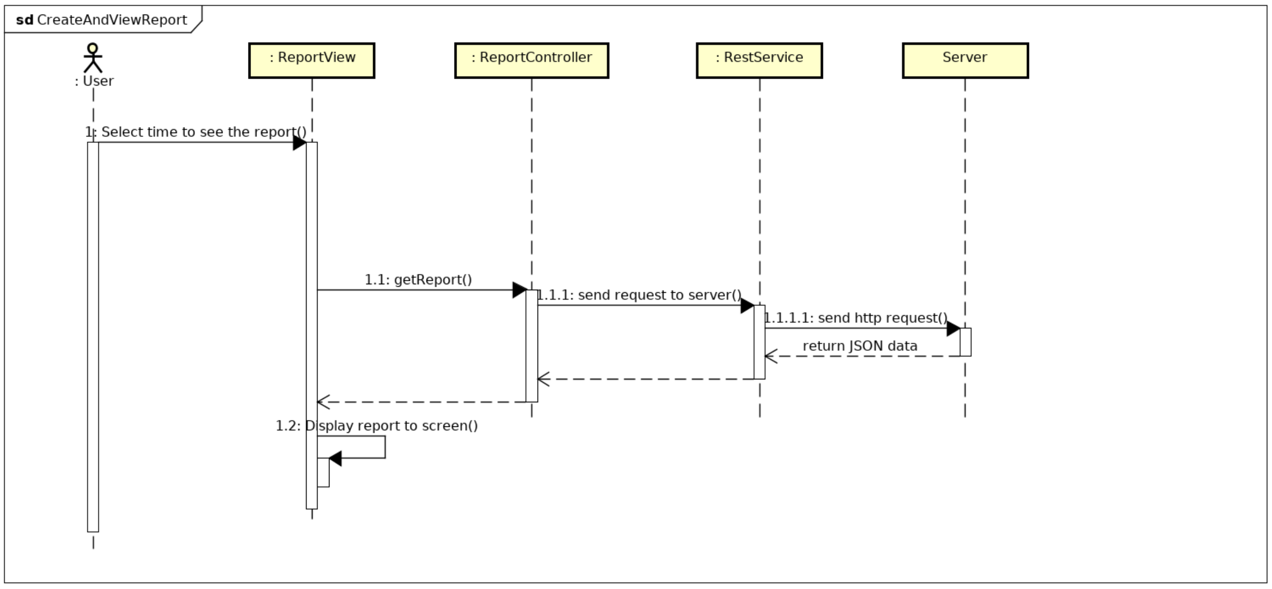
- Modify product


sd ModifyProduct
: User
: ProductView
: EditProductView
: ProductController
: RestService
Server
1: Click edit button()
1.1: Edit product(product)()
1.1.1: Display the old information of the product()
1.1.1.1: updateProductInfo()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data


sd DeleteProduct
User
: ProductView
: ProductController
: RestService
Server
1: Click delete button()
1.1: Display dialog to confirm the deletion()
1.1.1: deleteProduct()
1.1.1.1: send request to delete product()
1.1.1.1.1: send request to server()
1.1.2: Display success dialog()

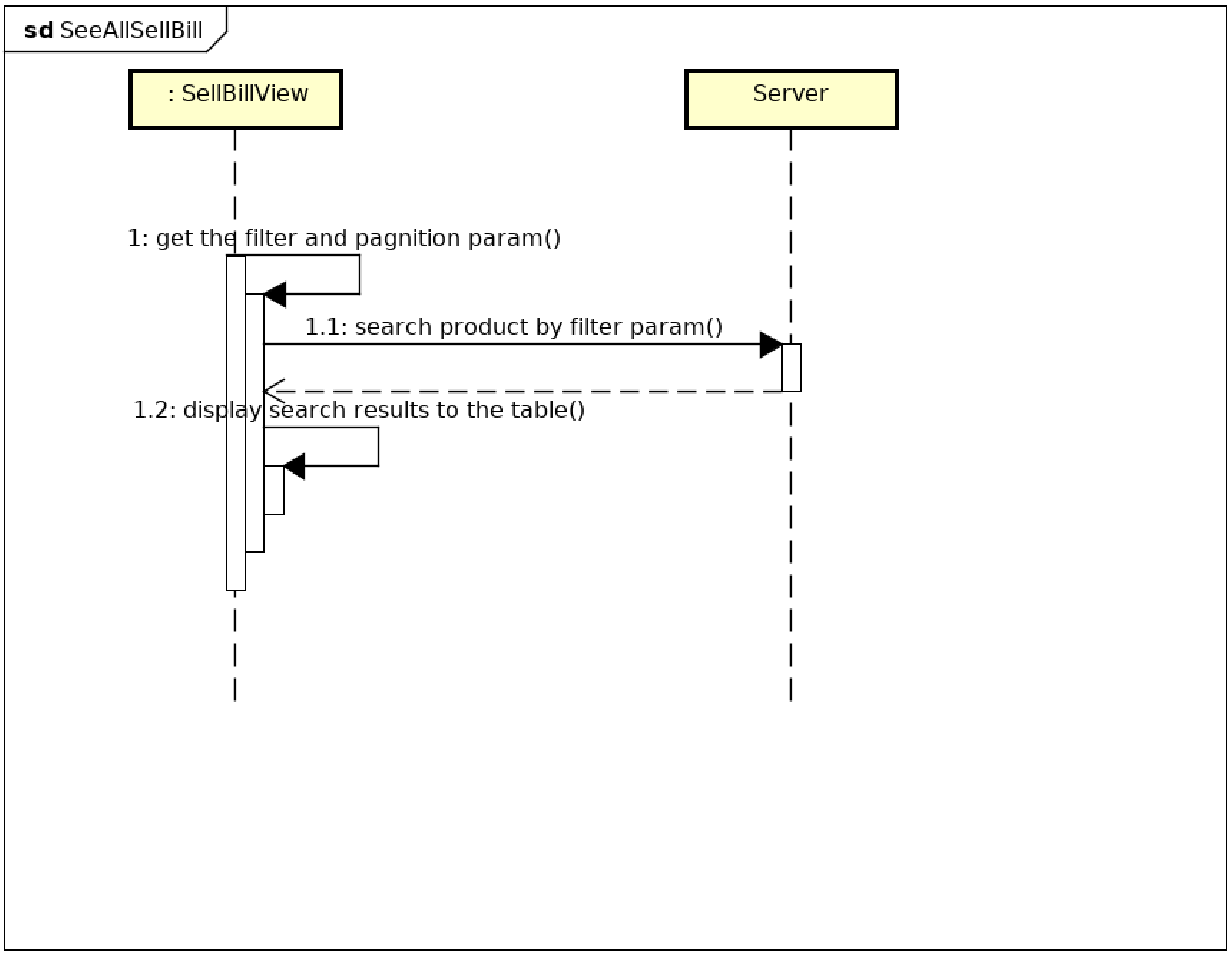
- Delete product

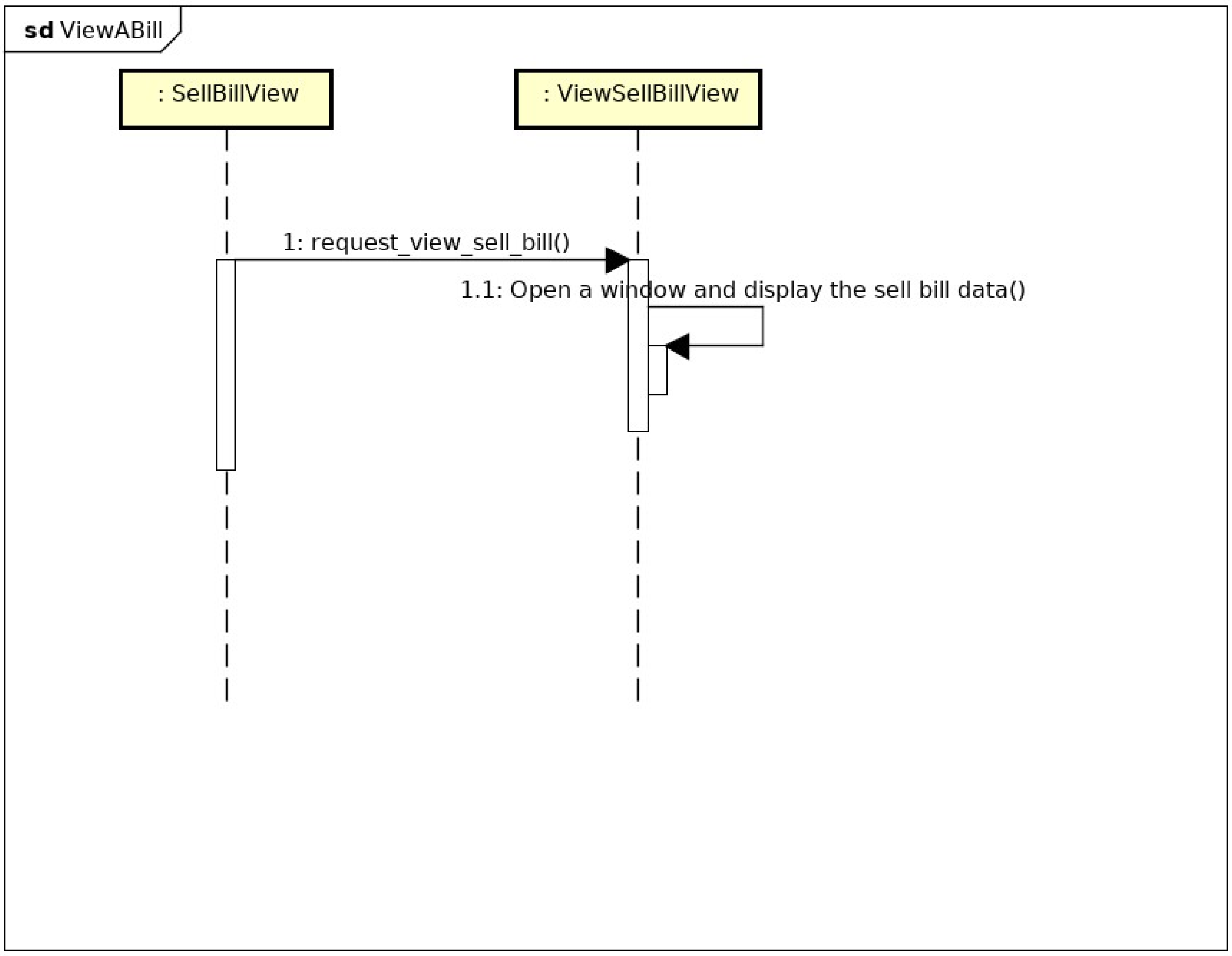
***** REPORTS**


sd CreateAndViewReport
: User
: ReportView
: ReportController
: RestService
Server
1: Select time to see the report()
1.1: getReport()
1.1.1: send request to server()
1.1.1.1: send http request()
return JSON data
1.2: Display report to screen()

***** SELL BILL MANAGEMENT**

- See all the sell bill

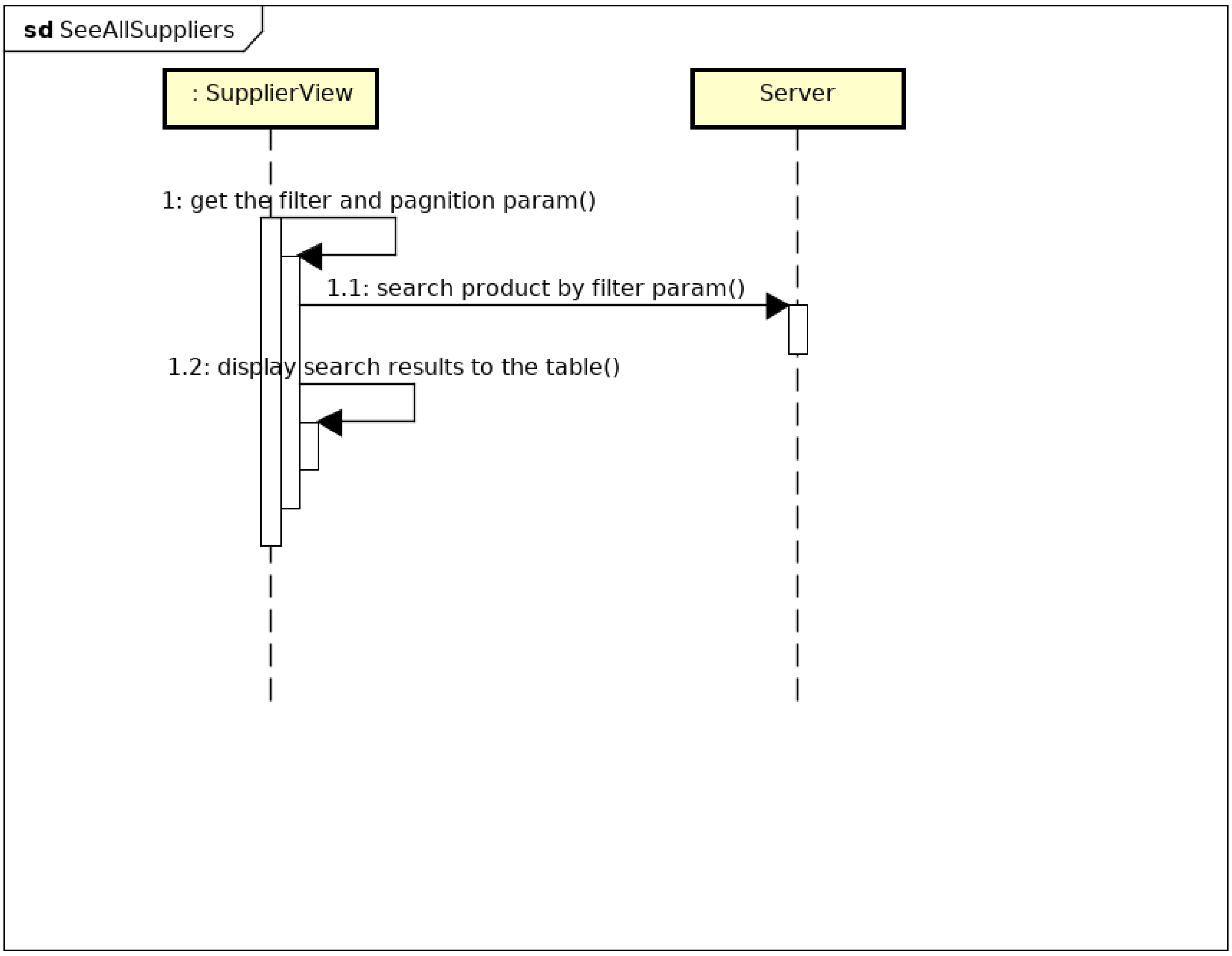

sd SeeAllSellBill
: SellBillView
Server
1: get the filter and pagnition param()
1.1: search product by filter param()
1.2: display search results to the table()


sd ViewABill
: SellBillView
: ViewSellBillView
1: request_view_sell_bill()
1.1: Open a window and display the sell bill data()

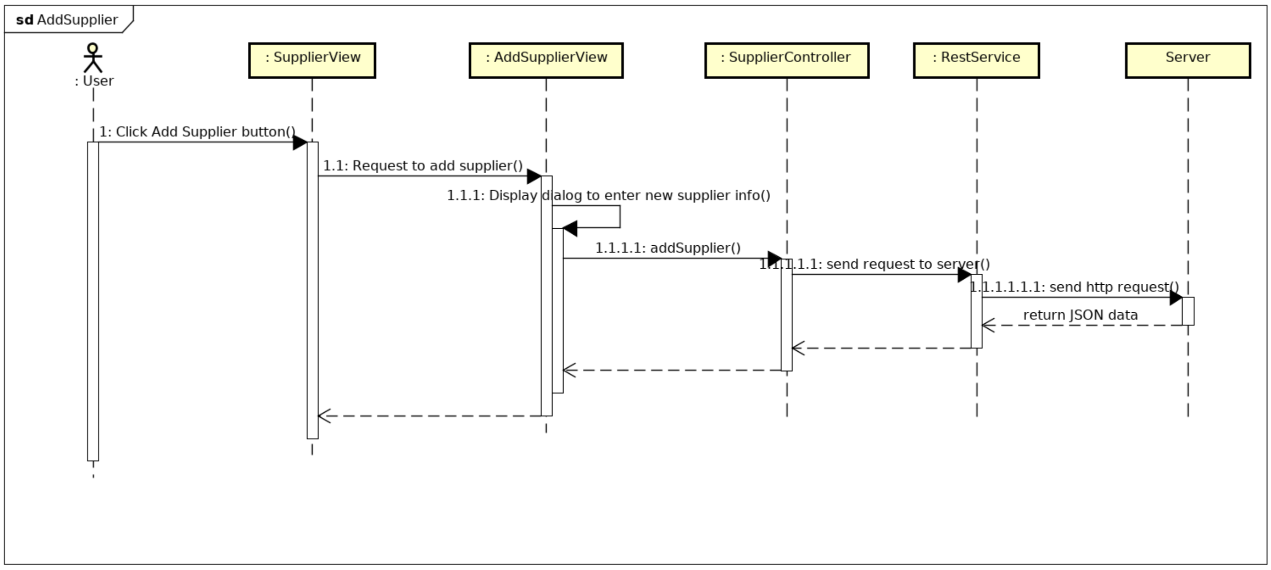
- View a bill

***** SUPPLIER MANAGEMENT**

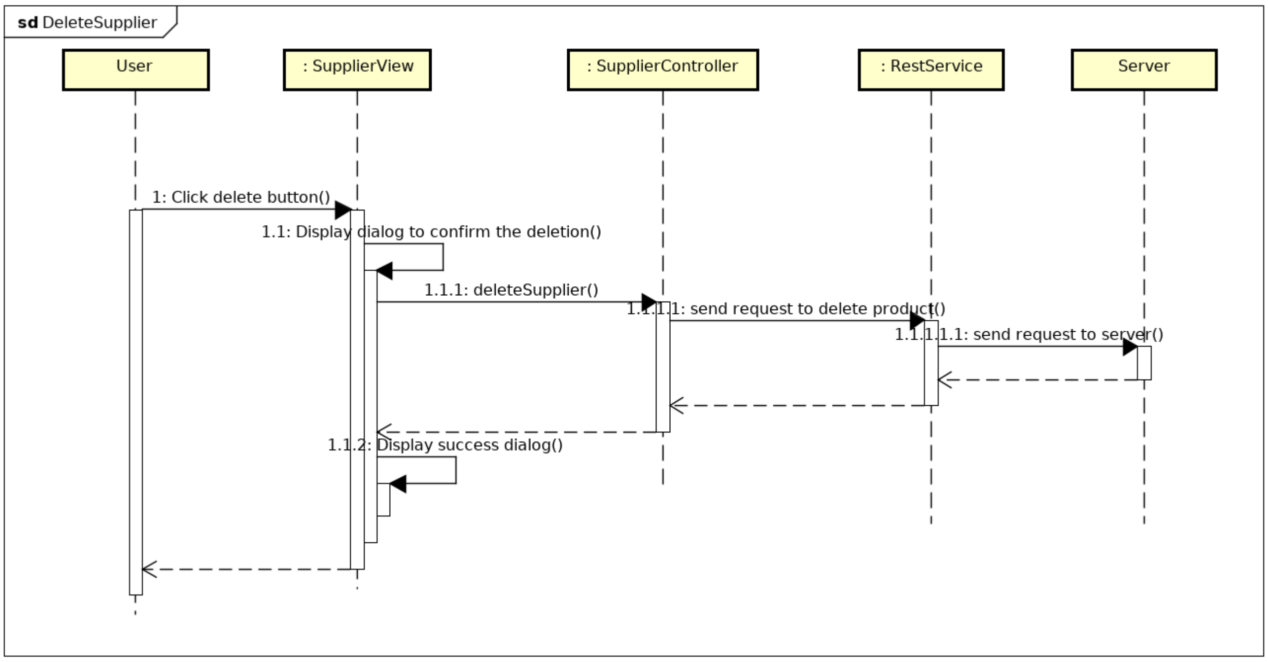
- See all supplier


sd SeeAllSuppliers
: SupplierView
Server
1: get the filter and pagnition param()
1.1: search product by filter param()
1.2: display search results to the table()

- Add supplier

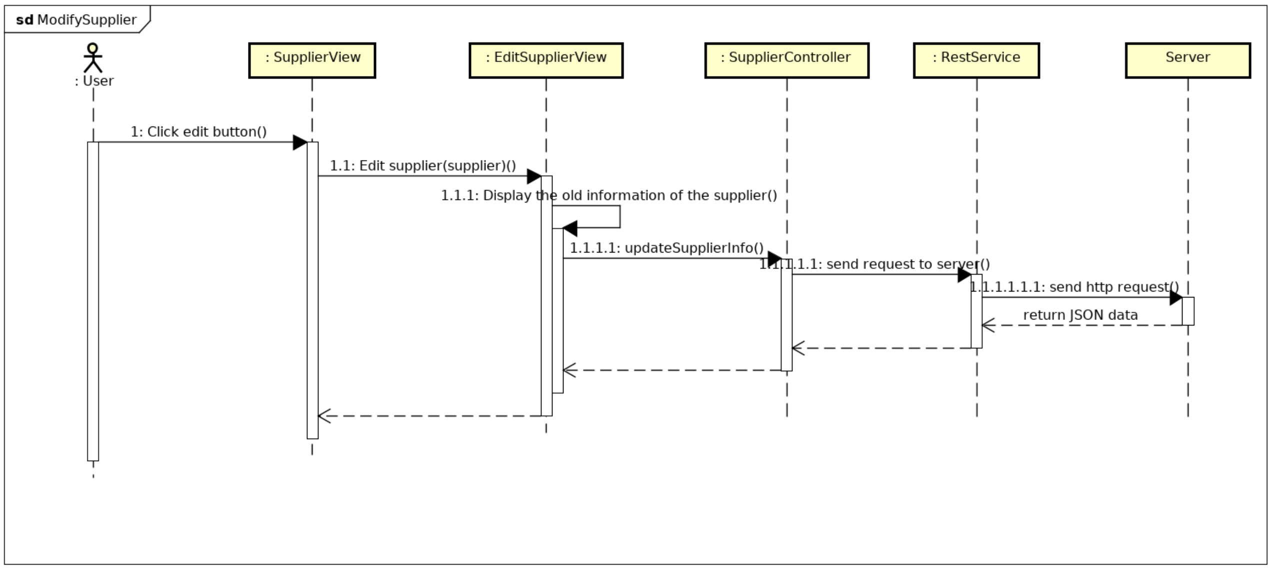

sd AddSupplier
: User
: SupplierView
: AddSupplierView
: SupplierController
: RestService
Server
1: Click Add Supplier button()
1.1: Request to add supplier()
1.1.1: Display dialog to enter new supplier info()
1.1.1.1: addSupplier()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

- Delete supplier



sd DeleteSupplier
User
: SupplierView
: SupplierController
: RestService
Server
1: Click delete button()
1.1: Display dialog to confirm the deletion()
1.1.1: deleteSupplier()
1.1.1.1: send request to delete product()
1.1.1.1.1: send request to server()
1.1.2: Display success dialog()

- **Edit supplier**

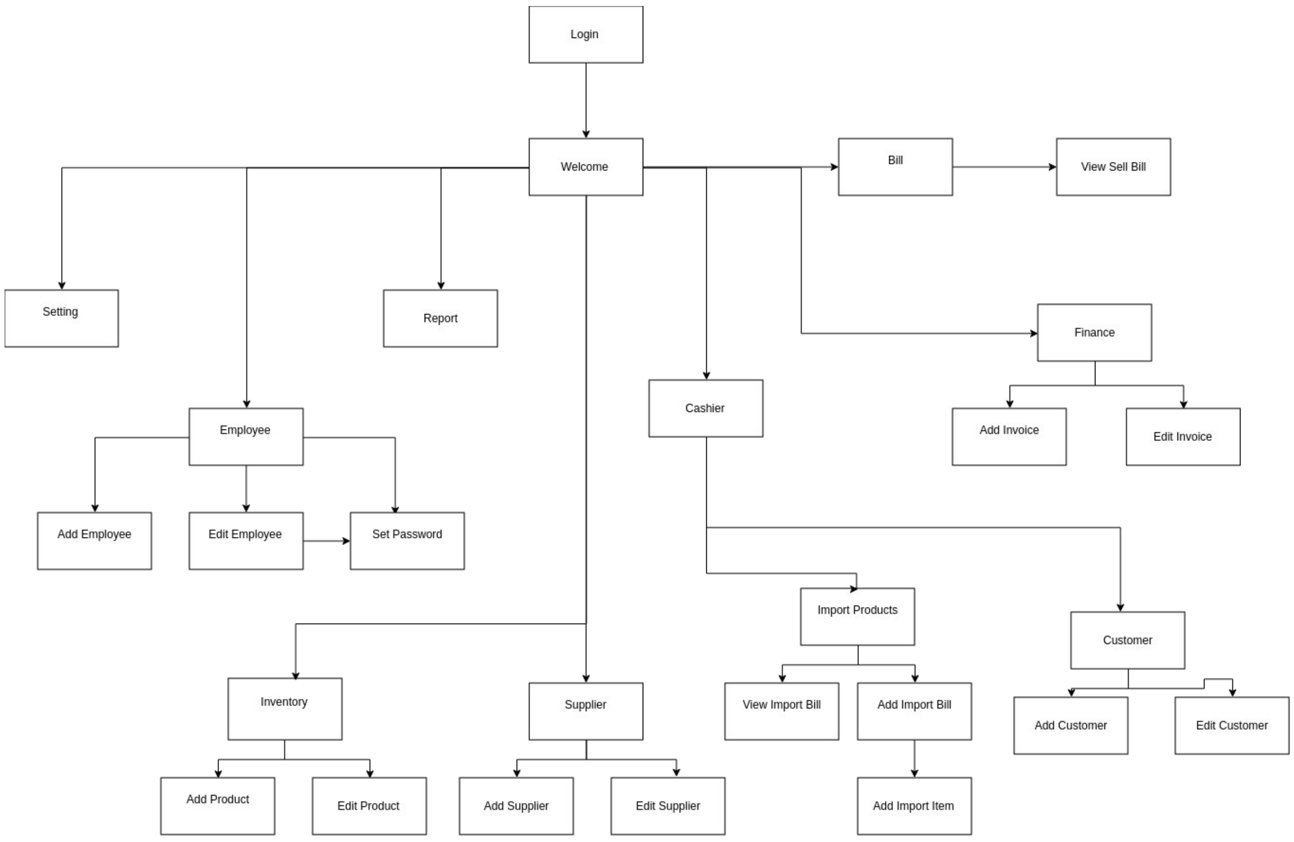


sd ModifySupplier
: User
: SupplierView
: EditSupplierView
: SupplierController
: RestService
Server
1: Click edit button()
1.1: Edit supplier(supplier)()
1.1.1: Display the old information of the supplier()
1.1.1.1: updateSupplierInfo()
1.1.1.1.1: send request to server()
1.1.1.1.1.1: send http request()
return JSON data

3 Interface Design

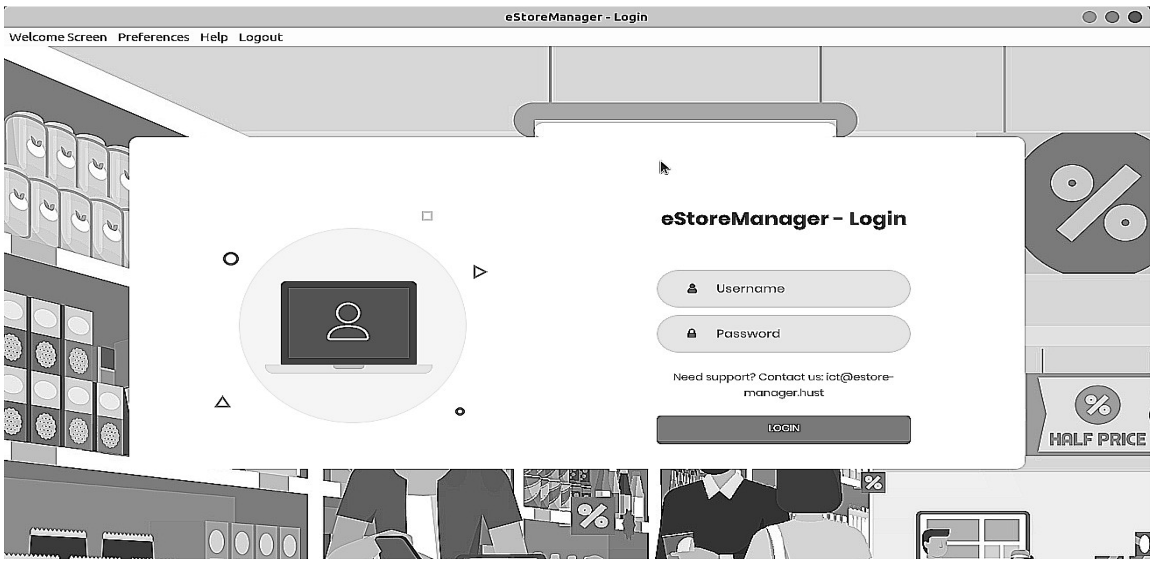
3.1 User Interface

3.1.1 Screen Transition



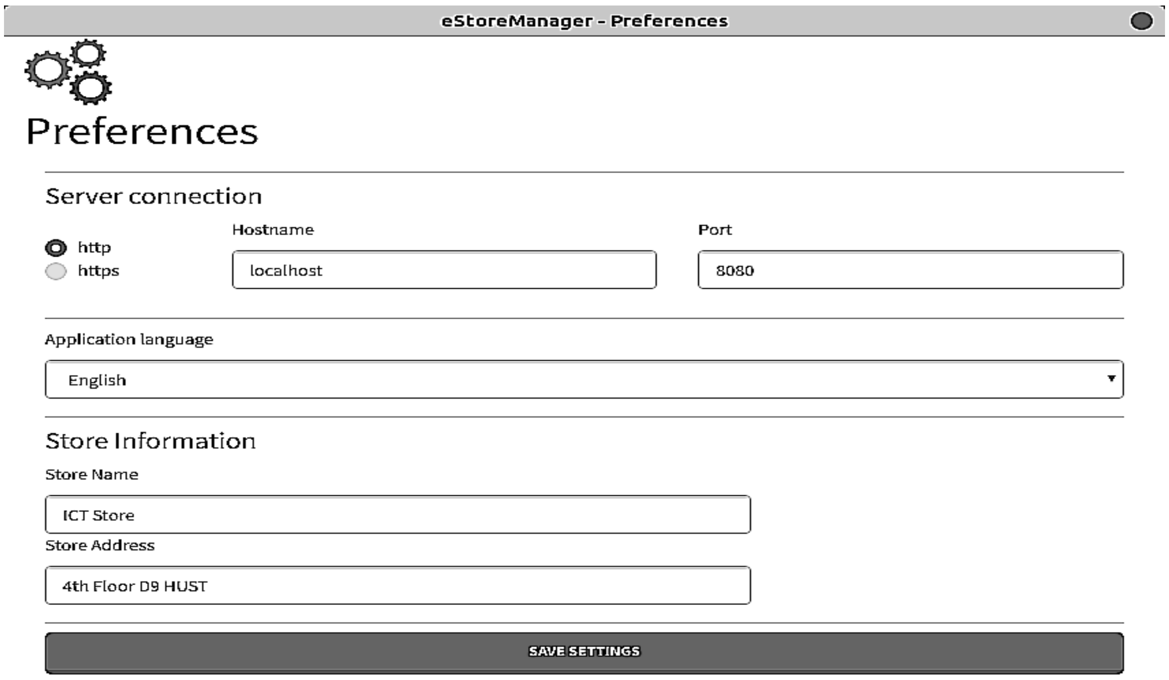
3.1.2 Screen design and screen specification

* Login



Need support? Contact us: ict@estore-manager.hust

* Setting


eStoreManager - Preferences

Preferences

Server connection

Hostname

Port

[✓ http] https

localhost

8080

Application language

English

Store Information

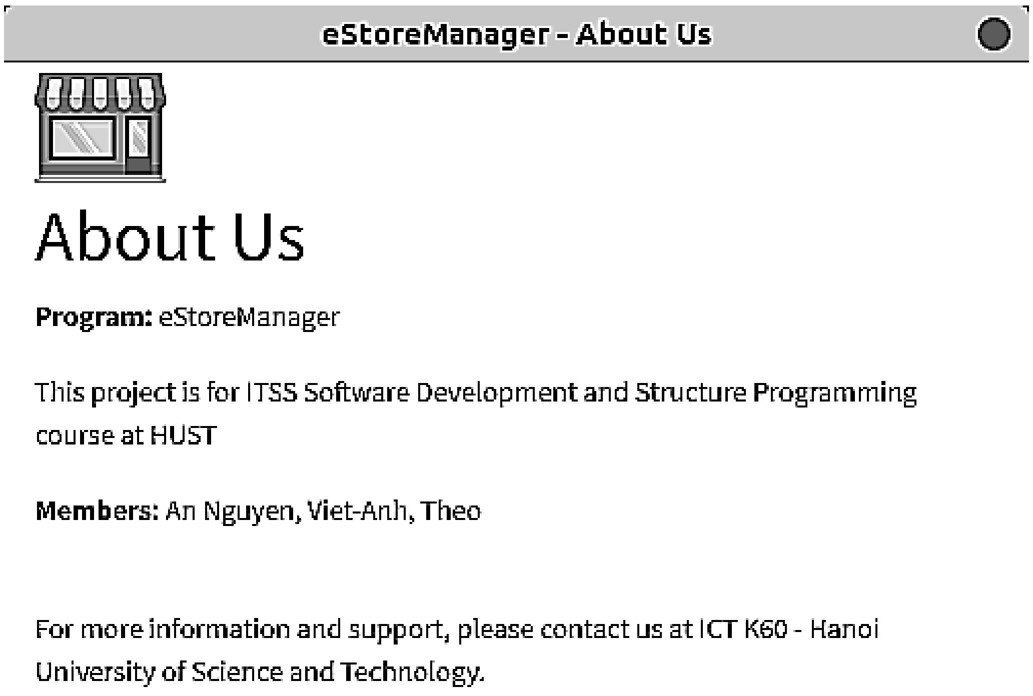
Store Name

ICT Store

Store Address

4th Floor D9 HUST

* About Us


eStoreManager - About Us

About Us

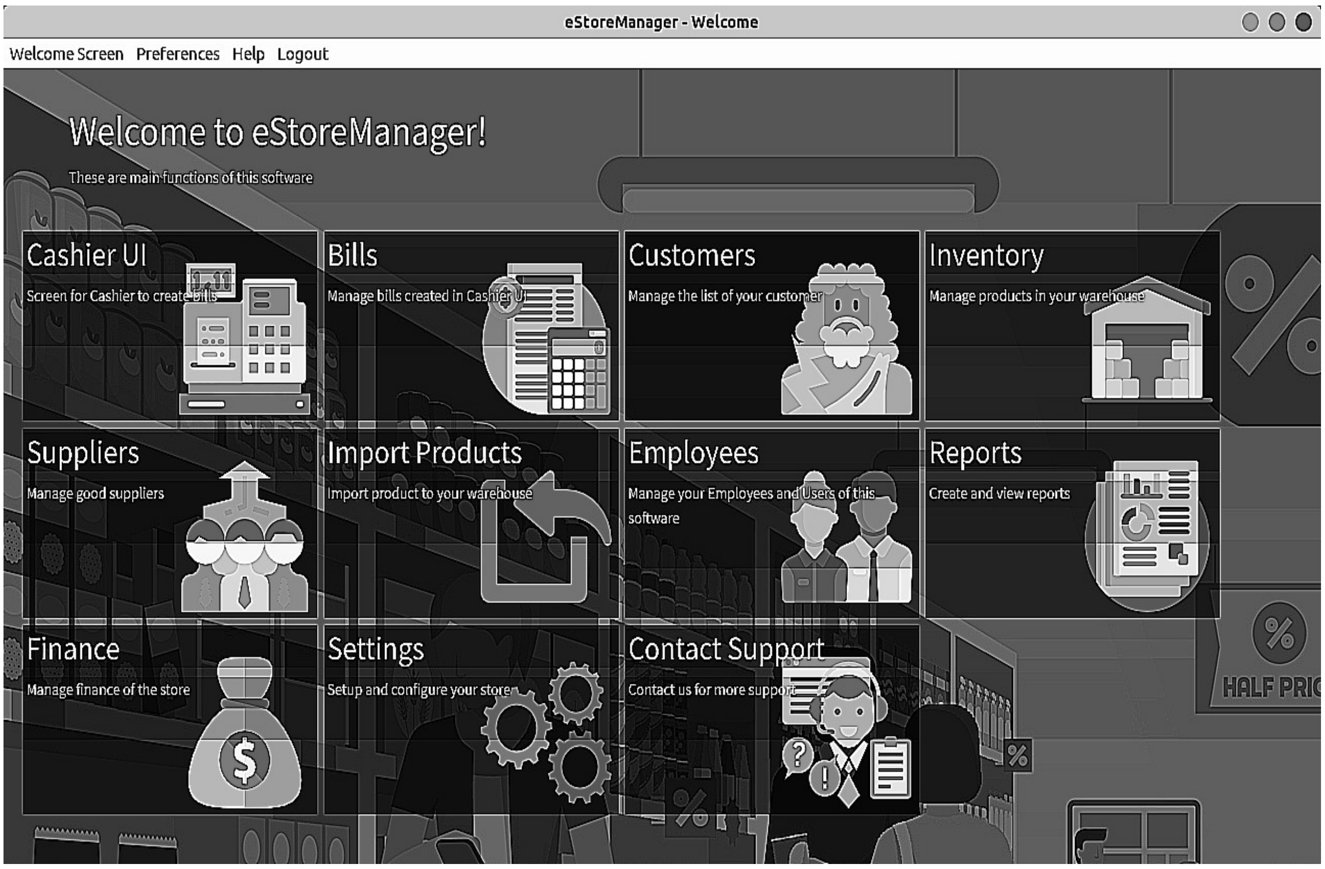
Program: eStoreManager

This project is for ITSS Software Development and Structure Programming course at HUST

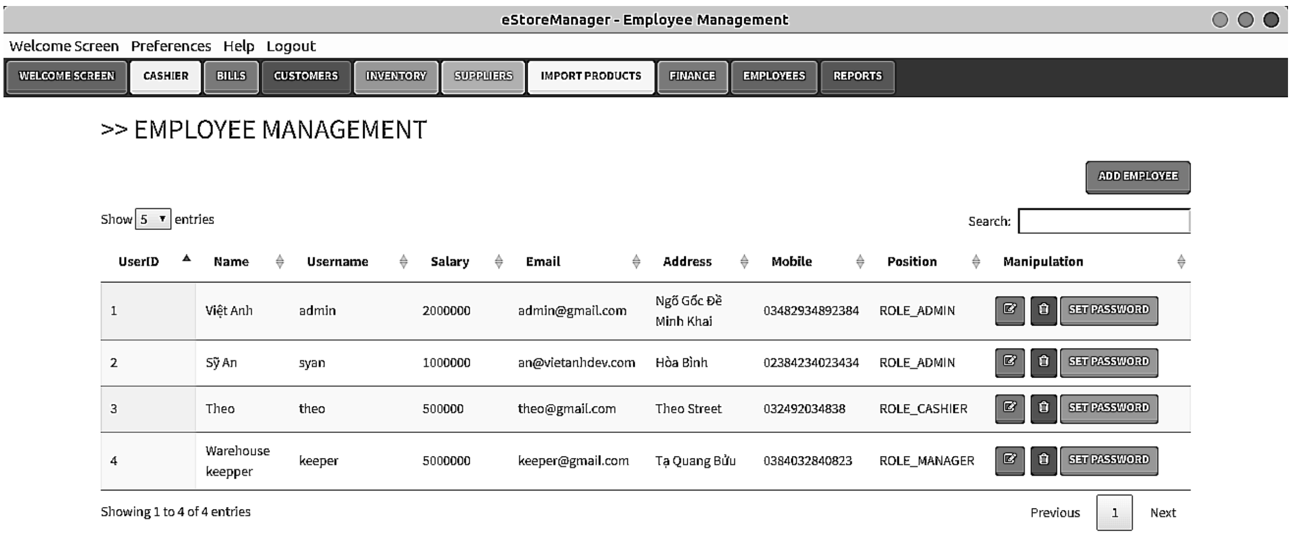
Members: An Nguyen, Viet-Anh, Theo

For more information and support, please contact us at ICT K60 - Hanoi University of Science and Technology.

* Welcome


eStoreManager - Welcome
Welcome Screen Preferences Help Logout
Welcome to eStoreManager!
These are main functions of this software
Cashier UI
Screen for Cashier to create bills
Bills
Manage bills created in Cashier UI
Customers
Manage the list of your customer
Inventory
Manage products in your warehouse
Suppliers
Manage good suppliers
Import Products
Import product to your warehouse
Employees
Manage your Employees and Users of this software
Reports
Create and view reports
Finance
Manage finance of the store
Settings
Setup and configure your store
Contact Support
Contact us for more support
HALF PRIC

* Employee



>> EMPLOYEE MANAGEMENT

Show 5 entries

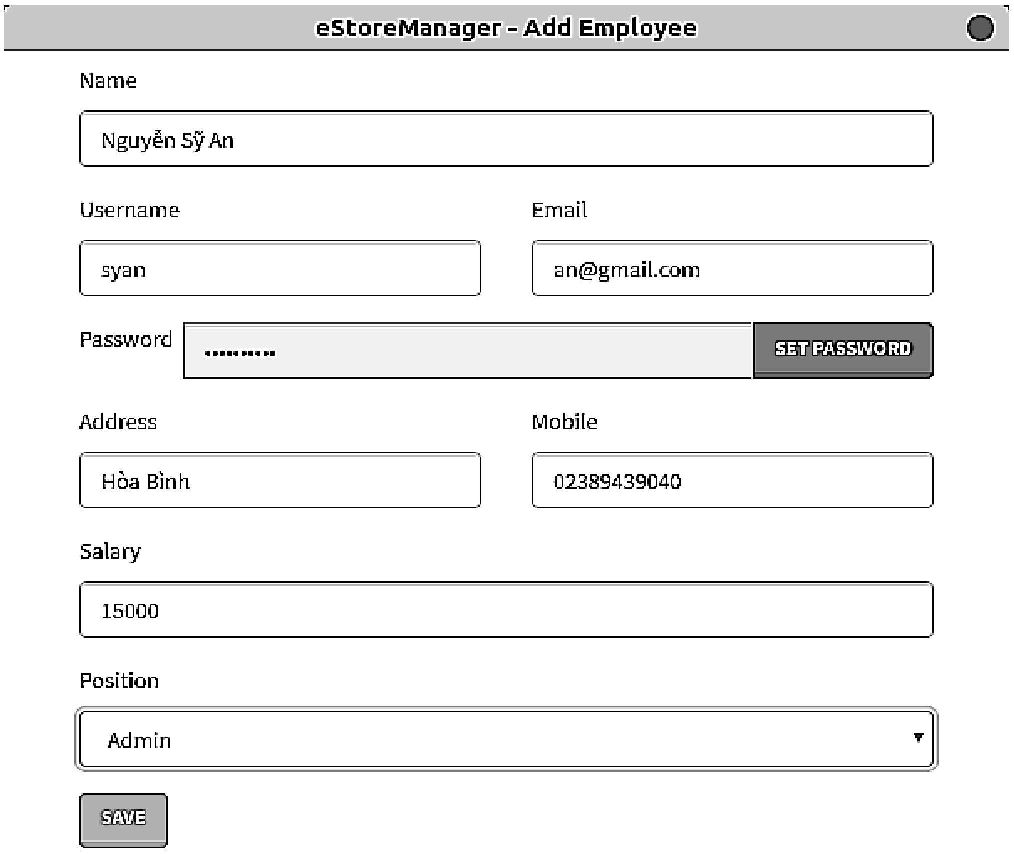
Search:

UserID	Name	Username	Salary	Email	Address	Mobile	Position	Manipulation
1	Việt Anh	admin	2000000	admin@gmail.com	Ngõ Gốc Đề Minh Khai	03482934892384	ROLE_ADMIN	SET PASSWORD
2	Sỹ An	syan	1000000	an@vietanhdev.com	Hòa Bình	02384234023434	ROLE_ADMIN	SET PASSWORD
3	Theo	theo	500000	theo@gmail.com	Theo Street	032492034838	ROLE_CASHIER	SET PASSWORD
4	Warehouse keepper	keeper	5000000	keeper@gmail.com	Tạ Quang Bửu	0384032840823	ROLE_MANAGER	SET PASSWORD

Showing 1 to 4 of 4 entries

Previous 1 Next

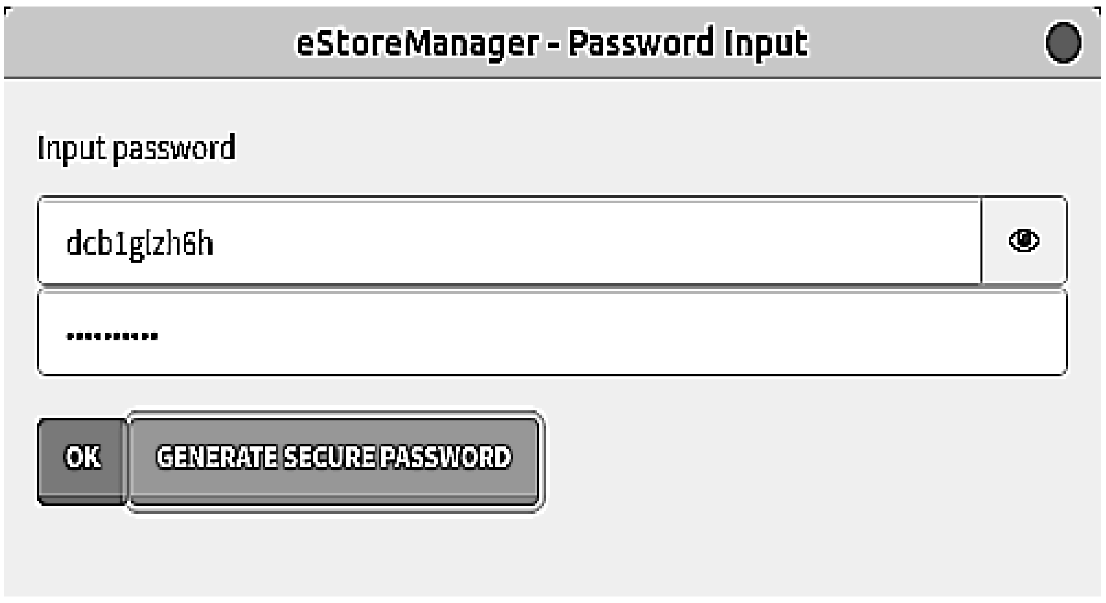
* Employee - Add employee



eStoreManager - Add Employee

Name	Nguyễn Sỹ An
Username	syan
Email	an@gmail.com
Password	••••••••••
Address	Hòa Bình
Mobile	02389439040
Salary	15000
Position	Admin

* Employee – Set password



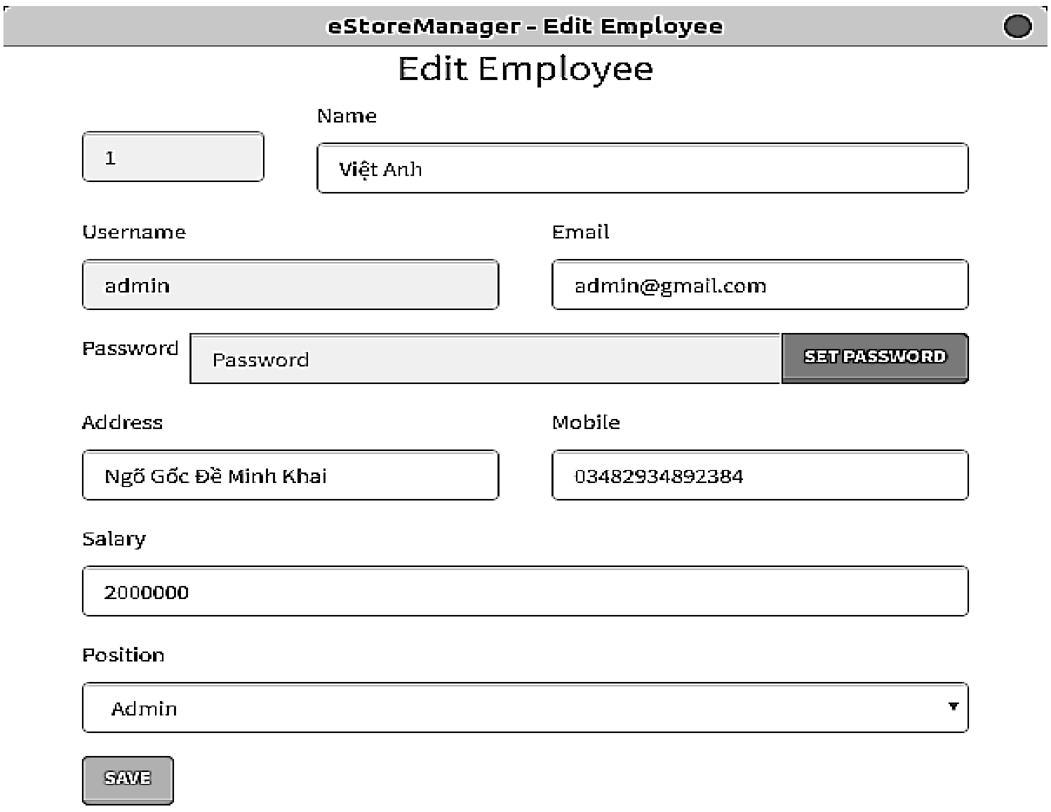
eStoreManager - Password Input

Input password

dcb1g[zh6h

••••••••••

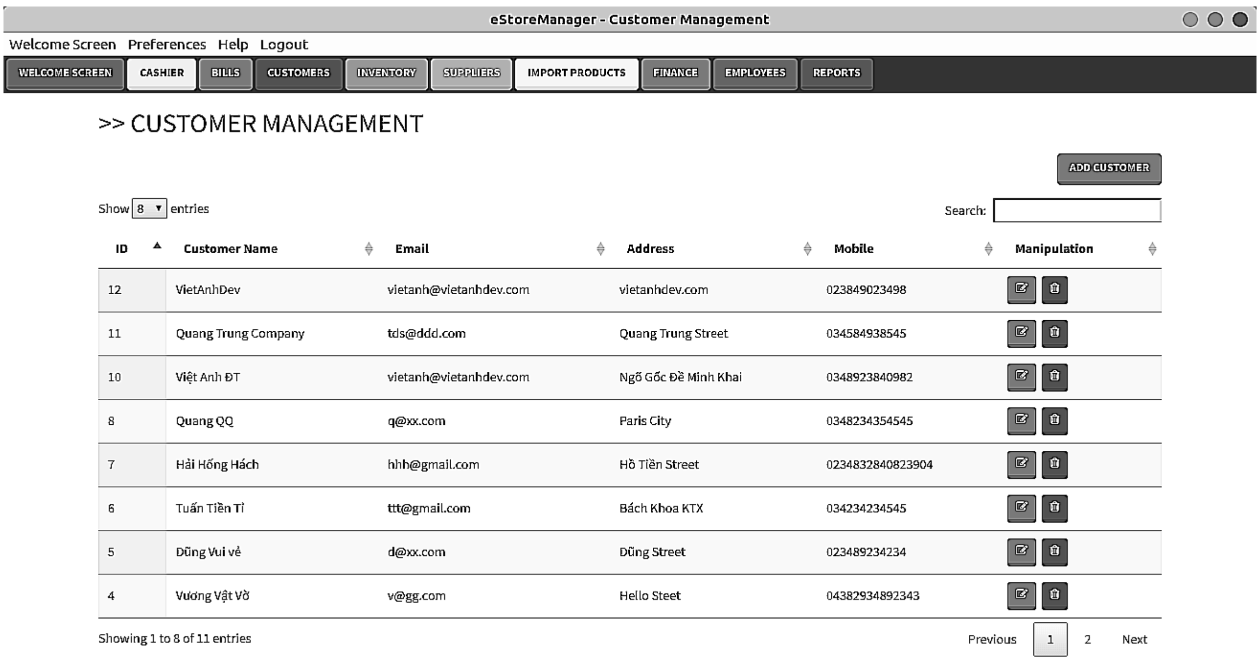
* Employee – Edit employee



Edit Employee

	1
Name	Việt Anh
Username	admin
Email	admin@gmail.com
Password	Password
Address	Ngõ Gốc Đề Minh Khai
Mobile	03482934892384
Salary	2000000
Position	Admin

* Customer



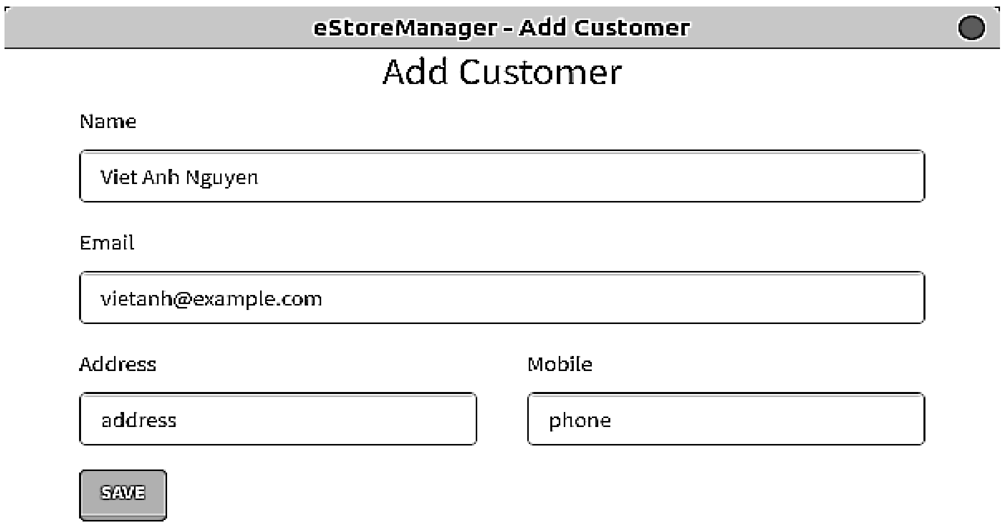
>> CUSTOMER MANAGEMENT

ID	Customer Name	Email	Address	Mobile	Manipulation
12	VietAnhDev	vietanh@vietanhdev.com	vietanhdev.com	023849023498	
11	Quang Trung Company	tds@ddd.com	Quang Trung Street	034584938545	
10	Việt Anh ĐT	vietanh@vietanhdev.com	Ngõ Gốc Đề Minh Khai	0348923840982	
8	Quang QQ	q@xx.com	Paris City	0348234354545	
7	Hải Hống Hách	hhh@gmail.com	Hồ Tiền Street	0234832840823904	
6	Tuấn Tiền Tỉ	ttt@gmail.com	Bách Khoa KTX	034234234545	
5	Dũng Vui vẻ	d@xx.com	Dũng Street	023489234234	
4	Vương Vật Vờ	v@gg.com	Hello Steet	04382934892343	

Showing 1 to 8 of 11 entries

Previous 1 2 Next

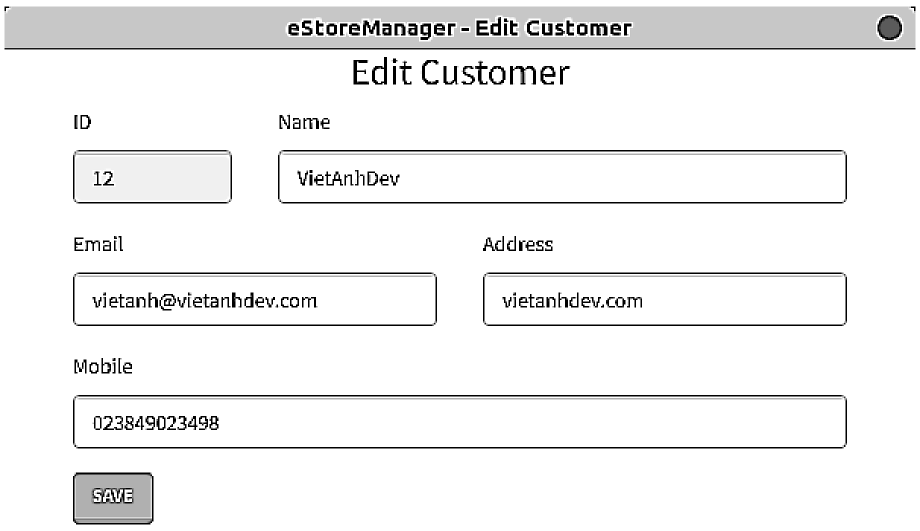
* Customer – Add customer


eStoreManager - Add Customer

Add Customer

Name	Viet Anh Nguyen
Email	vietanh@example.com
Address	address
Mobile	phone

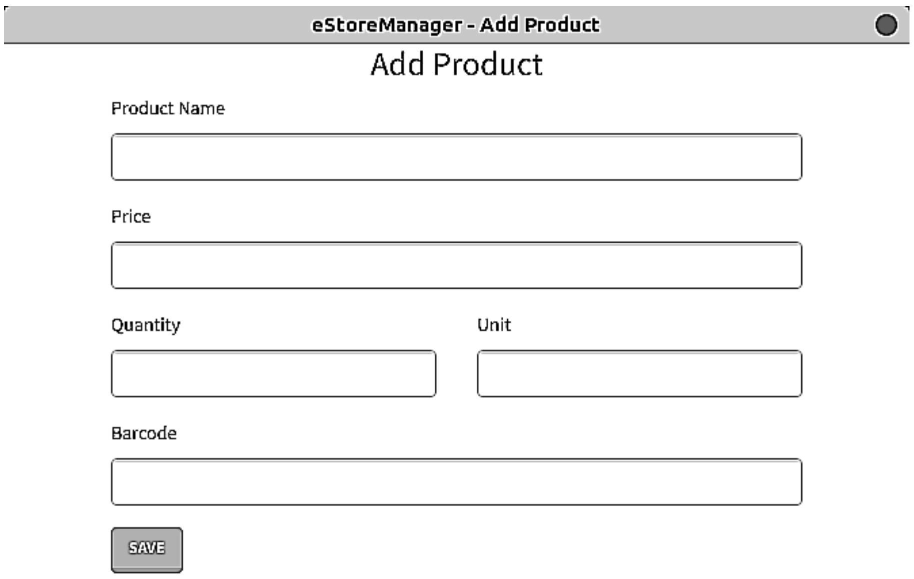
* Customer – Edit customer


eStoreManager - Edit Customer

Edit Customer

ID	12
Name	VietAnhDev
Email	vietanh@vietanhdev.com
Address	vietanhdev.com
Mobile	023849023498

* Inventory – Add product


eStoreManager - Add Product

Add Product

Product Name

Price

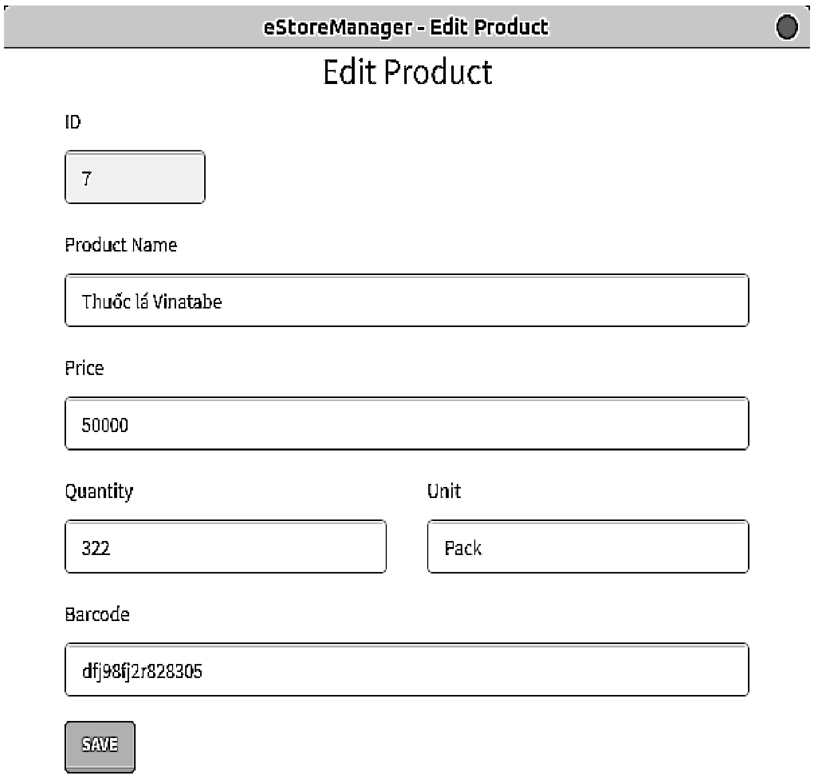
Quantity

Unit

Barcode

SAVE

* Inventory – Edit product


eStoreManager - Edit Product

Edit Product

ID

7

Product Name

Thuốc lá Vinatabe

Price

50000

Quantity

322

Unit

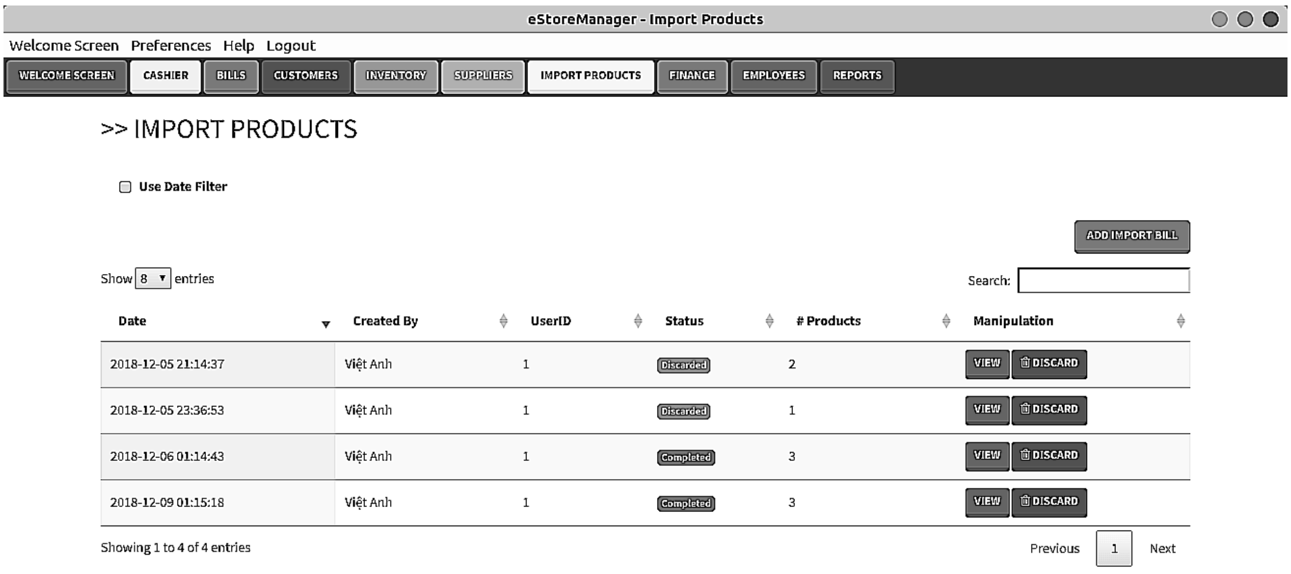
Pack

Barcode

dfj98fj2r828305

SAVE

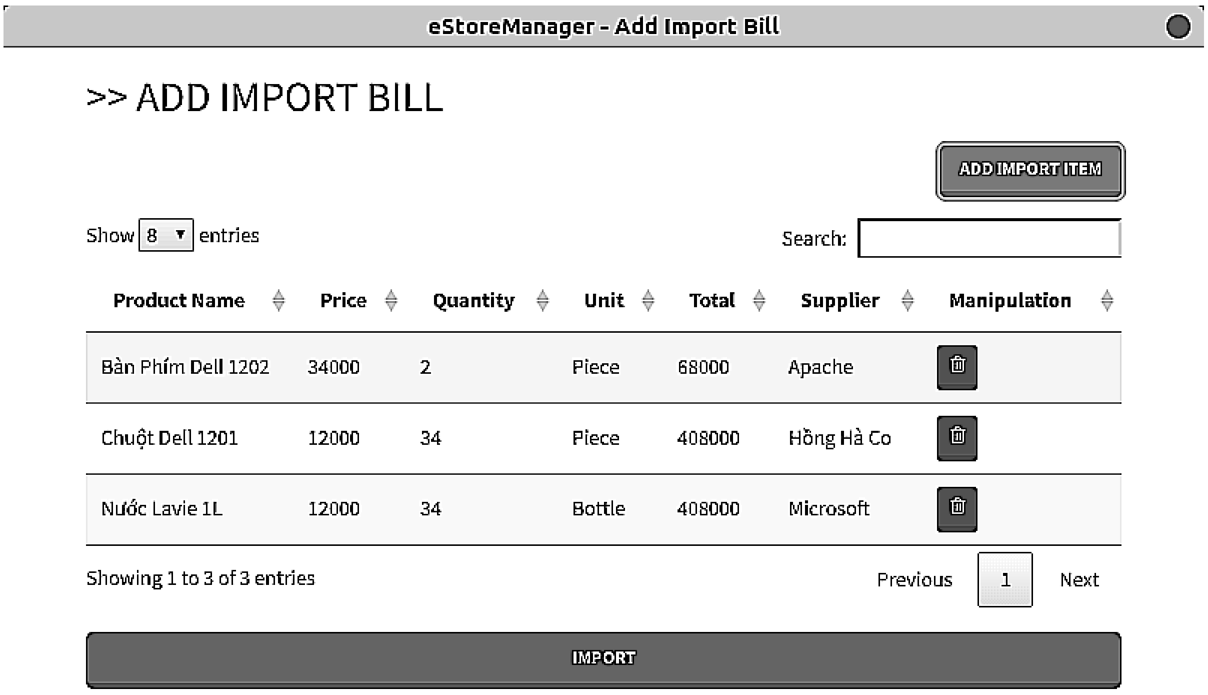
* Import products



Date	Created By	UserID	Status	# Products	Manipulation
2018-12-05 21:14:37	Việt Anh	1	Discarded	2	VIEW DISCARD
2018-12-05 23:36:53	Việt Anh	1	Discarded	1	VIEW DISCARD
2018-12-06 01:14:43	Việt Anh	1	Completed	3	VIEW DISCARD
2018-12-09 01:15:18	Việt Anh	1	Completed	3	VIEW DISCARD

Showing 1 to 4 of 4 entries

* Import products – Add import bill

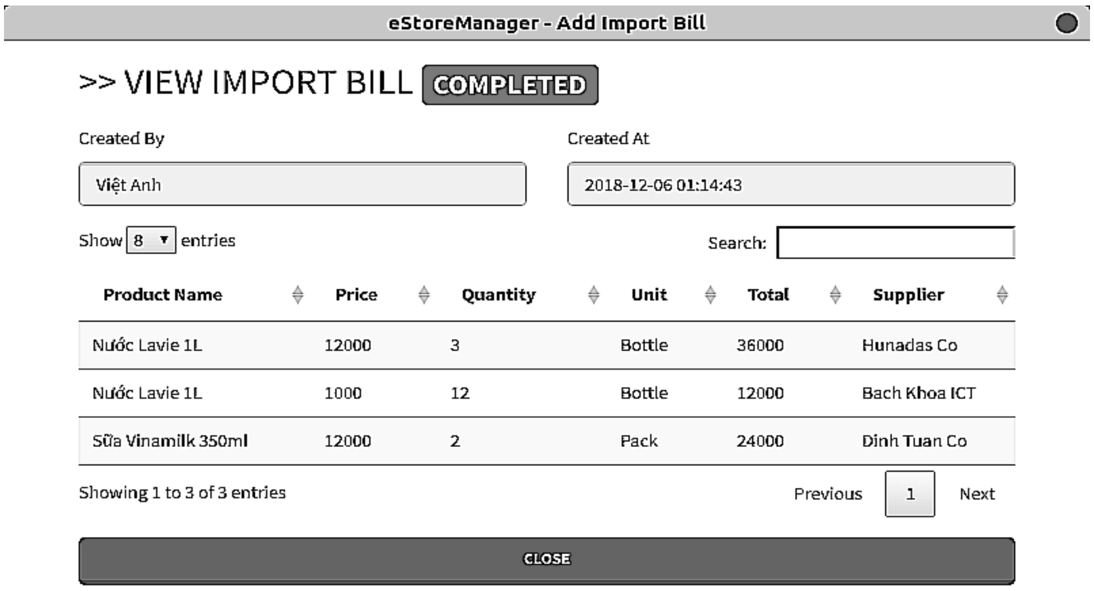


>> ADD IMPORT BILL

Product Name	Price	Quantity	Unit	Total	Supplier	Manipulation
Bàn Phím Dell 1202	34000	2	Piece	68000	Apache	🗑
Chuột Dell 1201	12000	34	Piece	408000	Hồng Hà Co	🗑
Nước Lavie 1L	12000	34	Bottle	408000	Microsoft	🗑

Showing 1 to 3 of 3 entries

* Import products – View import bill



>> VIEW IMPORT BILL COMPLETED

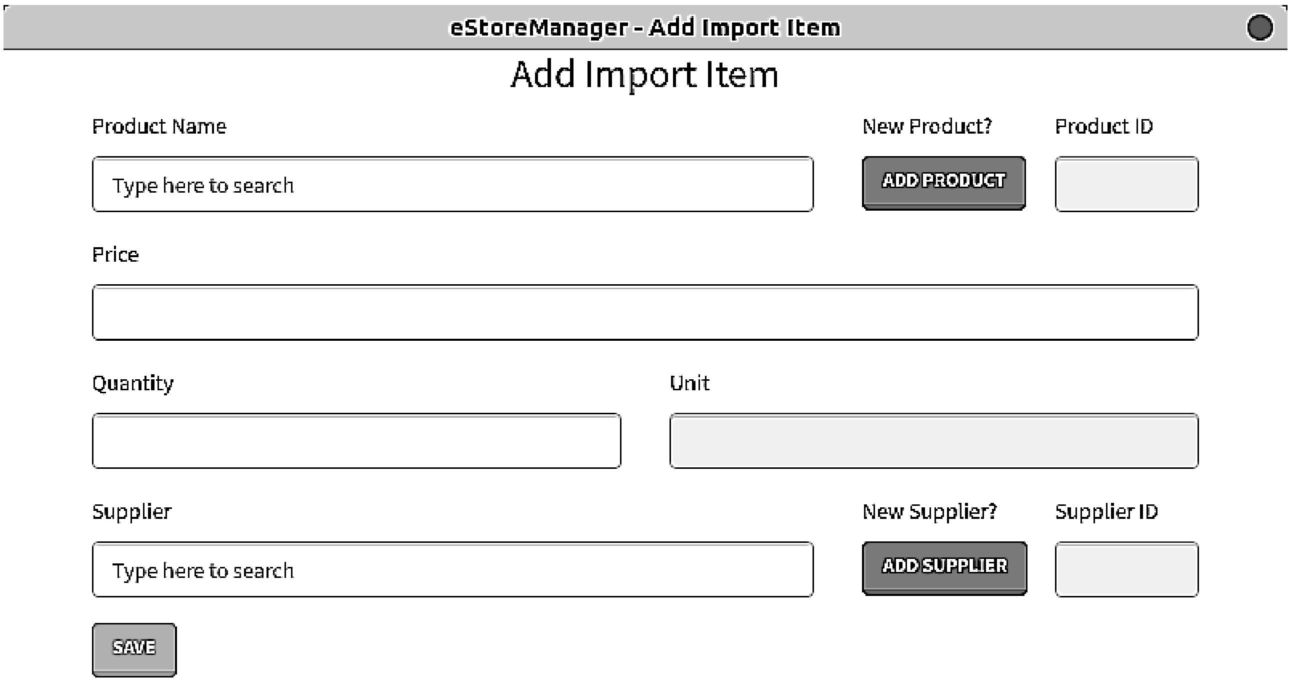
Created By: Việt Anh

Created At: 2018-12-06 01:14:43

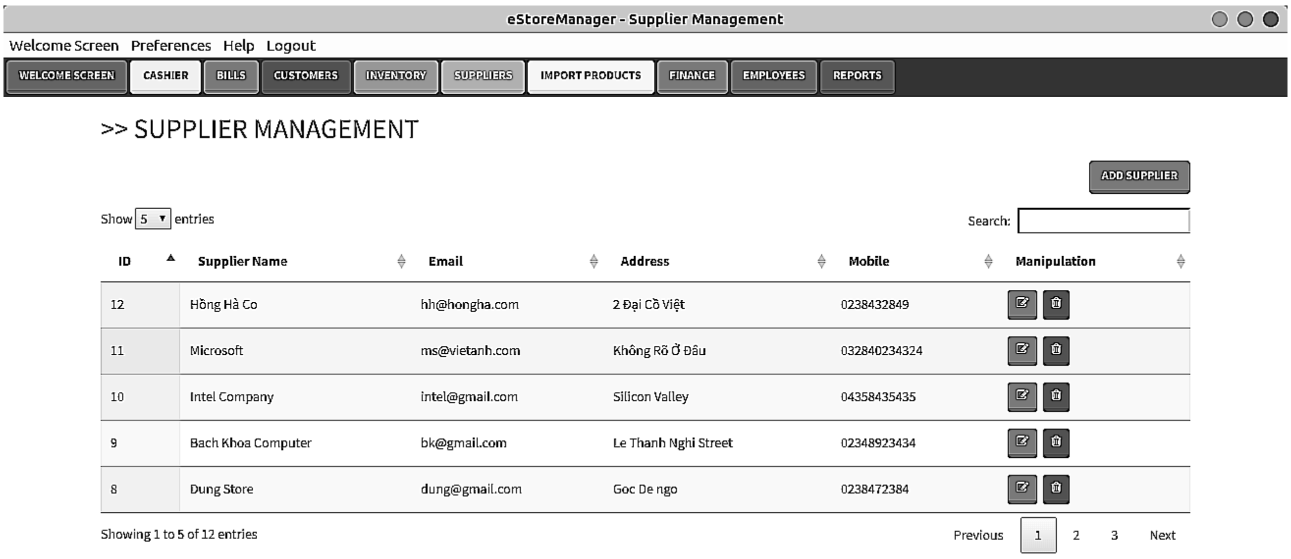
Product Name	Price	Quantity	Unit	Total	Supplier
Nước Lavie 1L	12000	3	Bottle	36000	Hunadas Co
Nước Lavie 1L	1000	12	Bottle	12000	Bach Khoa ICT
Sữa Vinamilk 350ml	12000	2	Pack	24000	Dinh Tuan Co

Showing 1 to 3 of 3 entries

* Import products – Add import bill – Add import item



* Suppliers

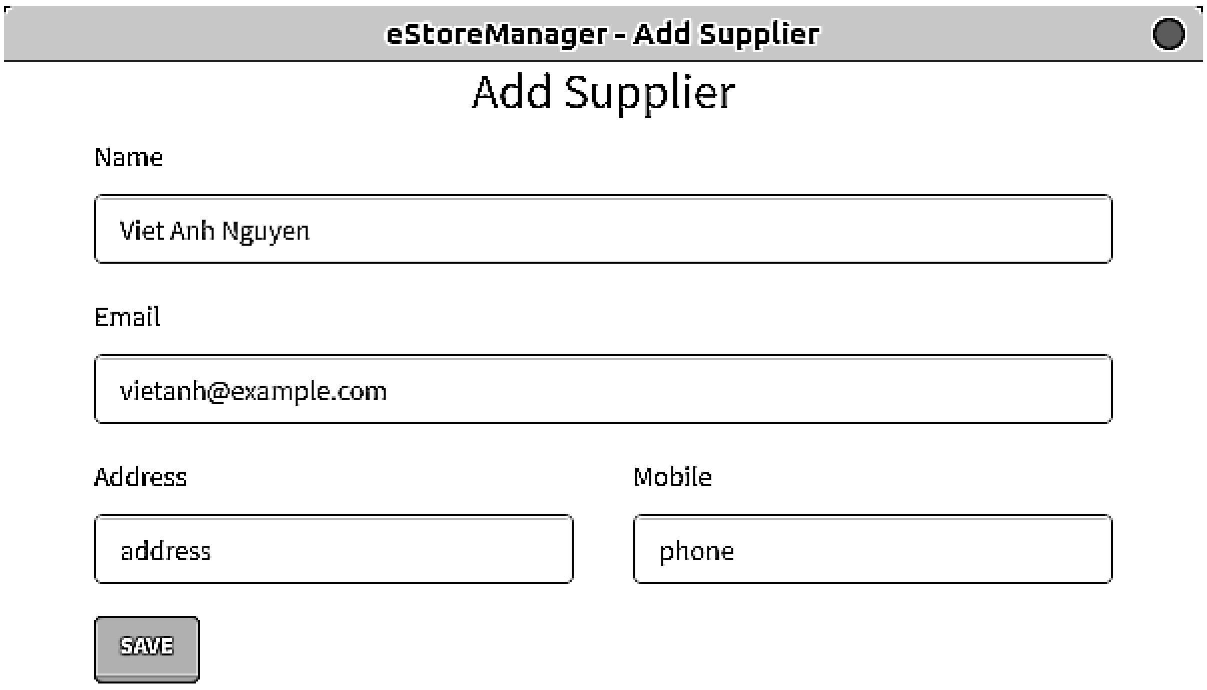


>> SUPPLIER MANAGEMENT

ID	Supplier Name	Email	Address	Mobile	Manipulation
12	Hồng Hà Co	hh@hongha.com	2 Đại Cồ Việt	0238432849	
11	Microsoft	ms@vietanh.com	Không Rõ Ở Đâu	032840234324	
10	Intel Company	intel@gmail.com	Silicon Valley	04358435435	
9	Bach Khoa Computer	bk@gmail.com	Le Thanh Nghi Street	02348923434	
8	Dung Store	dung@gmail.com	Goc De ngo	0238472384	

Showing 1 to 5 of 12 entries

* Suppliers – Add supplier



Add Supplier

Name

Viet Anh Nguyen

Email

vietanh@example.com

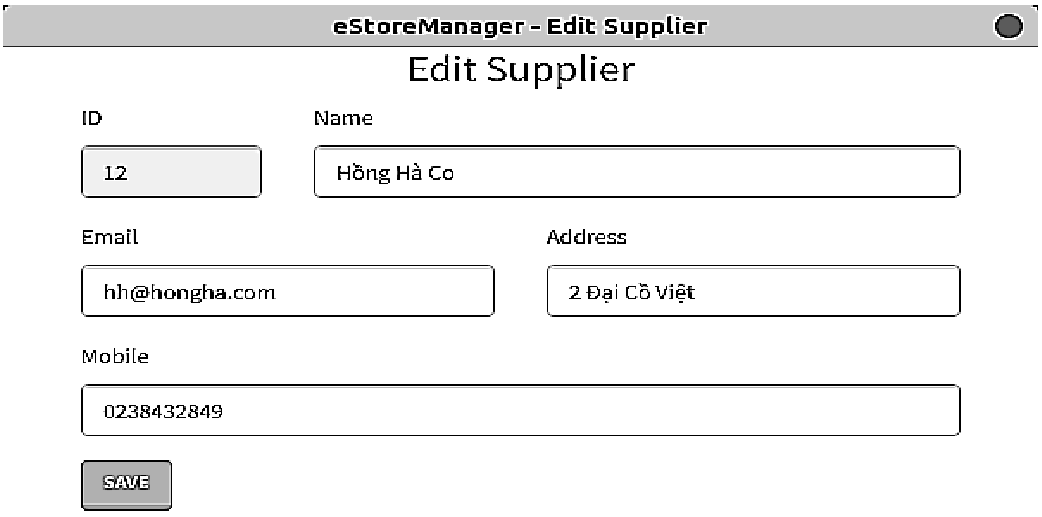
Address

address

Mobile

phone

* Suppliers – Edit supplier



Edit Supplier

ID: 12

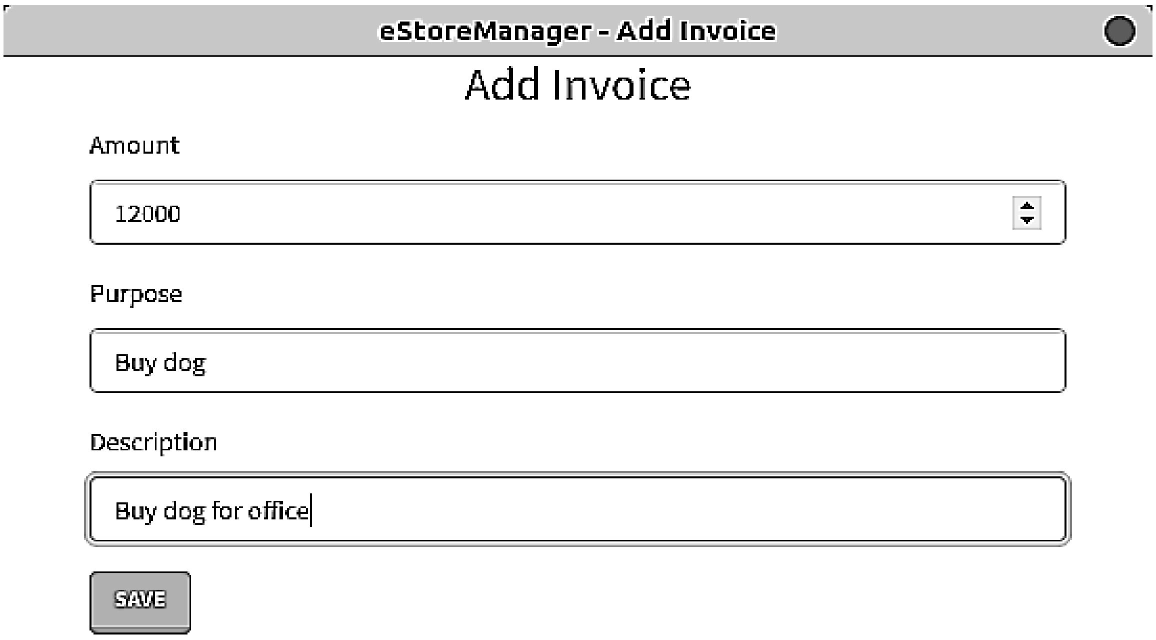
Name: Hồng Hà Co

Email: hh@hongha.com

Address: 2 Đại Cồ Việt

Mobile: 0238432849

* Finance – Add invoice



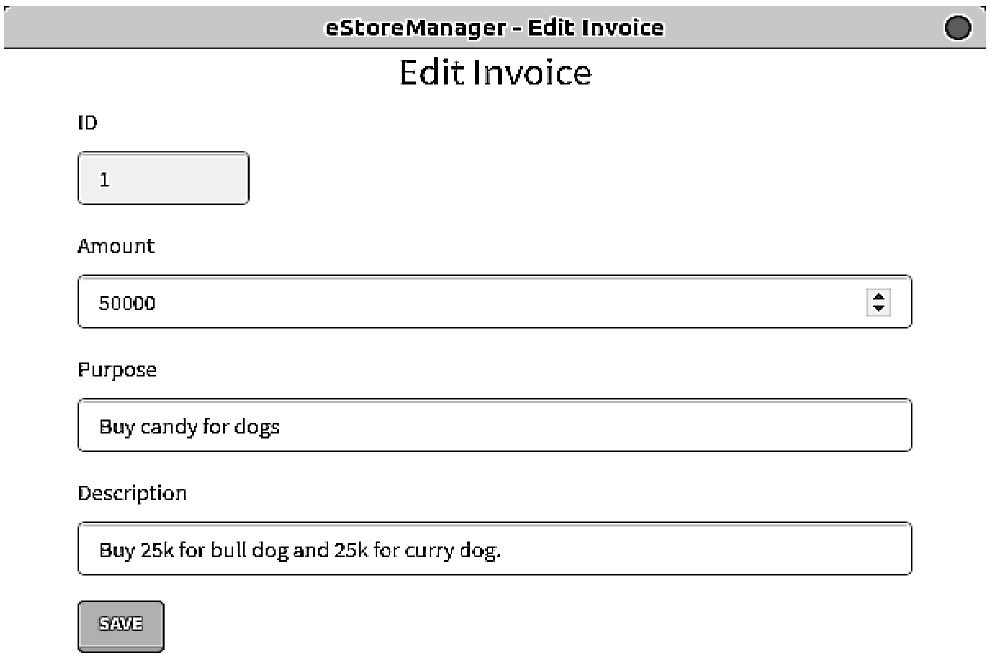
Add Invoice

Amount: 12000

Purpose: Buy dog

Description: Buy dog for office

* Finance – Edit invoice



Edit Invoice

ID

1

Amount

50000

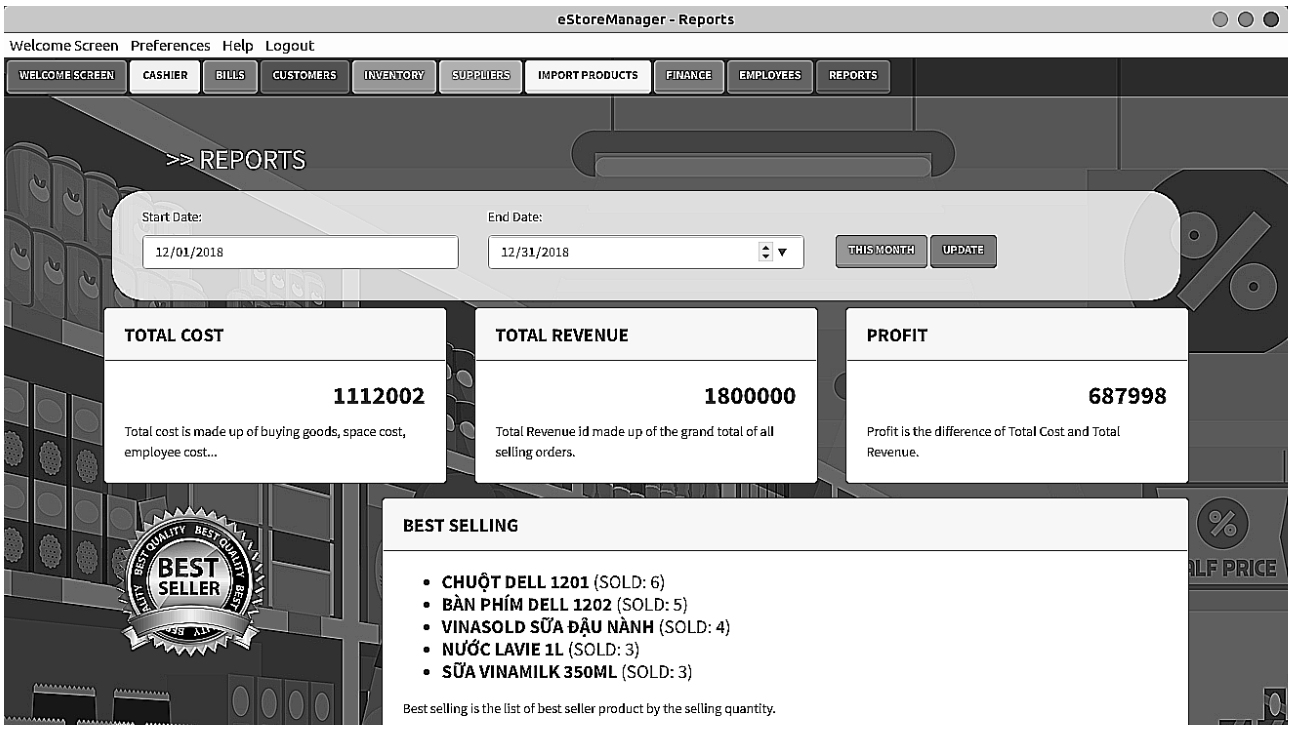
Purpose

Buy candy for dogs

Description

Buy 25k for bull dog and 25k for curry dog.

* Report



Start Date: 12/01/2018

End Date: 12/31/2018

TOTAL COST

1112002

Total cost is made up of buying goods, space cost, employee cost...

TOTAL REVENUE

1800000

Total Revenue id made up of the grand total of all selling orders.

PROFIT

687998

Profit is the difference of Total Cost and Total Revenue.

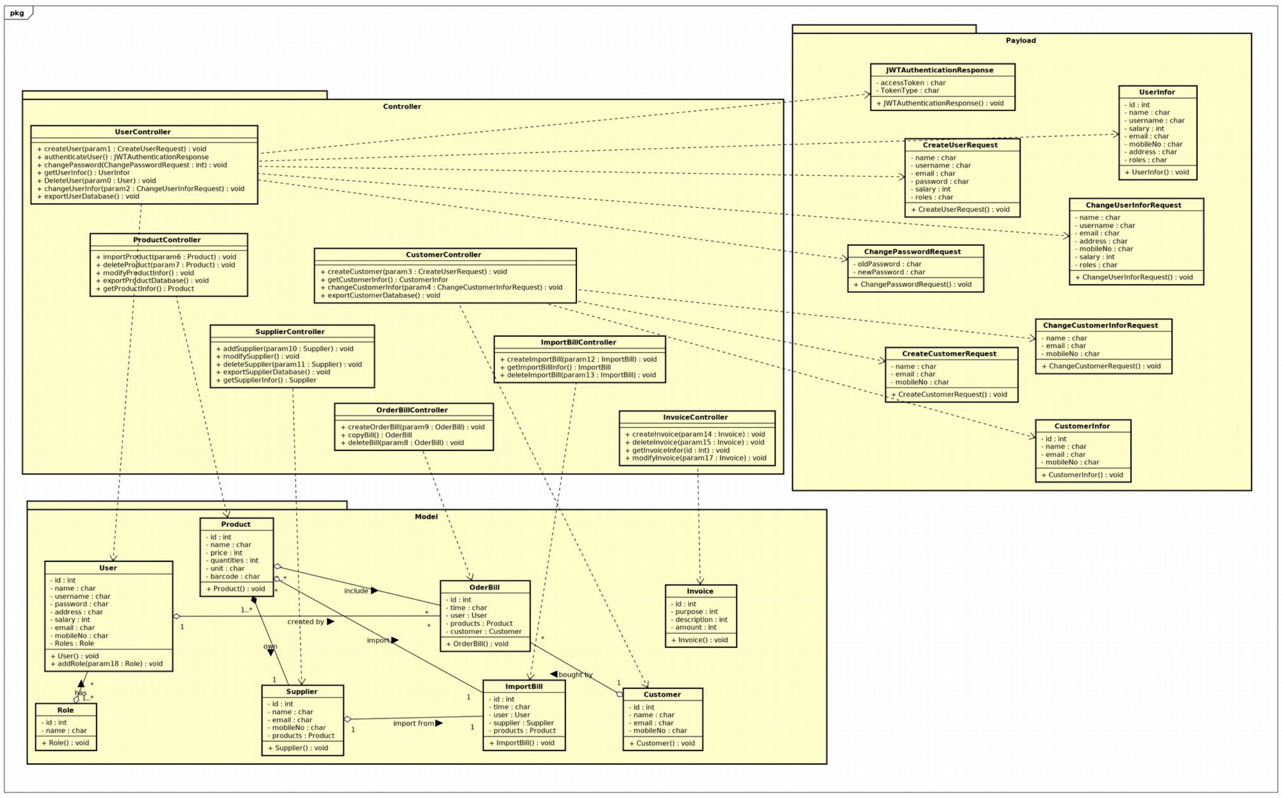
BEST SELLING

- **CHUỘT DELL 1201** (SOLD: 6)
- **BÀN PHÍM DELL 1202** (SOLD: 5)
- **VINASOLD SỮA ĐẬU NÀNH** (SOLD: 4)
- **NƯỚC LAVIE 1L** (SOLD: 3)
- **SỮA VINAMILK 350ML** (SOLD: 3)

Best selling is the list of best seller product by the selling quantity.

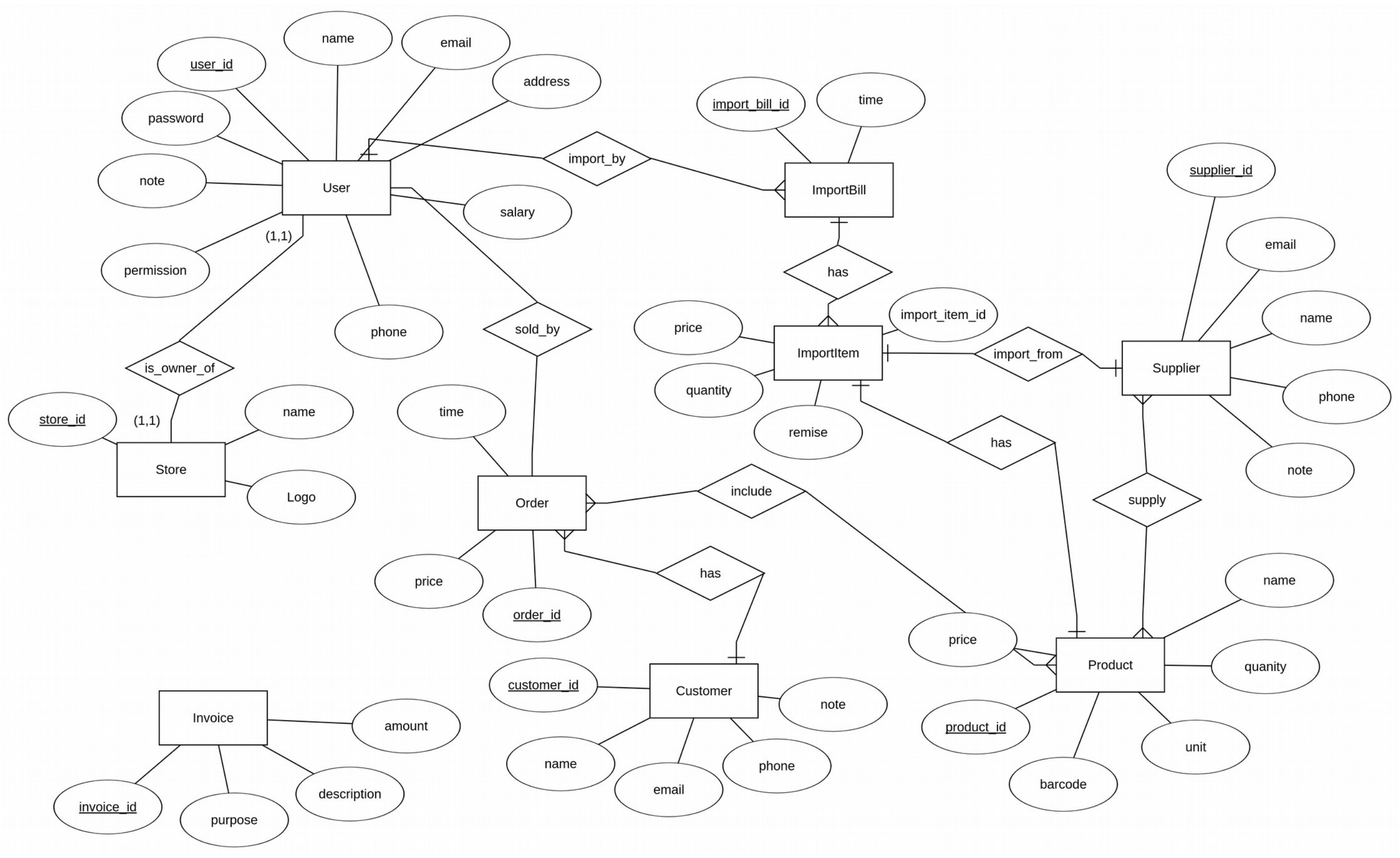
4 Class Design

4.1 Class Diagram

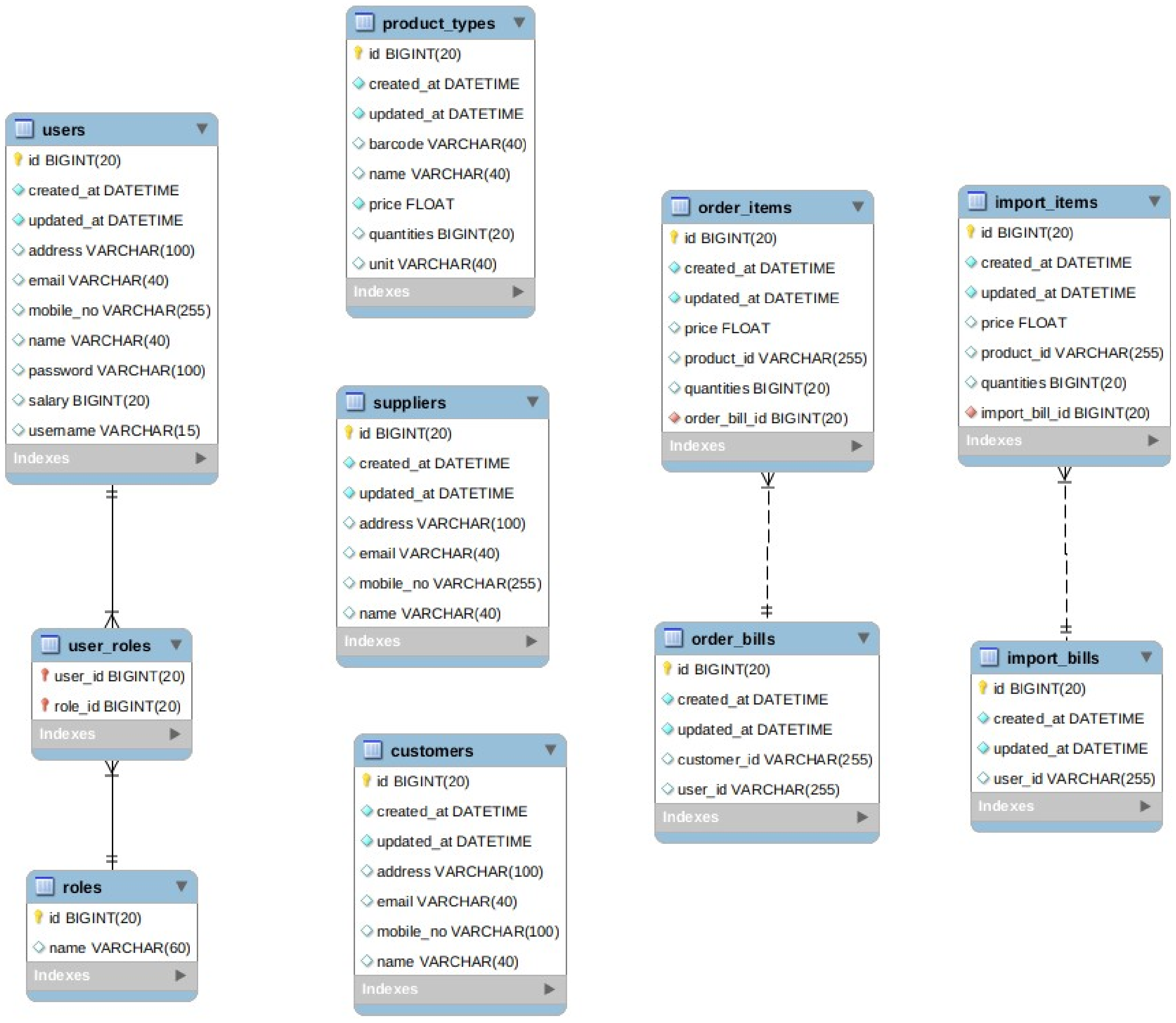


5 Data Model

5.1 Entity Relation Diagram



5.2 Logical Data Model



5.3 Detailed Design

5.3.1 Users

```
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| created_at  | datetime      | NO   |     | NULL    |                |
| updated_at  | datetime      | NO   |     | NULL    |                |
| address     | varchar(100)  | YES  |     | NULL    |                |
| email       | varchar(40)   | YES  | UNI | NULL    |                |
| mobile_no   | varchar(255)  | YES  |     | NULL    |                |
| name        | varchar(40)   | YES  |     | NULL    |                |
| password    | varchar(100)  | YES  |     | NULL    |                |
| salary      | bigint(20)    | YES  |     | NULL    |                |
| username    | varchar(15)   | YES  | UNI | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
```

5.3.2 Roles

```
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| name  | varchar(60)  | YES  | UNI | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
```

5.3.3 User_roles

```
+---------+------------+------+-----+---------+-------+
| Field   | Type       | Null | Key | Default | Extra |
+---------+------------+------+-----+---------+-------+
| user_id | bigint(20) | NO   | PRI | NULL    |       |
| role_id | bigint(20) | NO   | PRI | NULL    |       |
+---------+------------+------+-----+---------+-------+
```

5.3.4 Customers

```
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| created_at  | datetime      | NO   |     | NULL    |                |
| updated_at  | datetime      | NO   |     | NULL    |                |
| address     | varchar(100)  | YES  |     | NULL    |                |
| email       | varchar(40)   | YES  |     | NULL    |                |
| mobile_no   | varchar(100)  | YES  |     | NULL    |                |
| name        | varchar(40)   | YES  |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
```

5.3.5 Products

```
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| created_at | datetime    | NO   |     | NULL    |                |
| updated_at | datetime    | NO   |     | NULL    |                |
| barcode    | varchar(40) | YES  |     | NULL    |                |
| name       | varchar(40) | YES  |     | NULL    |                |
| price      | float       | NO   |     | NULL    |                |
| quantities | float       | YES  |     | NULL    |                |
| unit       | varchar(40) | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

5.3.6 Suppliers

```
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| created_at | datetime     | NO   |     | NULL    |                |
| updated_at | datetime     | NO   |     | NULL    |                |
| address    | varchar(100) | YES  |     | NULL    |                |
| email      | varchar(40)  | YES  |     | NULL    |                |
| mobile_no  | varchar(255) | YES  |     | NULL    |                |
| name       | varchar(40)  | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

5.3.7 Buys

```
+------------+------------+------+-----+---------+----------------+
| Field      | Type       | Null | Key | Default | Extra          |
+------------+------------+------+-----+---------+----------------+
| id         | bigint(20) | NO   | PRI | NULL    | auto_increment |
| created_at | datetime   | NO   |     | NULL    |                |
| updated_at | datetime   | NO   |     | NULL    |                |
| active     | bit(1)     | YES  |     | NULL    |                |
| user_id    | bigint(20) | YES  |     | NULL    |                |
+------------+------------+------+-----+---------+----------------+
```

5.3.8 Buy_items

```
+-------------+------------+------+-----+---------+----------------+
| Field       | Type       | Null | Key | Default | Extra          |
+-------------+------------+------+-----+---------+----------------+
| id          | bigint(20) | NO   | PRI | NULL    | auto_increment |
| created_at  | datetime   | NO   |     | NULL    |                |
| updated_at  | datetime   | NO   |     | NULL    |                |
| price       | float      | YES  |     | NULL    |                |
| product_id  | bigint(20) | YES  |     | NULL    |                |
| quantities  | float      | YES  |     | NULL    |                |
| supplier_id | bigint(20) | YES  |     | NULL    |                |
| buy_id      | bigint(20) | NO   | MUL | NULL    |                |
+-------------+------------+------+-----+---------+----------------+
```

5.3.9 Sells

```
+-------------+------------+------+-----+---------+----------------+
| Field       | Type       | Null | Key | Default | Extra          |
+-------------+------------+------+-----+---------+----------------+
| id          | bigint(20) | NO   | PRI | NULL    | auto_increment |
| created_at  | datetime   | NO   |     | NULL    |                |
| updated_at  | datetime   | NO   |     | NULL    |                |
| active      | bit(1)     | YES  |     | NULL    |                |
| customer_id | bigint(20) | YES  |     | NULL    |                |
| tax         | float      | YES  |     | NULL    |                |
| user_id     | bigint(20) | YES  |     | NULL    |                |
| total       | float      | YES  |     | NULL    |                |
+-------------+------------+------+-----+---------+----------------+
```

5.3.10 Sell_items

```
+------------+------------+------+-----+---------+----------------+
| Field      | Type       | Null | Key | Default | Extra          |
+------------+------------+------+-----+---------+----------------+
| id         | bigint(20) | NO   | PRI | NULL    | auto_increment |
| created_at | datetime   | NO   |     | NULL    |                |
| updated_at | datetime   | NO   |     | NULL    |                |
| price      | float      | YES  |     | NULL    |                |
| product_id | bigint(20) | YES  |     | NULL    |                |
| quantities | float      | YES  |     | NULL    |                |
| sell_id    | bigint(20) | NO   | MUL | NULL    |                |
+------------+------------+------+-----+---------+----------------+
```

5.3.11 Invoices

```
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| created_at  | datetime     | NO   |     | NULL    |                |
| updated_at  | datetime     | NO   |     | NULL    |                |
| amount      | float        | YES  |     | NULL    |                |
| description | varchar(255) | YES  |     | NULL    |                |
| purpose     | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```