Project I Report

Supermarket Chain Management



# Introduction

The goal of this assignment is to know as much as possible about creating the necessary structures for a Project creation, and with those, the necessity of the subjects Databases and Software Engineering. I hope that i’m able to take what’s essential on them to apply it here, filling up gaps such as the Relationship Model, Triggers, Selects, Use Cases, Contracts, Diagrams (of Communication, of Classes, etc) and such.

# Project Description

The Project applies to supermarket chains in the real world and can be considered as one. The customer/client purchases the product through the system, but not without registering his/herself first. The Store Manager takes care of everything related with the products. He can refill the stock when the ammounts are low and can add or remove products if the need arises. The General Manager is in charge of the discounts and promotions applied to the products and will add those as he sees fit. This constitutes a supermarket chain which will make it easier for the customer to purchase the products.

# Requirements Specifications

The system must:

* Add products
* Remove products
* Add promotions
* Remove Promotions
* Add discounts
* Remove discounts
* Allow the client to register, and consequently login.
* Allow the client to purchase products through the Internet, with methods such as Paypal, Credit Card or Purchase on Arrival
* Refill the stock of products if the ammounts are low

# Actors Functions

## General Manager:

* Insert Discount
* Remove Discount
* Insert Promotion
* Remove Promotion

## Store Manager:

* Insert Product
* Remove Product
* Refill Stock

## Client:

* Do Purchase
* Register

# Use Cases

## Use Case 1: Insert Discount

Domain: System

Level: Summary

Primary Actor: General Manager

Pre-condition: General Manager must be registered

Initiator: Discount is recognized by the system for entry

Primary success scenery:

* General Manager enters his credentials in the system
* System recognizes and informs Manager that there is a new discount ready to be inserted on the system
* Manager selects discount and requests system to add it to the list of discounts
* System asks for confirmation
* Manager confirms adition
* System validates and gives the insertion as complete

## Use Case 2: Remove Discount

Domain: System

Level: Summary

Primary Actor: General Manager

Pre-condition: General Manager must be registered

Initiator: Discount obsolete

Primary success scenery:

* General manager enters his credentials on the system
* System recognizes credentials and informs manager that the discount is obsolute and must be removed
* General manager asks the system to remove the discount
* System asks for removal confirmation
* Manager confirms
* System validates and removes discount

## Use Case 3: Insert Promotion

Domain: System

Level: Summary

Primary Actor: General Manager

Pre-condition: General Manager must be registered

Initiator: Promotion is recognized by the system for entry

Primary success scenery:

* General Manager enters his credentials in the system
* System recognizes and informs Manager that there is a new promotion ready to be inserted on the system
* Manager selects promotion and requests system to add it to the list of promotions
* System asks for confirmation
* Manager confirms adition
* System validates and gives the insertion as complete

## Use Case 4: Remove Promotion

Domain: System

Level: Summary

Primary Actor: General Manager

Pre-condition: General Manager must be registered

Initiator: Promotion obsolete

Primary success scenery:

* General manager enters his credentials on the system
* System recognizes credentials and informs manager that the promotion is obsolute and must be removed
* General manager asks the system to remove the promotion
* System asks for removal confirmation
* Manager confirms
* System validates and removes promotion

## Use Case 5: Insert Product

Domain: System

Level: Summary

Primary Actor: Store Manager

Pre-condition: Store Manager must be registered

Initiator: Product is recognized by the system for entry

Primary success scenery:

* Store Manager enters his credentials in the system
* System recognizes and informs Manager that there is a new product ready to be inserted on the system
* Manager selects product and requests system to add it to the list of products
* System asks for confirmation
* Manager confirms adition
* System validates and gives the insertion as complete

## Use Case 6: Remove Product

Domain: System

Level: Summary

Primary Actor: Store Manager

Pre-condition: Store Manager must be registered

Initiator: Product obsolete

Primary success scenery:

* Store manager enters his credentials on the system
* System recognizes credentials and informs manager that the product is obsolute and must be removed
* Store manager asks the system to remove the product
* System asks for removal confirmation
* Manager confirms
* System validates and removes product

## Use Case 7: Refill Stock

Domain: System

Level: Summary

Primary Actor: Store Manager

Pre-condition: Store Manager must be registered

Initiator: Stock list is recognized by the system

Primary success scenery:

* Store manager enters his credentials on the system
* System recognizes Manager and informs him that some products are out of stock and they must be refilled with a given list
* Store Manager consults the list and asks the system to take care of the refill
* System asks for confirmation
* Store Manager confirms stock refill
* System validates and increases the ammounts of the products by the provided list

## Use Case 8: Do Purchase

Domain: System

Level: Summary

Primary Actor: Client

Pre-condition: Client must be registered

Initiator: Client wants to purchase a(several) product(s)

Primary success scenery:

* Client enters his credentials on the system
* System validates and informs client that there are products available for purchase
* Client selects the product(s) he wants to purchase and proceeds to the checkout
* System validates an asks Client for method of payment
* Client selects method of payment
* System validates and confirms purchase

## Use Case 9: Register

Domain: System

Level: Summary

Primary Actor: Client

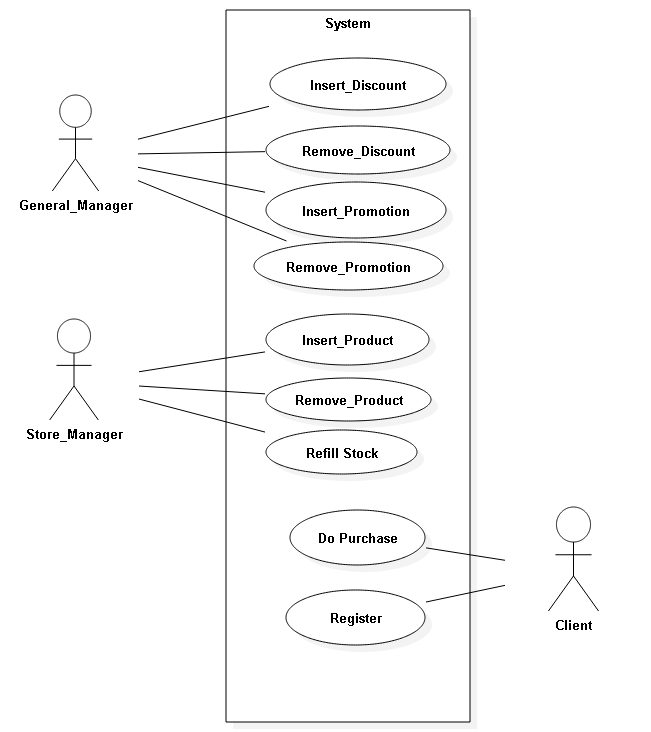
Pre-condition: None

Initiator: Client isn’t registered

Primary success scenery:

* Client decides to register himself on the system
* System acknowledges and gives the Client a form with the data he must fill to register himself successfully
* Client fills form and confirms registering
* System validates and confirms successfull registration

# Use Case Diagram



# Contracts

## Contract 1: Insert Discount

**System Operation:** Insert Discount (discount\_type\_id)

**Responsability:** The system must be able to insert a discount

**Type:** System

**Cross References**: Use Case 1: Insert Discount

**Notes:** Discount management is used in this contract

**Output:**

* Error message: Failed to insert discount – missing data
* Success message: Discount correctly inserted

**Pre-condition:**

* There must be a Discount\_Type Class
* There must be a Discount Class
* There must exist associations between these two classes
* There must exist at least one discount type

**Post-condition:**

* The Discount must be correctly inserted

## Contract 2: Remove Discount

**System Operation:** Remove Discount (discount\_type\_id)

**Responsability:** The system must be able to remove a discount

**Type:** System

**Cross References**: Use Case 2: Remove Discount

**Notes:** Discount management is used in this contract

**Output:**

* Success message: Discount correctly removed

**Pre-condition:**

* There must be a Discount\_Type Class
* There must be a Discount Class
* There must exist associations between these two classes
* There must exist at least one discount type

**Post-condition:**

* The Discount must be correctly removed
* The associations between the two classes and discount must be removed

## Contract 3: Insert Promotion

**System Operation:** Insert Promotion (promotion\_type\_id)

**Responsability:** The system must be able to insert a promotion

**Type:** System

**Cross References**: Use Case 3: Insert Promotion

**Notes:** Promotion management is used in this contract

**Output:**

* Error message: Failed to insert promotion – missing data
* Success message: Promotion correctly inserted

**Pre-condition:**

* There must be a Promotion\_Type Class
* There must be a Promotion Class
* There must exist associations between these two classes
* There must exist at least one promotion type

**Post-condition:**

* The Promotion must be correctly inserted

## Contract 4: Remove Promotion

**System Operation:** Remove Promotion (promotion\_type\_id)

**Responsability:** The system must be able to remove a promotion

**Type:** System

**Cross References**: Use Case 4: Remove Promotion

**Notes:** Promotion management is used in this contract

**Output:**

* Success message: Promotion correctly removed

**Pre-condition:**

* There must be a Promotion\_Type Class
* There must be a Promotion Class
* There must exist associations between these two classes
* There must exist at least one promotion type

**Post-condition:**

* The Promotion must be correctly removed
* The associations between these two classes and promotion must be removed

## Contract 5: Insert Product

**System Operation:** Insert Product (discount\_id,promotion\_id,product\_type\_id)

**Responsability:** The system must be able to insert a product

**Type:** System

**Cross References**: Use Case 5: Insert Product

**Notes:** Product management, Discount management and Promotion management are used in this contract

**Output:**

* Error message: Failed to insert product – missing data
* Success message: Product correctly inserted

**Pre-condition:**

* There must be a Discount Class
* There must be a Promotion Class
* There must be a Product\_Type Class
* There must exist associations between these three classes
* There must exist at least one product type

**Post-condition:**

* The Product must be correctly inserted

## Contract 6: Remove Product

**System Operation:** Remove Product (discount\_id,promotion\_id,product\_type\_id)

**Responsability:** The system must be able to remove a product

**Type:** System

**Cross References**: Use Case 6: Remove Product

**Notes:** Product management, Discount management and Promotion management are used in this contract.

**Output:**

* Success message: Product correctly removed

**Pre-condition:**

* There must be a Discount Class
* There must be a Promotion Class
* There must be a Product\_Type Class
* There must exist associations between these three classes
* There must exist at least one product type

**Post-condition:**

* The Product must be correctly removed
* The associations between the three classes and product must be removed

## Contract 7: Refill Stock

**System Operation:** Refill Stock (ammount)

**Responsability:** The system must be able to refill the products stock when it’s low

**Type:** System

**Cross References**: Use Case 7 : Refill Stock

**Notes:** Product management is used in this contract

**Output:**

* Error message: Stock list not specified
* Success message: Stock refilled

**Pre-condition:**

* There must be a Product class
* There must be products with a low ammount

**Post-condition:**

* The stock must be correctly refilled

## Contract 8: Do Purchase

**System Operation:** Do Purchase (product\_id,client\_id)

**Responsability:** The System must be able to allow the client to purchase products

**Type:** System

**Cross References**: Use Case 8: Do Purchase

**Notes:** Client management, Product Management and Purchase management are used in this contract.

**Output:**

* Success message: Item successfully purchased
* Error message: Not enough money on your account
* Error message: Wrong or unexistant zipcode

**Pre-condition:**

* There must be a Client Class
* There must be a Product Class
* There must be a Purchase Class
* All of the above classes must be associated with eachother
* The client must have a correct address

**Post-condition:**

* The purchase must be successfull
* The product ammount must decrease during the purchase of the products

## Contract 9: Register

**System Operation:** Register()

**Responsability:** The system must be able to let the Client register successfully (or Store Manager or General Manager)

**Type:** System

**Cross References**: Use Case 9: Register

**Notes:** Person management, Login management, Client(or others) management

**Output:**

* Success message: Successfully registered
* Error message: Missing fields
* Error message: Validation wrong (password)

**Pre-condition:**

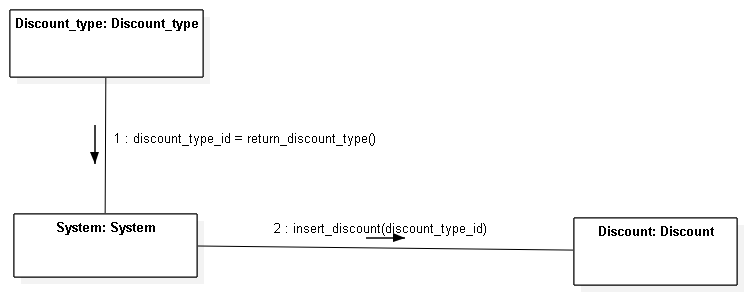
* There must be a Person Class
* There must be a Login Class
* There must be a Client class (or any other class tied with Person and Login)
* All of these must be associated

**Post-condition:**

* The client(or any other person) must be successfully registered

# Comunication Diagrams

## Diagram 1: Insert Discount



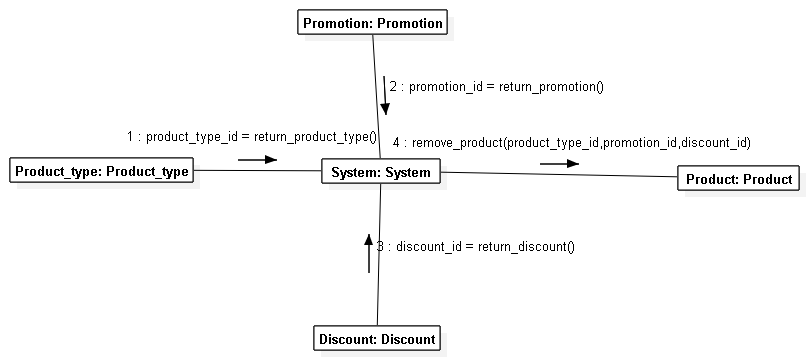
## Diagram 2: Remove Discount

## Diagram 3: Insert Promotion

## Diagram 4: Remove Promotion

## Diagram 5 : Insert Product

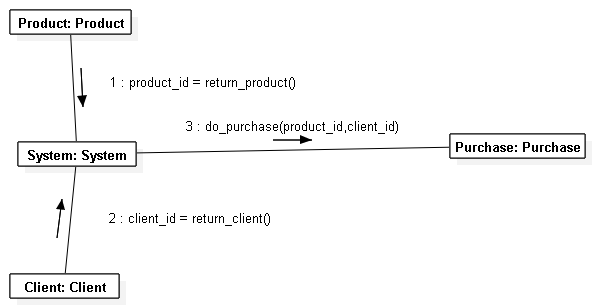
## Diagram 6: Remove Product



## Diagram 7: Refill Stock



## Diagram 8: Do Purchase



## Diagram 9: Register



# Class Diagram (No Associations)

# Class Diagram (Associations)

# Class Diagram (Direct Associations)