POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY 1

FINAL REPORT

INFORMATION SYSTEM ANALYSIS AND DESIGN

SUBJECT NO.25

Student Name : **Phạm Anh Minh**
Student ID : **B22DCAT192**
Class : **E22HTTT**

Hà Nội – 11/2025

Subject No. 25

Duration: 90 minutes

A restaurant management system enables management staff, sale staff and customers to use:

- Management staff: view statistics: dishes, ingredients, customers and suppliers. Manage dish information, make combo menus.
- Warehouse staff: import materials from suppliers, manage supplier information
- Sale staff: receive customers, take orders, receive payments at the table, make membership cards for customers, confirm table reservation information and order online.
- Customer: search, book a table and order food online.

Considering two modules and answering questions

- ***Management staff edits dish information***: selects the menu to manage dish information → selects the function to edit dish information → searches for dishes by name to edit → selects dishes from the list of results to edit → enters information dishes and clicks save → saves to the database.
- ***Management staff view supplier statistics***: selects the menu to view the statistical report → selects the supplier statistics according to the imported quantity → selects the start and end time of statistics → views supplier statistics → selects a supplier view details → views list of imported invoice of the supplier → selects view 1 imported invoice → views the detail of selected imported invoice.

Question 1 (2 points)

a. Build the use case diagram for the two modules

b. Present scenarios for the two modules

Question 2 (2 points)

a. Extract the entity classes (class name, attributes) related to the two modules

b. Build the entity class diagram for the extracted classes.

Question 3 (2 points)

a. Build the communication diagram for the standard scenario of the two modules.

b. Build the detail design class diagram for the two modules.

Question 4 (2 points)

a. Based on the entity classes of Question 2, design the database related to the two modules

b. Generate Java code (class template, declaration of attributes and methods, explain how methods work) for classes in the diagram of Question 3.b.

Question 5 (2 points)

a. Build the package diagram for classes in the diagram of Question 3.b.

b. Build the deployment diagram for MVC model based on the J2EE technology.

A. Specification

I. Keyword table

Actor	Activities	Object
Staff Management Staff Warehouse Staff Sale Staff Customer Supplier	View statistics Manage dish information Edit dish information View supplier statistics Make combo menu Import ingredients Manage supplier information Receive customer Take orders Receive payment Make membership card Confirm table reservation Order online Search table Book table online	Dish Ingredient Combo Menu Order Payment Table Membership card Imported Invoice Reservation Invoice

II. Glossary list

No.	Name	Meaning
Related to actor		
1	Staff	A person that works in the restaurant and has an account to use the system
2	Management Staff	A person with the highest authority, including viewing statistics, managing dish information, and making combo menus
3	Warehouse Staff	A person responsible for importing ingredients from suppliers and managing supplier information
4	Sale Staff	A person who receives customers, takes orders, receives payments at the table, makes membership cards for customers, and confirms table reservation information and online orders
5	Customer	A person who benefits from the service, can search, book a table, and order food online
6	Supplier	A partner providing ingredients

Related to activity		
7	View statistics	An action of manager staff that involves viewing the information of dishes, ingredients, customers, and suppliers
8	Manage dish information	An action of manager staff that includes adding a new dish, or changing or deleting existing dish information (name, description, price) in the database
9	Make combo menu	An action of manager staff that combines two or more dishes into a combo menu (name, description, list of dishes, price) and inserts its information into the database
10	Import ingredients from suppliers	An action of warehouse staff that confirms all the ingredients brought from suppliers and inserts new ingredients or updates the stock of existing ingredients in the database
11	Manage supplier information	An action of warehouse staff that includes adding new supplier information (name, phone, email) and changing or deleting existing supplier information
12	Receive customer	An action of sale staff that involves meeting a customer at the reception hall, checking their information, and guiding them
13	Take orders	An action of sale staff that involves taking note of a customer's orders (list of dishes, beverages) face-to-face and informing the chefs
14	Receive payment at the table	An action of sale staff that involves receiving a customer's payment after they finish their meal and printing a bill
15	Make membership card for customer	An action of sale staff that involves creating and issuing a card for a customer to earn loyalty points
16	Confirm table reservation information	An action of sale staff to check a customer's reservation details (name, phone, table) when they arrive at the restaurant
17	Order online	An action of sales staff. The customer calls, and the staff checks available tables and saves the reservation
18	Add new dish	An action of management staff that is included in the "managing dish information" action. The staff fills in the dish information (name, description, price) to be saved into the database
19	Search table	An action of the customer to search for an available table at a specific time online
20	Edit dish information	**(Module 1)** An action of Management staff, part of "Manage dish information". It includes searching for a dish by name, selecting the dish, modifying its information (name,

		description, price), and saving the changes.
21	View supplier statistics	**(Module 2)** An action of Management staff. It includes selecting a time period to view statistics on suppliers based on imported quantity, viewing a list of suppliers, and drilling down to see detailed import invoices.
Related to object		
22	Dish	A specific type of food that has been prepared and is ready to be served. Its information will be saved into the database (id, name, description, price)
23	Ingredient	A specific food item used to make a dish. Its information will be saved into the database (id, name, price, stock)
24	Combo	Short for "combination," a package deal that bundles several individual items together. Its information will be saved into the database (id, name, list of dishes, price)
25	Menu	A list of the food and beverages that are available for purchase
26	Order	A request made by a customer to a restaurant for food and drinks
27	Payment	The process of a customer settling the cost of their order
28	Table	A physical location in the restaurant where customers are seated
29	Membership card	A physical or digital card issued by the restaurant to its customers, linked to a loyalty account
30	Imported Invoice	**(Module 2)** A document or record generated when ingredients are imported from a supplier, detailing the supplier, date, ingredients, quantities, and price.
31	Reservation	An arrangement made by a customer in advance to book a specific table at a specific time.
32	Invoice	A document (bill) given to a customer detailing the dishes, combos, and total price for their order. It confirms the sales transaction.

II. System description

1. System purpose

The Restaurant Management System is a web-based application designed to support and manage the operational activities of a restaurant. This includes:

- Managing dish information, ingredients, schedules, and suppliers.
- Processing sales and orders for customers.

- Providing comprehensive statistical reports on dishes, ingredients, customers, and suppliers.

The system supports both in-house customer service and online functionalities such as table booking and food ordering.

2. System scope

The users allowed to access the system and the functions they can perform are defined as follows:

2.1. Staff (General):

- Login to the system.
- Logout from the system.
- Change the password.

2.2. Customer:

- Search for dishes and tables.
- Book a table online.
- Order food online.

2.3. Sale staff:

- Receive customers.
- Take orders at the table.
- Receiving payments at the table.
- Create and issue membership cards for customers.
- Confirm table reservation information.
- Process online orders.

2.4. Warehouse staff:

- Import ingredients from suppliers.
- Manage supplier information (add, edit, delete).

2.5. Management staff:

- View statistics (dishes, ingredients, customers, suppliers).
- Manage dish information (add, edit, delete).
- Create and manage combo menus.

3. Business operations

Module 1: Management staff edits dish information

The manager selects the menu to manage dish information. → The manager selects the function to edit dish information. → The manager searches for dishes by name to edit. → The manager selects a dish from the list of results to edit. → The manager enters the new information for the dish and clicks “save”. → The system saves the new information to the database.

Module 2: Management staff view supplier statistics

The manager selects the menu to view the statistical report. → The manager selects the “supplier statistics” according to the “imported quantity” criteria. → The manager selects the start and end time for the statistics. → The system shows the supplier statistics (list of suppliers). → The manager selects a supplier from the list to view details. → The system shows the list of imported invoices for that supplier. → The manager selects one imported invoice to view. → The system shows the details of the selected imported invoice (e.g., the ingredients included).

4. Information about objects

4.1. User information:

- Staff: username, password, name, phone, email, role (Management, Warehouse, Sale).

- Customer: customerID, name, phone, email.
- Membership: cardID, customer, type, points, startDate, dueDate.

4.2. Restaurant Entities:

- Dish: dishID, name, description, price.
- Combo: comboID, name, price, description, listofDishes.
- Table: tableID, type, available

4.3. Warehouse and Supplier Entities:

- Supplier: supplierID, name, phone, email, address.
- Ingredients: ingredientID, name, stock, price.
- Imported Invoice: importInvoiceID, warehouseStaff, supplier, importDate, totalPrice.
- ImportedInvoiceDetail (association class): importInvoice, ingredient, quantity, price.

4..4. Transaction Entities:

- Reservation: reservationID, customer, table, reservationTime, status
- Order: orderID, reservation, date, price.
- Invoice: invoiceID, reservation, order, saleStaff, paymentDate, status.

4.5. Statistical Information:

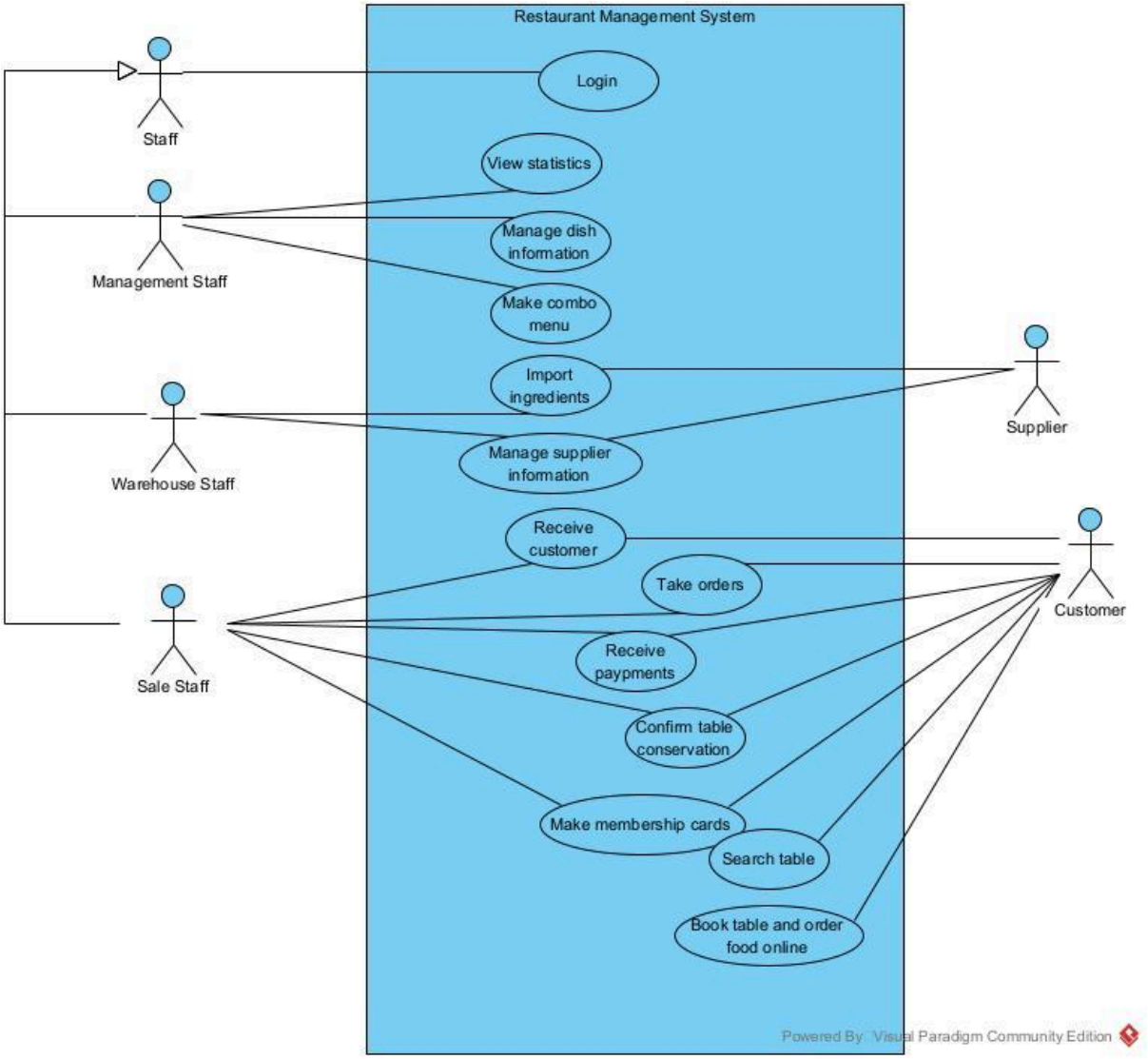
- SupplierStatistic: supplierName, totalImportedQuantity, totalCost.

5. Relationships among objects

- Customer - Membership: 1 - 1
- Customer - Reservation: 1 - n
- Reservation - Order: 1 - n
- Reservation - Table: n - 1
- Reservation - Invoice: 1 - n
- Order - Invoice: n - 1
- Order - Dishes: n - n
- Order - Combo: n - n
- Dish - Combo: n - n
- Sale Staff - Invoice: 1 - n
- Supplier - Imported Invoice: 1 - n
- Warehouse Staff - Imported Invoice: 1 - n
- Imported Invoice - Ingredient: n - n
- Management Staff - Dishes: 1 - n

IV. Use Case Diagram

1. General Use Case Diagram



General use case diagram

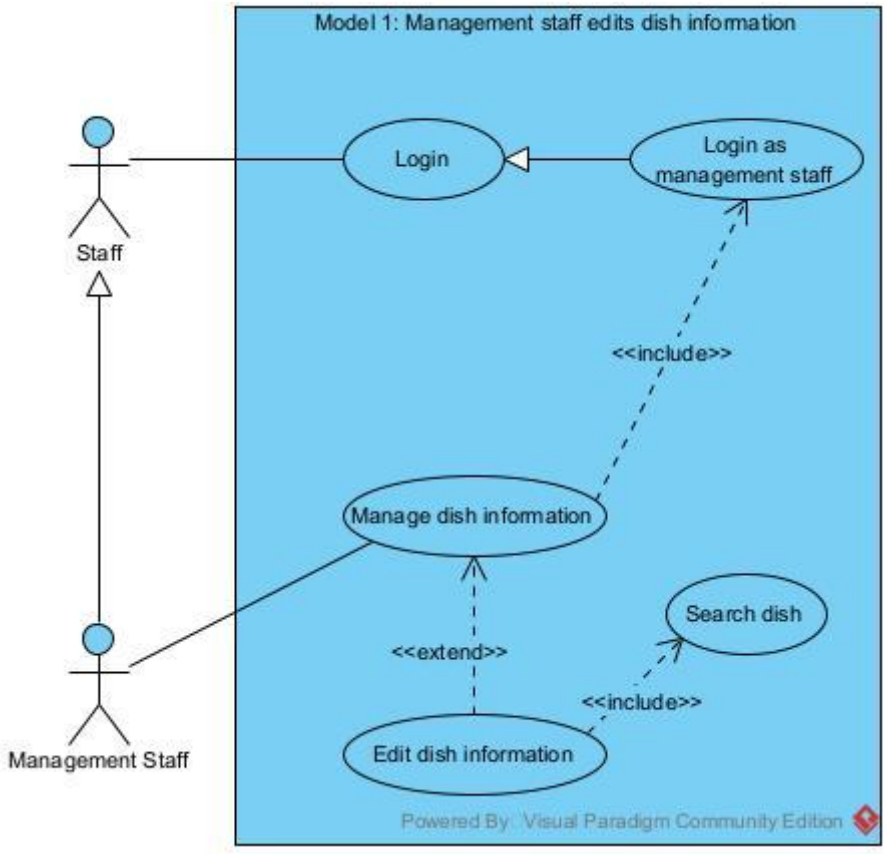
Descriptions:

- Login: This use case allows all Staff members to log in to the system with their accounts.
- View statistics: This use case allows the Management Staff to view various statistical reports (e.g., on dishes, customers, suppliers).
- Manage dish information: This use case allows the Management Staff to add, edit, and delete dish information in the system.
- Make combo menus: This use case allows the Management Staff to create and manage special combo offers.
- Import ingredients: This use case allows the Warehouse Staff to record new ingredients received from a Supplier.
- Manage supplier information: This use case allows the Warehouse Staff to add, edit, or delete supplier details.

- Receive customer: This use case allows the Sale Staff to greet and check in a Customer.
- Take orders: This use case allows the Sale Staff to take a Customer's food and drink order at the table.
- Receive payments: This use case allows the Sale Staff to process a Customer's payment and print an invoice.
- Confirm table reservation: This use case allows the Sale Staff to check and confirm a Customer's existing booking.
- Make membership cards: This use case allows the Sale Staff to create a new membership account and issue a card to a Customer.
- Search table: This use case allows a Customer to search for available tables online.
- Book table and order food online: This use case allows a Customer to create a new reservation and pre-order food through the website.

2. Detailed Use Case Diagram

2.1. Module 1: Management staff edits dish information



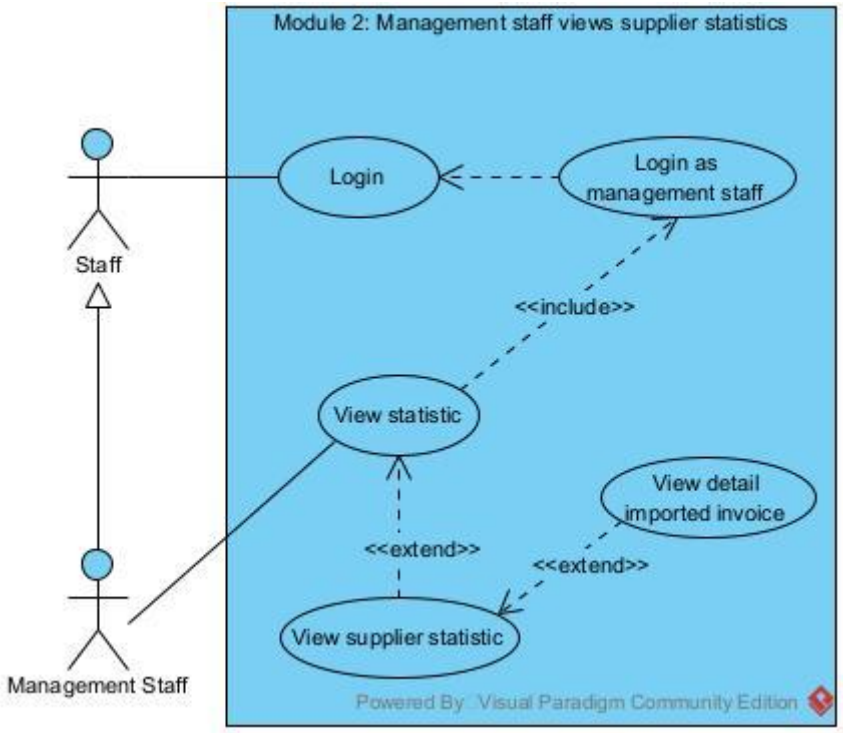
Use Case Diagram for Module 1

Descriptions:

- Login: This use case allows all Staff members to log in to the system with their accounts.
- Login as management staff: This use case is a specialization of Login that grants access to management-level functions.
- Manage dish information: This use case is the main function for managing all dishes. It <<includes>> logging in as a manager.

- Edit dish information: This use case <<extends>> the "Manage dish information" function, allowing the manager to modify the details of an existing dish.
- Search dish: This use case is <<included>> by the "Edit dish information" process, as the manager must first find the dish they want to edit.

2.2. Module 2: Management staff view supplier statistics



Use Case Diagram for Module 2

Descriptions:

- Login: This use case allows all Staff members to log in to the system with their accounts.
- Login as management staff: (Same as above)
- View statistic: This use case is the main dashboard for viewing all reports. It <<includes>> logging in as a manager.
- View supplier statistic: This use case <<extends>> the "View statistic" function, allowing the manager to see a specific report on suppliers based on imported quantity.
- View detail imported invoice: This use case <<extends>> the "View supplier statistic" function, allowing the manager to optionally drill down to see the contents of a single invoice.

B. Analysis phase

I. Entity Class Diagram

1. System description

The Restaurant Management System is a web application that supports the management of restaurant operations. This includes managing information about dishes, combos, ingredients, and suppliers. The Management Staff can manage dish information and view statistics. The Warehouse Staff can import ingredients and manage supplier information. The Sale Staff is responsible for taking orders, confirming reservations, issuing membership cards, and processing payments at the table. Customers can book tables and order food online. The system generates Invoices for customer sales and Imported Invoices for supplier imports.

2. Extract nouns

- Nouns related to Actors: Management Staff, Warehouse Staff, Sale Staff, Customer, Supplier.
- Nouns related to Objects/Entities: dishes, combos, ingredients, table, membership cards, Imported Invoices, Invoices.
- Nouns related to Information/Transactions: information, statistics, orders, reservations, payments.

3. Evaluate and select Entity Classes

a. General nouns (Eliminated):

- information: Too general.
- statistics: This is a function, not a single data entity. (Note: We will have a SupplierStat class later, but it's a result of this function, not the function itself).
- payments: This is an action. The entity that records this action is the Invoice.

b. Nouns related to Actors (Kept as Entities):

- Staff (Proposed as an abstract parent class for ManagementStaff, WarehouseStaff, SaleStaff).
- Customer (Kept).
- Supplier (Kept).

c. Nouns related to Objects/Entities (Kept as Entities):

- Dish (Kept).
- Combo (Kept).
- Ingredient (Kept).
- Table (Kept).
- Membership (Kept, renamed from membership cards).
- ImportedInvoice (Kept).
- Invoice (Kept).
- Reservation (Kept).
- Order (Kept).

d. Proposed Association Classes (from n-n relationships):

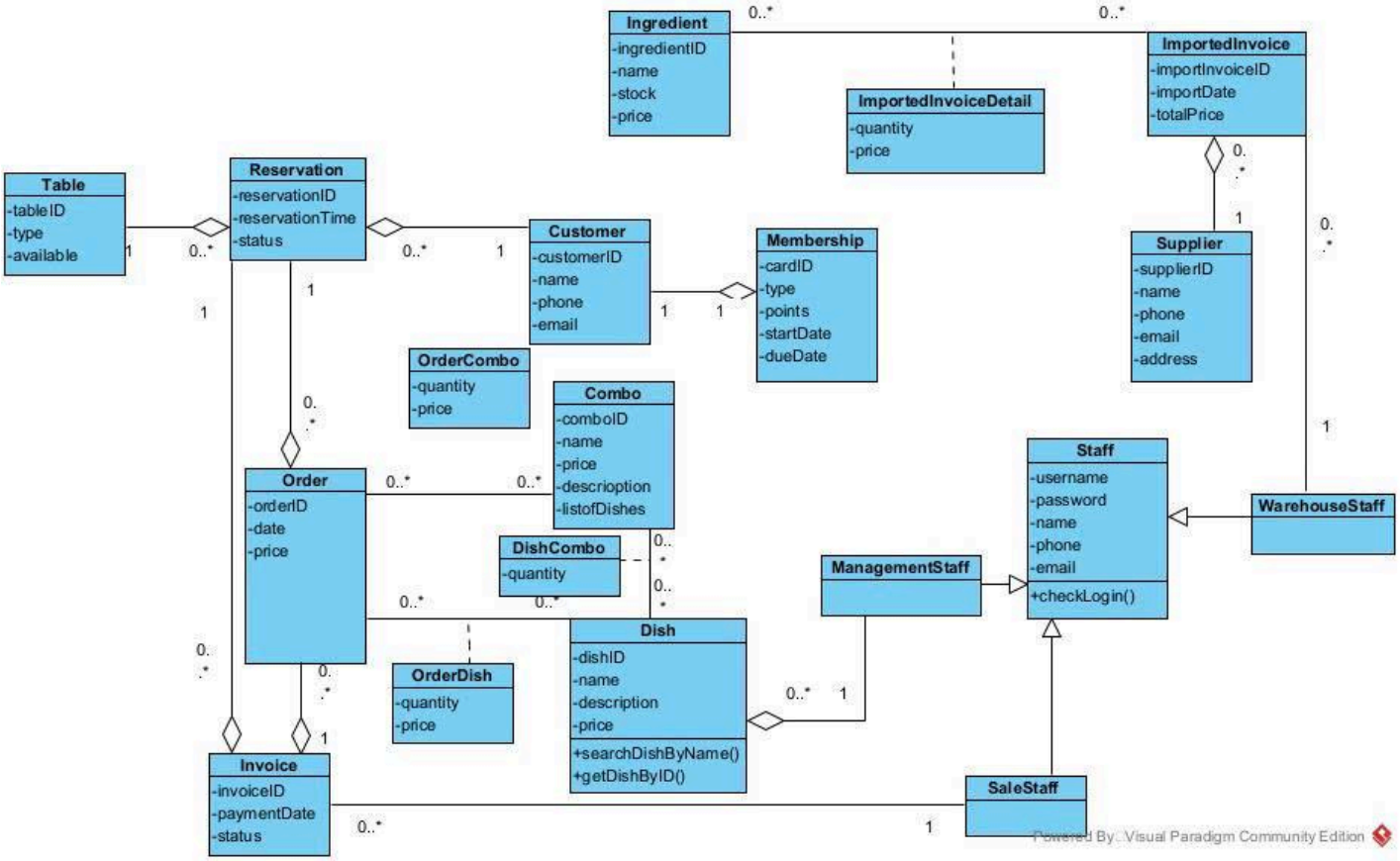
- Order (n-n) Dish ➔ Propose OrderDish.
- Order (n-n) Combo ➔ Propose OrderCombo.
- Dish (n-n) Combo ➔ Propose DishCombo.
- ImportedInvoice (n-n) Ingredient ➔ Propose ImportedInvoiceDetail.

4. Determine Relationships

- Customer - Membership: 1 - 1

- Customer - Reservation: 1 - n
- Reservation - Order: 1 - n
- Reservation - Table: n - 1
- Reservation - Invoice: 1 - n
- Order - Invoice: n - 1
- Order - Dishes: n - n
- Order - Combo: n - n
- Dish - Combo: n - n
- Sale Staff - Invoice: 1 - n
- Supplier - Imported Invoice: 1 - n
- Warehouse Staff - Imported Invoice: 1 - n
- Imported Invoice - Ingredient: n - n
- Management Staff - Dishes: 1 - n
- The ImportedInvoice - Ingredient relationship is linked via ImportedInvoiceDetail.
- The Order - Dish relationship is linked via OrderDish.
- The Order - Combo relationship is linked via OrderCombo.
- The Dish - Combo relationship is linked via DishCombo.

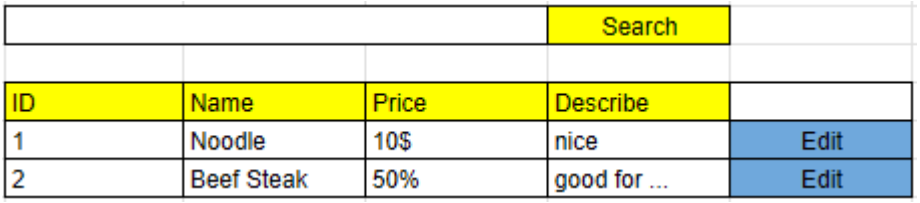
5. Result

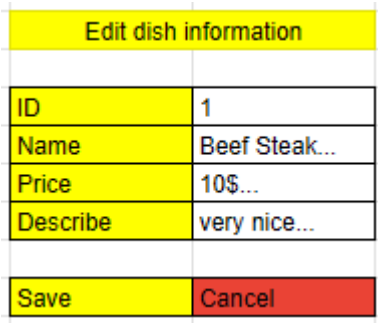


Entity Class Diagram

II. Module 1: Management staff edits dish information

1. Scenario

<table>
<tr><td>Use case</td><td>Edit dish information</td></tr>
<tr><td>Actor</td><td>Management staff</td></tr>
<tr><td>Pre-condition</td><td>The management staff has successfully logged into the system.
The management staff has permission to access the “Manage dish information” function.</td></tr>
<tr><td>Post-condition</td><td>The information for the selected dish is updated in the database.</td></tr>
<tr><td>Main events</td><td>1. After login, the management staff selects the "Manage dish information” function from the main menu.
2. The system shows the dish management interface.
 

3. The management staff enters a dish name and click “Search”.
4. The system refreshes the list to shows only the matching results.
5. The management staff clicks the “Edit” button.
6. The system shows the “Edit dish” interface, with all fields…, pre-filled…
 

7. The management staff modifies the information… and clicks the “Save” button.
8. The system validates the data, saves the changes to the database, and shows a "Update successful" message on the EditDishView interface.</td></tr>
<tr><td>Exception</td><td>Step 4: If no dish matches the search term.
4.1. The system shows the message “No dish found”.
Step 8: If the new information is invalid.
8.1. (Invalid data) If the price is not a number (e.g., "abc"). The system shows an error message (e.g., "Invalid price. Please enter a number.") and remains on the EditDishView interface.
8.2. (Duplicate data) If the manager changes the name to a name that already exists in the database (e.g., renames "Pho Bo" to "Com Rang",</td></tr>
</table>

	but "Com Rang" already exists). The system shows an error message (e.g., "A dish with this name already exists.") and remains on the EditDishView interface.

2. Class Diagram

a. Boundary Classes:

- LoginView: (Identified from checkLogin() step) .
- ManagementHomeView: (Identified from "select Manage dish" step) .
- ManageDishView: (Identified from "search dish" step) This class is proposed to hold the search controls and results list.
- EditDishView: (Identified from "enter information" step) This class is proposed to hold the form for editing.

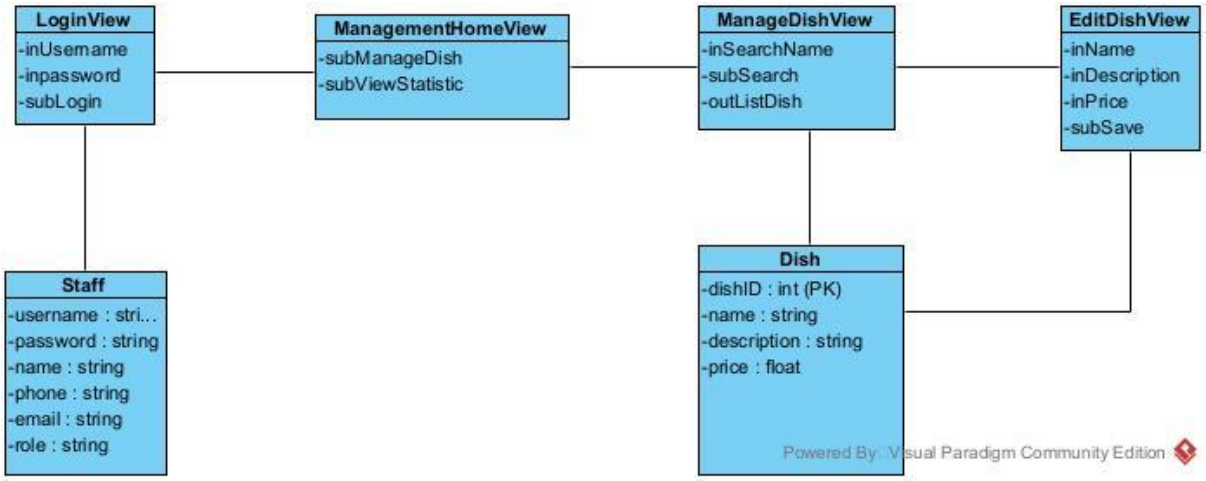
b. Entity Classes:

- Staff: (Identified from checkLogin() step).
- Dish: (Identified from "search," "edit," and "save" steps).

c. Control logic (Assigned methods):

- +checkLogin() is assigned to Staff.
- +searchDishByName(name): Dish[]
- +getDishByID(id): Dish
- +updateDish(dish): boolean

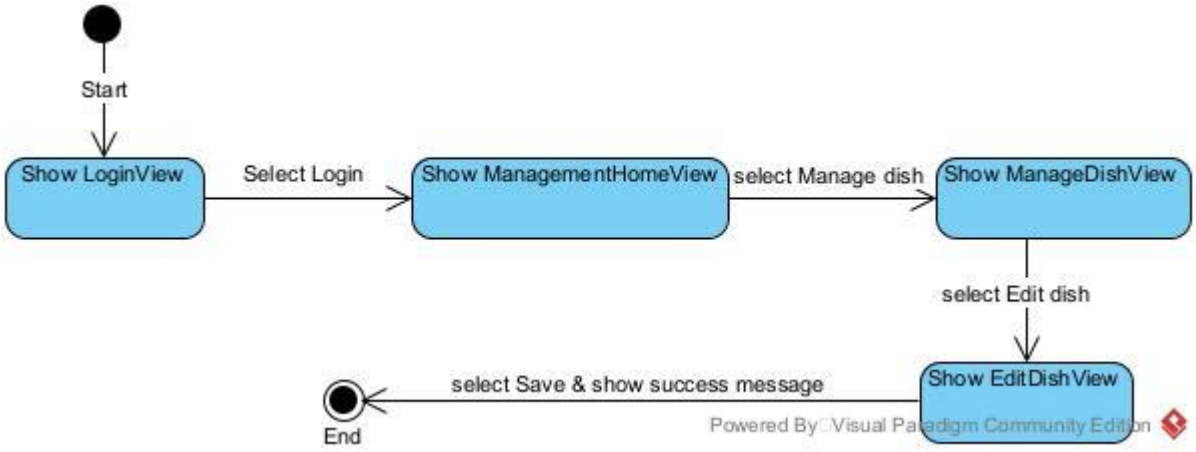
d. Result:



Class Diagram for Module 1

3. State Diagram

- From Show LoginView, after the user enters username/password and selects "Login", the system will move to Show ManagementHomeView.
- From Show ManagementHomeView, selecting the "select Manage dish" option will move the system to Show ManageDishView.
- From Show ManageDishView, after the user searches and selects an item to "select Edit dish", the system will show the Show EditDishView (this replaces the show AddDishView step).
- "From Show EditDishView, after the user enters the information and selects "Save", the system will show the success message (within the same view) and then End the flow."

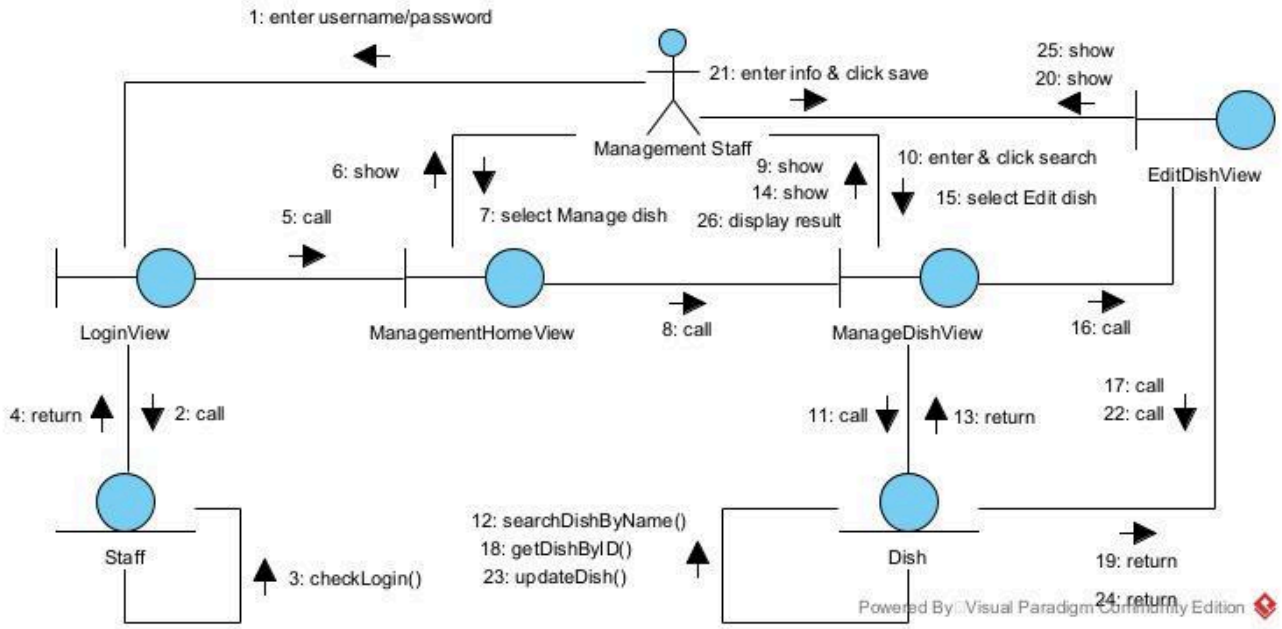


State Diagram for Module 1

4. Scenario ver 2

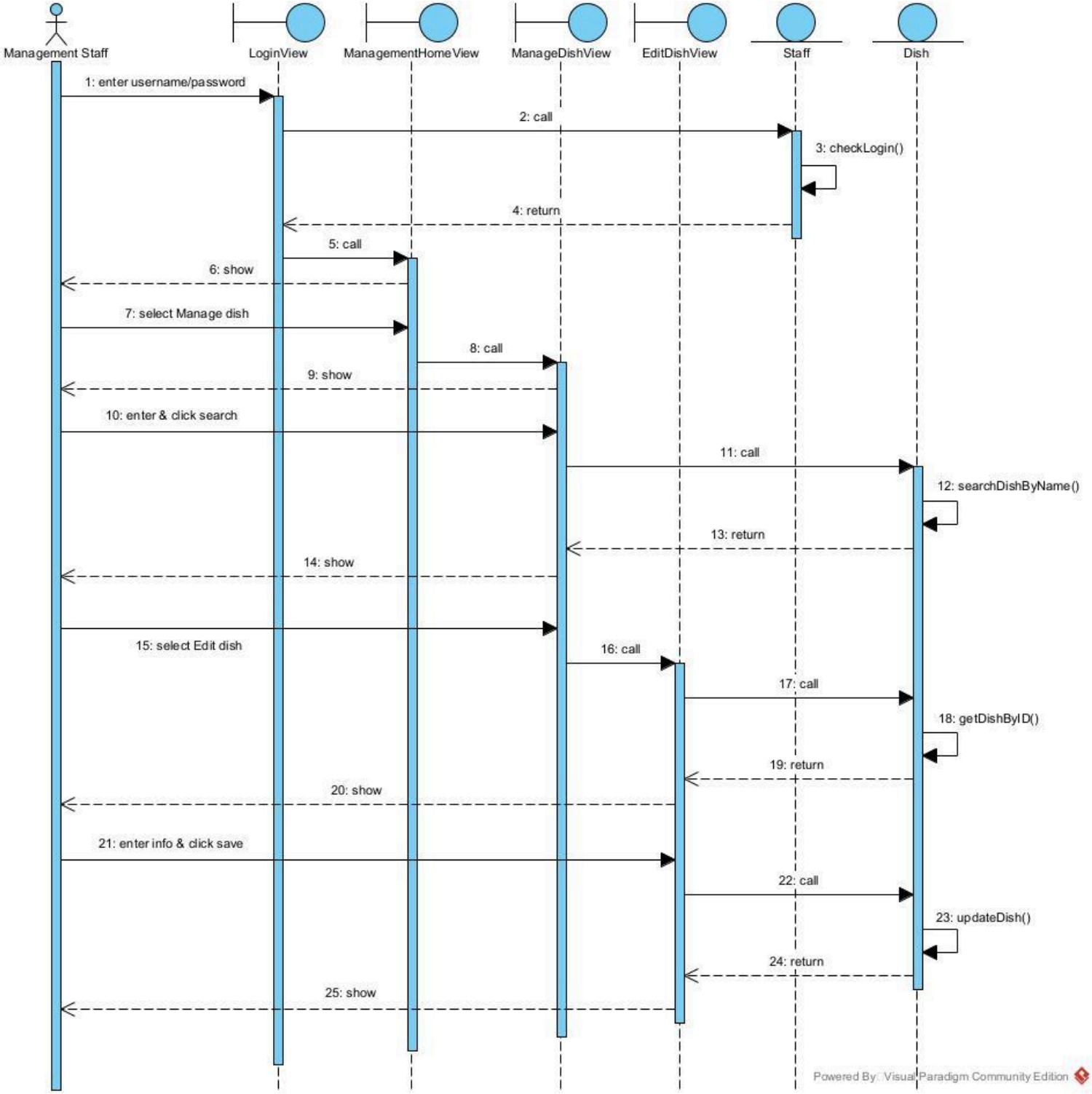
1. The management staff enters username/password on LoginView and clicks the "Login" button.
2. The class LoginView calls the class Staff to process.
3. The class Staff calls the method checkLogin().
4. The class Staff returns the results (Staff object) to the class LoginView.
5. The class LoginView calls the class ManagementHomeView.
6. The class ManagementHomeView shows itself to the management staff.
7. The management staff clicks on the "Manage dish" button.
8. The class ManagementHomeView calls the class ManageDishView.
9. The class ManageDishView shows itself (showing the search form).
10. The management staff enters a name and clicks the "Search" button.
11. The class ManageDishView calls the class Dish to process.
12. The class Dish calls the method searchDishByName().
13. The class Dish returns the list of Dish objects to the class ManageDishView.
14. The class ManageDishView shows the search results to the staff.
15. The management staff selects a dish and clicks the "Edit" button.
16. The class ManageDishView calls the class EditDishView.
17. The class EditDishView calls the class Dish to get data.
18. The class Dish calls the method getDishByID().
19. The class Dish returns the selected Dish object to EditDishView.
20. The class EditDishView shows itself, with the form fields populated.
21. The management staff enters the new information (name, description, price) and clicks "Save".
22. The class EditDishView calls the class Dish to process.
23. The class Dish calls the method updateDish().
24. The class Dish returns the result to the class EditDishView.
25. The class EditDishView shows a success message to the management staff.

5. Communication Diagram



Communication Diagram for Module 1

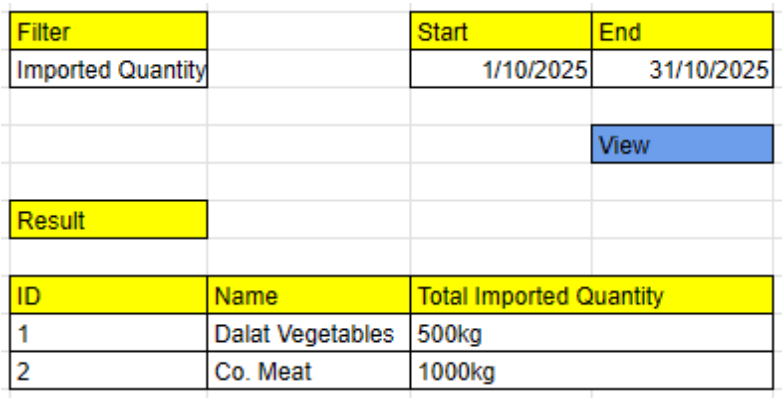
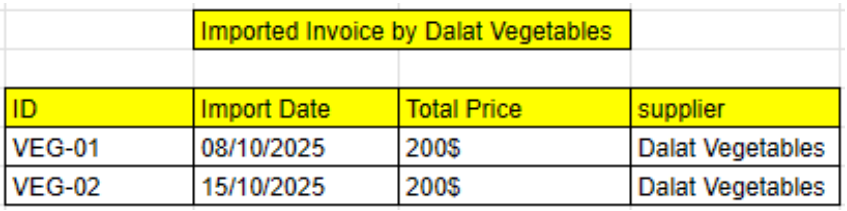
6. Sequence Diagram

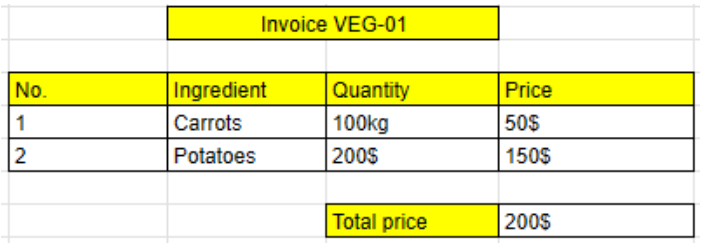


Sequence Diagram for Module 1

III. Module 2: Management staff view supplier statistics

1. Scenario

Use case	View supplier statistics
Actor	Management staff
Pre-condition	The management staff has successfully logged into the system. The management staff has permission to access the “View statistics” function.
Post-condition	The management staff has viewed the desired statistical report.
Main events	1. After login, the management staff selects “View statistics” from the main menu. 2. The system shows the statistics dashboards. The manager clicks “Supplier Statistics”. 3. The system shows an interface with filter options. The manager selects the criteria "according to the imported quantity", enters start time = "01/10/2025" and end time = "31/10/2025", and clicks "View". 4. The system shows a list of suppliers ranked by imported quantity.  Filter: Imported Quantity; Start: 1/10/2025; End: 31/10/2025; View Result ID | Name | Total Imported Quantity 1 | Dalat Vegetables | 500kg 2 | Co. Meat | 1000kg 5. The management staff clicks on a supplier from the list (e.g., "Dalat Vegetables"). 6. The system shows a new list showing all "Imported Invoices" from that supplier.  Imported Invoice by Dalat Vegetables ID | Import Date | Total Price | supplier VEG-01 | 08/10/2025 | 200$ | Dalat Vegetables VEG-02 | 15/10/2025 | 200$ | Dalat Vegetables 7. The management staff clicks on an invoice (e.g., "1"). 8. The system shows the details of that invoice, including a list of all imported ingredients.

	
Exception	Step 4: (No suppliers found) No suppliers have imported data in this period. 4.1. The system shows an empty list with the message "No supplier data found for this period." Step 6: (No invoices found) The selected supplier has no imported invoices in this period. 6.1. The system shows an empty list with the message "No invoices found for this supplier." Step 8: (Invoice is empty) The selected invoice has no ingredient details (e.g., data entry error). 8.1. The system shows an empty list with the message "This invoice contains no ingredient details."

2. Class Diagram

a. Boundary Classes:

- ManagementHomeView: (Identified from "select View statistics" step).
- SelectStatisticView: (Identified from "select Supplier Statistics" step).
- SupplierStatView: (Identified from "enter start and end time" step).
- ListImportInvoiceView: (Identified from "select a supplier" step)
  ImportInvoiceDetailView: (Identified from "select an invoice" step).

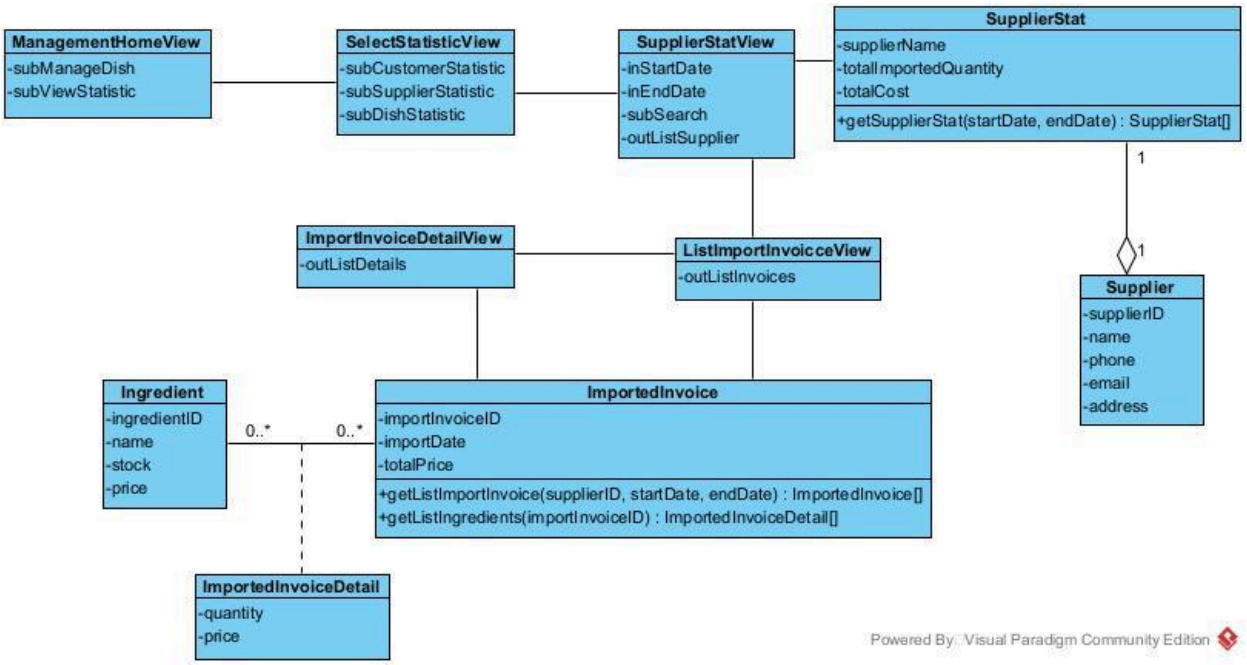
b. Entity Classes:

- Supplier: Identified from the "selects a supplier" step.
- SupplierStat: Identified from the "views supplier statistics" step. This class is proposed to hold the computed results of the statistical report (e.g., supplierName, totalImportedQuantity).
- ImportedInvoice: Identified from the "views list of imported invoice" step.
- ImportedInvoiceDetail: Identified from the "views the detail of selected imported invoice" step. This is the association class that links an invoice to its specific ingredients.
- Ingredient: Identified from the "views the detail... (e.g., the ingredients included)" step.

c. Control logic (Assigned methods):

- +getSupplierStat() is assigned to SupplierStat.
- +getListImportInvoice() and +getListIngredients() are assigned to ImportedInvoice.

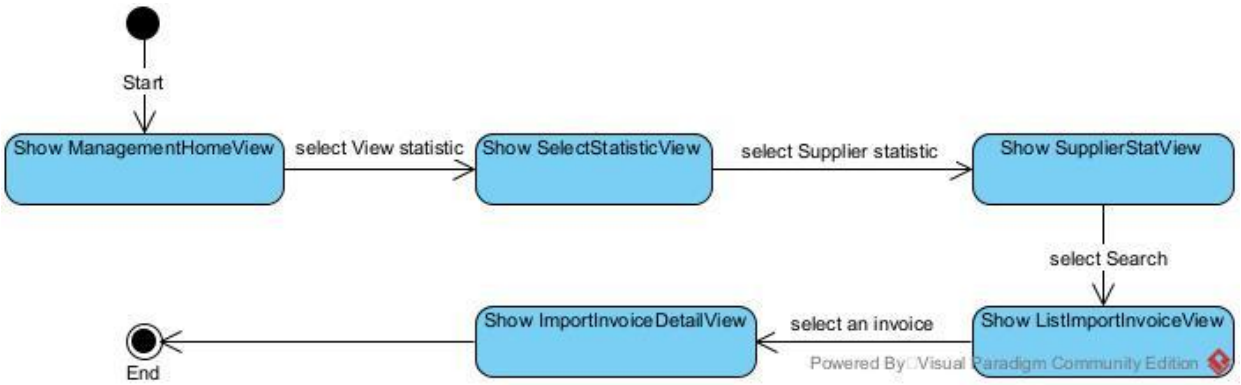
d. Result:



Class Diagram for Module 2

3. State Diagram

- From Show ManagementHomeView, if the "select View statistic" option is chosen, the system will show Show SelectStatisticView.
- From Show SelectStatisticView, if the "select Supplier statistic" button is chosen (replacing "Customer statistic"), the system will show the Show SupplierStatView.
- From Show SupplierStatView, after the user enters the start/end date and clicks "select Search", the system will show Show ListImportInvoiceView (replacing ListOrderView).
- From Show ListImportInvoiceView, after the user "select an invoice" (replacing "select an order"), the system will show Show ImportInvoiceDetailView (replacing OrderStatView) and End the flow.



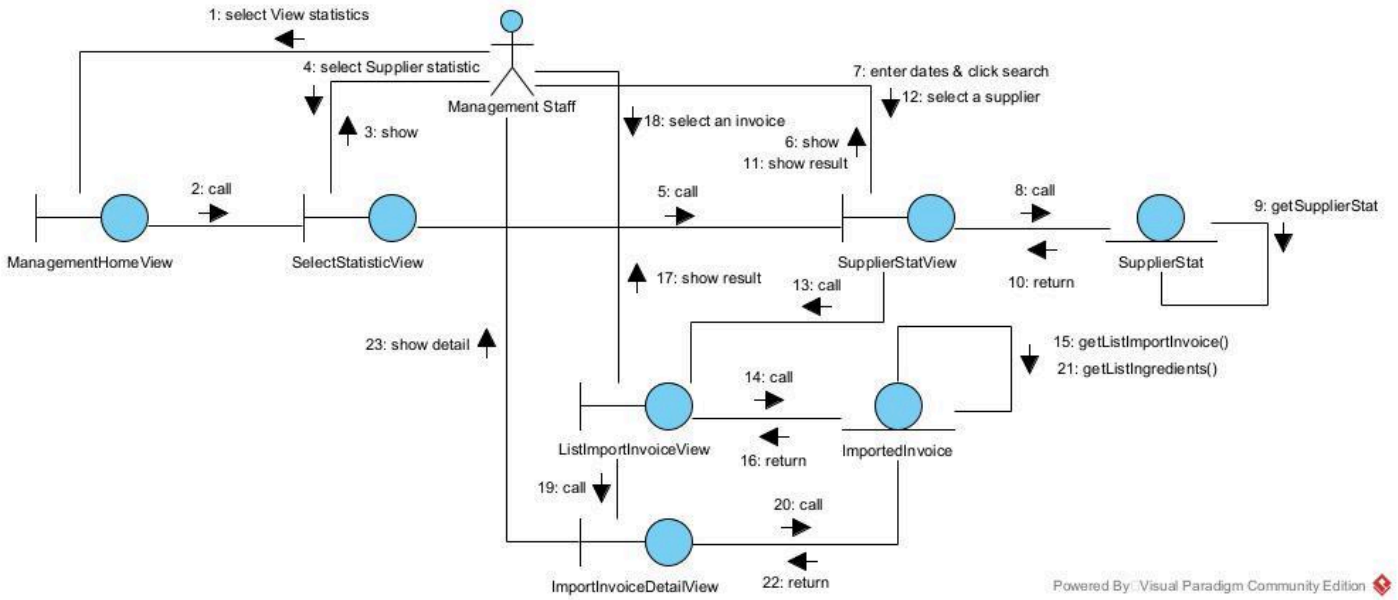
State Diagram for Module 2

4. Scenario ver 2

1. The management staff selects the "View statistics" button on ManagementHomeView.
2. The class ManagementHomeView calls the class SelectStatisticView.
3. The class SelectStatisticView shows itself to the management staff.

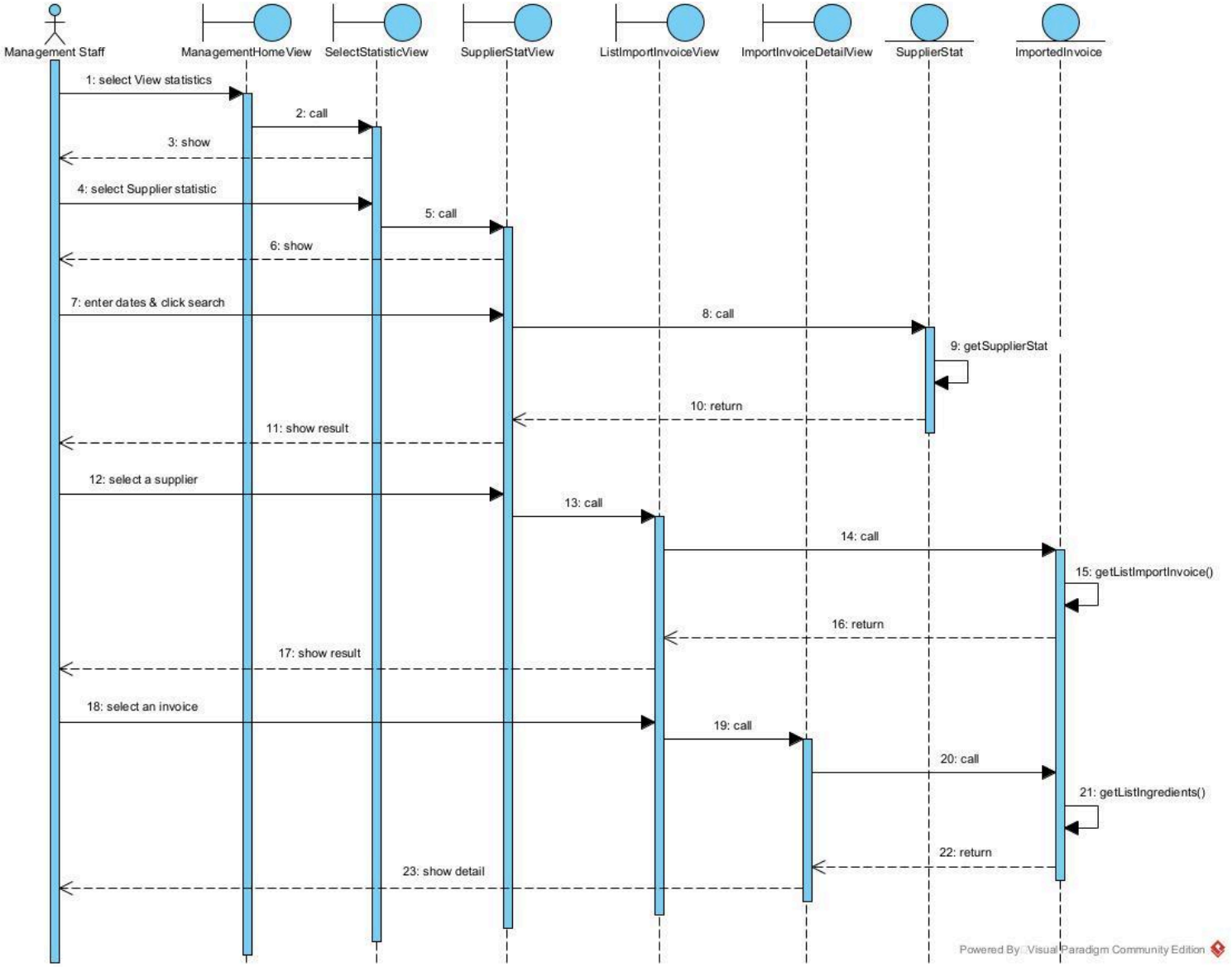
4. The management staff clicks on the "View supplier statistics" button.
5. The class SelectStatisticView calls the class SupplierStatView.
6. The class SupplierStatView shows itself to the management staff (showing date filters).
7. The management staff enters start date, end date and clicks "Search".
8. The class SupplierStatView calls the class SupplierStat to process.
9. The class SupplierStat calls the method getSupplierStat().
10. The class SupplierStat returns the result (list of SupplierStat objects) to the class SupplierStatView.
11. The class SupplierStatView shows the results (supplier list) to the management staff.
12. The management staff selects a supplier from the list.
13. The class SupplierStatView calls the class ListImportInvoiceView.
14. The class ListImportInvoiceView calls the class ImportedInvoice to get data.
15. The class ImportedInvoice calls the method getListImportInvoice().
16. The class ImportedInvoice returns the results (list of invoices) to the class ListImportInvoiceView.
17. The class ListImportInvoiceView shows the results to the management staff.
18. The management staff selects an invoice from the list.
19. The class ListImportInvoiceView calls the class ImportInvoiceDetailView.
20. The class ImportInvoiceDetailView calls the class ImportedInvoice to get data.
21. The class ImportedInvoice calls the method getListIngredients().
22. The class ImportedInvoice returns the results (list of ingredient details) to the class ImportInvoiceDetailView.
23. The class ImportInvoiceDetailView shows the final results to the management staff.

5. Communication Diagram



Communication Diagram for Module 2

6. Sequence Diagram

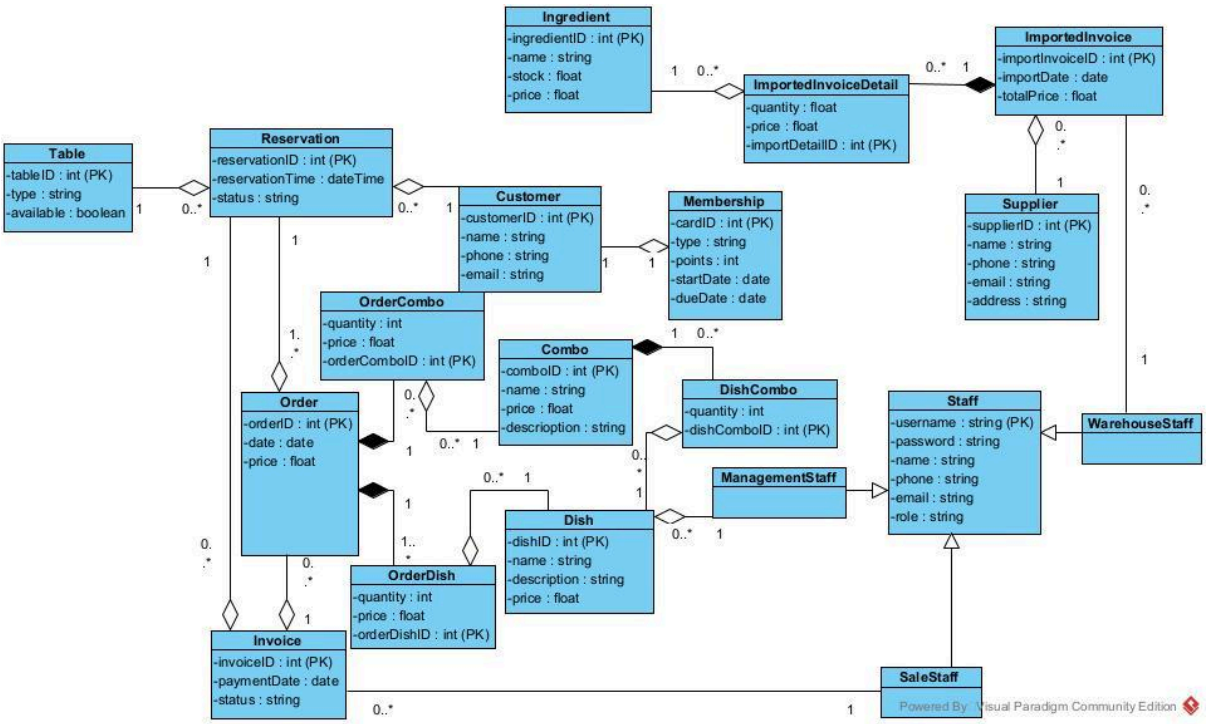


Sequence Diagram for Module 2

C. Design Phase

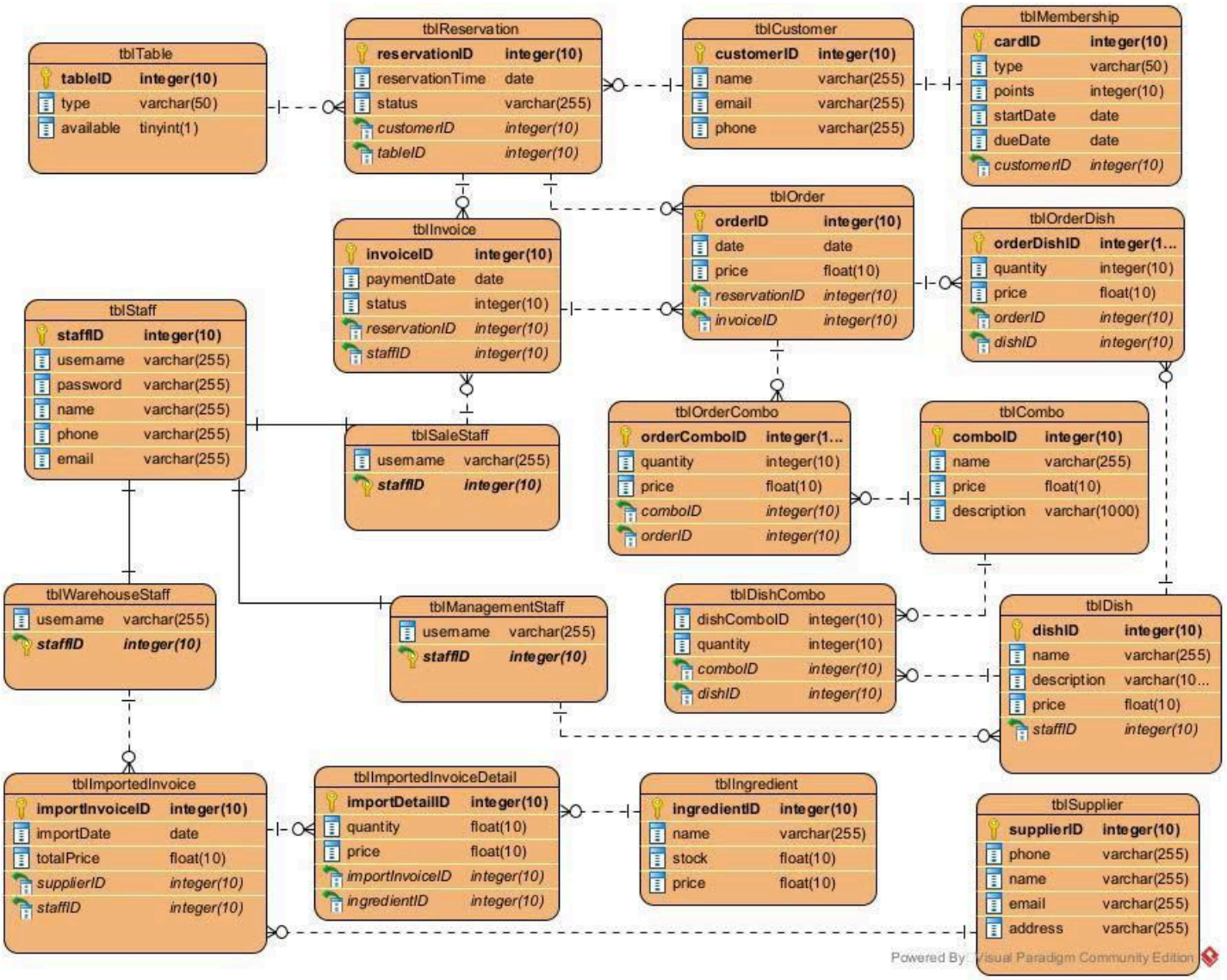
I. Entity Class Diagram

- Step 1: Add ID attributes.
- Step 2: Add type of attribute according to Java.
- Step 3: Change association into composition and aggregation relationship.



II. Database design

- Step 1: Name the tables. (tblCustomer, tblImportedInvoiceDetail,...)
- Step 2: Take the original attribute only in the entity class diagram and change the type to SQL type.
- Step 3: Consider the relationship between tables (1-1, 1-n)
- Step 4: Add primary key, foreign key
    + Primary Keys: The (PK) attribute in each class becomes the PRIMARY KEY of its table.
    + Foreign Keys (1-n): For a 1-n relationship (e.g., Supplier (1) -- ImportedInvoice (n)), the n side table (tblImportedInvoice) gets a new supplierID (FK) column that references tblSupplier(supplierID).
    + Foreign Keys (n-n): The junction tables (e.g., tblOrderDish) receive two Foreign Key columns (e.g., orderID (FK) referencing tblOrder and dishID (FK) referencing tblDish).
- Step 5: Remove redundant attributes.

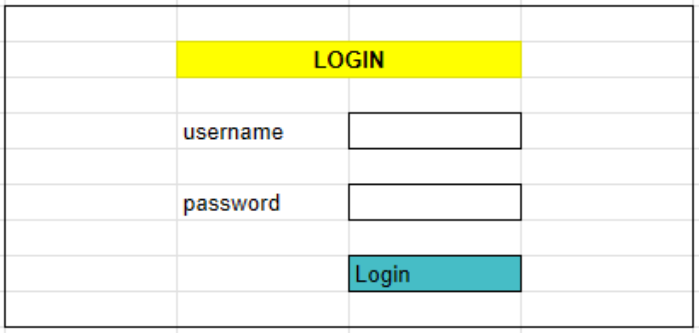


Entity Relationship Diagram

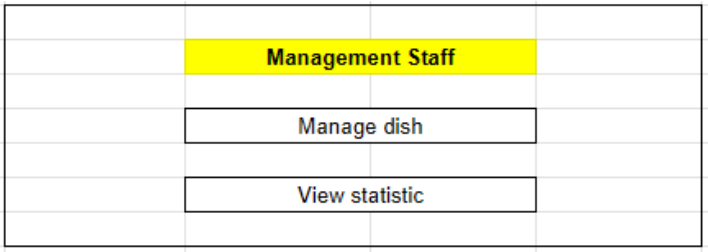
III. Module 1: Management staff edits dish information

1. Class Diagram

Preview interface

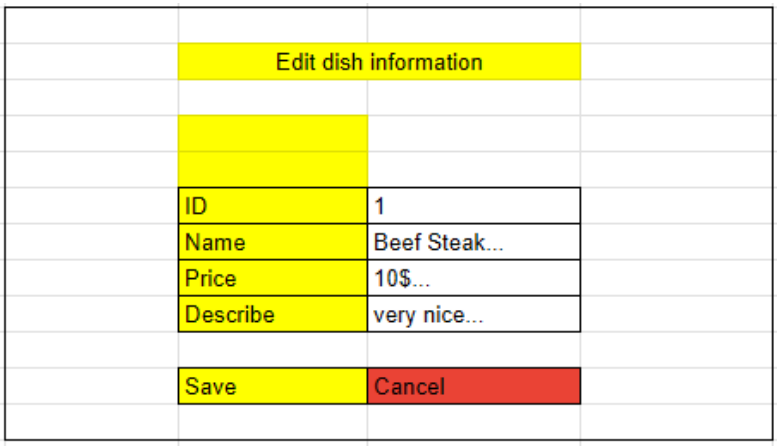


LoginView

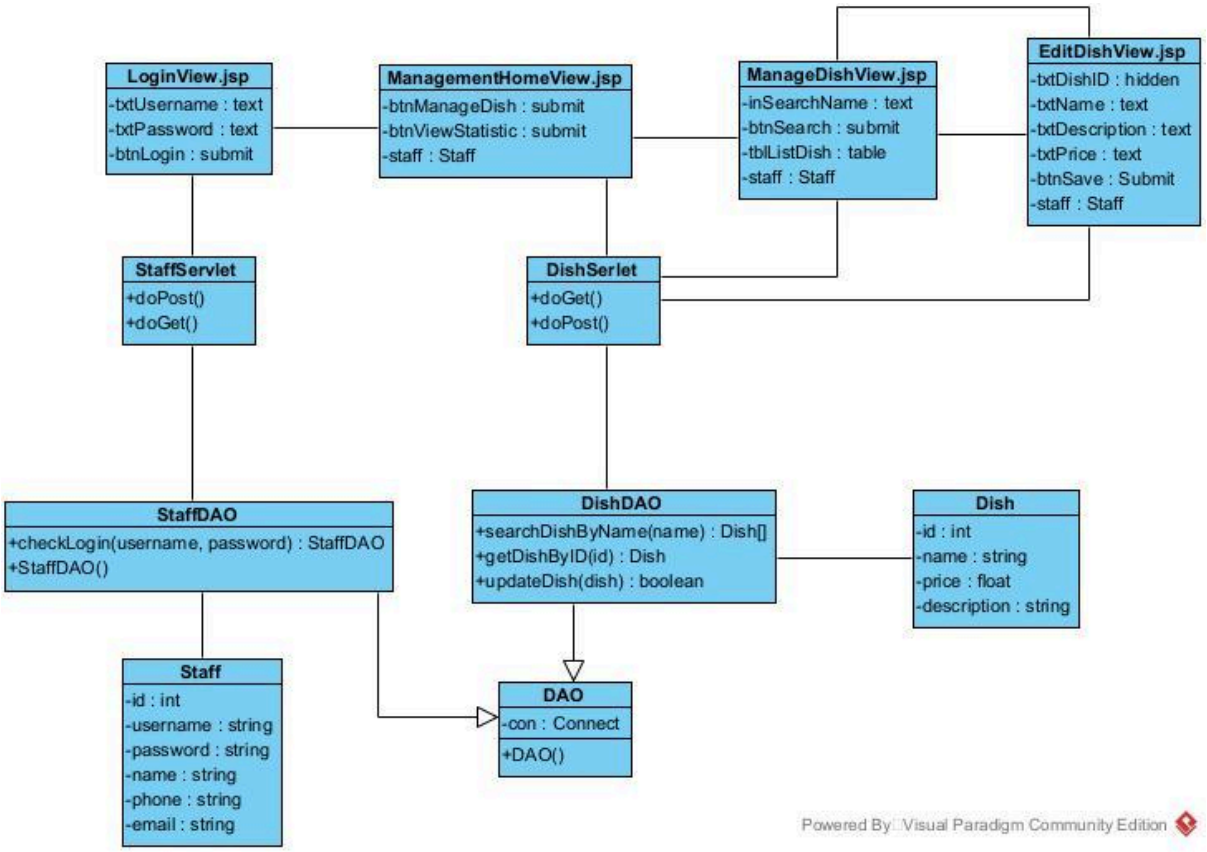


ManagementHomeView

ManageDishView

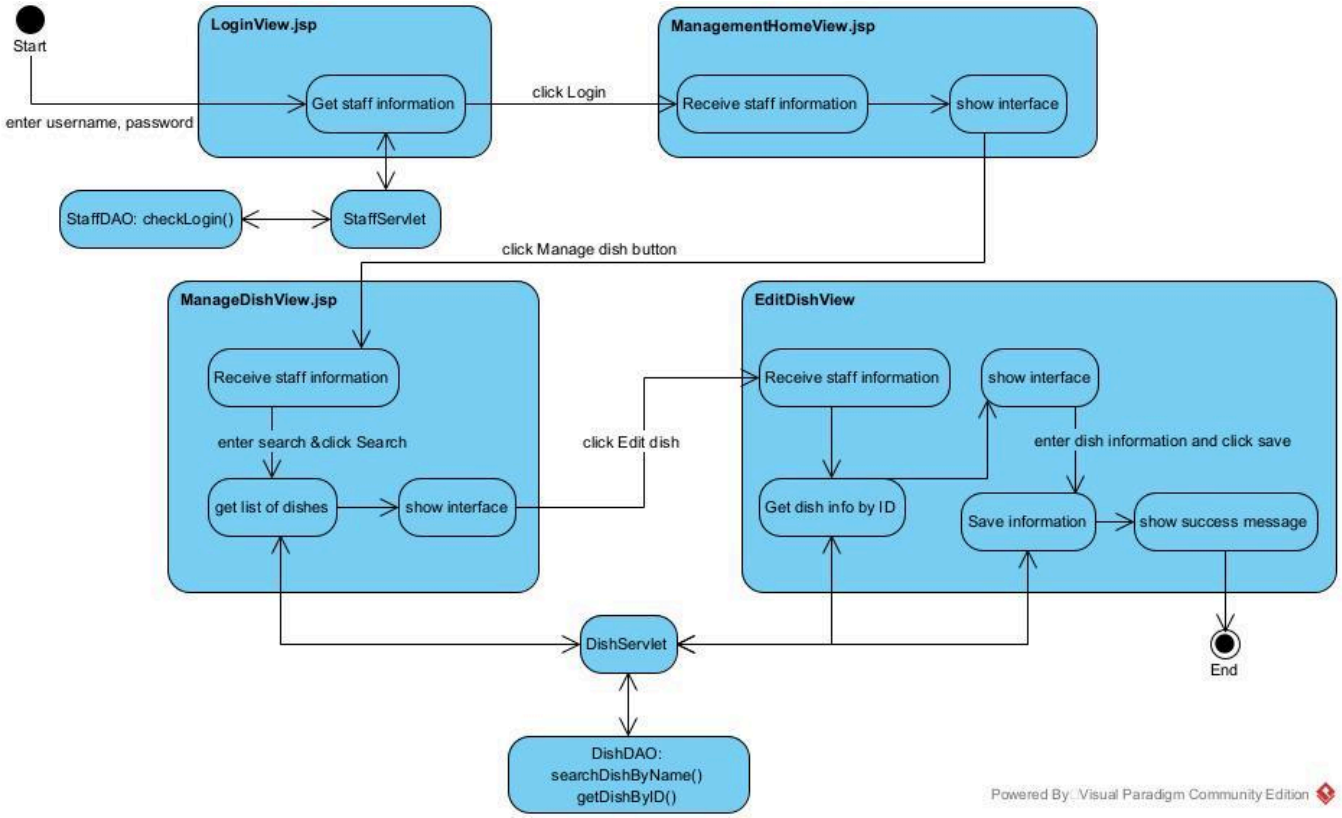


EditDishView



Class Diagram for Module 1

2. Activity Diagram



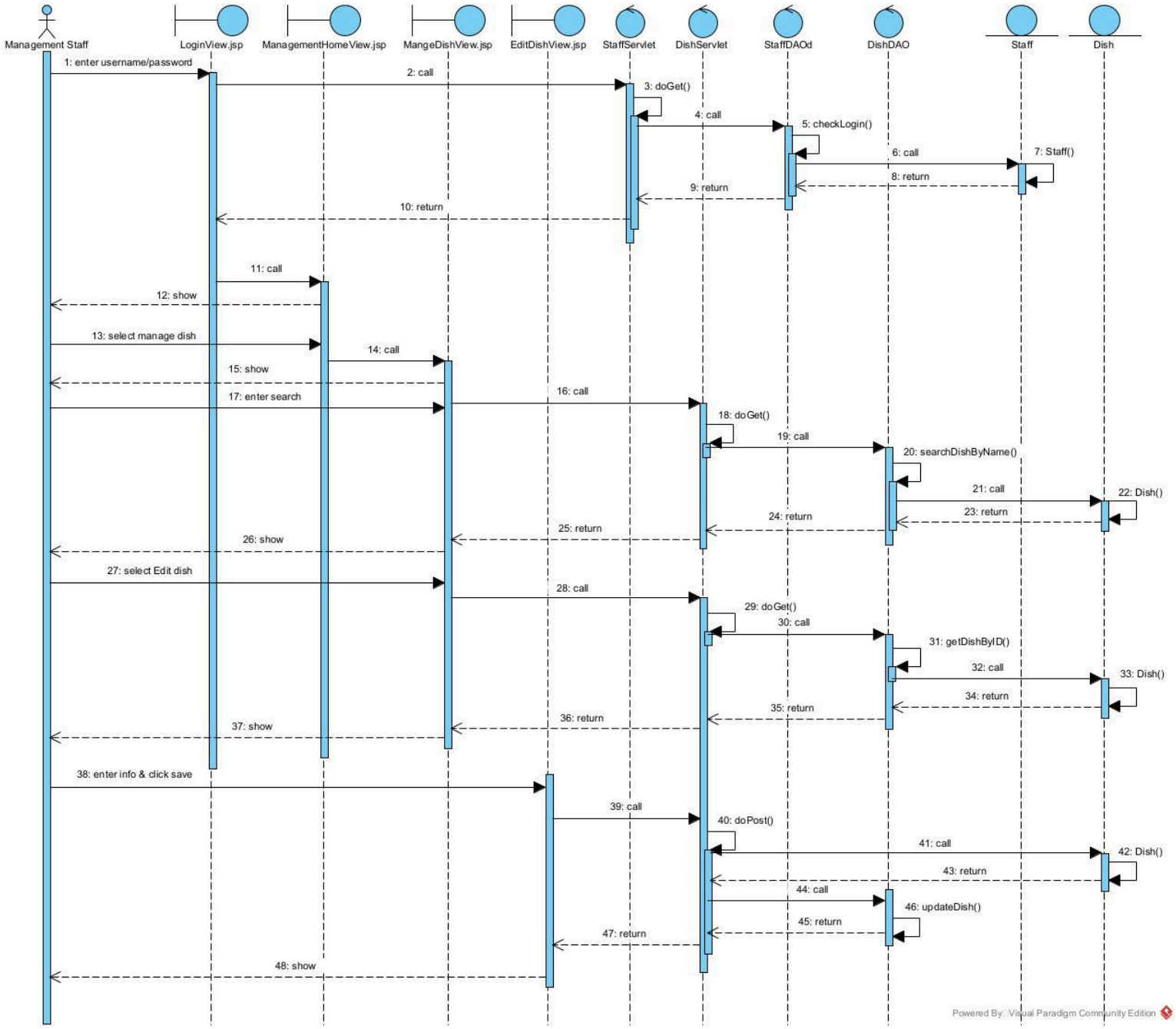
Activity Diagram for Module 1

3. Scenario ver 3

1. A management staff enters username, password and clicks on the login button on LoginView.jsp.
2. The interface LoginView.jsp calls StaffServlet.
3. The class StaffServlet calls the method doGet().
4. The method doGet() calls the class StaffDAO.
5. The class StaffDAO calls the method checkLogin().
6. The method checkLogin() calls the class Staff (Entity) to pack data.
7. The class Staff calls its constructor Staff() (self-call) to pack the information.
8. The class Staff returns the Staff object to the method checkLogin().
9. The method checkLogin() returns the result to the method doGet().
10. The method doGet() returns the result to the interface LoginView.jsp.
11. The interface LoginView.jsp calls the interface ManagementHomeView.jsp.
12. The interface ManagementHomeView.jsp shows itself to the management staff.
13. The management staff selects the "manage dish" button.
14. The interface ManagementHomeView.jsp calls the interface ManageDishView.jsp.
15. The interface ManageDishView.jsp shows itself (with search form) to the management staff.
16. The management staff enters a search term and clicks "Search".
17. The interface ManageDishView.jsp calls the class DishServlet (with action="search").
18. The class DishServlet calls the method doGet().
19. The method doGet() calls the class DishDAO.
20. The class DishDAO calls the method searchDishByName().
21. The method searchDishByName() calls the class Dish (Entity) to pack the data.
22. The class Dish calls its constructor Dish() (self-call) to pack each object.
23. The class Dish returns the Dish objects to the method searchDishByName().
24. The method searchDishByName() returns the list to the method doGet().
25. The method doGet() returns the list to the interface ManageDishView.jsp.
26. The interface ManageDishView.jsp shows the results to the management staff.
27. The management staff selects an "Edit" button for a dish.
28. The interface ManageDishView.jsp calls the class DishServlet (with action="load_edit").
29. The class DishServlet calls the method doGet().
30. The method doGet() calls the class DishDAO.
31. The class DishDAO calls the method getDishByID().
32. The method getDishByID() calls the class Dish (Entity) to pack data.
33. The class Dish calls its constructor Dish() (self-call) to pack the object.
34. The class Dish returns the Dish object to the method getDishByID().
35. The method getDishByID() returns the object to the method doGet().
36. The method doGet() returns the object to the interface EditDishView.jsp.
37. The interface EditDishView.jsp shows itself, populated with the dish data.
38. The management staff enters the new dish information and clicks the "Save" button.
39. The interface EditDishView.jsp calls the class DishServlet (with action="update").
40. The class DishServlet calls the method doPost().
41. The method doPost() calls the class Dish (Entity) to pack the information into an object.
42. The class Dish calls its constructor Dish() (self-call) to pack the new data.
43. The class Dish returns the new Dish object to the method doPost().

44. The method doPost() calls the class DishDAO, passing the new Dish object.
45. The class DishDAO calls the method updateDish().
46. The method updateDish() returns the result (true/false) to the method doPost().
47. The method doPost() returns the result to the interfaceEditDishView.jsp.
48. The interface EditDishView.jsp shows the success message to the management staff.

4. Sequence Diagram

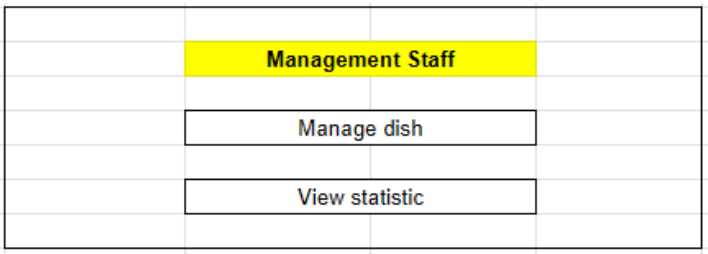


Sequence Diagram for Module 1

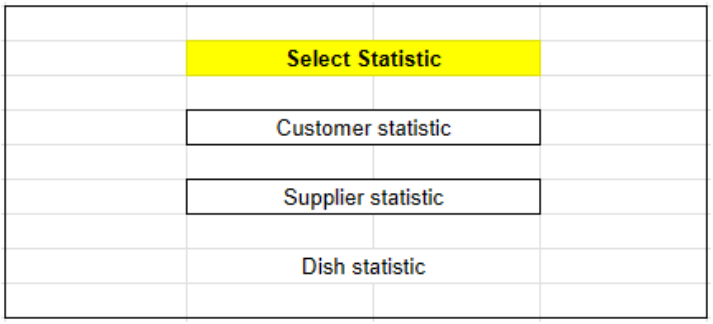
IV. Module 2: Management staff view supplier statistics

1. Class Diagram

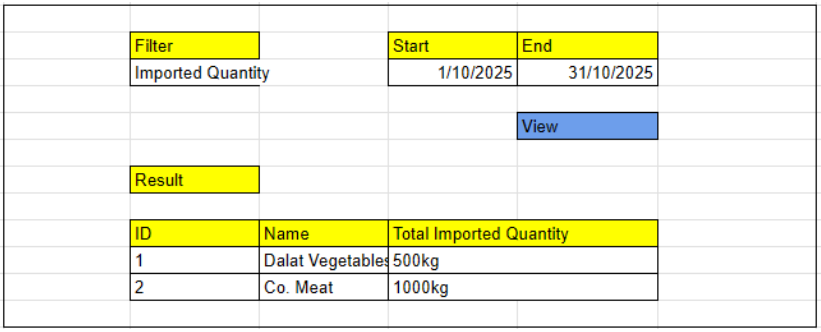
Preview interface:



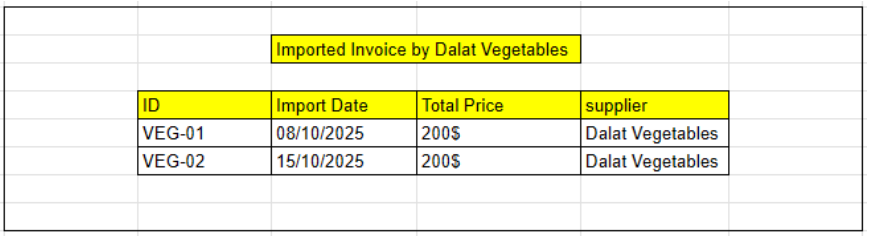
ManagementHomeView



SelectStatisticView



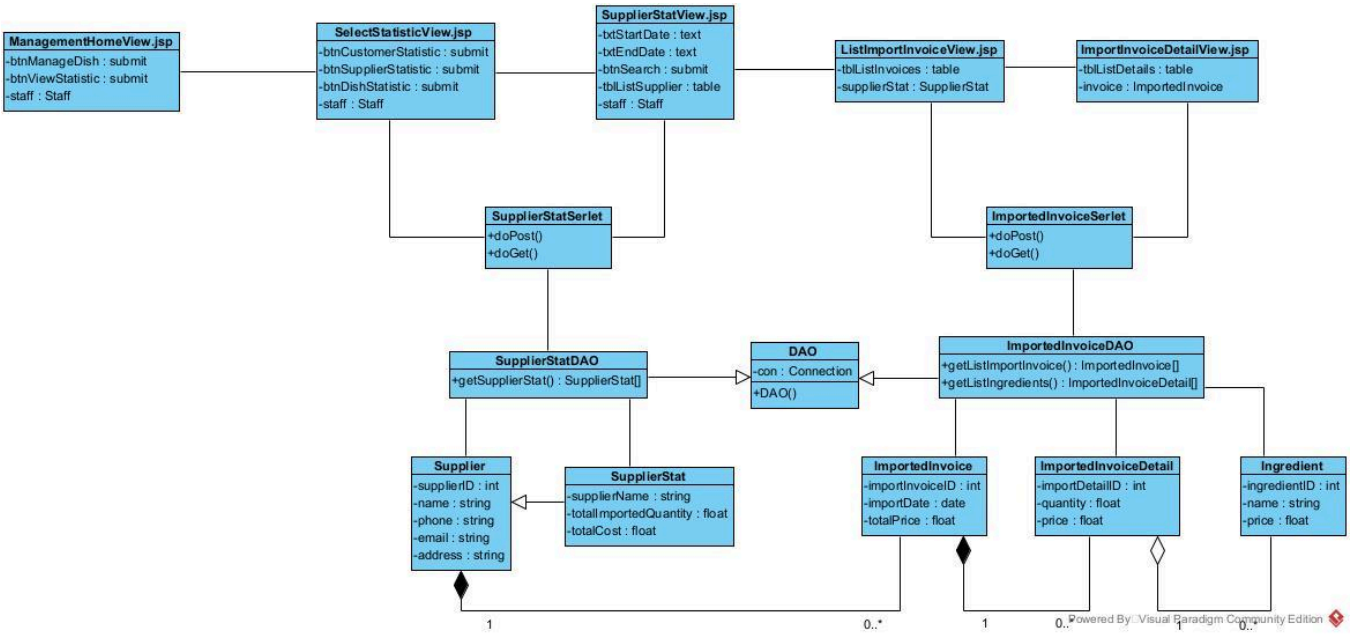
SupplierStatView



ListImportInvoiceView

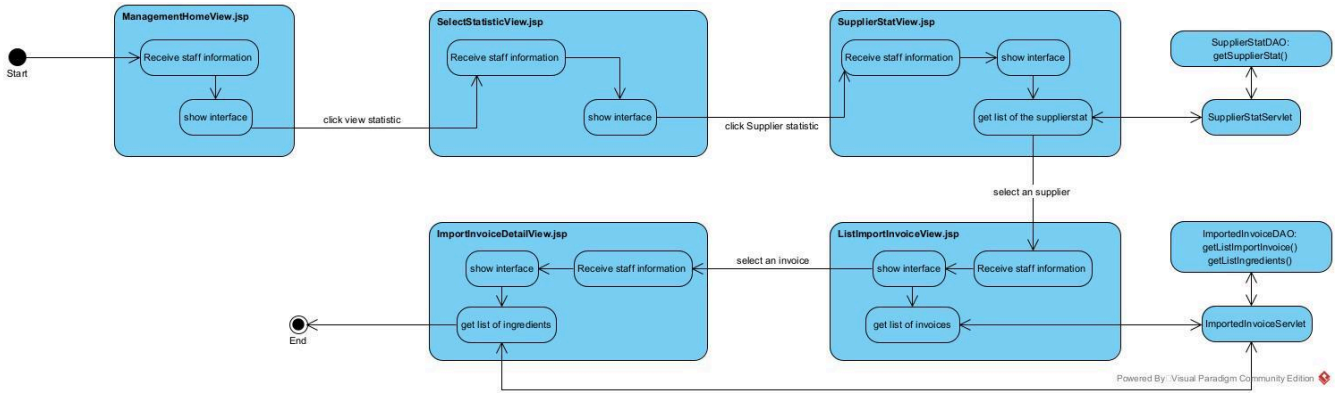
Invoice VEG-01			
No.	Ingredient	Quantity	Price
1	Carrots	100kg	50$
2	Potatoes	200$	150$
		Total price	200$

ImporInvoiceDetailView



Class Diagram for Module 2

2. Activity Diagram



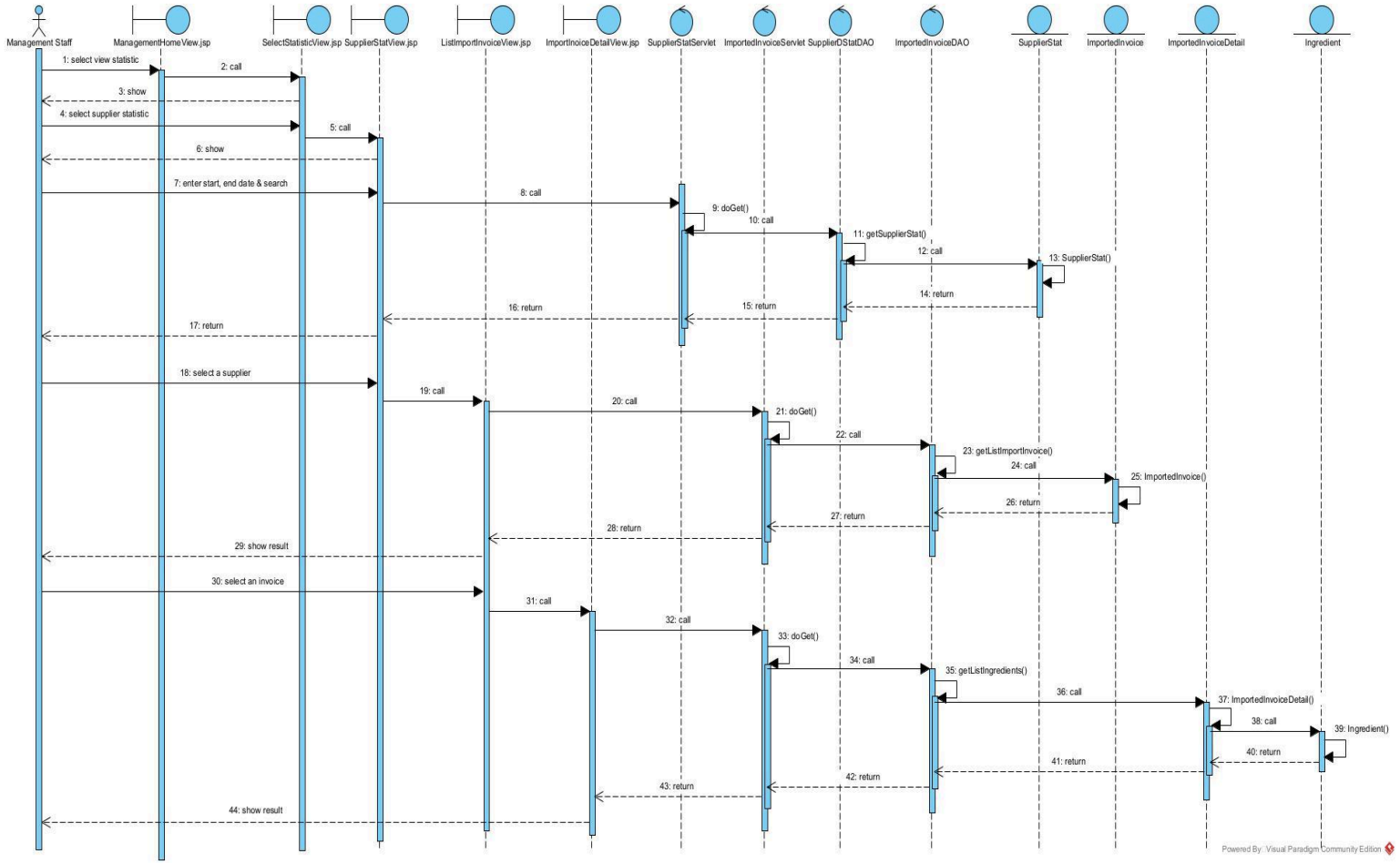
Activity Diagram for Module 2

3. Scenario ver 3

1. The management staff selects the "view statistic" button in the interface ManagementHomeView.jsp.
2. The interface ManagementHomeView.jsp calls the interface SelectStatisticView.jsp.
3. The interface SelectStatisticView.jsp shows itself to the management staff.
4. The management staff selects the "supplier statistic" button.
5. The interface SelectStatisticView.jsp calls the interface SupplierStatView.jsp.

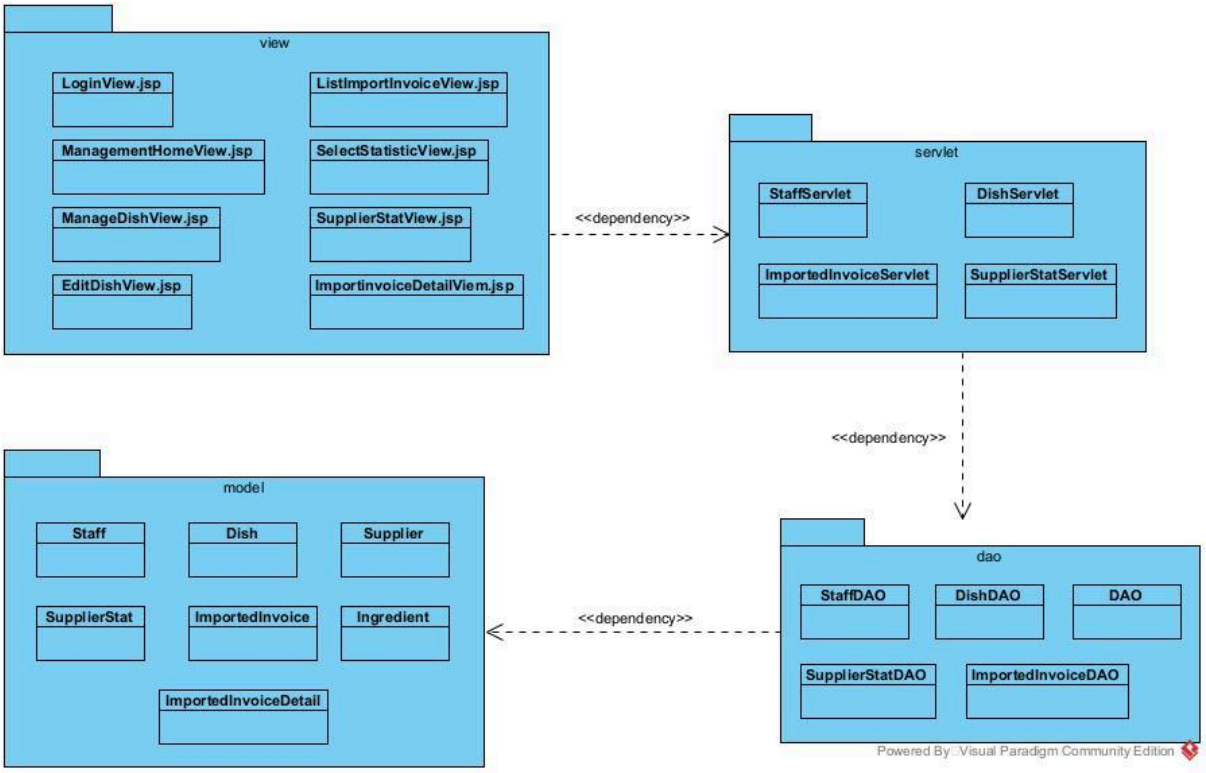
6. The interface SupplierStatView.jsp shows itself to the management staff.
7. The management staff enters start, end date and clicks "search".
8. The interface SupplierStatView.jsp calls the class SupplierStatServlet.
9. The class SupplierStatServlet calls the method doGet().
10. The method doGet() calls the class SupplierStatDAO.
11. The class SupplierStatDAO calls the method getSupplierStat().
12. The method getSupplierStat() calls the class SupplierStat (Entity) to pack data.
13. The class SupplierStat calls its constructor SupplierStat() (self-call).
14. The class SupplierStat returns the SupplierStat object(s) to the method getSupplierStat().
15. The method getSupplierStat() returns the result to the method doGet().
16. The method doGet() returns the result to the interface SupplierStatView.jsp.
17. The interface SupplierStatView.jsp shows the result to the management staff.
18. The management staff selects a supplier in the list on SupplierStatView.jsp.
19. The interface SupplierStatView.jsp calls the interface ListImportInvoiceView.jsp.
20. The interface ListImportInvoiceView.jsp calls the class ImportedInvoiceServlet.
21. The class ImportedInvoiceServlet calls the method doGet().
22. The method doGet() calls the class ImportedInvoiceDAO.
23. The class ImportedInvoiceDAO calls the method getListImportInvoice().
24. The method getListImportInvoice() calls the class ImportedInvoice (Entity) to pack data.
25. The class ImportedInvoice calls its constructor ImportedInvoice() (self-call).
26. The class ImportedInvoice returns the object(s) to the method getListImportInvoice().
27. The method getListImportInvoice() returns the result to the method doGet().
28. The method doGet() returns the result to the interface ListImportInvoiceView.jsp.
29. The interface ListImportInvoiceView.jsp shows the result to the management staff.
30. The management staff selects an invoice on ListImportInvoiceView.jsp.
31. The interface ListImportInvoiceView.jsp calls the interface ImportInvoiceDetailView.jsp.
32. The interface ImportInvoiceDetailView.jsp calls the class ImportedInvoiceServlet.
33. The class ImportedInvoiceServlet calls the method doGet().
34. The method doGet() calls the class ImportedInvoiceDAO.
35. The class ImportedInvoiceDAO calls the method getListIngredients().
36. The method getListIngredients() calls the class ImportedInvoiceDetail (Entity) to pack data.
37. The class ImportedInvoiceDetail calls its constructor ImportedInvoiceDetail() (self-call).
38. The method getListIngredients() calls the class Ingredient (Entity) to get related data.
39. The class Ingredient calls its constructor Ingredient() (self-call).
40. The class Ingredient returns its object to the method getListIngredients().
41. The class ImportedInvoiceDetail returns its object to the method getListIngredients().
42. The method getListIngredients() returns the result to the method doGet().
43. The method doGet() returns the result to the interface ImportInvoiceDetailView.jsp.
44. The interface ImportInvoiceDetailView.jsp shows the result to the management staff.

4. Sequence Diagram



Sequence Diagram for Module 2

V. Package Diagram



Package Diagram

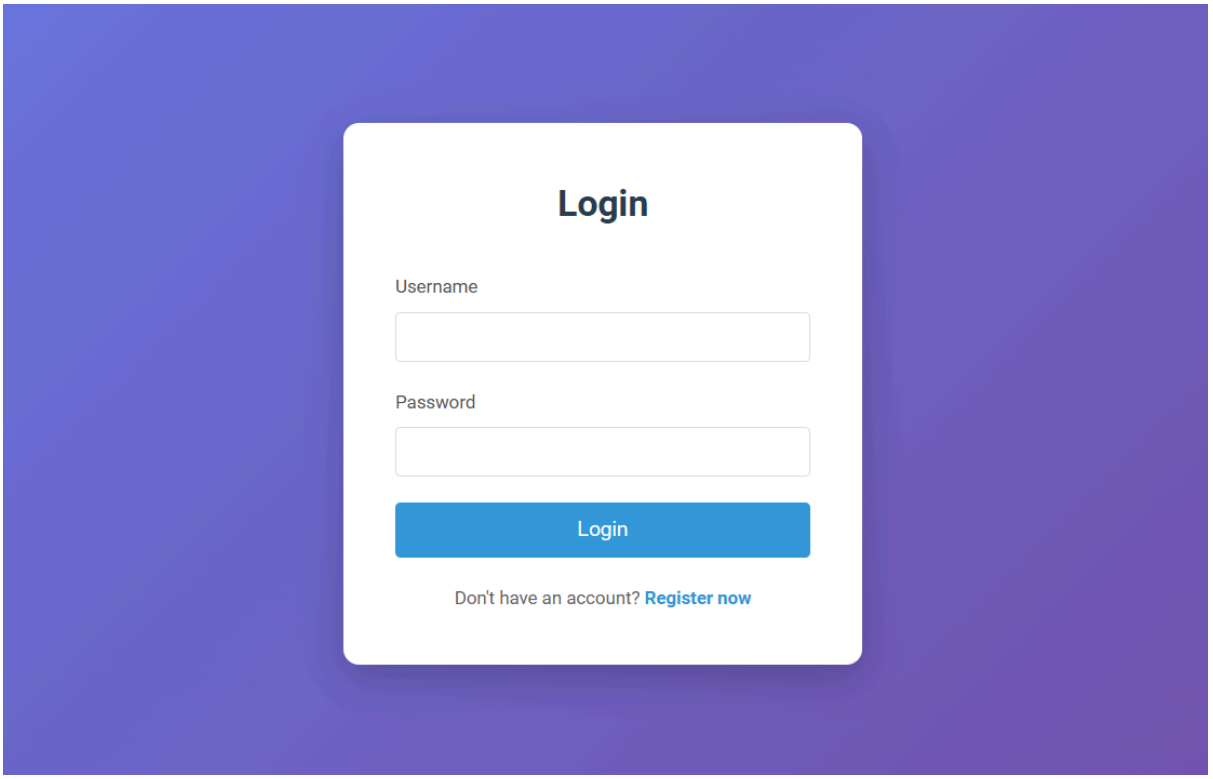
VI. Deployment Diagram

Deployment Diagram

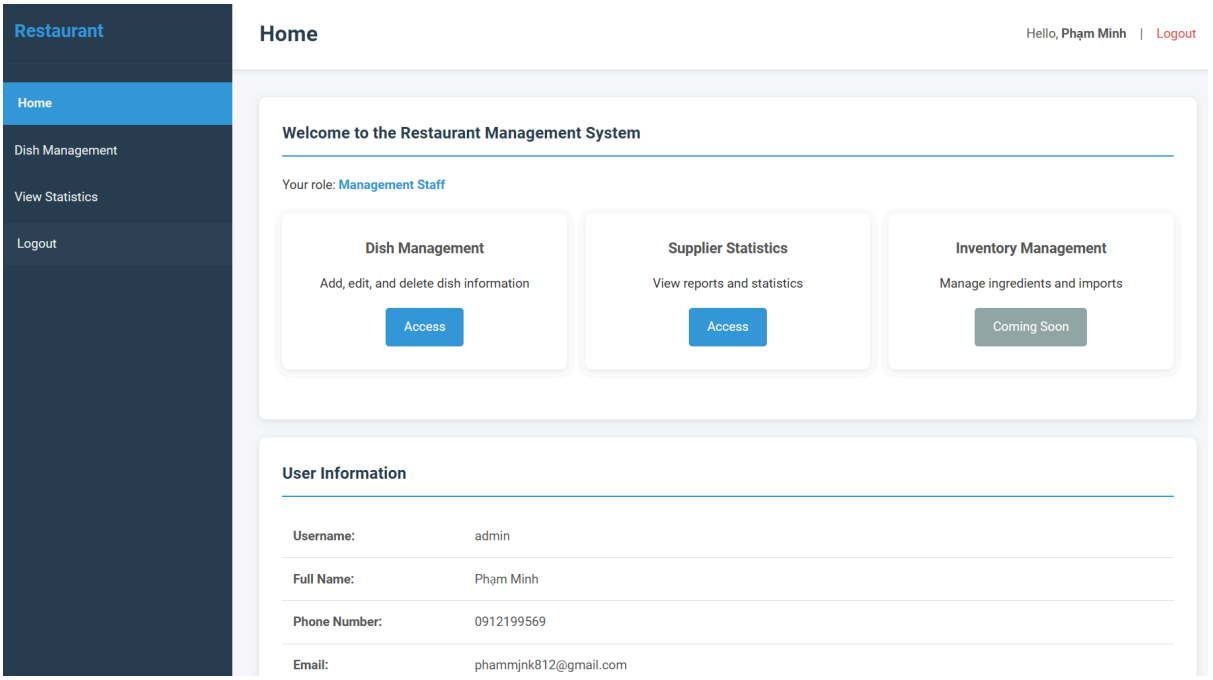
D. Programming

I. User Interface

1. General UI

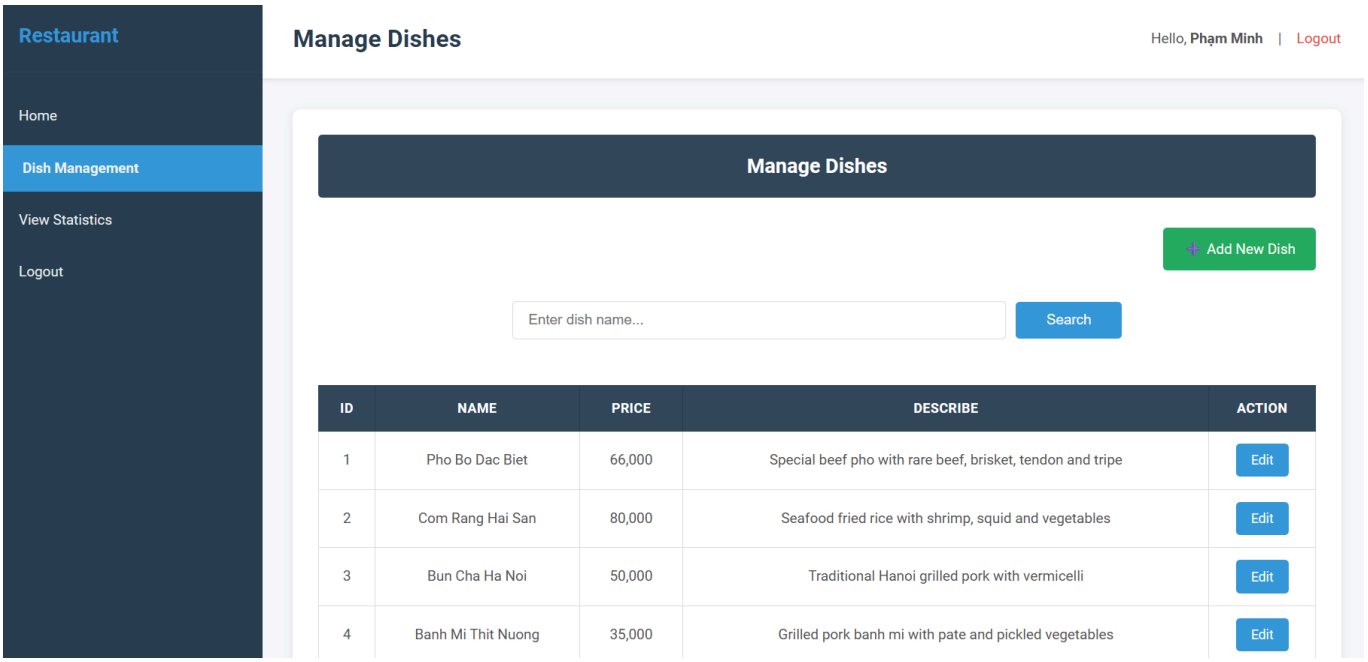


LoginView.jsp

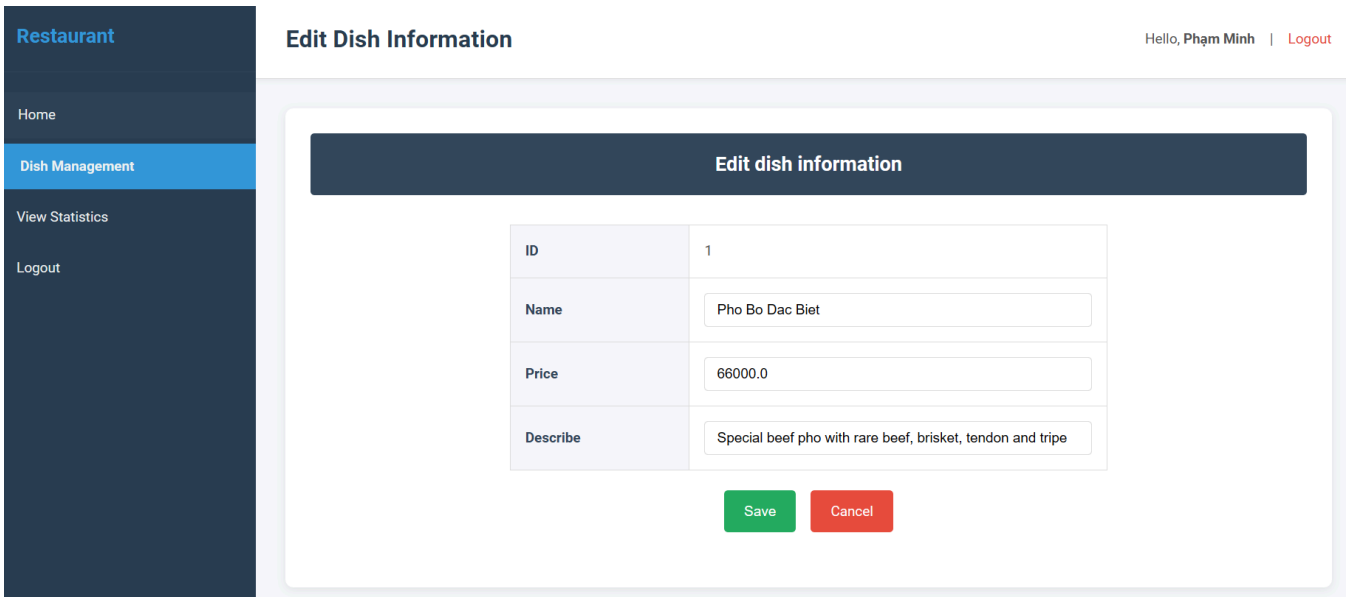


ManagementHomeView.jsp

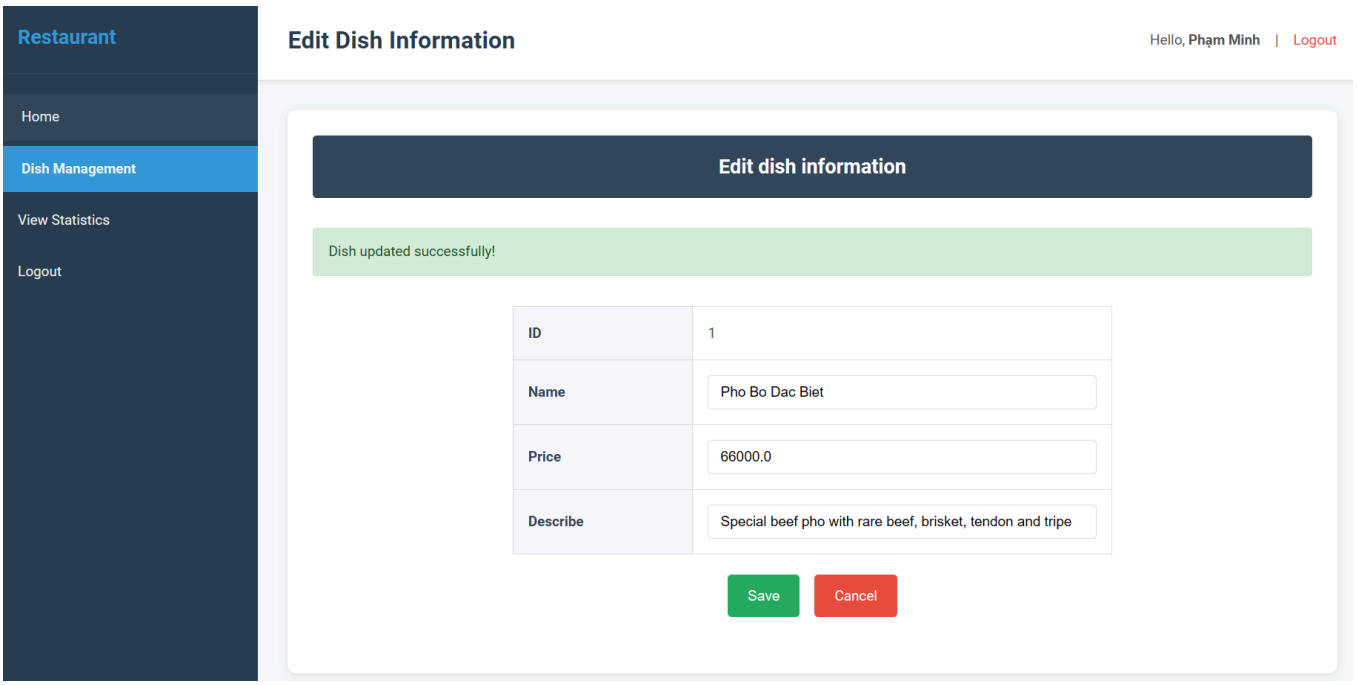
2. Module 1: Management staff edits dish information



ManageDishView.jsp

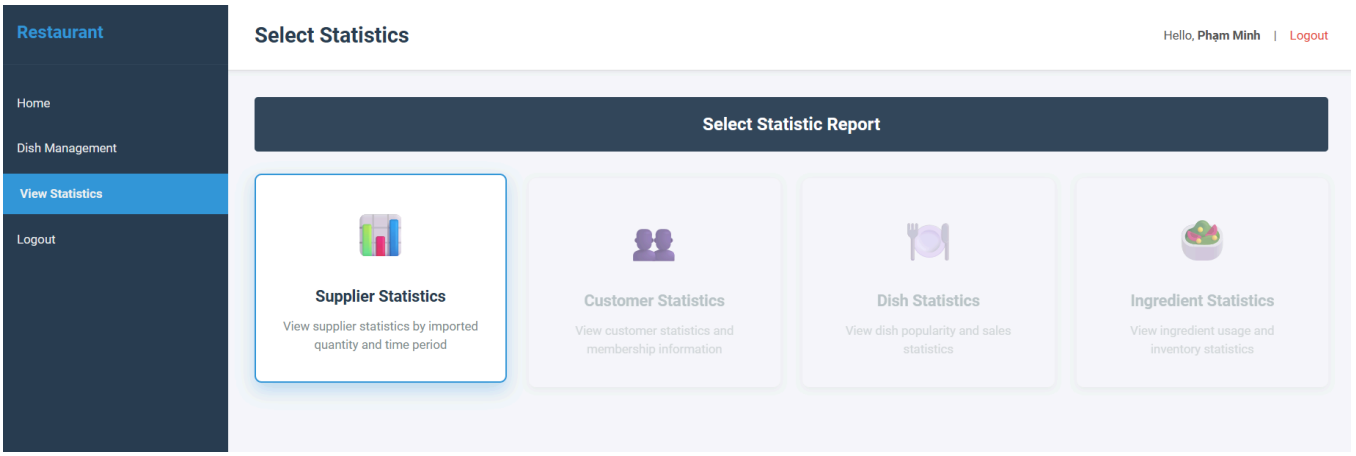


EditDishView.jsp

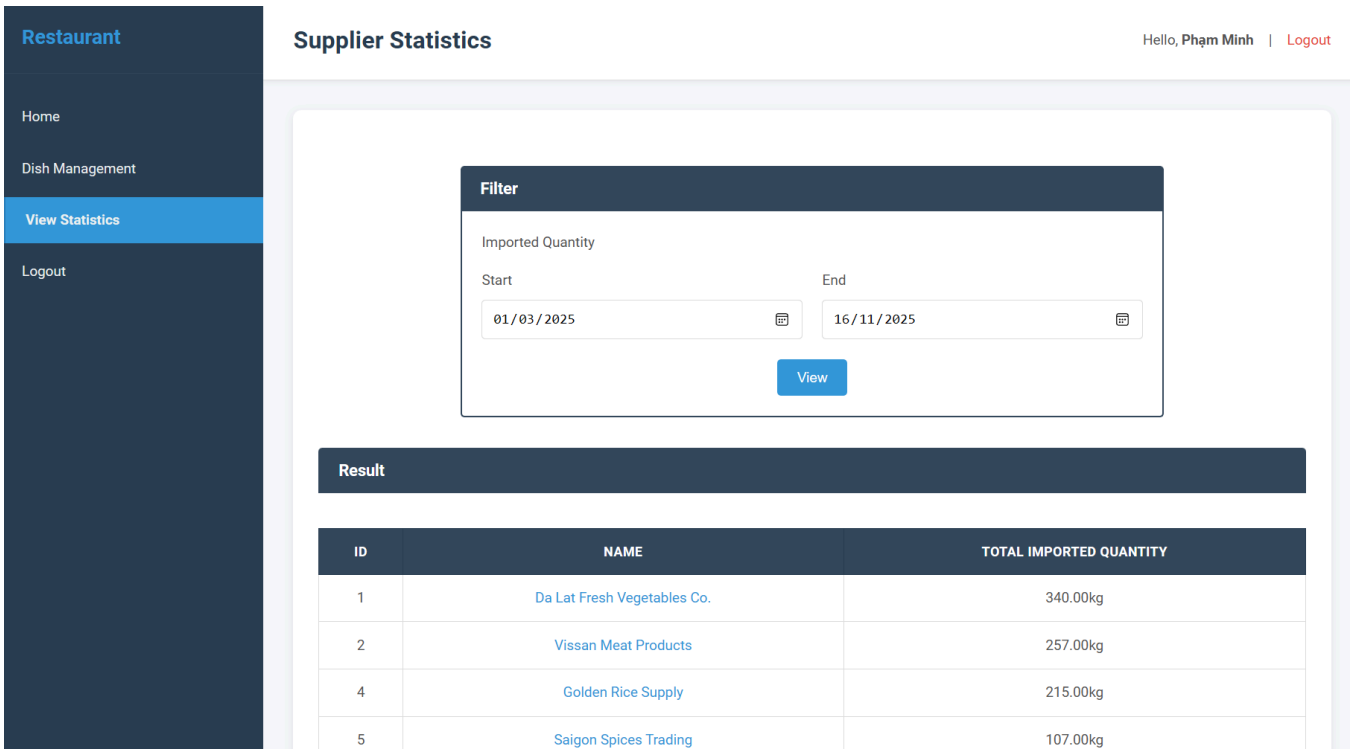


EditDishView.jsp (Updated successfully)

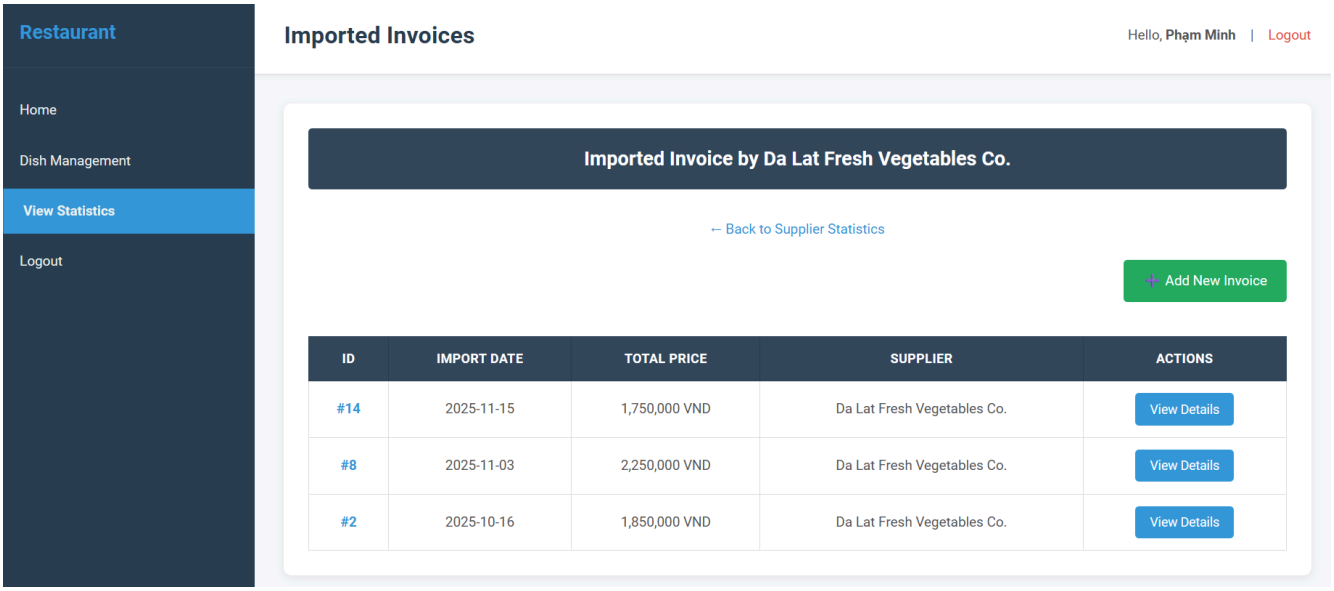
3. Module 2: Management staff view supplier statistics



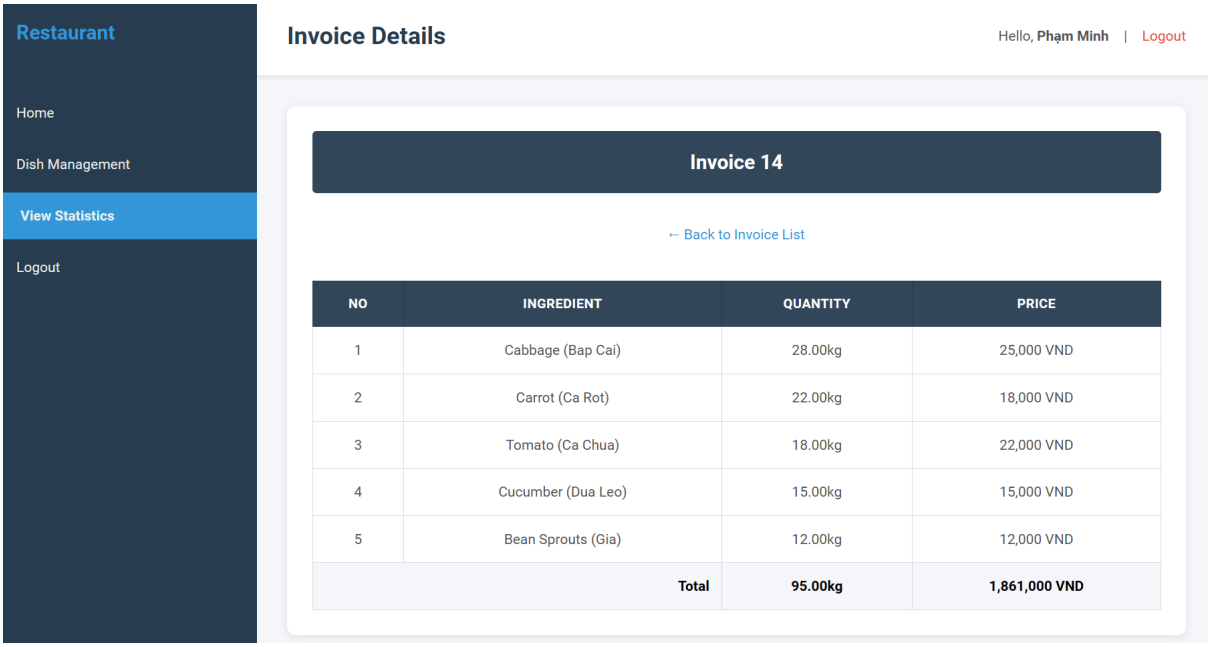
SelectStatisticView.jsp



SupplierStatView.jsp



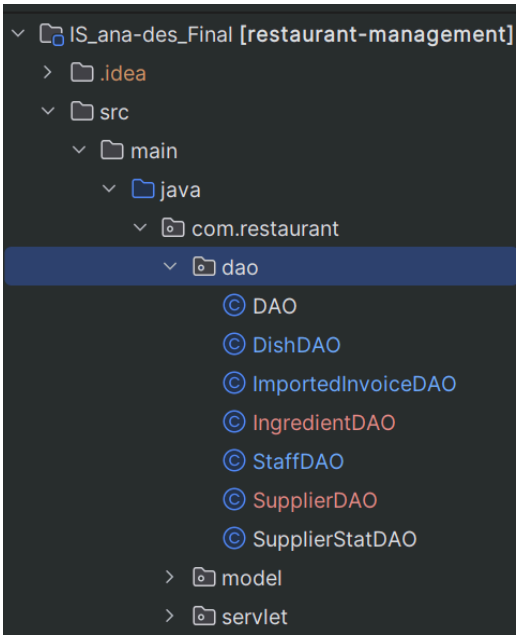
ListImportInvoiceView.jsp



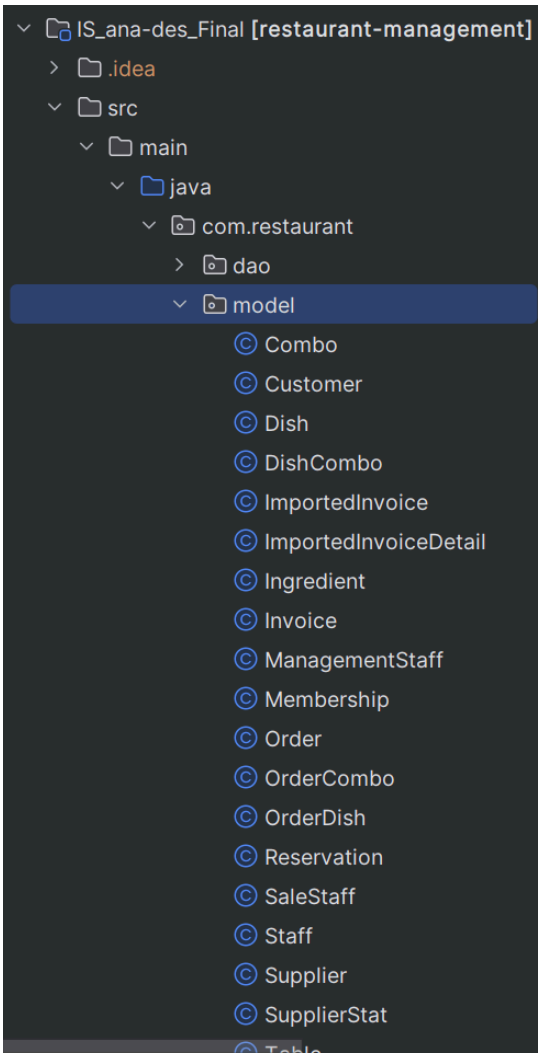
ImportInvocieDetailView.jsp

II. Directory Structure

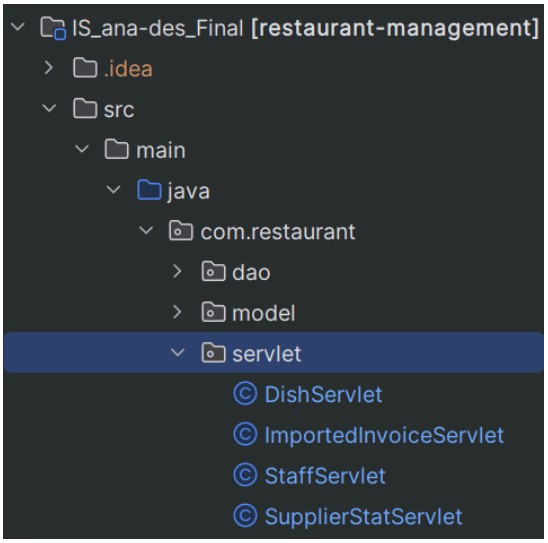
1. Folder DAO



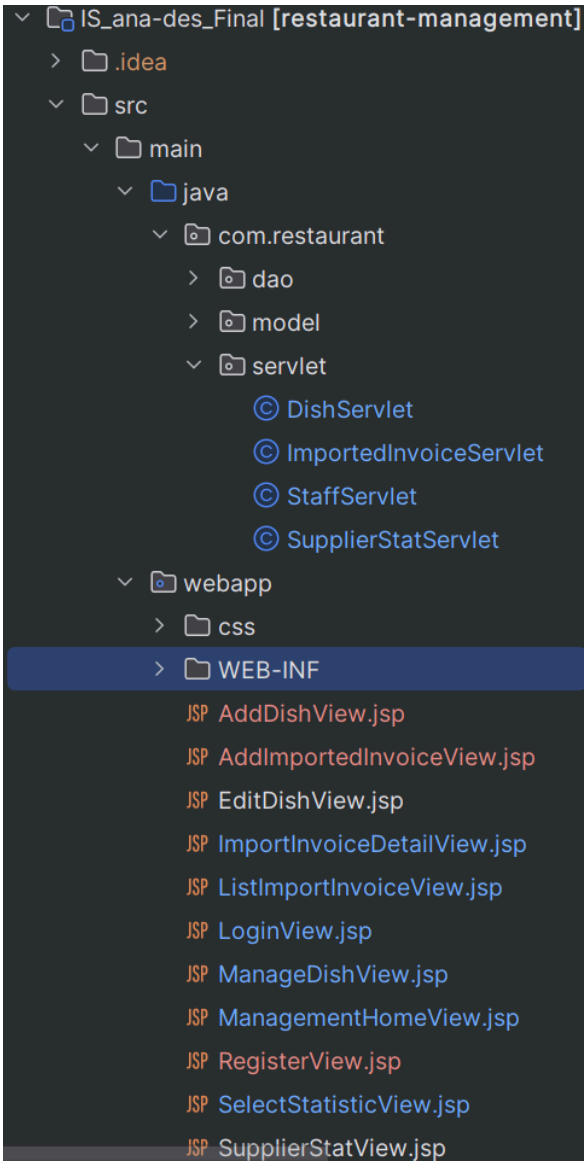
2. Folder Model



3. Folder Servlet



4. Folder WEB-INF (include Views)



III. Source code
GitHub URL: [phmiinh/IS_ana-des_Final]