TAT Restaurant

Food Management System
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
<24/12/2020>	<1.0>	Final Draft	Nguyen Thi Ngoc Anh Nguyen Thu Trang

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document applies to the Food Management System which will be developed by TAT Restaurant.

1.3 Definitions, Acronyms, and Abbreviations

User – a person who use the system, can be customer or admin.

1.4 References

None.

1.5 Overview

In the following section, architectural design of the Food Management System is provided in detail. First, the primary software architecture of the system will be defined. Then, there are further discussion about the goals and constraints that will be imposed upon the quality of the final product, which including but not limited to security, distribution and reuse. In the precedence sections, the key views of the system are demonstrated to depict different aspects of the system. Lastly, criteria concerning with size, performance and quality of the system will be proposed.

2. Architectural Representation

This documents presents the architectural as a series of mandatory views: Use-Case View, Logical View, Deployment View and Data View. These views are presented as Visual Paradigm Community Edition Models , StarUML and use the Unified Modeling Language (UML).

Use-Case View

- *Audience*: all the stakeholders of the system, including the end-users.
- *Area*: describes the set of scenarios and/or use cases that represent significant, central functionality to the system.
- *Related artifacts*: Use-Case Model, Analysis Model, Use-Case-Realization documents.

Logical View

- *Audience*: designers, programmers.
- *Area*: functional requirements: describes the design's object model.
- *Related artifacts*: Design Model.

Deployment View

- *Audience*: deployment managers, system administrators.
- *Area*: topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.
- *Related artifacts*: Deployment Model.

Data View

- *Audience*: data specialists, database administrators.
- *Area*: persistence: describes the architecturally significant persistent elements in the data model.
- *Related artifacts*: Data Model.

3. Architectural Goals and Constraints

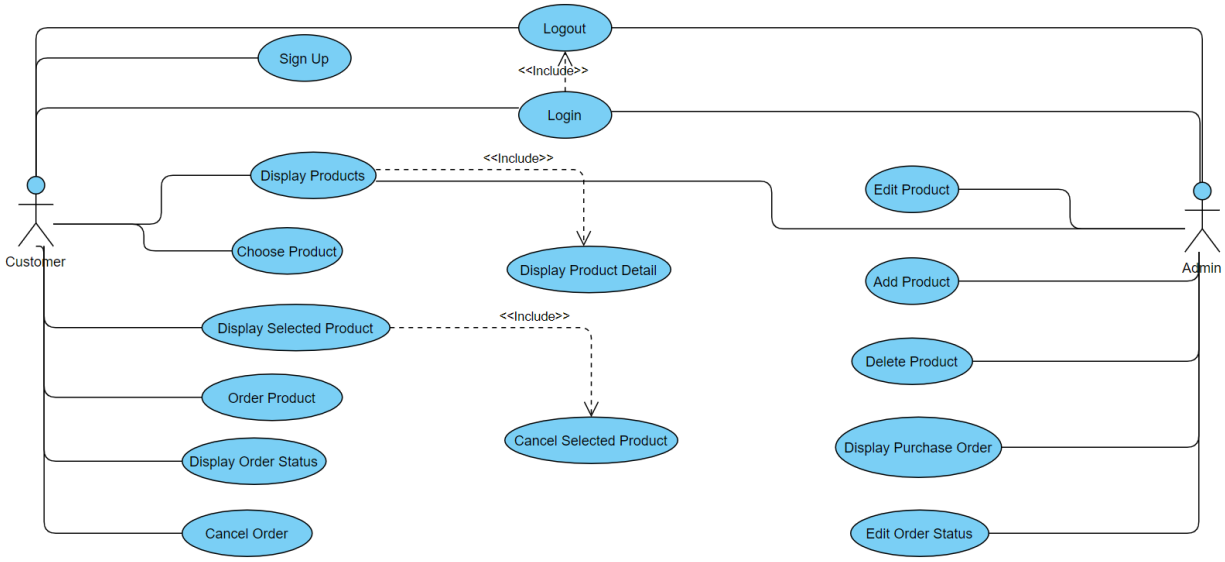
There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The Food Management System must be designed to fulfills all system requirements specified in requirements definition.
- The Food Management design must be structured to be robust, easy to change if and when functional requirements change.
- The Food Management System must be designed to allow the re-use of business logic across applications; therefore, the design separate the three components: model, view and controller.
- The separation of the three components: model, view and controller are also necessary to provide a convenient cooperation between different development teams.
- The Food Management System will run on a dedicated platform with access to a database.
- The Food Management website provides most of the content display. An interface to this system must be capable of handling large traffic volumes.

4. Use-Case View

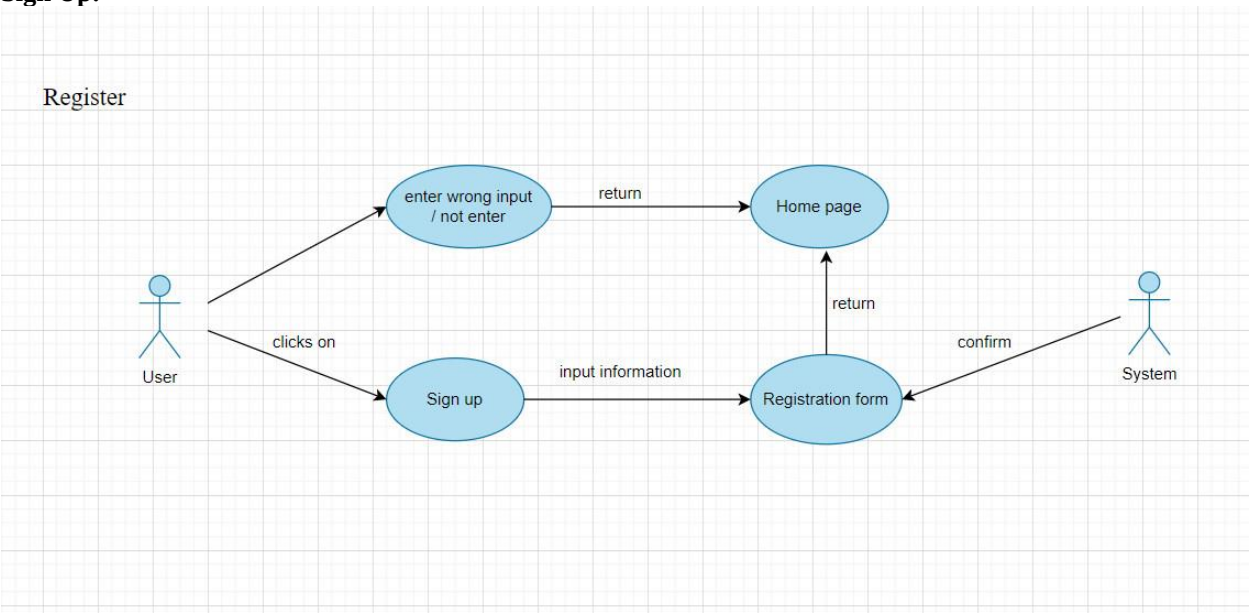
A description of the Use-Case View of the system architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture. The significant use cases (16 Use Cases) in this system are listed below:

- Login
- Logout
- Sign Up
- Display Products
- Display Product Detail
- Display Purchase Order
- Display Selected Product
- Display Order Status
- Choose Product
- Cancel Selected Product
- Order Product
- Cancel Order
- Add Product
- Delete Product
- Edit Product
- Edit Order Status



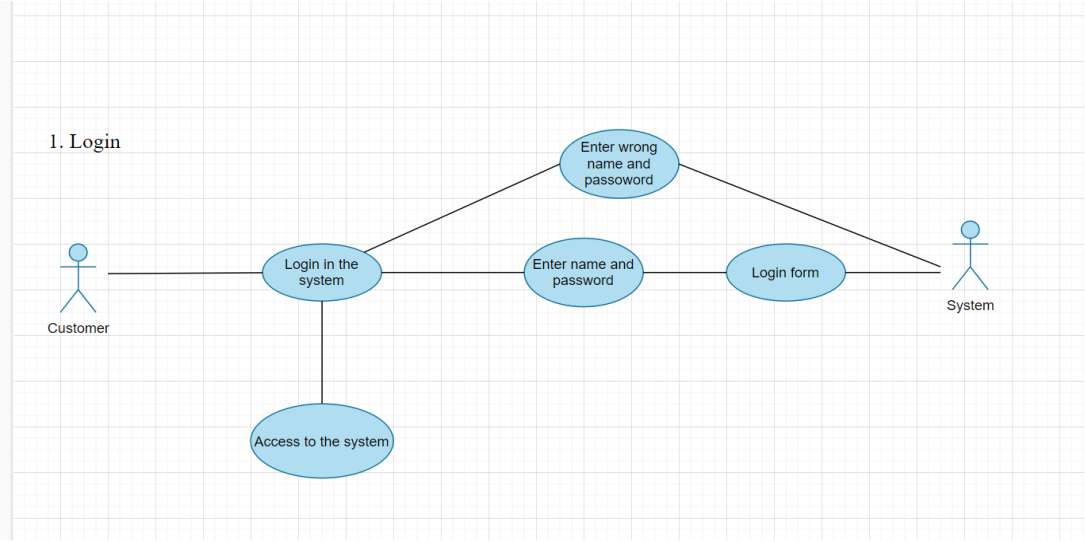
4.1 Use-Case Realizations

Sign Up:



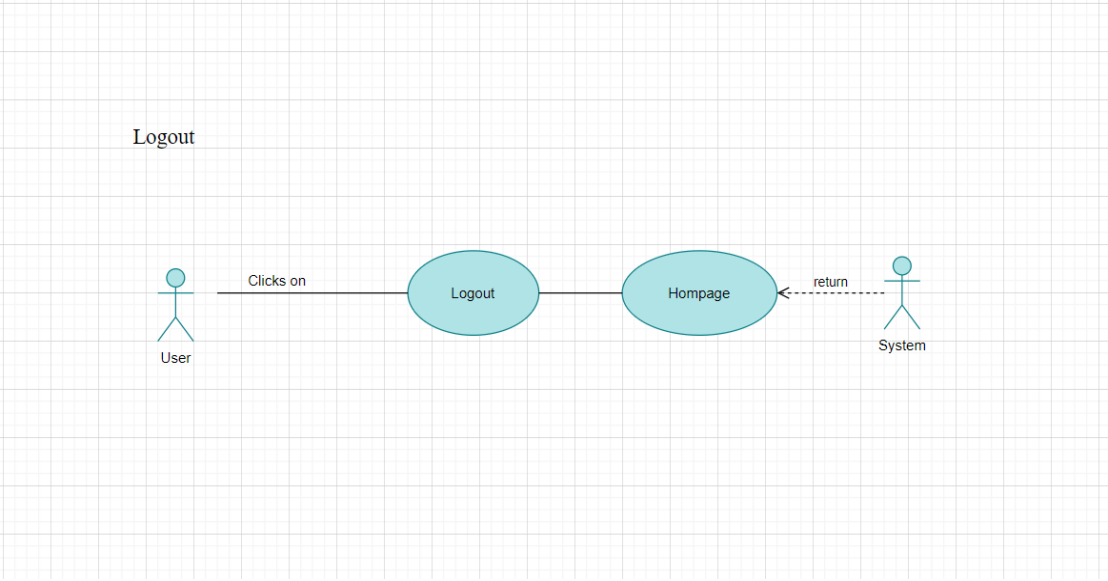
- **Brief Description:** A user creates an account.
- **Specification:** See Use-Case-Realization Specification: Sign Up

Login:



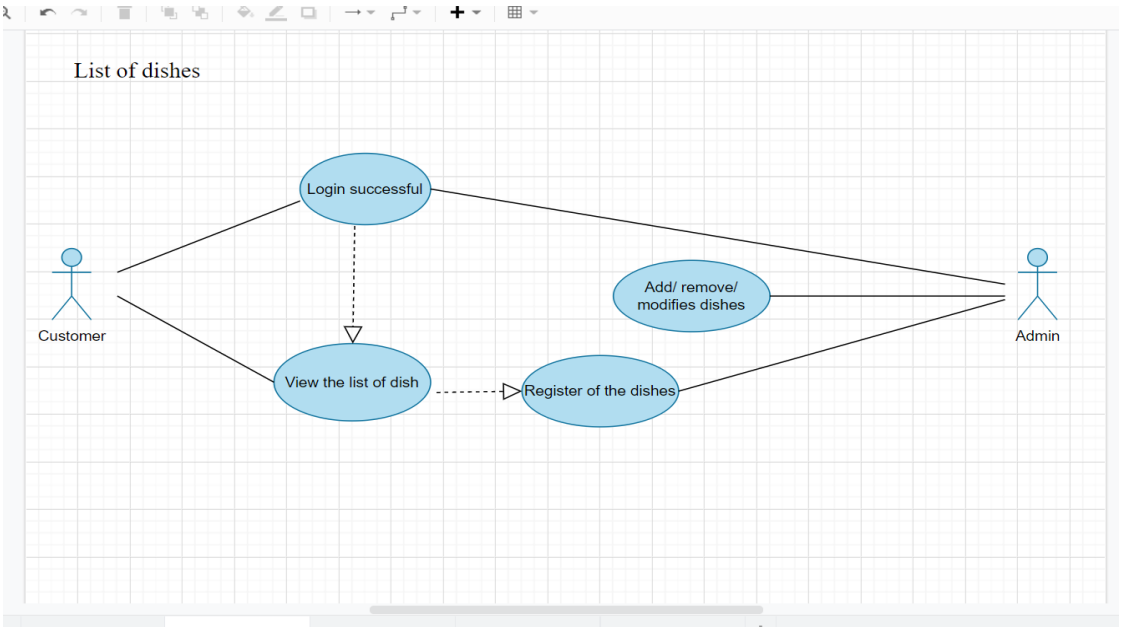
- **Brief Description:** A user logging in to the system.
- **Specification:** See Use-Case-Realization Specification: Login

Logout:



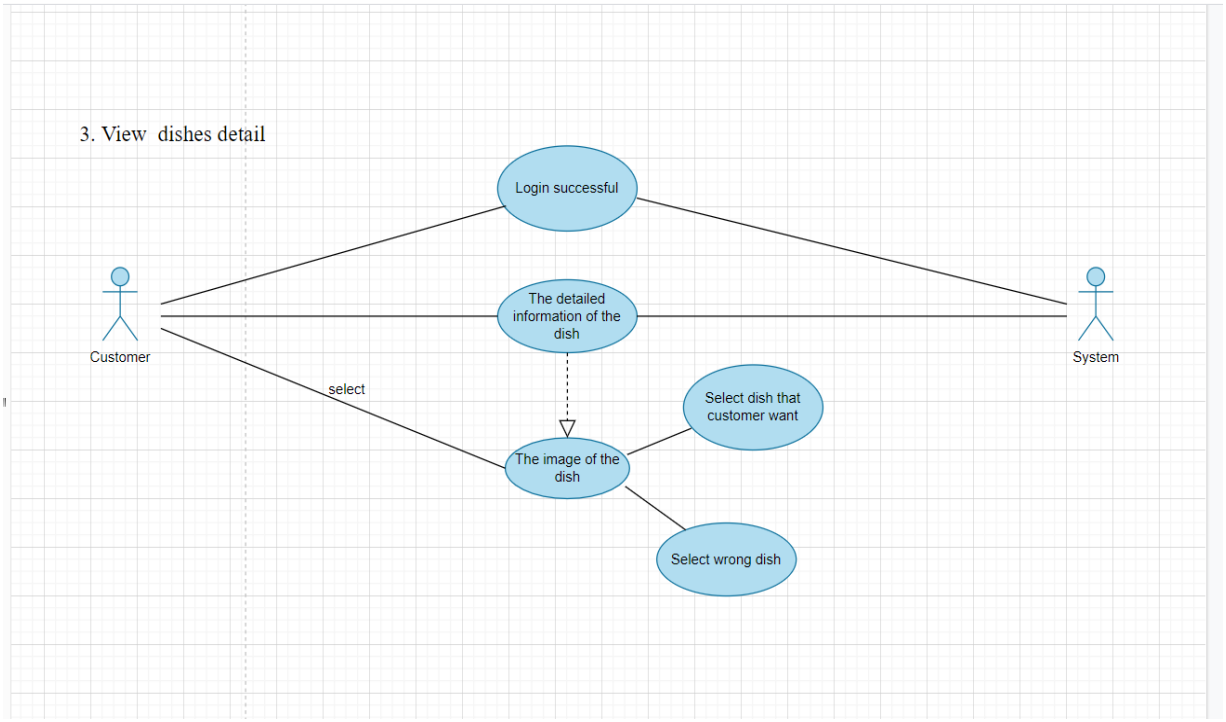
- **Brief Description:** A user logging out the system.
- **Specification:** See Use-Case-Realization Specification: Logout.

Display Products:



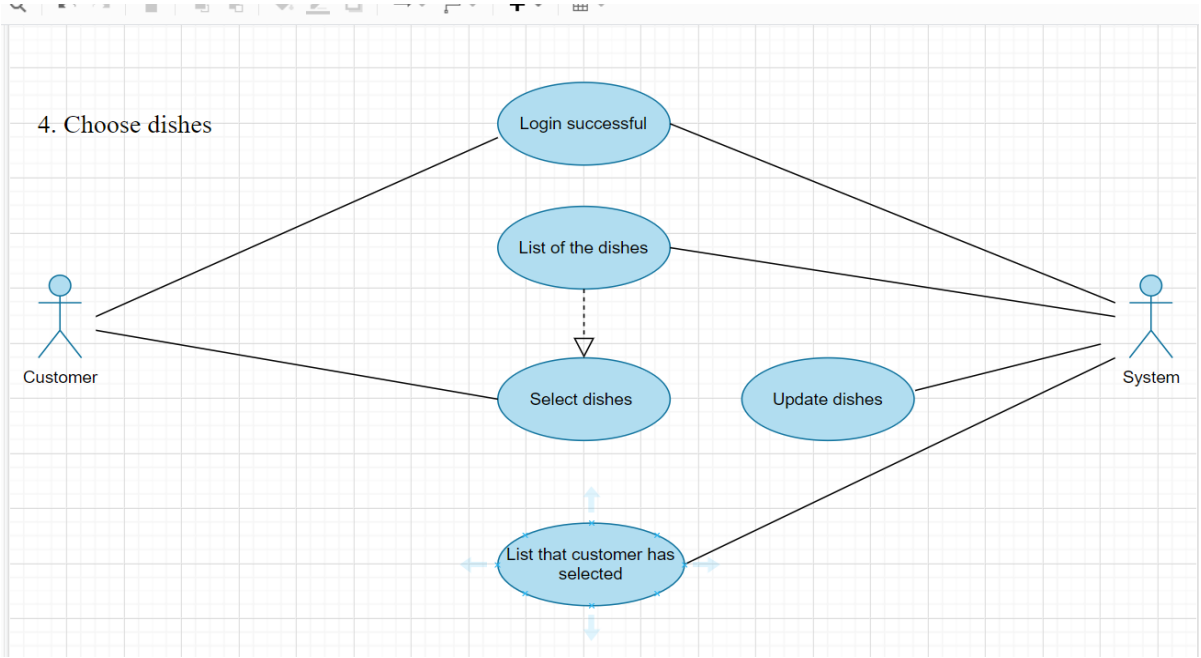
- **Brief Description:** A user displays all available products of the system
- **Specification:** See Use-Case-Realization Specification: Display Products

Display Product Detail:



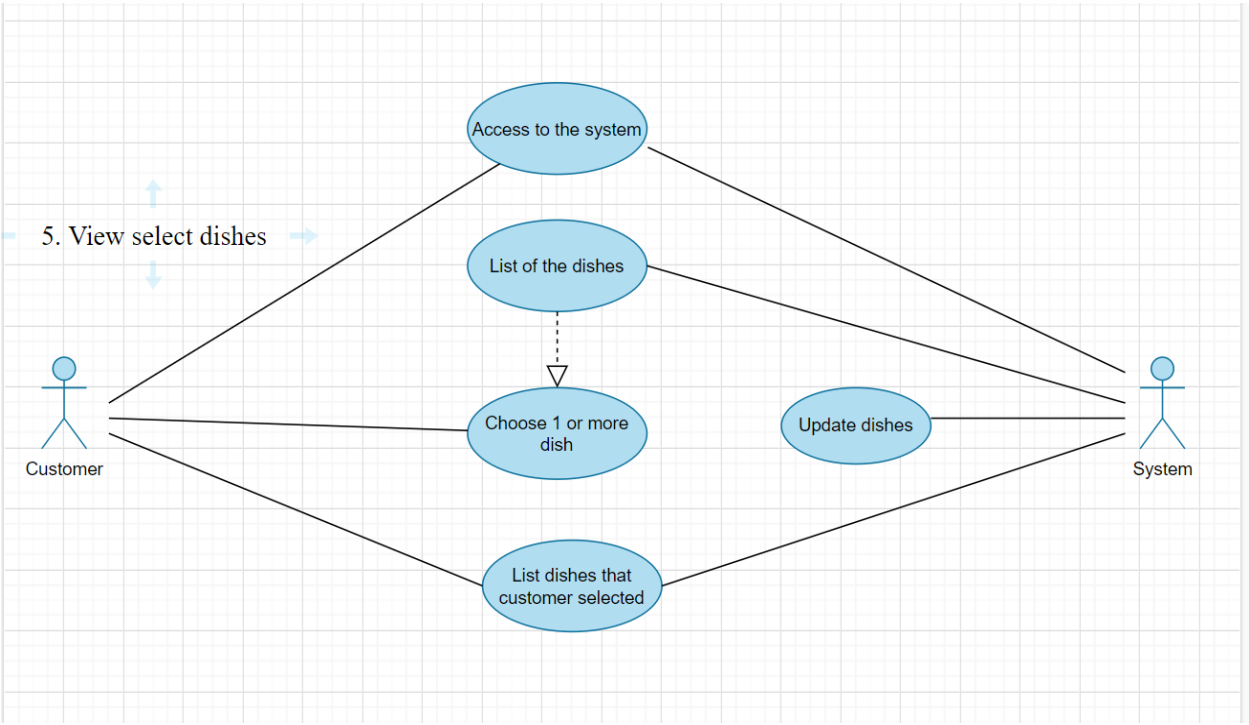
- **Brief Description:** A user displays detailed information of a product
- **Specification:** See Use-Case-Realization Specification: Display Product Detail

Choose Product:



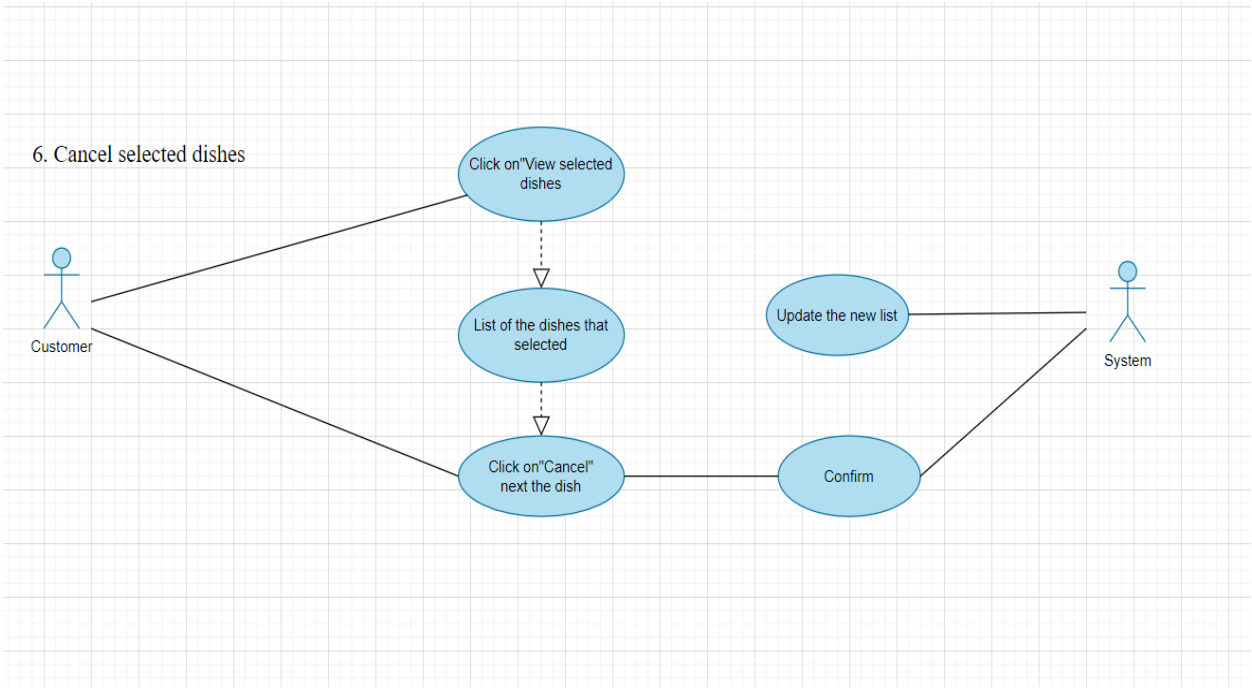
- **Brief Description:** A user chooses products based on search and/or filtering options.
- **Specification:** See Use-Case-Realization Specification: Choose Product

Display Selected Product:



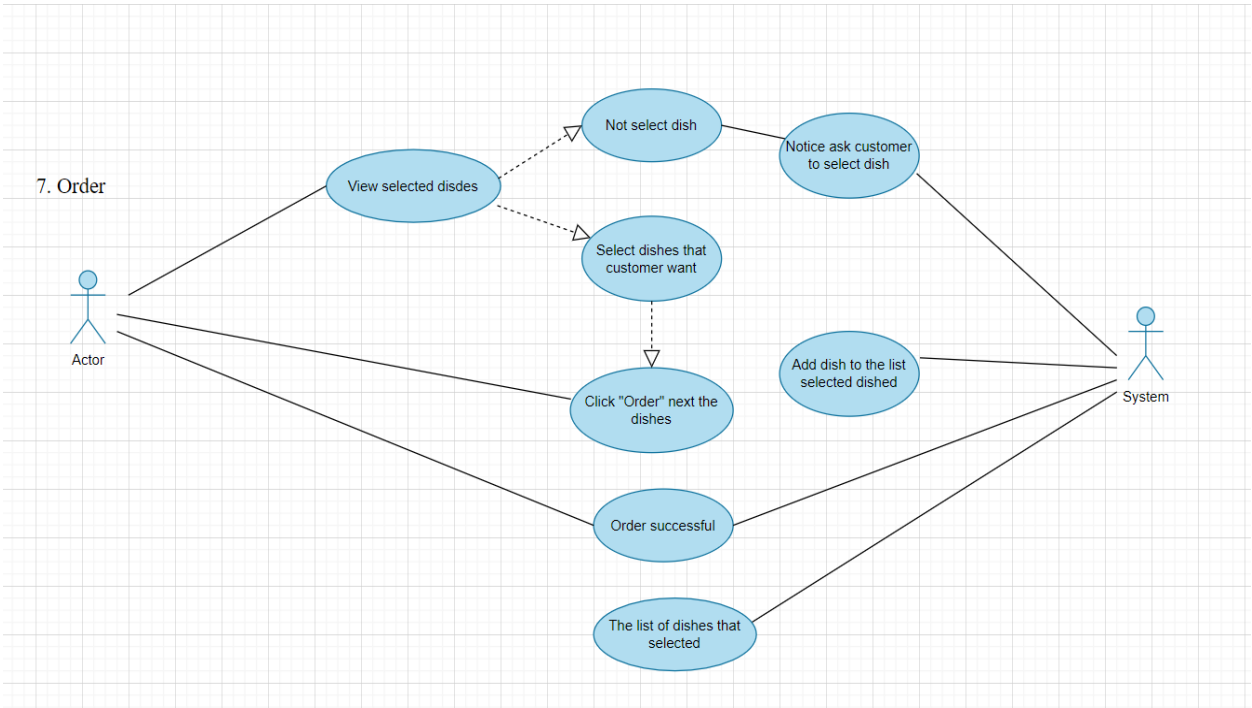
- **Brief Description:** A user displays list of selected products
- **Specification:** See Use-Case-Realization Specification: Display Selected Product

Cancel Selected Product:



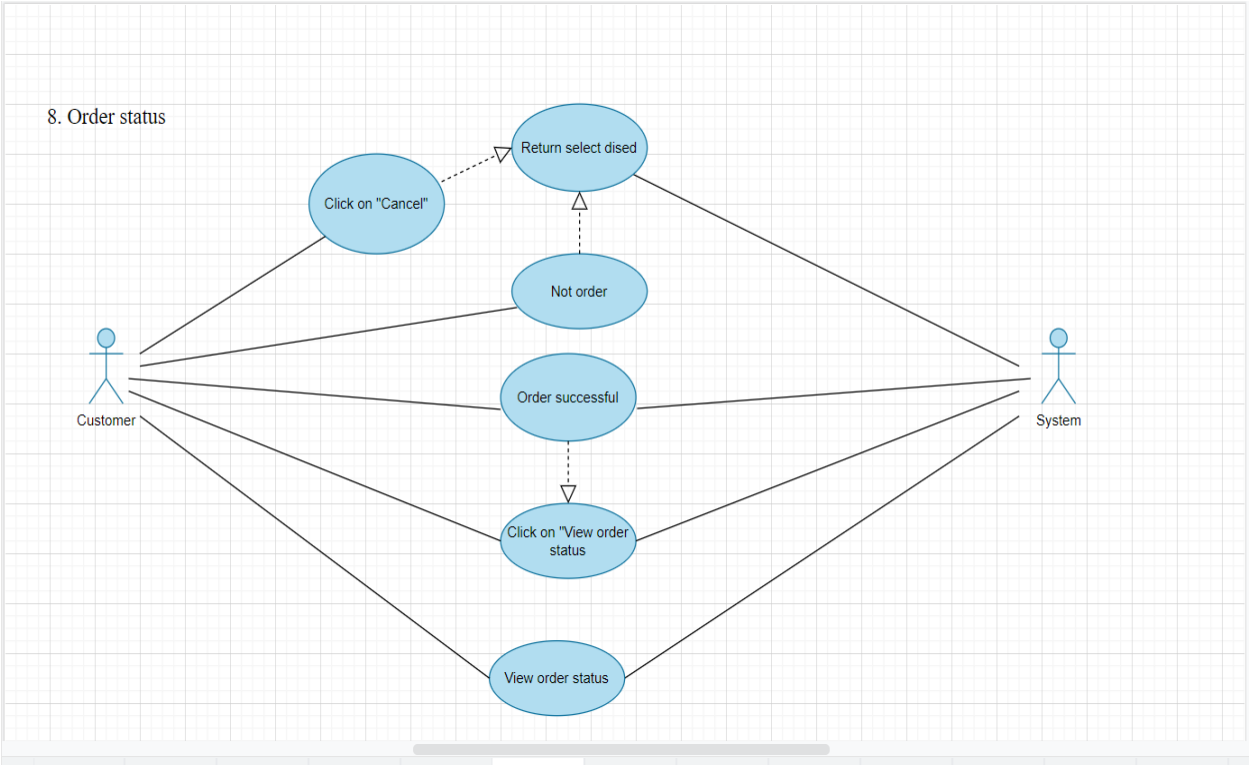
- **Brief Description:** A user unchecks the selected products
- **Specification:** See Use-Case-Realization Specification: Cancel Selected Product

Order Product:



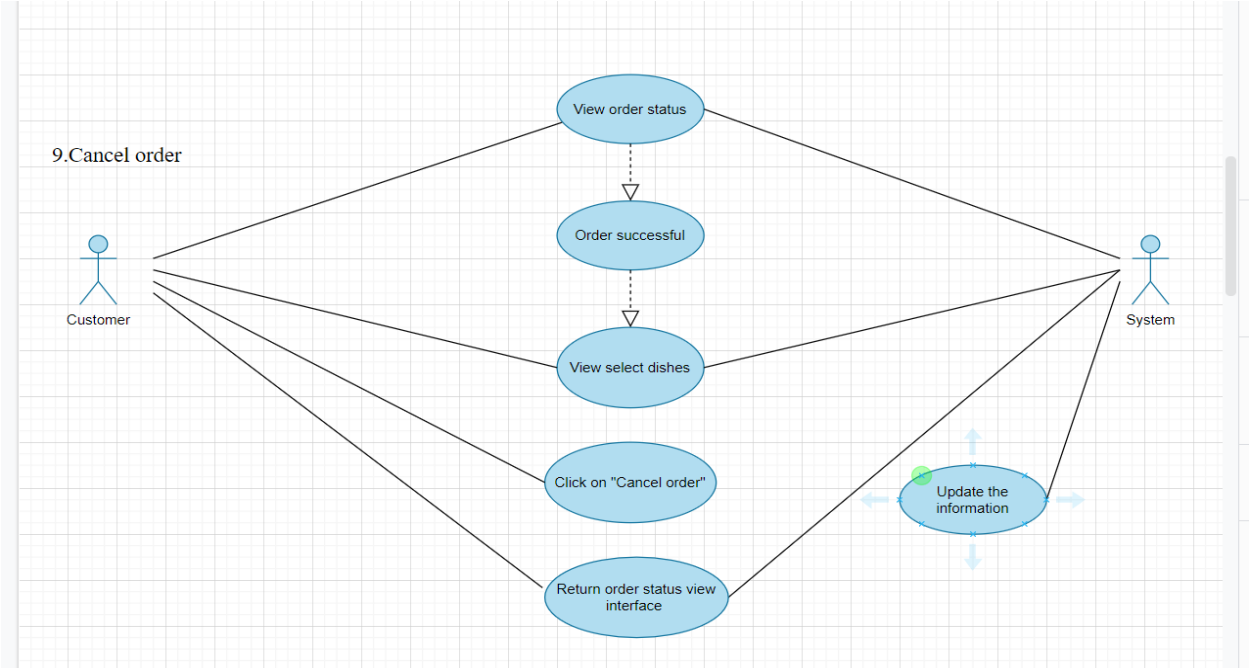
- **Brief Description:** A user orders the selected products online through the system
- **Specification:** See Use-Case-Realization Specification: Order Product

Display Order Status:



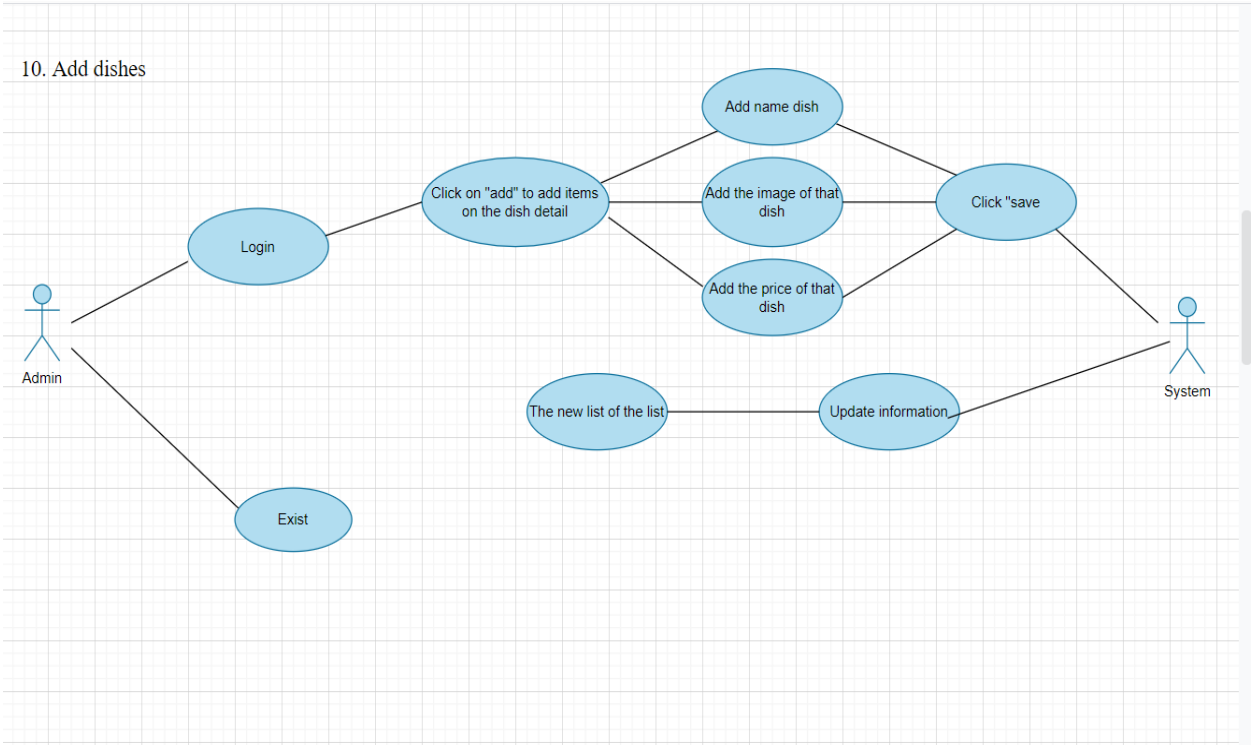
- **Brief Description:** A user displays order status after ordering
- **Specification:** See Use-Case-Realization Specification: Display Order Status

Cancel Order:



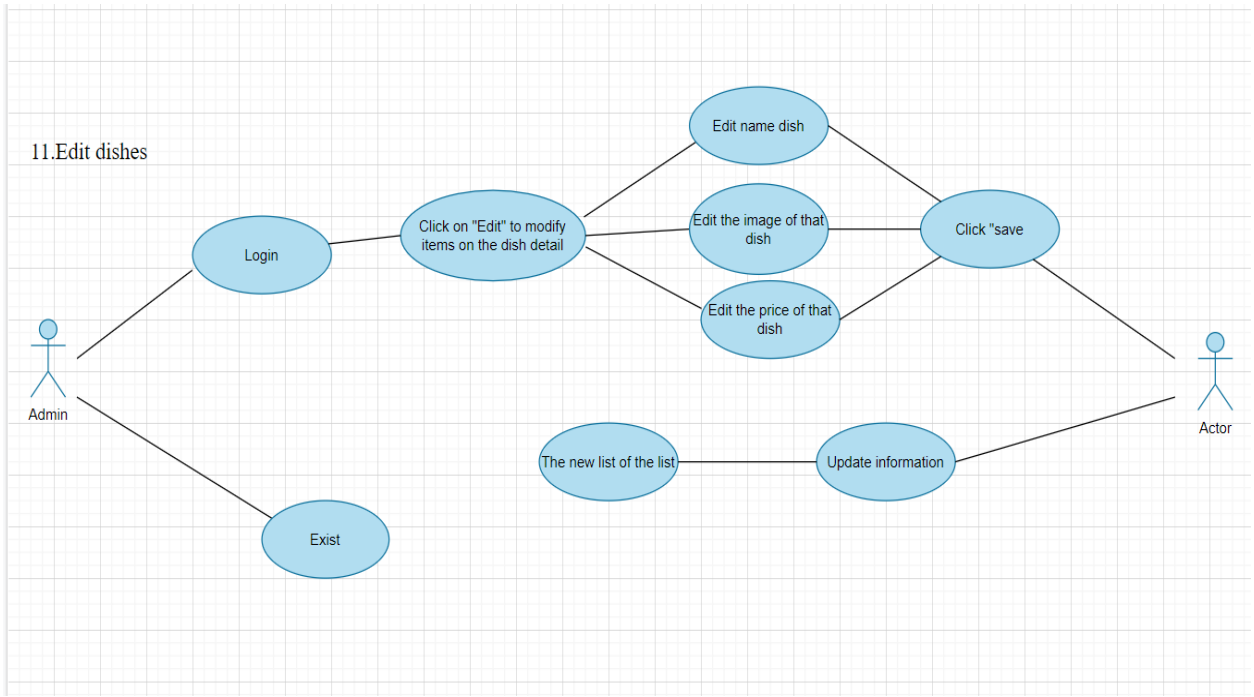
- **Brief Description:** A customer cancel purchase order.
- **Specification:** See Use-Case-Realization Specification: Cancel Order

Add Product:



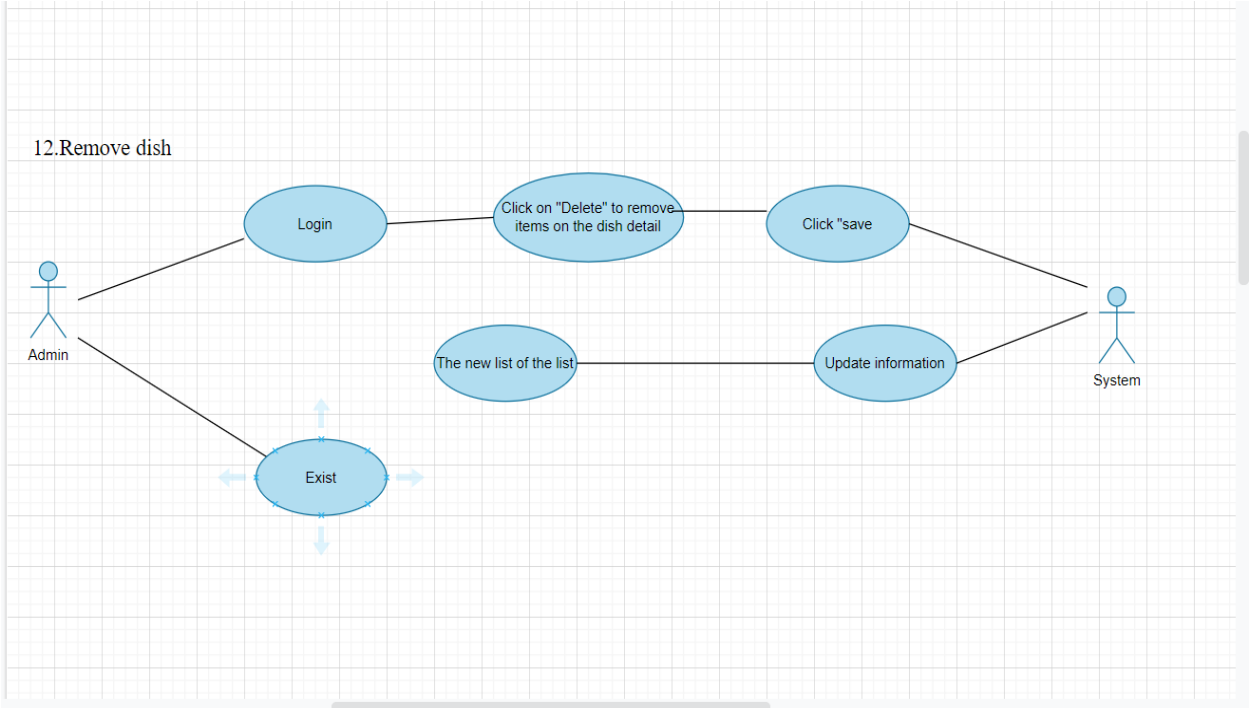
- **Brief Description:** Admin adds new product(s) to the database
- **Specification:** See Use-Case-Realization Specification: Add Product

Edit Product:



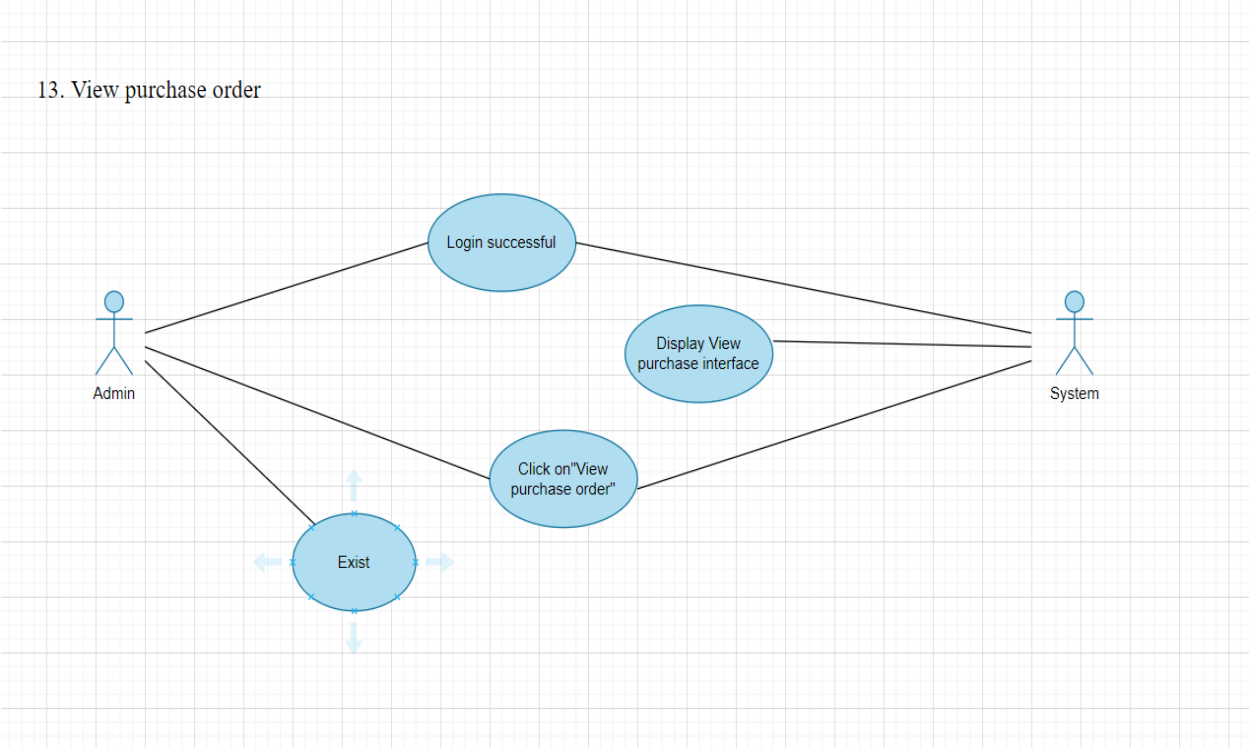
- **Brief Description:** Admin edits existing product(s) in the database
- **Specification:** See Use-Case-Realization Specification: Edit Product

Delete Product:



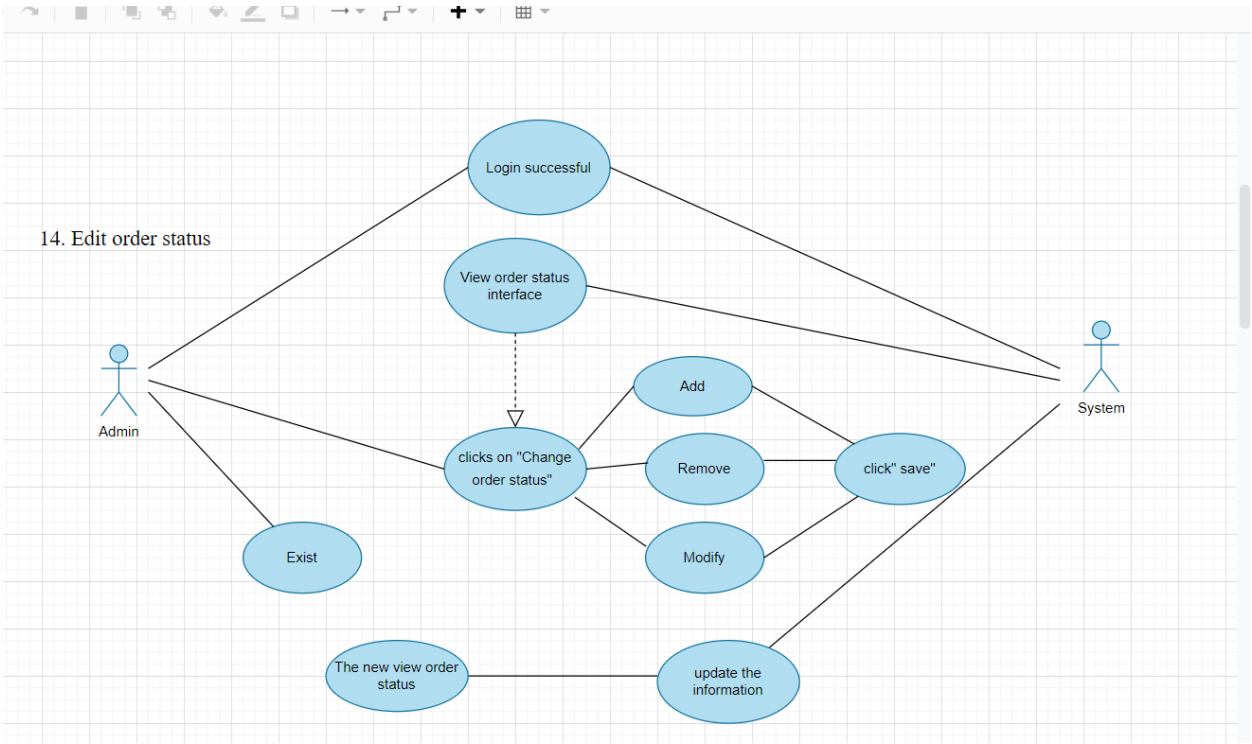
- **Brief Description:** Admin deletes product(s) from the database
- **Specification:** See Use-Case-Realization Specification: Delete Product

Display Purchase Order:



- **Brief Description:** Admin display the orders that have been ordered by customer
- **Specification:** See Use-Case-Realization Specification: Display Purchase Order

Edit Order Status:



- **Brief Description:** Admin edits purchase order status in the system
- **Specification:** See Use-Case-Realization Specification: Edit Order Status

5. Logical View

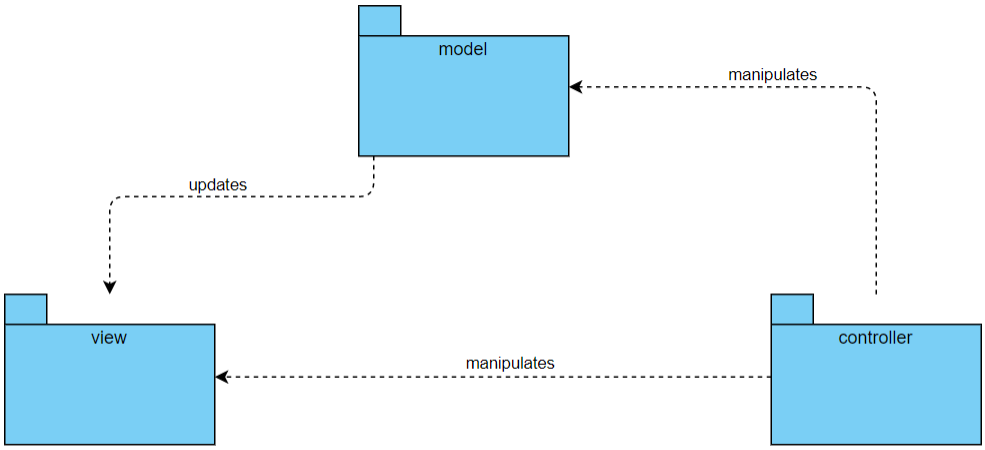
5.1 Overview

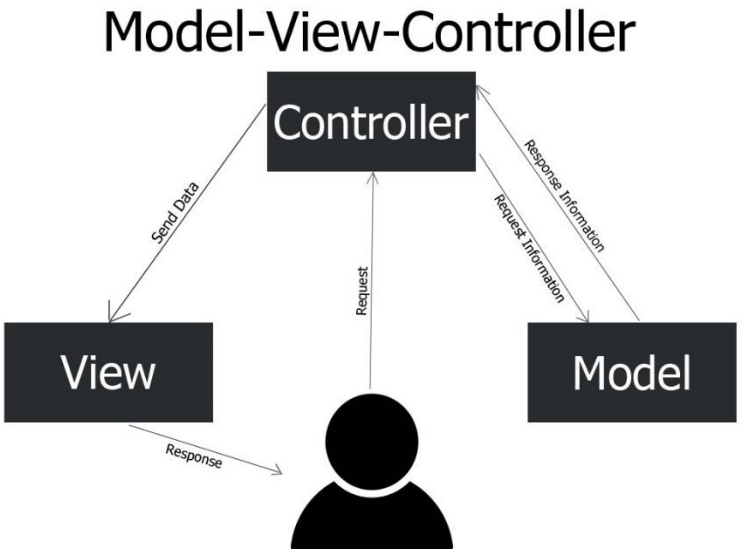
A description of the logical view of the architecture. Describes the overall decomposition of the design model in terms of package hierarchy and layers. The logical view of the Computer Shop Management System is comprised of 3 significant packages:

- **model:** contains classes that directly manages the data, logic and rules of the Computer Shop Management System and displayed in the view.

- **view:** contains classes that generates output representation of information to the user based on changes in the model.

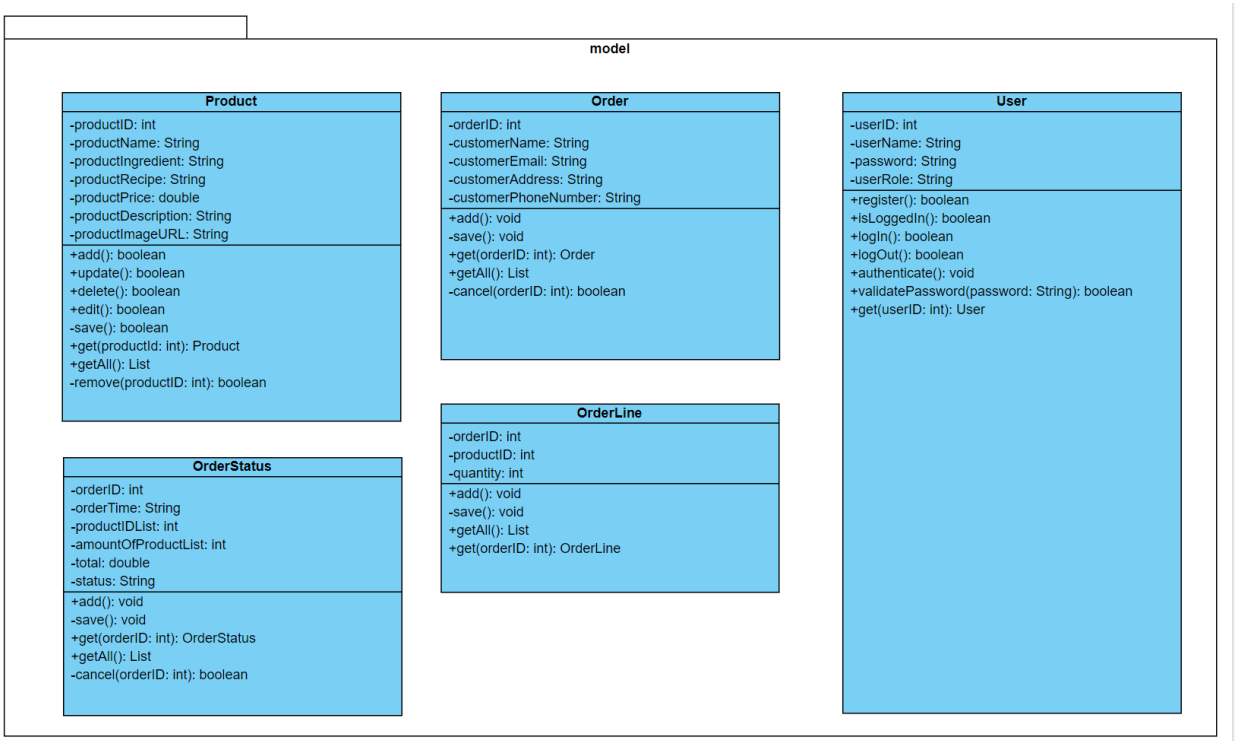
- **controller:** contains classes that can send commands to the model to update the model's state (e.g., add a new computer); it can also send commands to its associated view to change the view's presentation of the model (e.g., scrolling through computer's reviews).





5.2 Architecturally Significant Design Packages

5.2.1. Package model:



Name	model
Brief Description	Contains classes that directly manages the data, logic and rules of the Food Management System and displayed in the view.
Classes	Product, Order, OrderLine, OrderStatus.

Class Product:

Name	Product						
Brief Description	Data model for product table in database.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
productID	int	Private	False	False	N/A	1	N/A
productName	String	Private	True	False	200	N/A	N/A
productIngredient	String	Private	True	False	200	N/A	N/A
productRecipe	String	Private	True	False	200	N/A	N/A
productPrice	double	Private	True	False	N/A	1	N/A
productDescription	String	Private	True	False	550	N/A	N/A
productImageURL	String	Private	True	False	550	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
add()	boolean	Public	Instance	Add the computer this represent to database. Return true if success.

update()	boolean	Public	Instance	Update the product this represent to database. Return true if success.
delete()	boolean	Public	Instance	Delete the product this represent from database. Return true if success.
edit()	boolean	Public	Instance	Edit the product this represent to database. Return true if success.
save()	boolean	Private	Instance	Save changes from this to database.
get(int)	Product	Public	Classifier	Return a product in database with specified identifier.
getAll()	List	Public	Classifier	Return all products in database as a List.
remove(int)	boolean	Private	Classifier	Remove product in database with specified identifier. Return true if success.

Class Order:

Name	Order						
Brief Description	Data model for order table in database.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
orderID	int	Private	False	False	N/A	1	N/A
customerName	String	Private	True	False	50	N/A	N/A
customerEmail	String	Private	True	False	200	N/A	N/A
customerAddress	String	Private	True	False	200	N/A	N/A
customerPhoneNumber	String	Private	True	False	15	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
add()	void	Public	Instance	Add the order this represent to database. Return true if success.
save()	boolean	Private	Instance	Save changes from this to database.
get(int)	Order	Public	Classifier	Return an order in database with specified identifier.
getAll()	List	Public	Classifier	Return all orders in database as a List
cancel(int)	boolean	Private	Classifier	Remove orders in database with specified identifier.

Class OrderLine:

Name	OrderLine						
Brief Description	Data model for order_line table in database.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**

orderID	int	Private	False	False	N/A	1	N/A
productID	int	Private	False	False	N/A	1	N/A
quantity	int	Private	True	False	N/A	1	N/A

Operations

Header	Return Type	Access	Scope	Specification
add()	void	Public	Instance	Add the order line this represent to database. Return true if success.
save()	void	Private	Instance	Save changes from this to database.
get(int)	OrderLine	Public	Classifier	Return an order line in database with specified identifier.
getAll()	List	Public	Classifier	Return all order lines in database as a List

Class OrderStatus:

Name	OrderStatus
Brief Description	Data model for orders table in database.

Attributes

Name	Type	Access	Mutable	Optional	Length	Min	Max
orderID	int	Private	False	False	N/A	1	N/A
orderTime	String	Private	True	False	50	N/A	N/A
productIDList	int	Private	True	False	N/A	1	N/A
amountOfProductList	int	Private	True	False	N/A	1	N/A
total	double	Private	True	False	N/A	1	N/A
status	String	Private	True	False	45	N/A	N/A

Operations

Header	Return Type	Access	Scope	Specification
add()	void	Public	Instance	Add the order status this represent to database. Return true if success.
save()	void	Private	Instance	Save changes from this to database.
get(int)	OrderStatus	Public	Classifier	Return an order status in database with specified identifier.
getAll()	List	Public	Classifier	Return all order status in database as a List
cancel(int)	boolean	Private	Classifier	Remove orders in database with specified identifier.

Class User:

Name	User
Brief Description	Data model for user table in database.

Attributes

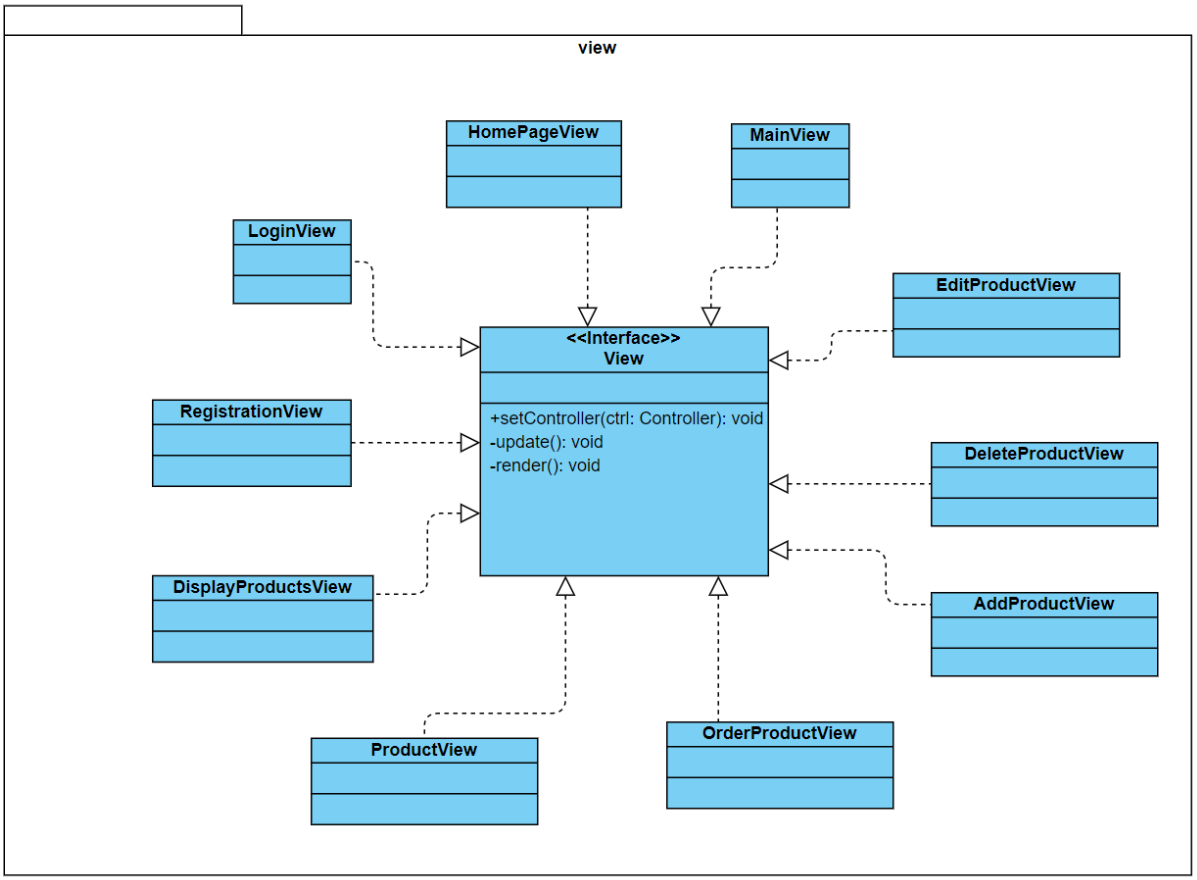
Name	Type	Access	Mutable	Optional	Length	Min	Max

userID	int	Private	False	False	N/A	1	N/A
userName	String	Private	True	False	50	N/A	N/A
password	String	Private	True	False	15	N/A	N/A
userRole	String	Private	True	False	20	N/A	N/A

Operations

Header	**Return Type**	**Access**	**Scope**	**Specification**
register()	boolean	Public	Instance	Register the user this represent and save to database. Return true if success.
isLoggedIn()	boolean	Public	Instance	Return true if the current user is logged in the system.
logIn()	boolean	Public	Instance	Log in the user this represent. Return true if success.
logOut()	boolean	Public	Instance	Log out the user this represent. Return true if success.
authenticate()	void	Private	Instance	Authenticate the user this represent. Grant the user an authentication token if the user is authenticated.
validatePassword(String)	boolean	Private	Instance	Validate the password of authenticating user by comparing hash and salt value with existing database hash and salt value in database.
get(int)	User	Public	Classifier	Return an user in database with specified identifier.

5.2.2. Package view:



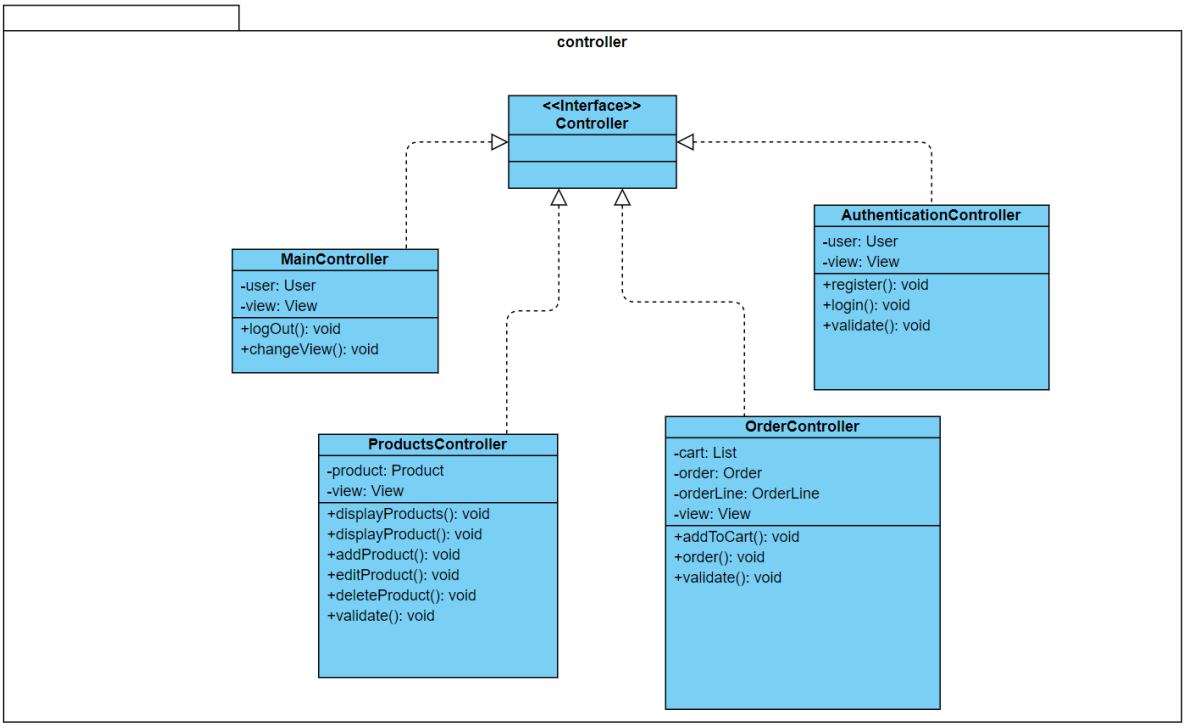
Name	view
Brief Description	Contains classes that generates output representation of Information to the user based on changes in the model.
Interfaces	View.
Classes	MainView, HomepageView, RegistrationView, LoginView, DisplayProductsView, ProductView, AddProductView, EditProductView, DeleteProductView, Order ProductView

Interface View:

Name	View			
Brief Description	Represents the visualization of the data that model contains.			
Implementing Classes	MainView, HomepageView, RegistrationView, LoginView, DisplayProductsView, ProductView, AddProductView, EditProductView, DeleteProductView, Order ProductView			
Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**

setController(Controller)	void	Public	Instance	Map this view with the specified controller.
update()	void	Public	Instance	Update this view based on changes in model
render()	void	Public	Instance	Render this view.

5.2.3. Package Controller:



Name	controller
Brief Description	Contains classes that directly manages the data, logic and rules of the Food Management System and displayed in the view.
Interfaces	Controller.
Classes	MainController, ProductsController, OrderController, AuthenticationController

Interface Controller:

Name	Controller
Brief Description	Controls the data flow into model object and updates the view whenever data changes.
Implementing Classes	MainController, ProductsController, OrderController,

	AuthenticationController

Class MainController:

Name	MainController						
Brief Description	Controller for the main functionality of the system.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
user	User	Private	True	False	N/A	N/A	N/A
view	View	Private	True	False	N/A	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
logOut()	void	Public	Instance	Handling log out request.
changeView()	void	Public	Instance	Handling change view request.

Class ProductsController:

Name	ProductsController						
Brief Description	Controller for handling operations related to computers.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
product	Product	Private	True	False	N/A	N/A	N/A
view	View	Private	True	False	N/A	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
displayProducts()	void	Public	Instance	Handling display products request.
displayProduct()	void	Public	Instance	Handling display product request.
addProduct()	void	Public	Instance	Handling add product request.
editProduct()	void	Public	Instance	Handling edit product request.
deleteProduct()	void	Public	Instance	Handling delete product request.
validate()	void	Private	Instance	Validate form inputs.

Class OrderController:

Name	OrderController
Brief Description	Controller for handling operations related to order.
Attributes	

Name	Type	Access	Mutable	Optional	Length	Min	Max
order	Order	Private	True	False	N/A	N/A	N/A
orderLine	OrderLine	Private	True	False	N/A	N/A	N/A
view	View	Private	True	False	N/A	N/A	N/A

Operations

Header	Return Type	Access	Scope	Specification
order()	void	Public	Instance	Handling customer order product request.
validate()	void	Private	Instance	Validate form inputs.

Class AuthenticationController:

Name	AuthenticationController
Brief Description	Controller for handling operations related to authentication.

Attributes

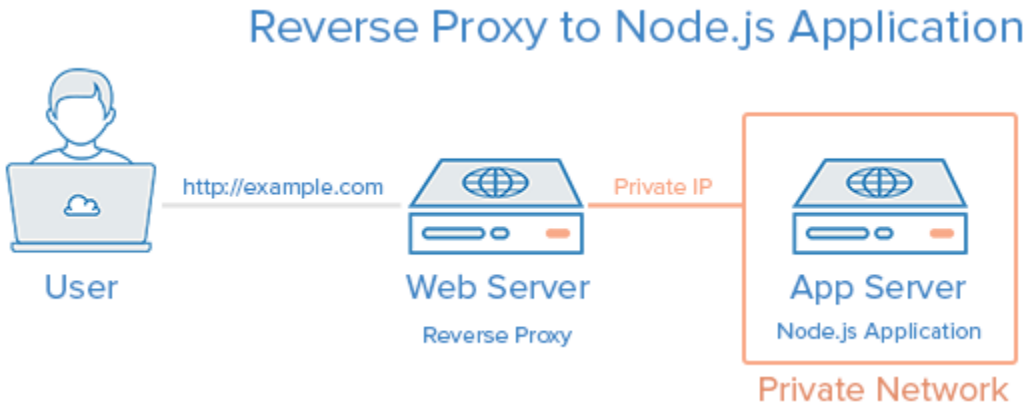
Name	Type	Access	Mutable	Optional	Length	Min	Max
user	User	Private	True	False	N/A	N/A	N/A
view	View	Private	True	False	N/A	N/A	N/A

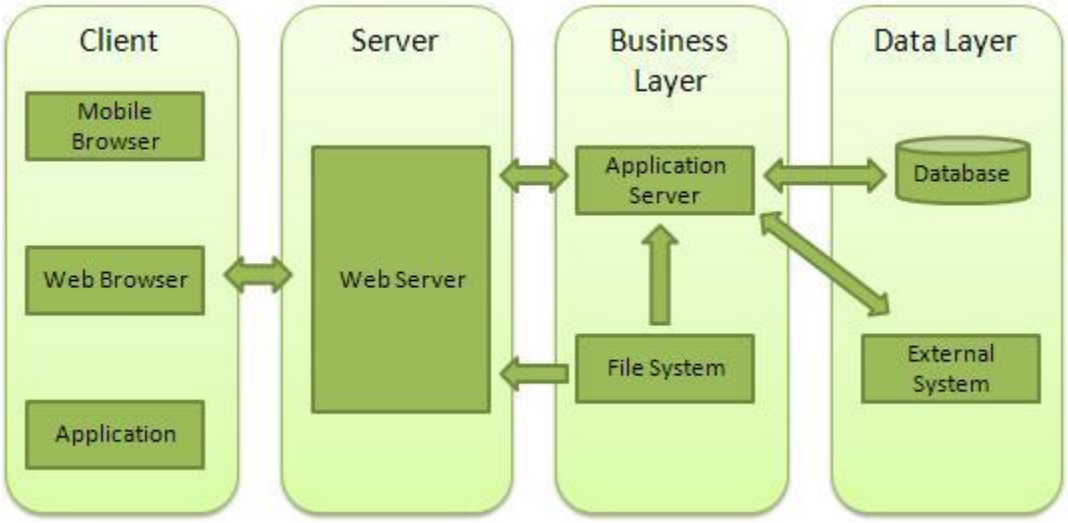
Operations

Header	Return Type	Access	Scope	Specification
register()	void	Public	Instance	Handling register request.
logIn()	void	Public	Instance	Handling log out request.
validate()	void	Private	Instance	Validate form inputs.

6. Process View

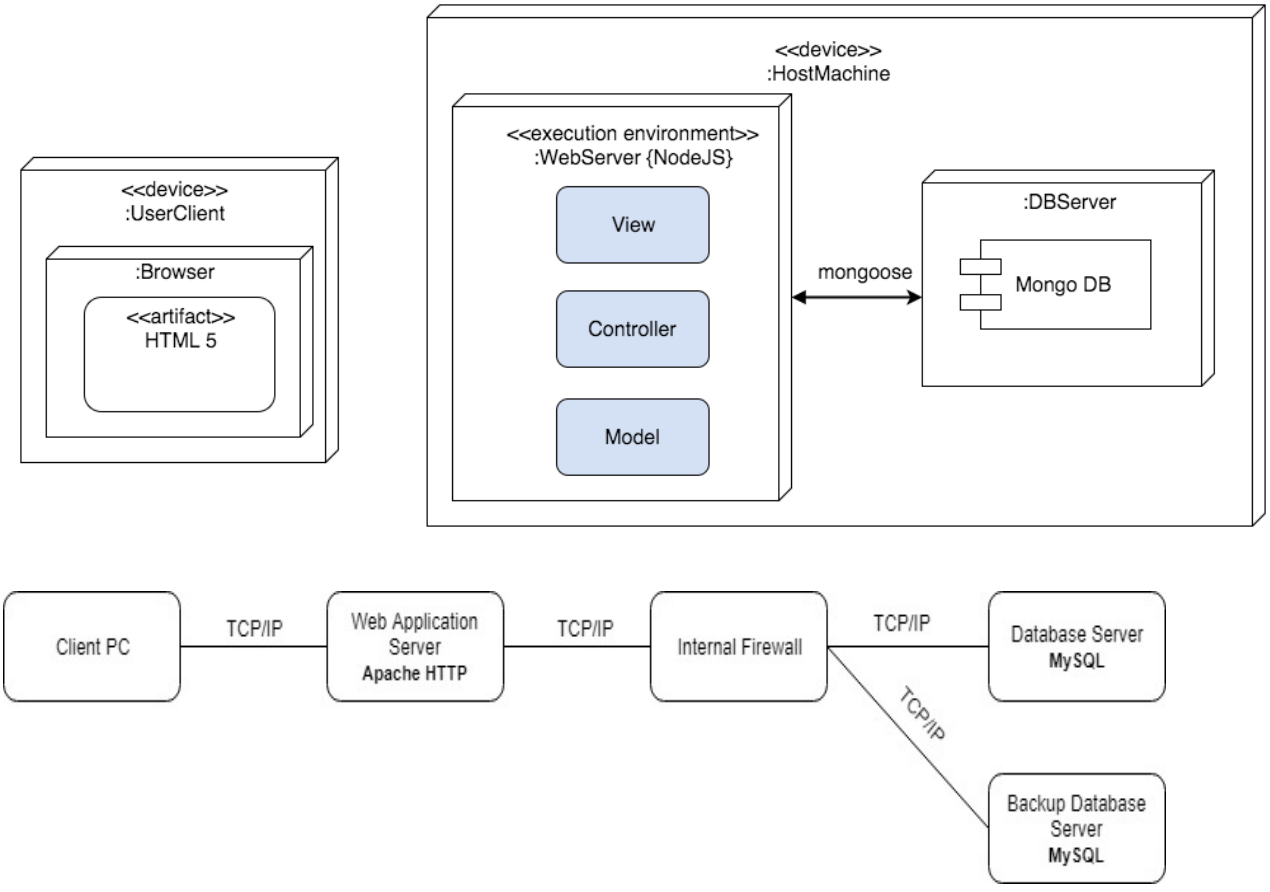
The Food Management System is designed to be implemented on Node.js server which support single-threaded asynchronous event handling (even loop); therefore, concurrency issues will not be considered in this document.





7. Deployment View

This section describes one or more physical network (hardware) configurations on which the Computer Shop Management System is deployed and run. The system is comprised of these mandatory physical nodes: two firewalls (internal and external), a web server, a database server and a backup database server. The diagram below is the simplicity version of the Food Management System deployment view.



8. Implementation View

The implementation of the system is strictly driven from the design; therefore, the implementation view will not be considered in this document

8.1 Overview

- The Implementation view depicts the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files). Usually, the layers of the Implementation view do fit the layering defined in the Logical view

8.2 Layers

8.2.1 Presentation Layer

- The Presentation layer contains all the components needed to allow interactions with an end-user. It encompasses the user interface

8.2.2 Control Layer

- The Control layer contains all the components used to access the domain layer or directly the resource layer when this is appropriate

8.2.3 Resource Layer

- The Resource layer contains the components needed to enable communication between the business tier and the enterprise information systems (Database, external services, ERP, etc…)

8.2.4 Domain layer

- The Domain layer contains all the components related to the business logic. It gathers all the subsystems that meet the needs of a particular business domain. It also contains the business object model.

8.2.5 Common Element Layer

- The Common Element layer contains the components re-used within several layers.

9. Data View (optional)

A description of the persistent data storage perspective of the Food Management System

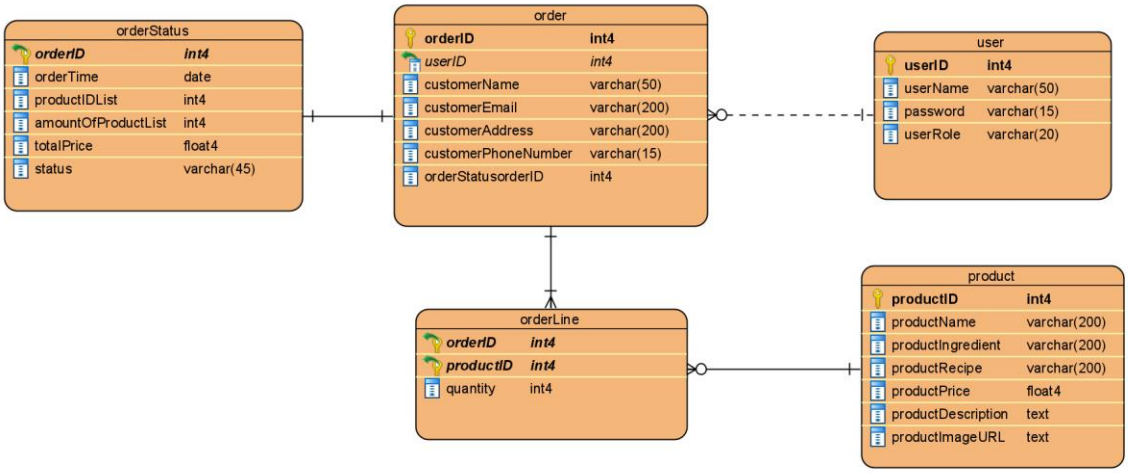


Table	Column	Type	Description	Length	Incl. in PK	Nullable	Unique
product	productID	integer	Unique identifier for each product	N/A	True	False	True

	productName	varchar	Name of product	200	False	False	False
	productIngredient	varchar	Ingredient of product	200	False	False	False
	productRecipe	varchar	Recipe of product	200	False	False	False
	productPrice	double	Price of product	N/A	False	False	False
	productDescription	text	Description of product	N/A	False	False	False
	productImageURL	text	Image of product	N/A	False	False	False
order	orderID	integer	Unique identifier for each order	N/A	True	False	False
	userID	integer	Unique identifier for each user	N/A	True	False	False
	customerName	varchar	Name of customer	50	False	False	False
	customerEmail	varchar	Email of customer	200	False	False	False
	customerAddress	varchar	Address of customer	200	False	False	False
	customerPhoneNumber	varchar	Phone number of customer	15	False	False	False
orderLine	orderID	integer	Reference to an order that this order line belong to	N/A	True	False	False
	productID	integer	Reference to an product that this order line belong to	N/A	True	False	False
	quantity	integer	Reference to product quantity that this order line belong to	N/A	False	False	False
orderStatus	orderID	integer	Reference to an order that this order status belong to	N/A	True	False	False
	orderTime	Date	Order date time of user	N/A	False	False	False
	productIDList	integer	Reference to list of products that this order status belong to	N/A	False	False	False
	amountOfProductList	integer	the number of products	N/A	False	False	False
	total	double	price total of products on	N/A	False	False	False

			order				
	status	varchar	Reference to status that this order belong to	45	False	False	False
user	userID	integer	Unique identifier for each user	N/A	True	False	False
	userName	varchar	Name of user	50	False	False	False
	password	varchar	Password for username	15	False	False	False
	userRole	varchar	Role of user	20	False	False	False

10. Size and Performance

The major dimensioning characteristics of the software that impact the architecture and performance constraints:

- The system will support up to 1000 concurrent users against the primary database at any given time, and up to 500 concurrent users against the local servers at any one time.
- The system must perform all functions with minimal time delays.
- The system must also accurately save all information transactions.

11. Quality

The system architecture supports the quality requirements:

- In order to maintain the highest degree of system integrity, the system is capable of ensuring that all information transitions are saved.
- Databases will be backed up on a daily basis in concern with safety implications.
- The system website is capable of display correctly on different devices web browser of any screen size (i.e. responsive design).
- All system website functions are available through popular web browsers; for instance, Google Chrome, Mozilla Firefox, Opera, Safari, Microsoft Edge, Internet Explorer.