**Documentation**

**Purpose**

The purpose of this software is to renovate the system of the market for more comfortable usage.  
Using this software the staff of supermarket will do their job faster and effectively however they should pass the short tutorial how to use this system for further usage.

**Scope**

The software will be used by cashiers, managers, warehouse workers.

Cashiers will use this software to register products that were sold and make report about amount of money in the register at the end of his shift. Manager can check the list of products, from which he can select those that have expired or those that the store ran out of. If some goods need to be replaced the manager can place the order for them. He is also able to manage the employees: hire or fire them and change salary. Every month the system makes a sales report to the head office. Warehouse workers make records of the delivered products, inputting both the value received and the amount ordered. The system will keep track of the discrepancies and will display them to the manager when asked to.

This software can be used in almost every supermarket as it has simple principle. It does not require powerful computers for doing the requests. Everybody from the staff have their identification for login and their specific workplaces for making the tasks.

There is no any hard compound dependencies, so the software is easy to use by workers.

**Use case scenario (Cashier service client)**

Cashier is serving the client. Cashier has to ask if client has saving or discount card, if client has one of these cards then cashier starts to scan products. In the case if client do not have any cards then cashier suggests him/her if they would like to have one of them, if yes then cashier creates the card, which type will be chosen depends on the client`s decision and then the process of scanning products starts. If client is not satisfied with price of some product or other factors than cashier can delete the product from the purchase if everything is ok then cashier presses the payment button and enters the amount of given money in the required field and if the system confirms that given sum and calculated is the same than it will create the receipt and print it , in the case if the amount will be bigger than the system will show the amount of change and print the receipt if the given amount is smaller than system will show the error message .Cashier will inform client with this problem and cashier has to return to the previous window with the list of products and ask client whether to delete some products for needed amount of money or to delete the whole purchase. If client wants to delete some product items than cashier will repeat the action with pressing the payment. If client do not want to pay than cashier will delete whole purchase.

**Discussion about changes made in design diagram**

1.Added class **ClassAttribute**

I have used this class for writing my static variables. As in database there is no sense to save static attributes in every created object so it will be easier to represent this attributes in separate class for using them in methods.

2.Comment referred to **positions[1…\*] in Employee**

The comment makes the diagram more readable so in this case my comment shows the positions which can be used in this class such as: Cashier, Manager, WarehouseWorker.

3.Class **Employee**

Class Employee was change comparing to the Analytical diagram where Employee was an abstract class. In this case, I have made one class which has positions which identify the position of employee. Implementation made as single table as the difference between classes is methods and position of Employee so it will be easier to save it in one class. There will not be displayed many null values in database except of those fields which are optional.

4. Method **payThePurchase()**

I have added this method because during work with my project this method becomes required in my class Client.

5.Attribute **status**

I have added this attribute during the process with my use case which requires it. This attribute is gotten from enum{ blocked, non-active, unknown}. The role of attribute is to sort clients by this criteria and delete those who are non-active in some period.

6.Class **ProductAmount**

This class was involved in with an attribute association. Accordingly to the lecture in dynamic diagram such type of association has to be made as a separate class with appropriate cardinalities.

7.Class **ProductLastingType**

This class was created for the purpose of making code more readable. As in class product was attributes such as : productionDate , expirationDate. So I decided to make class which will be responsible for lasting type (ProductLastinType) it will consist of those attributes which were deleted from Product class, it means that my class becomes abstract and will have Perishable and LongLasting classes. During creation the product I will create it with specific type where ProductionDate and ExpirationDate will be included so in this purpose person will know if this product is perishable or long lasting.

8.Interface **Perishable** and **LongLasting**

As these classes have only methods so it is easier to make them as interfaces. In this case the LastingProductType will implements them.

9. Attribute such as **TotalSum , Receipt**

All derived attribute were rewritten into the getTotalSum and getReceipt correspondently as in the design diagram we represents changes which will be used in the implementation so derived attributes are presented as methods.

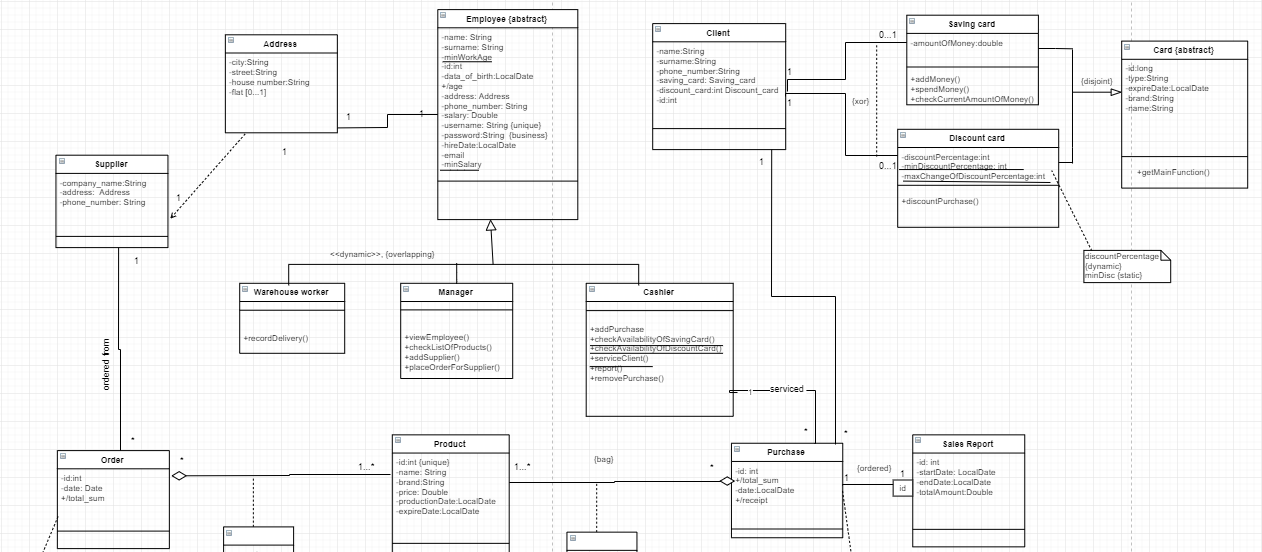
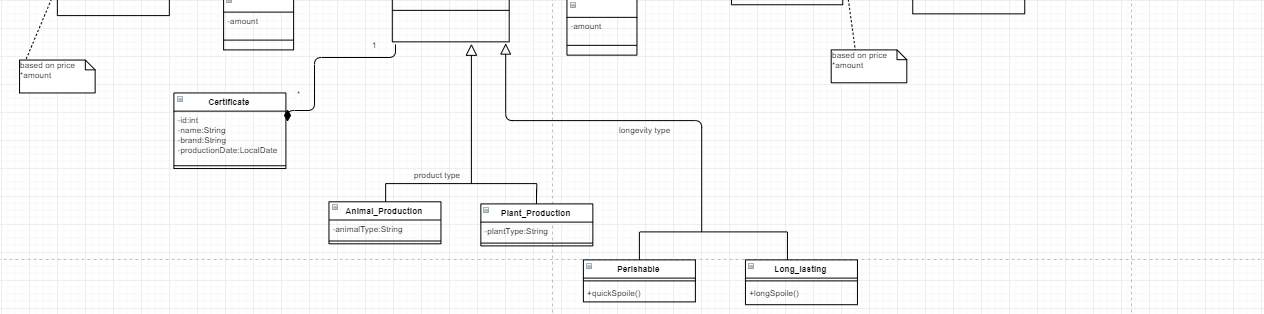
10.Method **makeDisount()**

I have created this method in class Product as during work with my state diagram I decided to make this method.

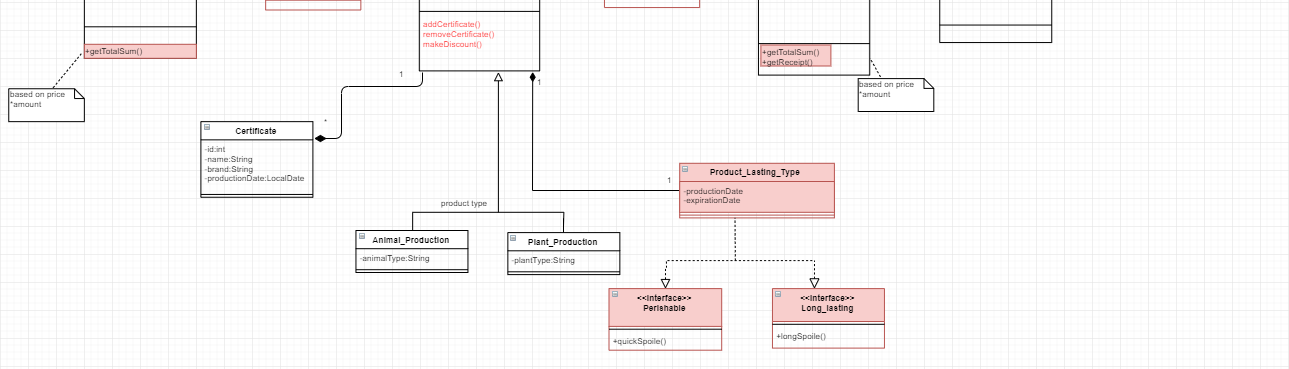
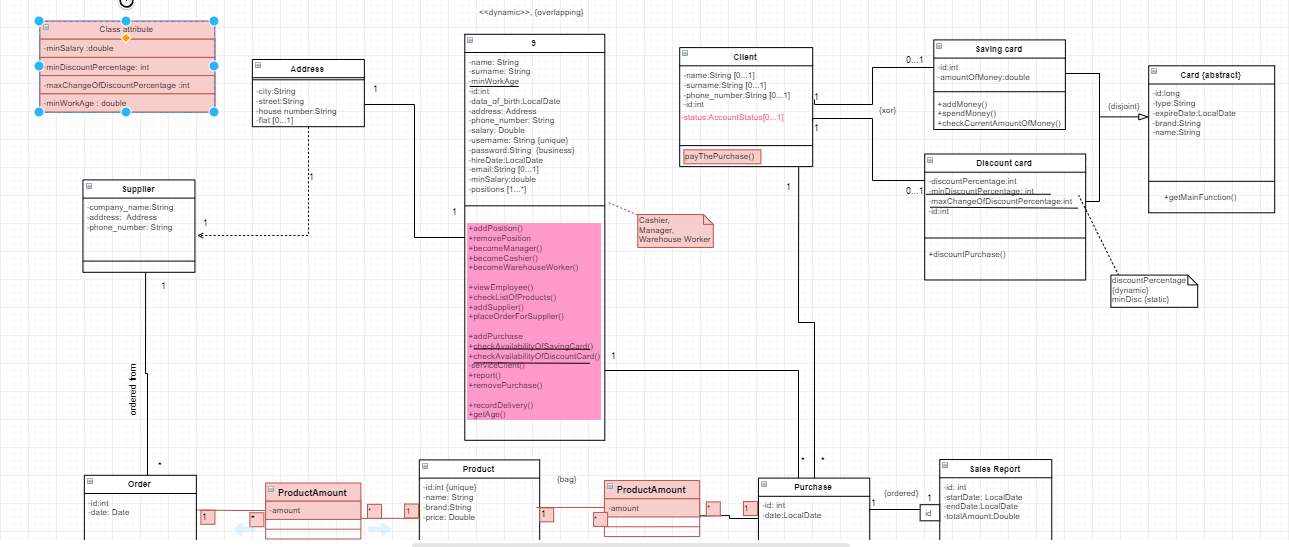
11.Class **Card**

Card is an abstract class which has two inherited classes such as DiscountCard and SavingCard.  
Implementation is made as single table, class Card will have column type which can identify to which type refers these inherited classes.

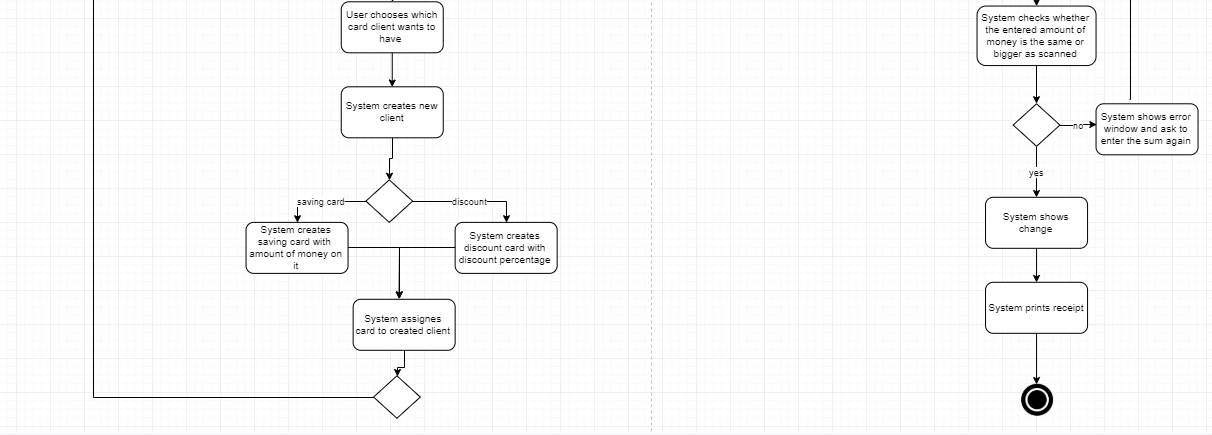
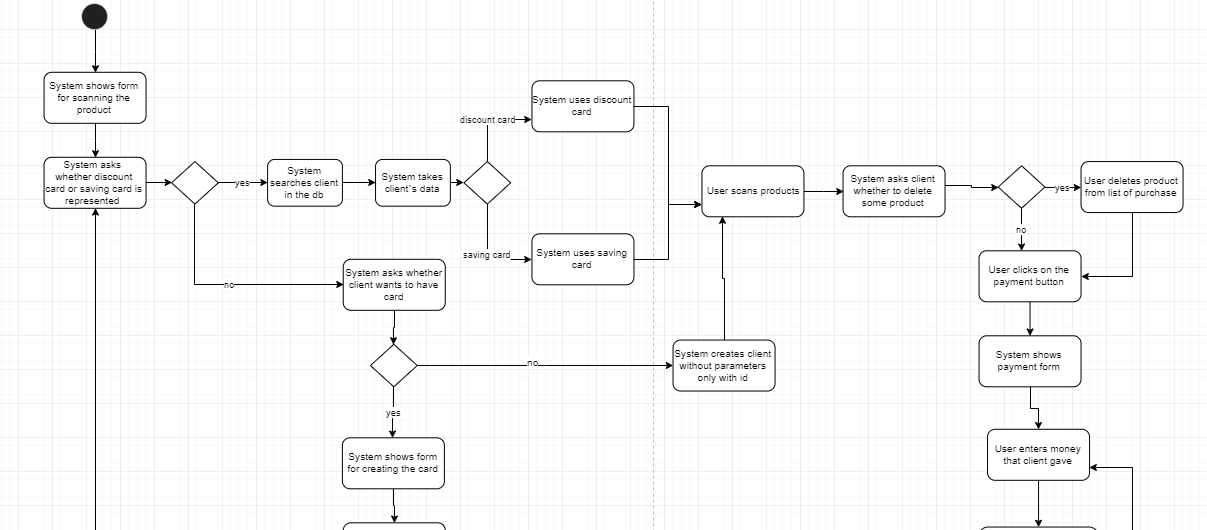
**Analytical Diagram**

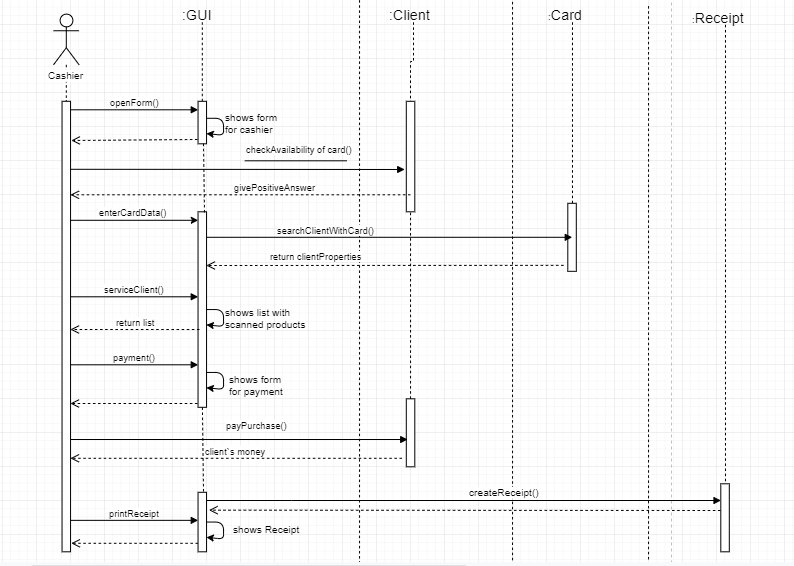
**Design Diagram**



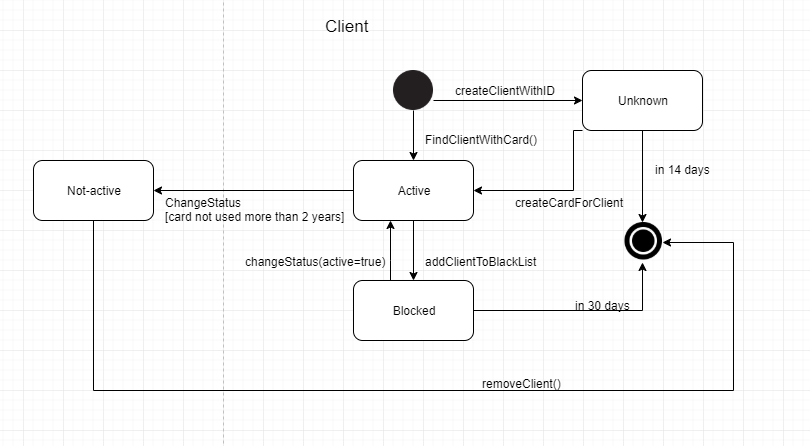
**Activity Diagram**



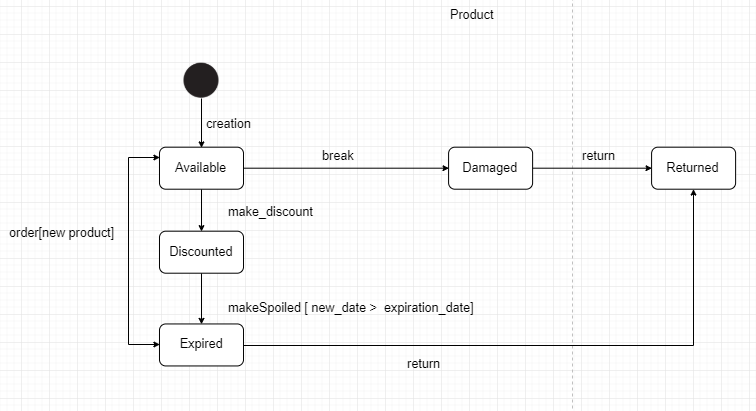
**Sequence Diagram**



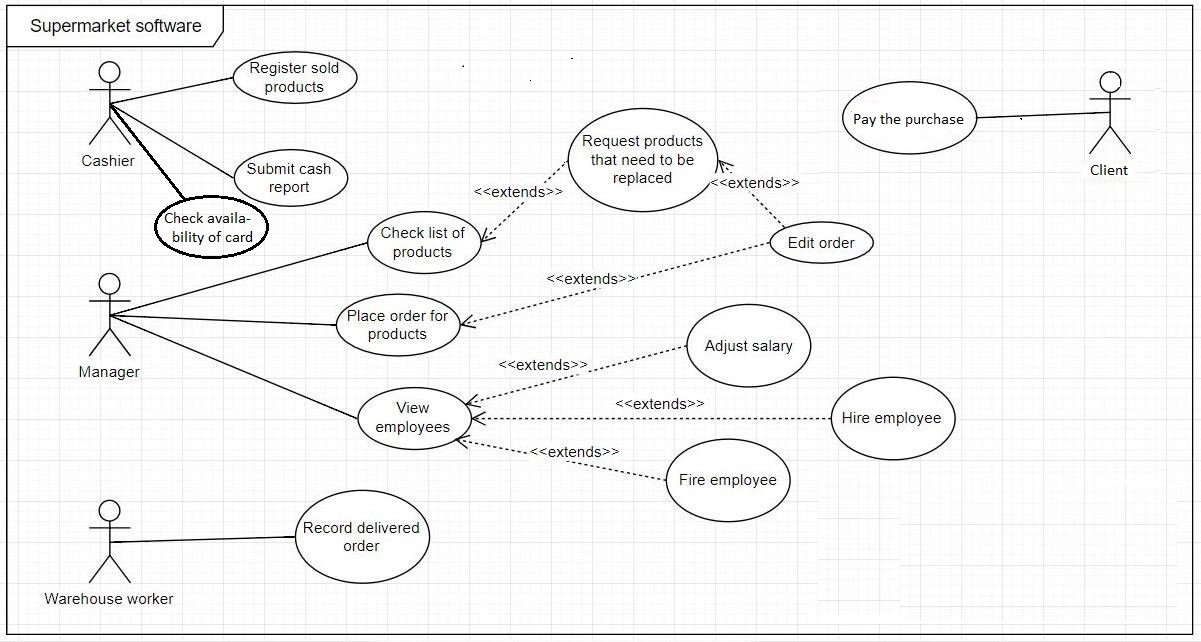
**State Diagram (Client)**



**State Diagram (Product)**

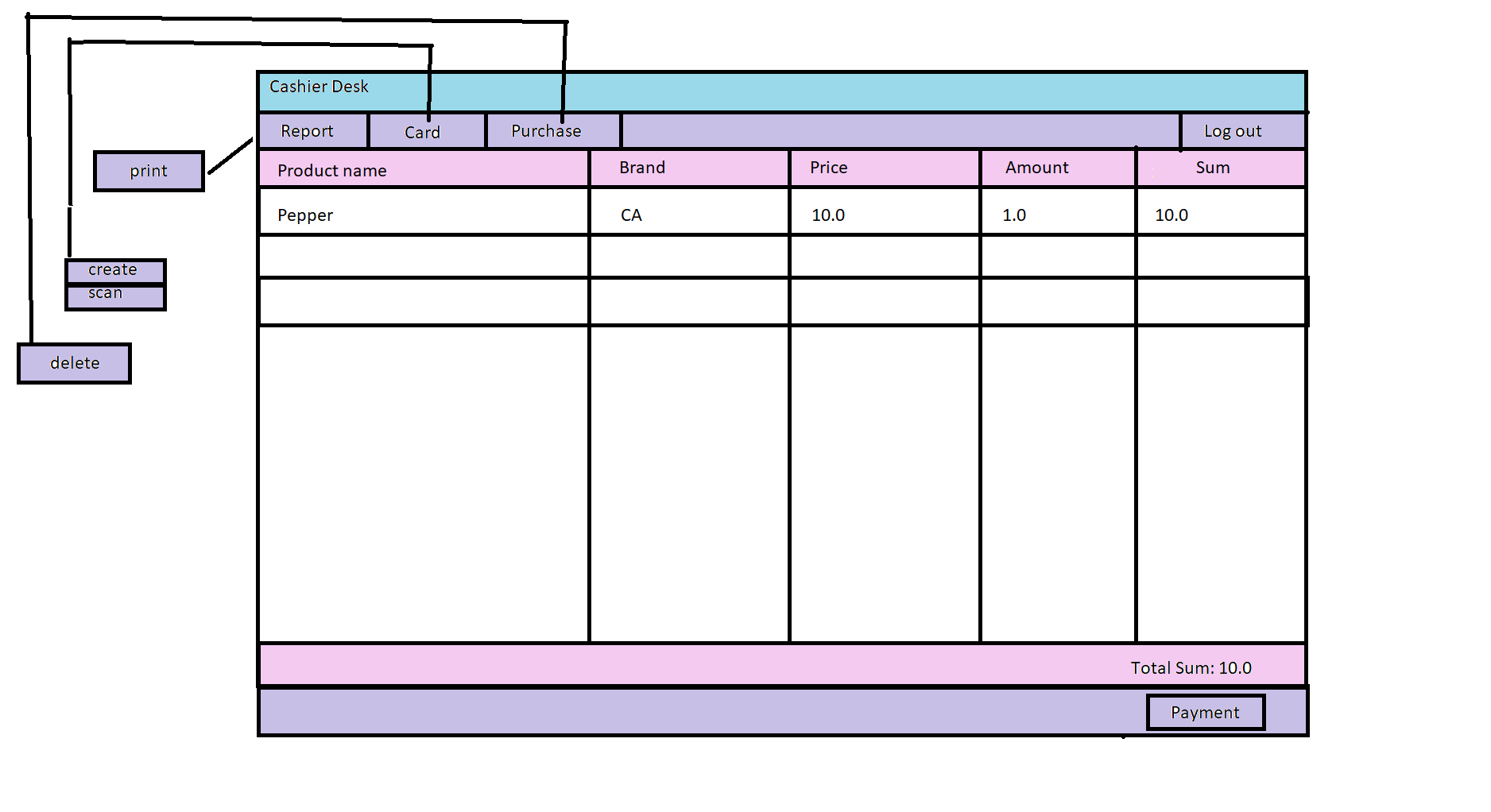


**Use case diagram**

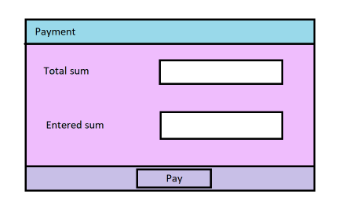


**GUI Design**

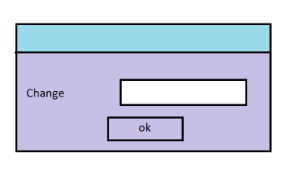
**Scan Product Form**



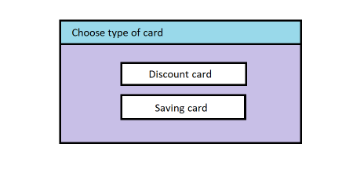
**Payment Form**



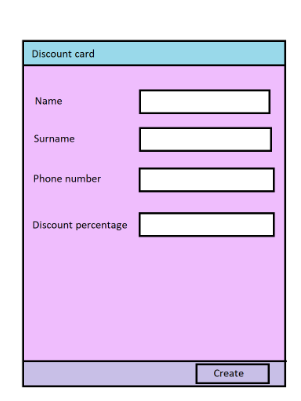
**Change window**



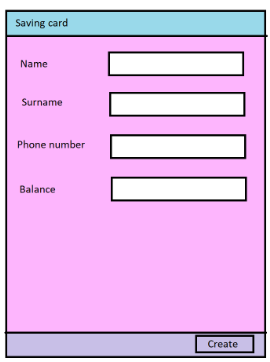
**Choose card to create window**



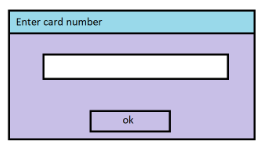
**Discount Card**



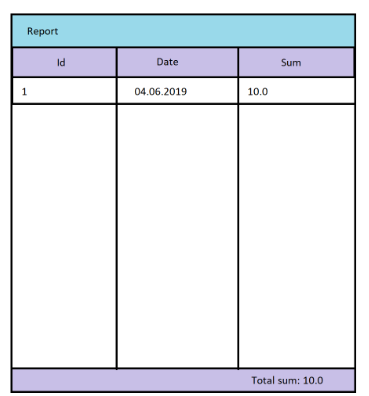
**Saving Card**



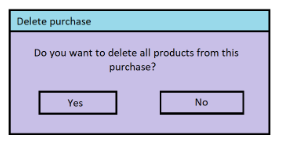
**Scan card form**



**Report form**



**Delete Purchase window**



**Log out window**

