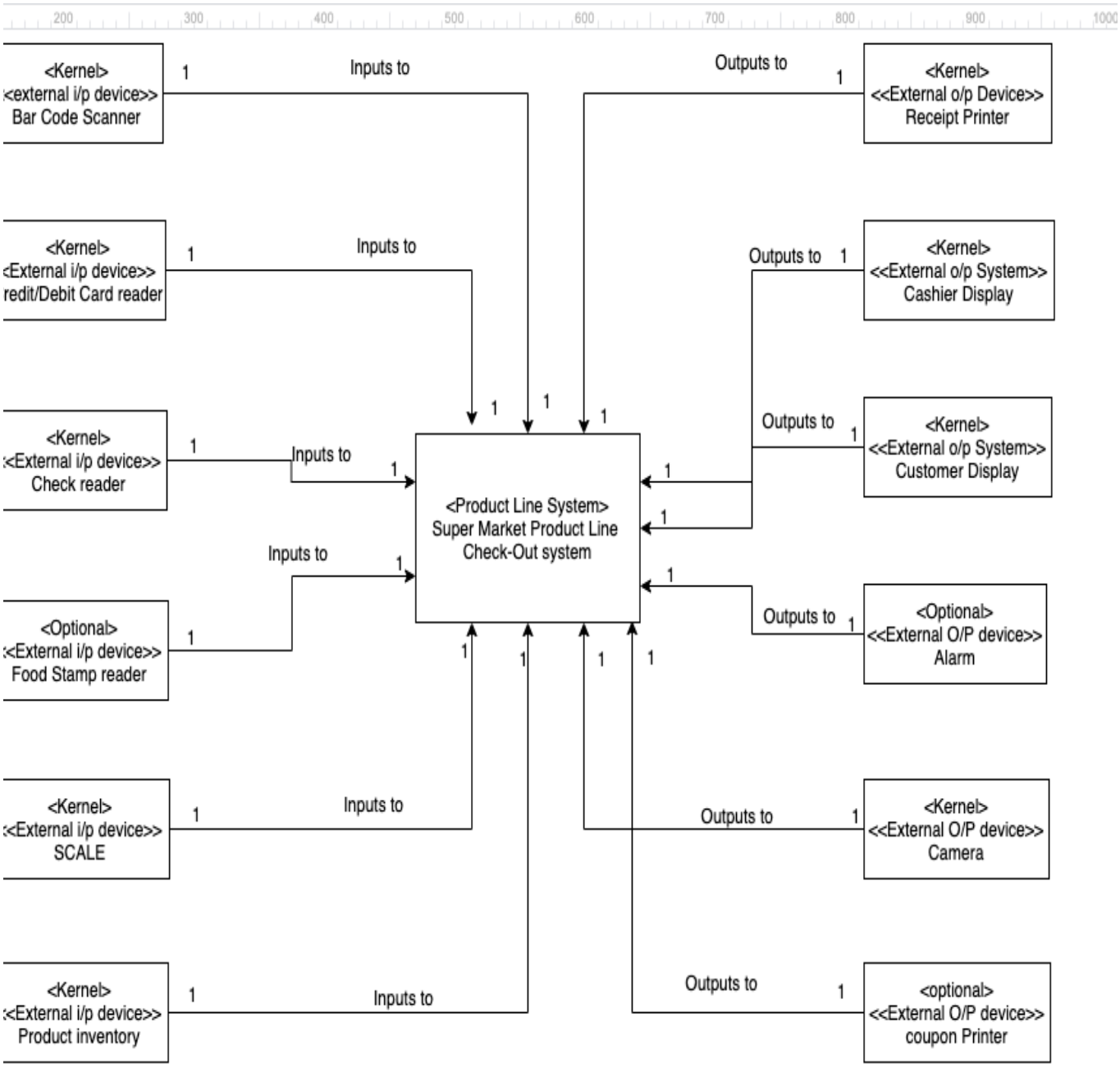
Supermarket Product Line Check Out System

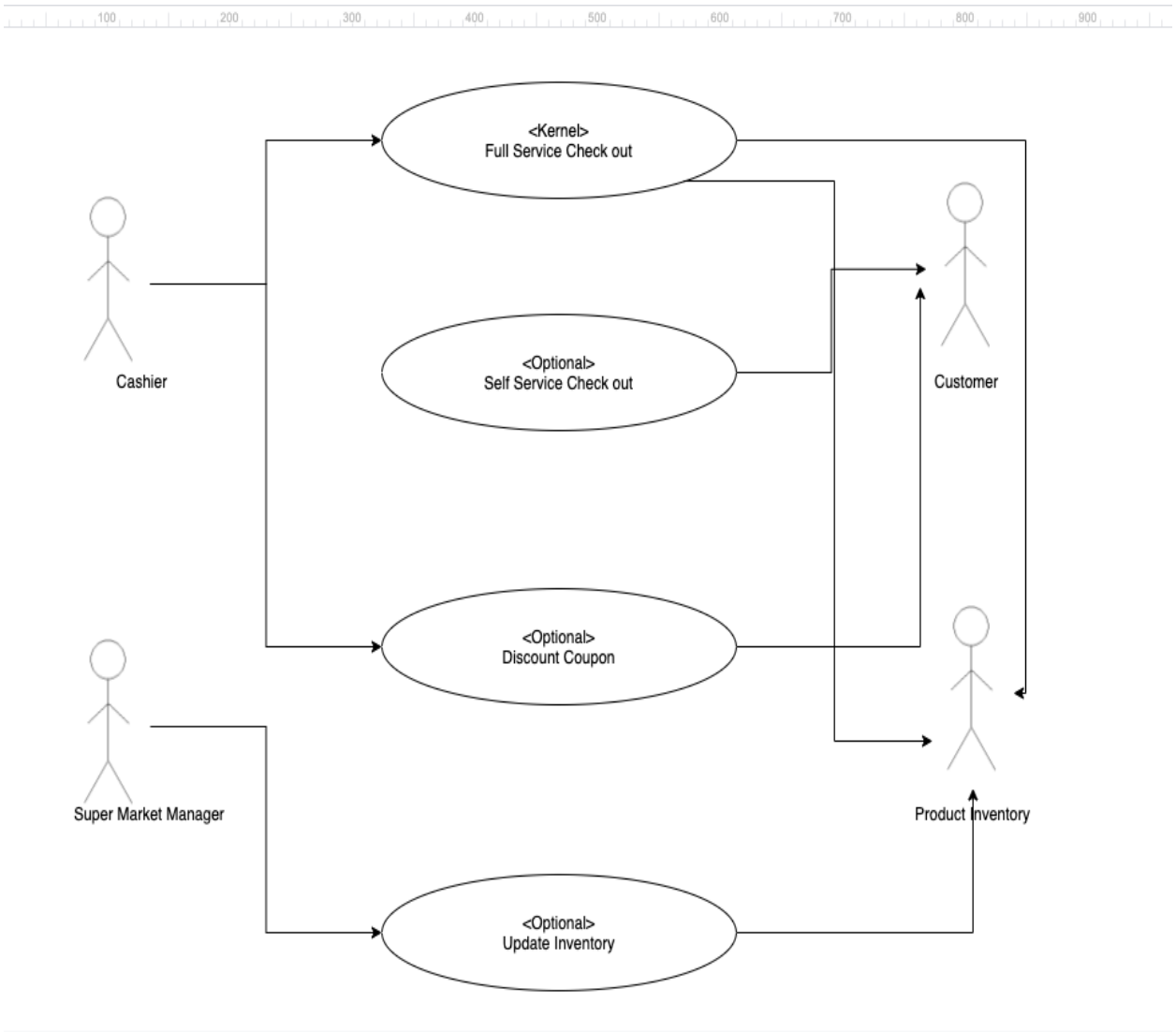
Context diagram for Supermarket Check-Out System



Context Diagram for Super Market Checkout Systems

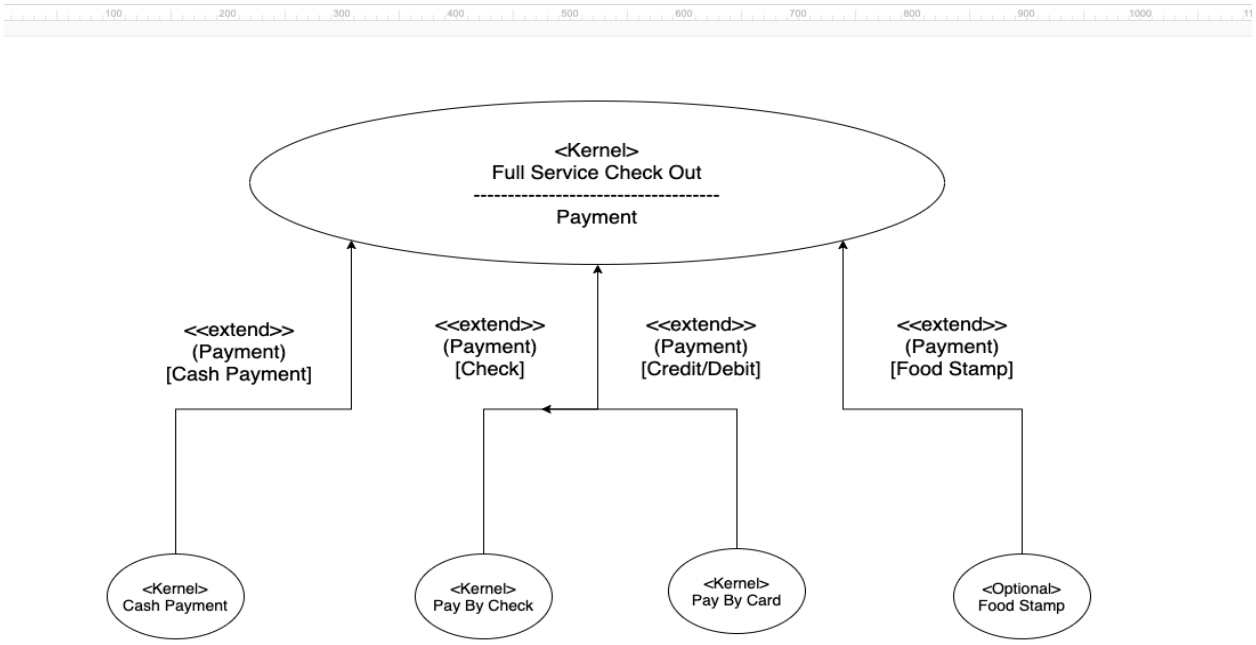
Use Case Analysis for SuperMarket Checkout System

Use Case diagram for Supermarket Checkout System

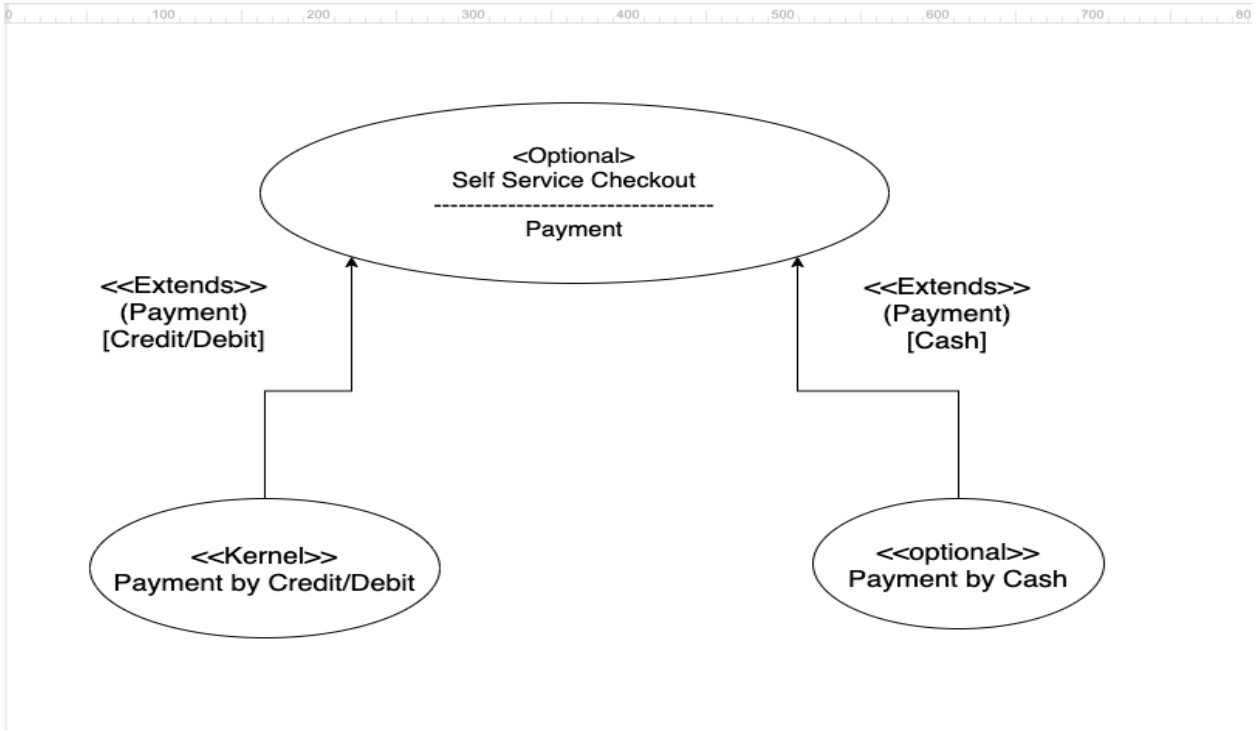


[cash-payment], [check], [credit/debit], [foodstamp]

<Payment> in Full service check-out



<Payment> in Self service checkout



Usecase Name - Full Service check-out
Reuse Category - Kernel
Summary - Cashier scans items,Calculate total price, collect bill from customer and customer checks out
Actors - Cashier, Customer, Product inventory
Precondition - system is Idle with welcome message
Description

1. Cashier scan the items
2. System display product information
3. Price,product information, discount information displayed on the screen
4. Repeat 1,2,3, for all items
5. Cashier hits total key
6. Customer selects payment method
7. <Payment>
8. System displays “Thank You” and customer checks out

Alternatives

Line 1 - If scan failed, then cashier manually enter ITEM_ID
Line 1 - If item is bulk item, then it can be measured by using SCALE
Line 2 - Retrieval of Product information fails from the inventory, order cancelled
Line 7 - Payment declined, order cancelled

Variation points

Name - Display language
Type of functionality - Mandatory alternative
Line numbers - 3,6,8
Description of functionality -
There is a choice of languages for displaying messages.The default one is english.
Alternative mutually exclusive languages are French,Spanish,German

Name - Memberships
Type of functionality - optional
Line number - Line 1
Description -

1. First item is scanned or item ID is entered,
2. The customer is prompted to enter the phone number and pin
3. If customer is loyal, the points are added to the customer account

Name - Store coupon
Type of functionality - optional
Line number - 2
Description -

1. Cash register obtains the product information
2. If it contains coupon then it will be printed on the store coupon printer

In base use case at step 7 <payment> is a placeholder that identifies the location at which extension use case is executed.

Cash payment use case

Use Case Name - Cash Payment
Reuse category - Kernel
Summary - Customer pays by cash
Actor - Cashier, Customer
Dependency - Extends full service check-out
Precondition - Customer has scanned items, but not yet paid
Description

1. Customer selects payment by cash
2. System prompts customer to deposit cash in bills/coins
3. Cashier enters cash amount
4. System prints total amount due, cash payment, change on receipt

Variation point

Name - Display language
Type - Mandatory alternative
Line number - 4
Description - Default is english, other languages are french, spanish, German

Check Payment Use Case

Use Case Name - Pay by check
Reuse category - Kernel
Summary - Customer pays by check
Actor - Cashier, Customer
Dependency - Extends full service check-out customer
Precondition - Customer has scanned items but not yet paid
Description -

1. Customers use a check for payment,
2. The check is scanned by the check reader,
3. The check is verified by the system.
4. If a check is accepted, the cashier is prompted to place it in the receipt printer. Date, time, store identity, cashier identity and order number is printed on check

Credit/Debit payment Use Case

Use Case Name - Pay by Card
Reuse Category - Kernel

Summary - Customer pay by either debit or credit card
Actor - Customer
Dependency - Extends full service check-out customer
Precondition - Customer has scanned items, but not yet paid
Description

1. Customer enters card into the credit/debit reader
2. If debit customer enters pin
3. Message is sent to the appropriate credit/debit authentication center
4. Authorization code is returned
5. Card number and authorization code are printed on the receipt

Alternatives

Line - 4 If authorization code not returned or card request rejected, then the order gets cancelled

Food Stamp Use Case

Use case Name - Food stamp
Reuse category - Optional
Summary - Customer pays by food stamp
Actor - Cashier, Customer
Dependency - Extends full service check-out customer
Pre condition - Customer has scanned items, but not yet paid
Description -

1. Cashier scans food stamp reader
2. Prints the receipt to customer

Use Case Name - Self Service Check out
Reuse category - Optional
Summary - Customer scans items, Pay bill and self check-out
Actors - Customer, Product - inventory
Precondition - System is idle with welcome message
Description

1. Customer press the start button and scans the items
2. Cash register obtains information from product inventory
3. Price,item-description and discount information are displayed on the screen
4. Repeat 1,2,3 for all the items
5. Calculated total displayed on the screen
6. <payment>
7. System displays “ Thank You “ message and customer checks out

Alternatives

Line 1 - If scan failed, customer can also enter ITEM_ID
Line 2 - If inventory system did not work, transaction gets failed
Line 6 - Payment declined, Cancel order

Variation Points

Name - Display language
Type of functionality - Mandatory alternative
Line number - 1,3,7
Description of functionality -

1. Choice of language for the display messages.
2. The default is english.
3. Alternative languages are french,spanish, German

Name - Memberships
Type of functionality - Optional
Line number - 1
Description -

1. First item is scanned or item ID is entered,
2. The customer is prompted to enter a phone number and pin.
3. If customer is loyal, the points are added to the customer account

Name - Camera
Type of functionality - optional
Line number - 1
Description -

1. The customer scans the items or starts the checkout process
2. Security Camera is activated

Name - Alarm
Type of functionality - Optional
Line number - 5
Description - Customer get extra discount of 5-10 % because they check out items by themselves

[Credit/Debit],[Cash]

Credit/Debit payment use case

Use Case Name - Payment by Credit/Debit
Reuse category - Kernel

Summary - Customer pay by using debit/credit card
Actor - Customer
Dependency - Extends self service check-out
Precondition - Customer scanned all items, but not yet paid
Description

1. Customer selects payment by card
2. Enters card into the card reader
3. Authorization of transaction
4. System prints receipt

Alternatives

Line 3 - If authorization of transaction failed, order gets cancelled

Cash Payment Use case

Use case Name - Payment by cash
Reuse category - optional
Summary - Customer pay by using cash
Actor - Customer
Dependency - Extends self service checkout
Precondition - Customer scanned all items, but not yet paid
Description

1. Customer selects payment by cash
2. Cash registers activates coin/cash reader device
3. Customer place the cash in the device
4. Change is returned and receipt is printed

Use Case Name - Discount coupon
Reuse category - optional
Summary - Cashier/ self service register accepts discount coupons from customers if coupon is valid, discount is applied
Actors - cashier, customer
Description

1. Cashier/customer scans discount coupon from customer
2. System verifies its expiration date
3. Discount applied

Alternatives
Line 1 - If scan fails, a coupon code can be manually entered
Line 2 - If verification fails, discount not applied

Use Case Name - Update Inventory
Reuse Category - optional
Summary - If items are less, inventory system orders required quantities of products automatically
Actors - Supermarket manager, Inventory system
Description

1. Inventory system keeps track of current no of items
2. If the items goes below a certain threshold an inventory message is created
3. Inventory orders predefined quantities of products and sent to suppliers and record in the system

Alternative
Line 2 - If the items are above threshold, cancel inventory message/ order

Variation points

Name - Update inventory with items
Type of functionality - alternative
Line number - 1
Description

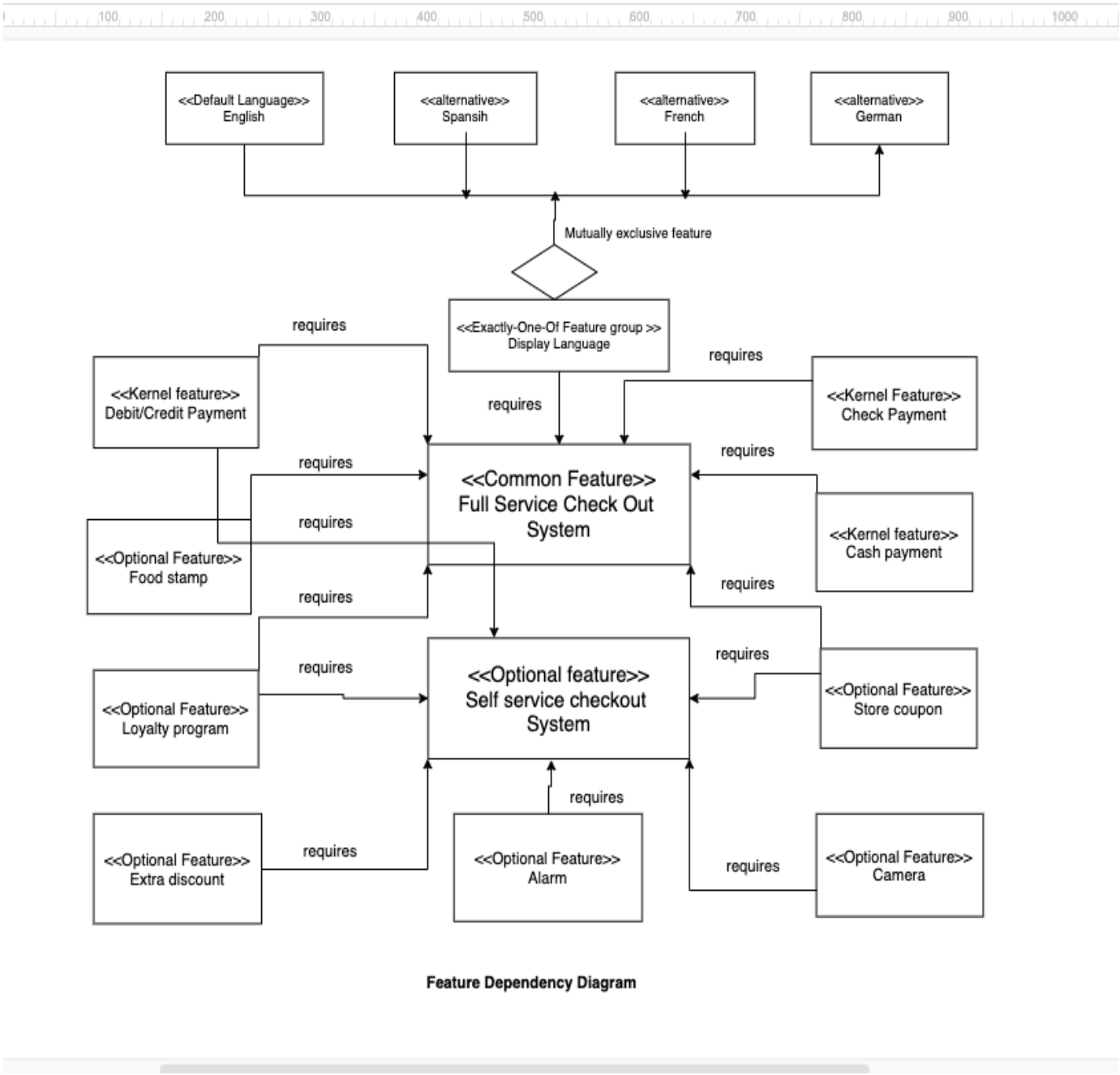
1. The supermarket manager run future inventory prediction algorithm to get no of quantities of each product to be ordered

Feature to Use Case relationships

Feature Name	**Feature Category**	**Use Case**	**Use Case Category/ Variation point**	**Variation Point Name**
Full Service checkout System	Common	Full Service checkout	Kernel	
English	default	Full Service/ Self service checkout	Variation point	Display Language
French	alternative	Full Service/ Self service checkout	Variation point	Display Language
Spanish	alternative	Full Service/ Self service checkout	Variation point	Display Language
German	alternative	Full Service/ Self service checkout	Variation point	Display Language
Loyalty Program	Optional	Full Service/ Self service checkout	Variation point	Membership
Store coupon	Optional	Full Service/ Self service	Variation point	Store Coupon

		checkout		
Cash Payment	Kernel	Full Service Checkout	Kernel	
Debit/Credit Payment	Kernel	Full Service Checkout	Kernel	
Check Payment	Kernel	Full Service Checkout	Kernel	
Food stamp	Optional	Full Service Checkout	Optional	
Self service checkout System	Optional	Self service checkout	Optional	
Camera	Optional	Self service checkout	Variation point	Camera
Alarm	Optional	Self service checkout	Variation point	Alarm
Extra discount	Optional	Self service checkout	Variation point	Extra discount
Inventory update	Optional	Update inventory	Optional	
Future inventory Prediction algorithm	Optional	Update inventory	Variation Point	Update inventory using algorithm

Feature Dependency Diagram



Feature Dependency Diagram

Analysis Modeling

Object and class structuring: Kernel use case Full service Checkout

Input device interface classes:

- Credit card reader interface class
- Barcode scanner interface class
- Check reader interface class
- Cash interface class
- Scale interface class

Input System interface classes:

- Product inventory system class

Output device interface class:

- Customer display interface
- Cash register display interface
- Receipt printer interface

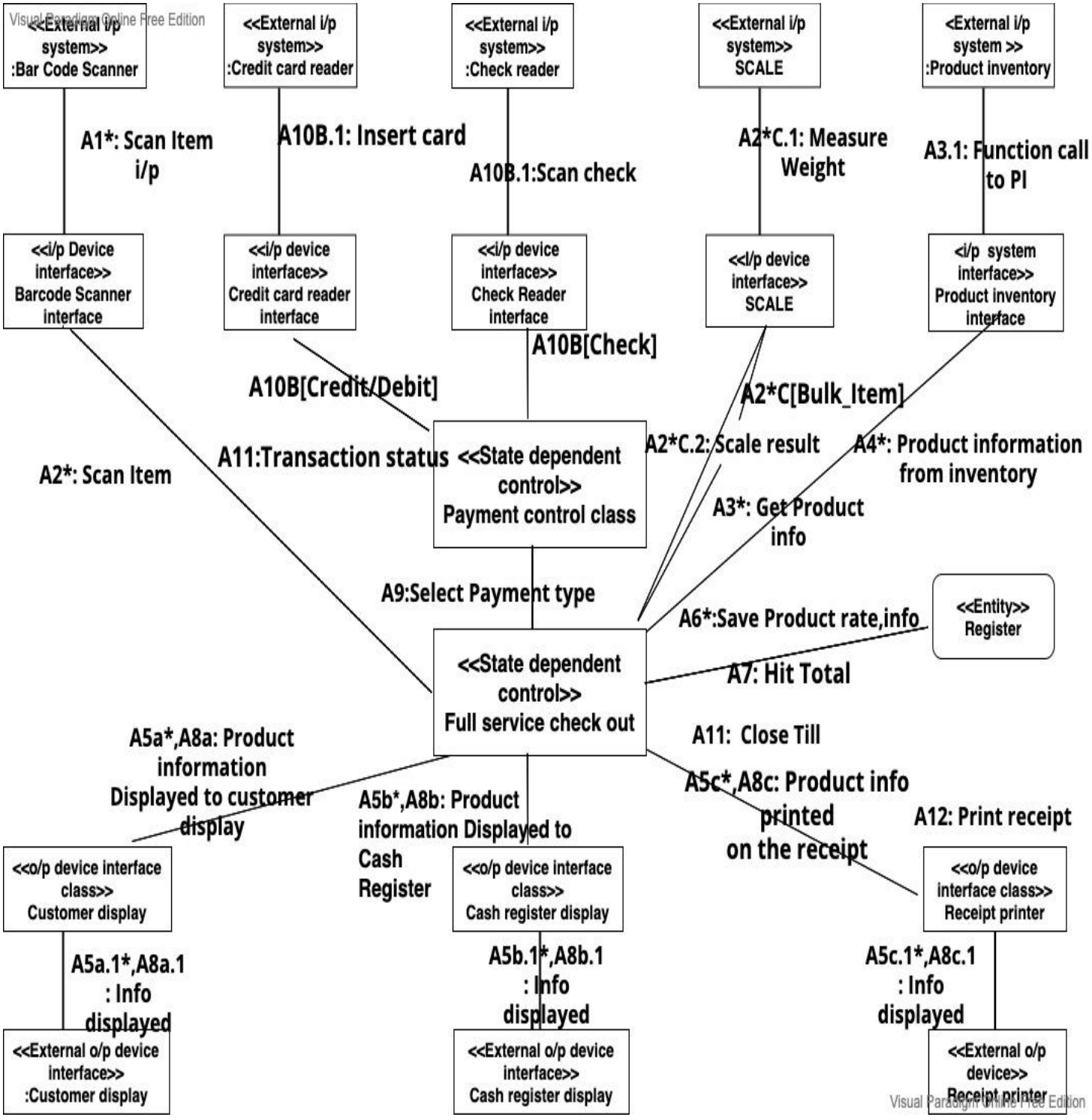
Control classes:

- Full service checkout by cashier control class
- Payment control class

Entity classes:

- Register

Communication diagram for Kernel Use Case Full Service Checkout System



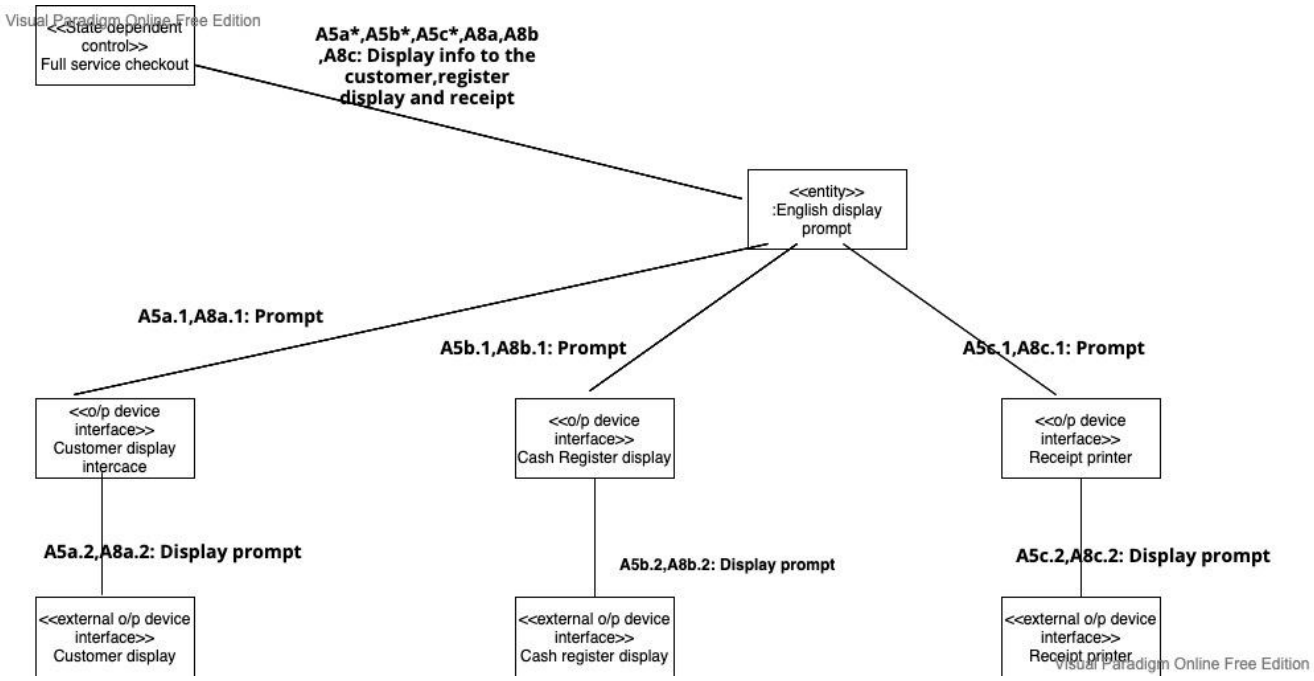
The following is the sequence of messages for the kernel communication diagram for full service checkout system

A1,A2: Scan Items using barcode scanner
A3,A4: Get Product information from inventory
A5a,A5b,A5c: Display product information on customer display, cashier display and to the receipt
A6: Save scanned product to the register list

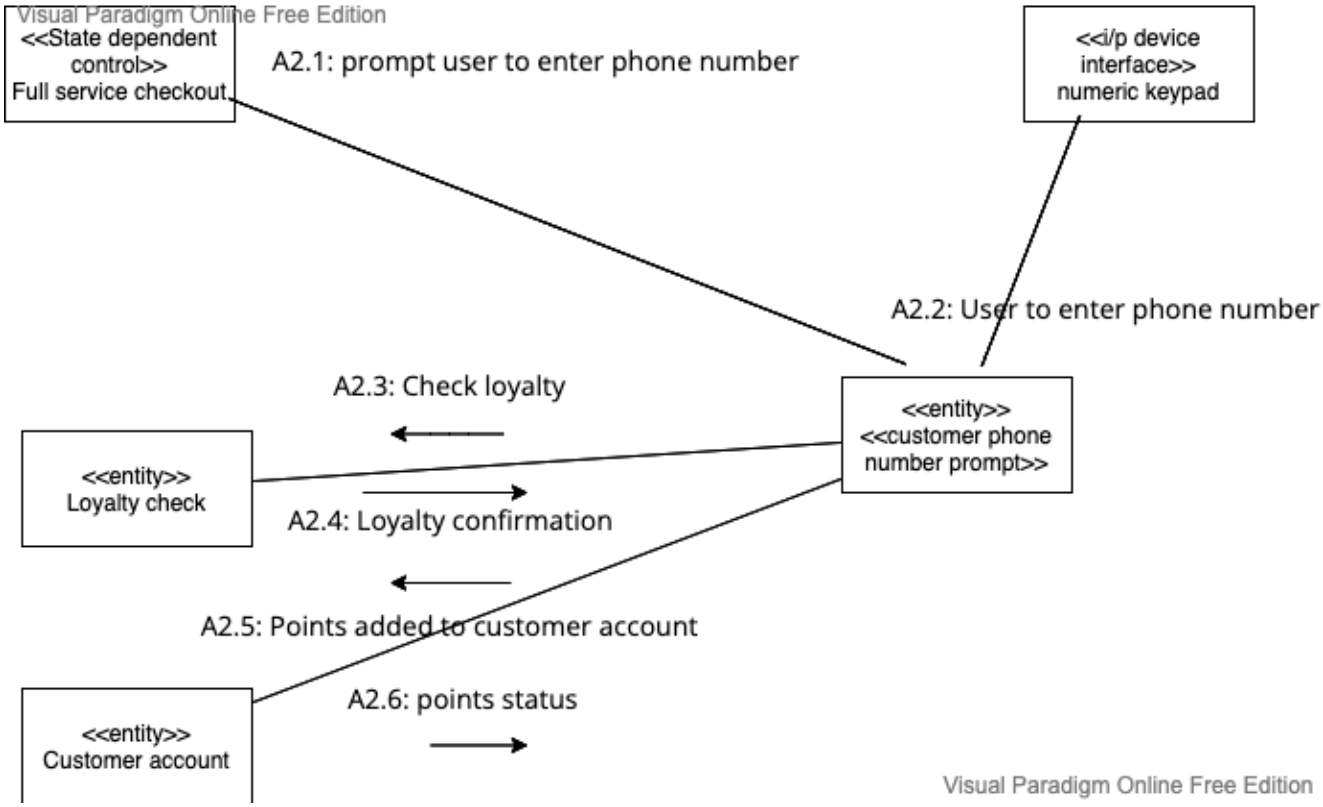
All the above steps will be repeated till the last item is scanned..So message sequence number ended with (*) on the diagram

A7: Hit total in the register
A7.1 : Computed total will be sent to the control class
A8a,A8b,A8c : Total will be displayed to the customer register, cashier register and to the receipt
A9 : Payment method is selected
A10[Credit/Debit],A10[Check] : Mandatory Alternative option..Only one payment method will be executed
A11: Close till
A12: Print receipt

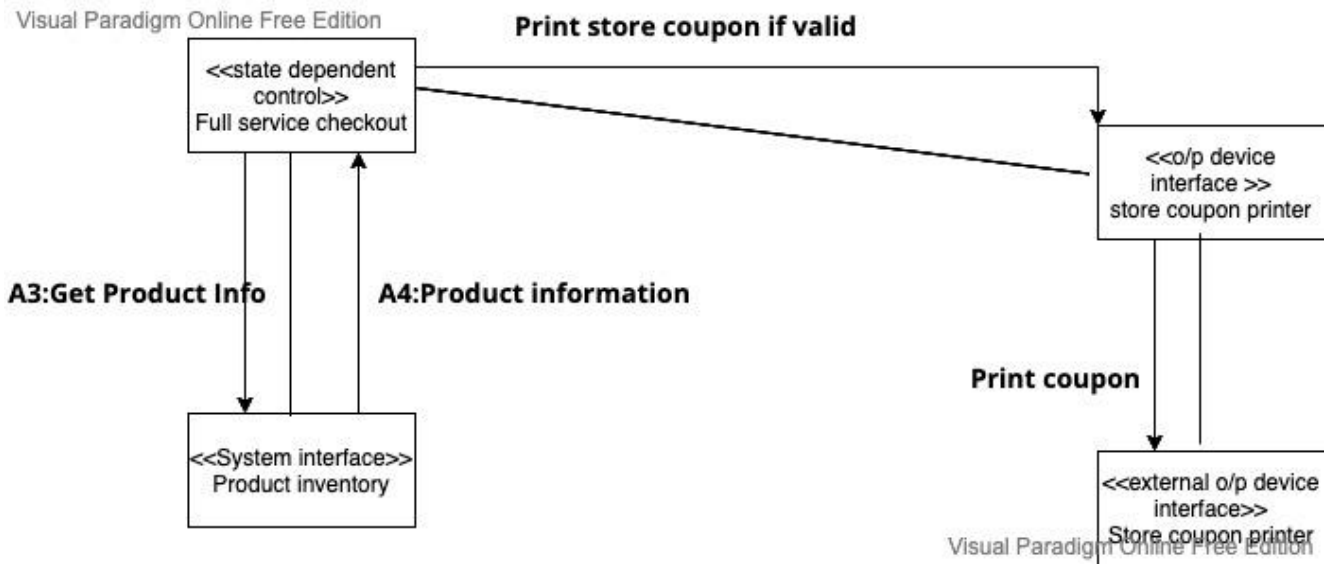
Variant **Display** **language** branch depicting the **impact of the display language variation point**



Variant Memberships and impact analysis of this variant for full service checkout



Variation point [store coupon] - impact analysis of store coupon on full service checkout system



Object and class structuring: Optional use case Self service Checkout

Input device interface classes:

- Credit card reader interface class
- Barcode scanner interface class
- Check reader interface class
- Cash interface class
- Scale interface class

Input System interface classes:

- Product inventory system class

Output device interface class:

- Customer display interface
- Cash register display interface
- Receipt printer interface

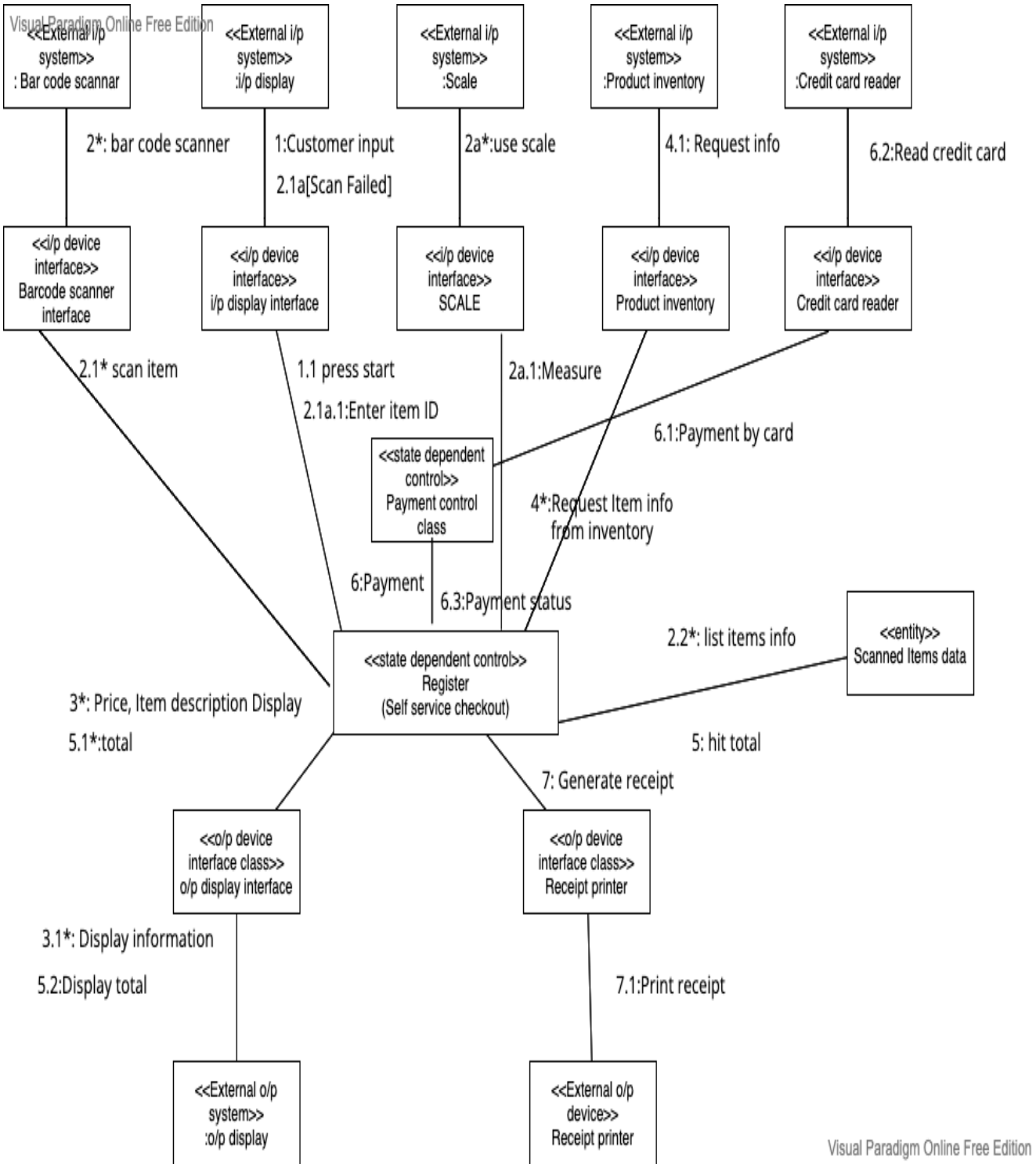
Control classes:

- Full service checkout by cashier control class
- Payment control class

Entity classes:

- Register

Communication diagram for optional use case self service check_out



The following is the sequence of messages for the kernel communication diagram for self service checkout system

1: Customer arrives at the billing point and input display screen shows welcome message

1.1: Customer press the start button

2*: barcode scanner is activated

2.1*: Customer starts scanning the items

2.2*: List scanned items info in the local entity

3*: Item, price description are sent to the output display

3.1*: Output display displays the information

4*: Request information from the inventory

(Repeat step 2 to step 4 till last item is scanned)

5: Hit total..Total will be computed in the entity “Scanned Items information”

5.1: Total to be displayed on the screen

6: payment request to the payment control class

6.1: Select payment type (Payment using card by default)

6.2: Read card (Card reader)

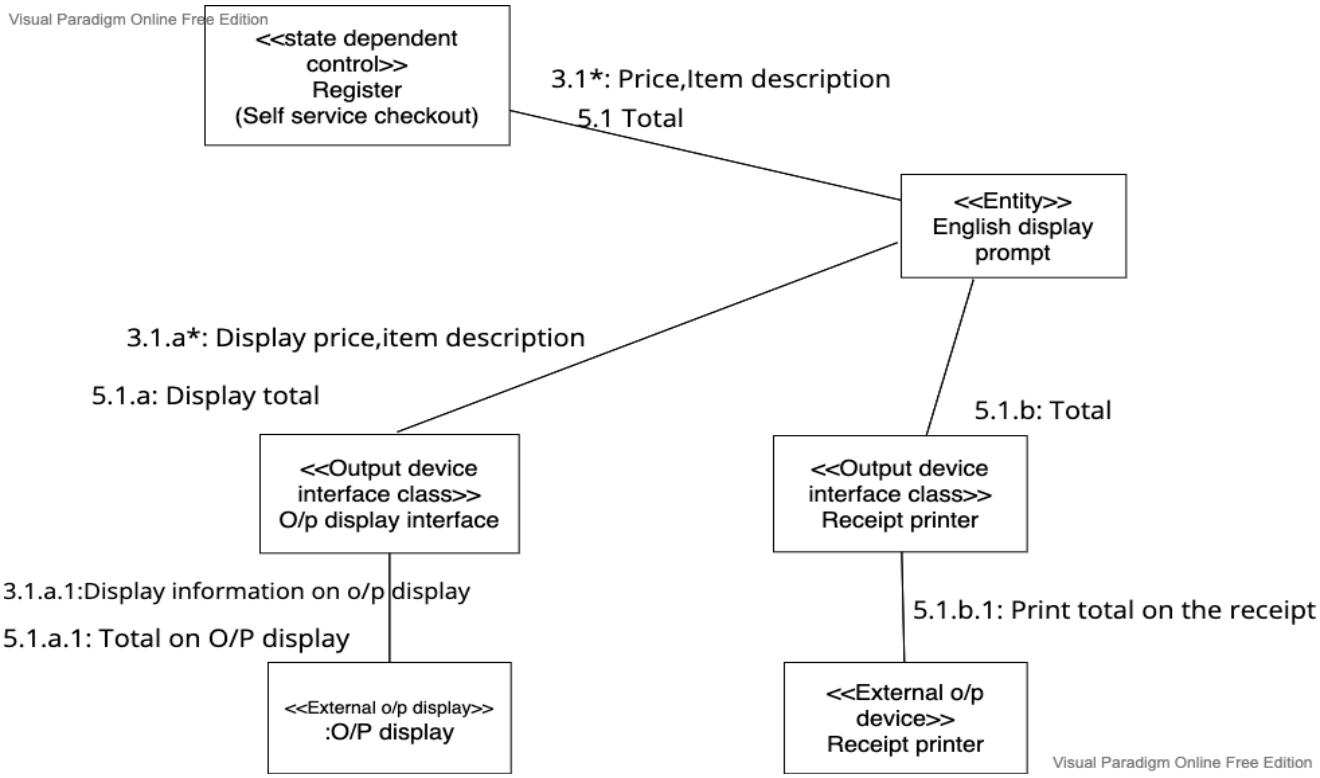
6.3: Payment confirmation

7: Generate receipt and send to the receipt printer

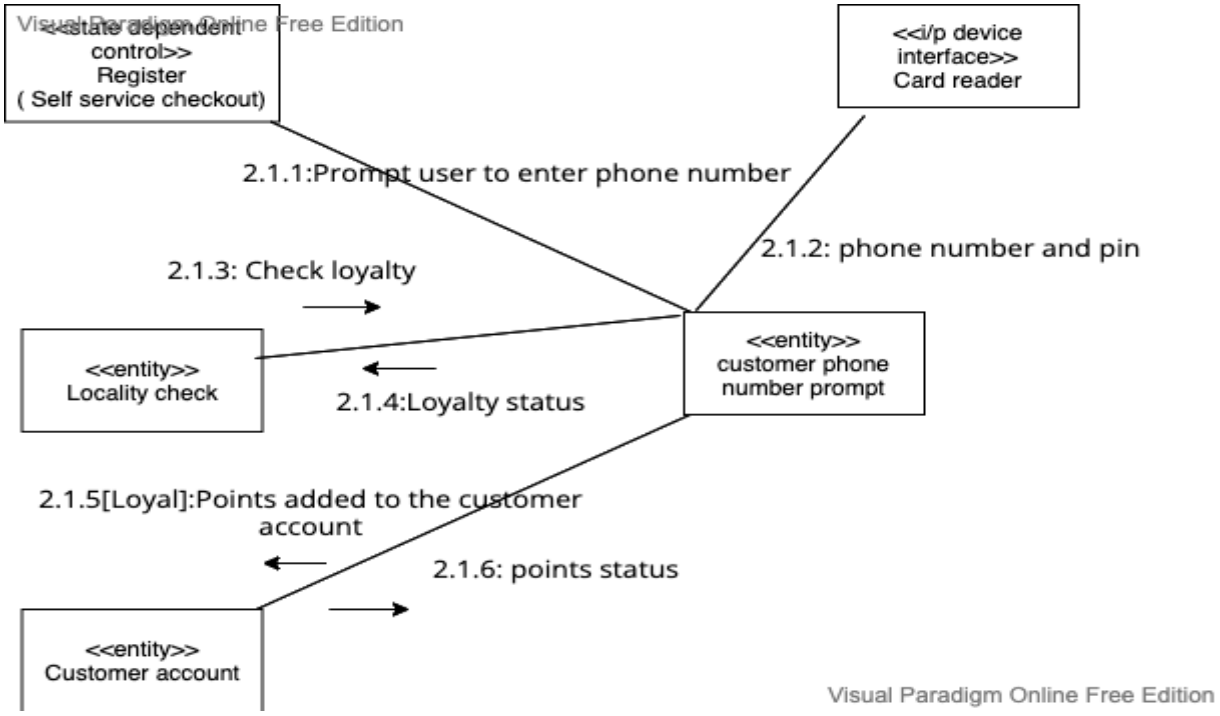
7.1 : Collect receipt

Variant **Display language** branch depicting the **impact of the display language variation point**

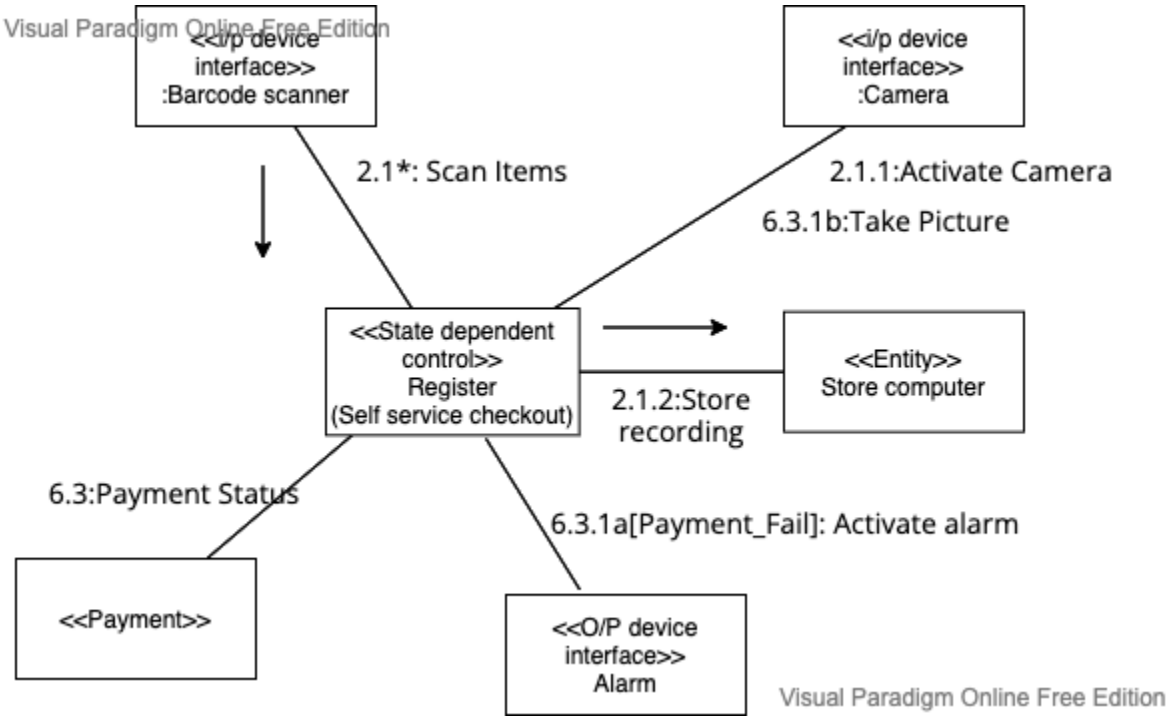
Default - English. Variations - Spanish, French and Germany



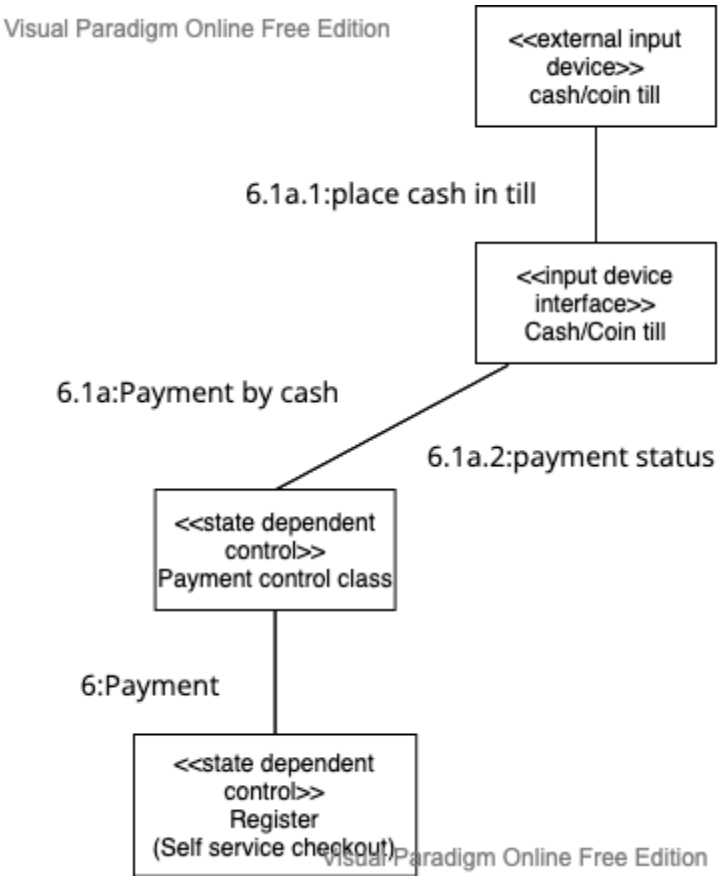
Variant **Memberships** branch depicting the **impact of the Memberships variation point**



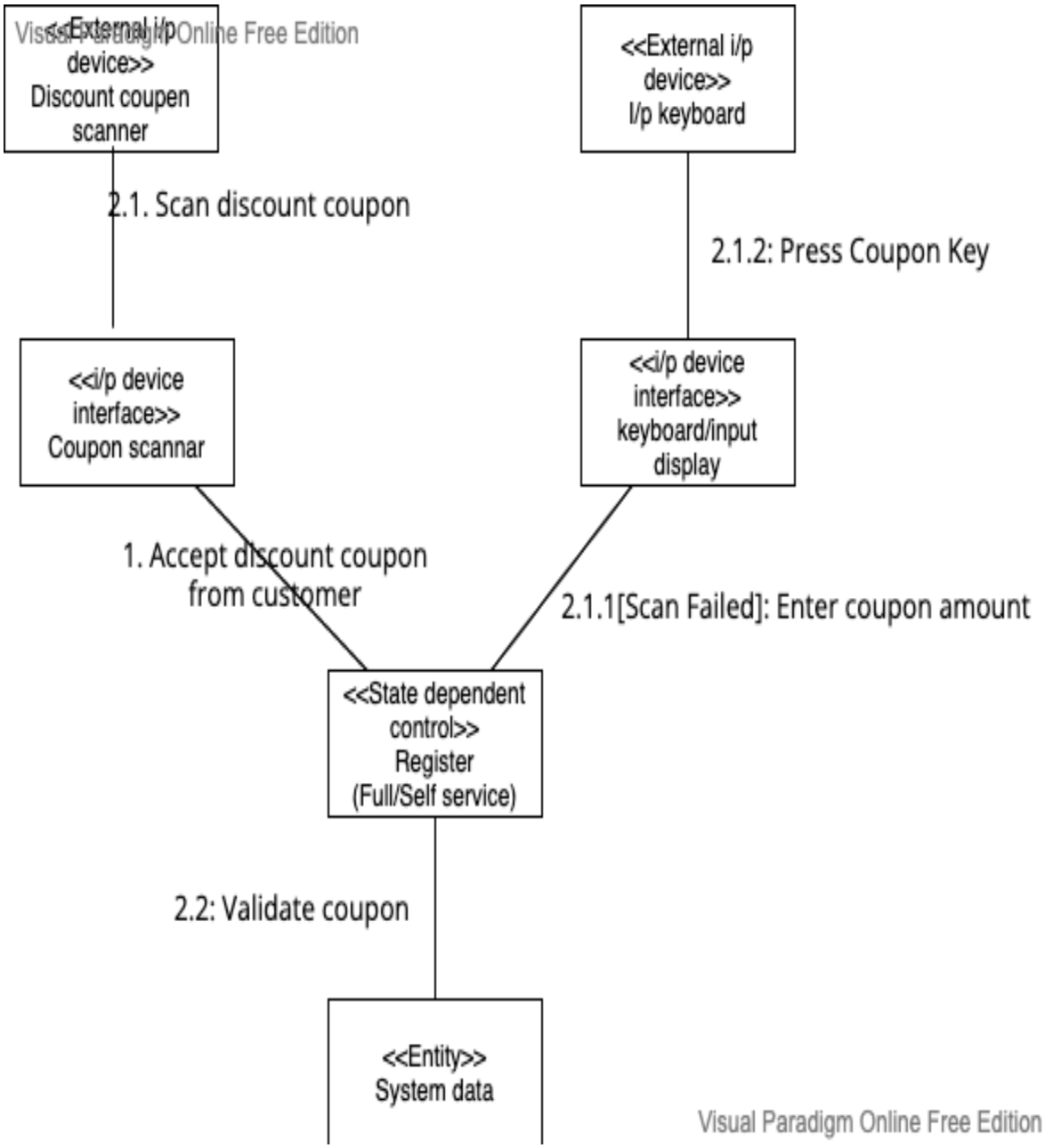
Variant **security** (**Camera and alarm**) in self service checkout system



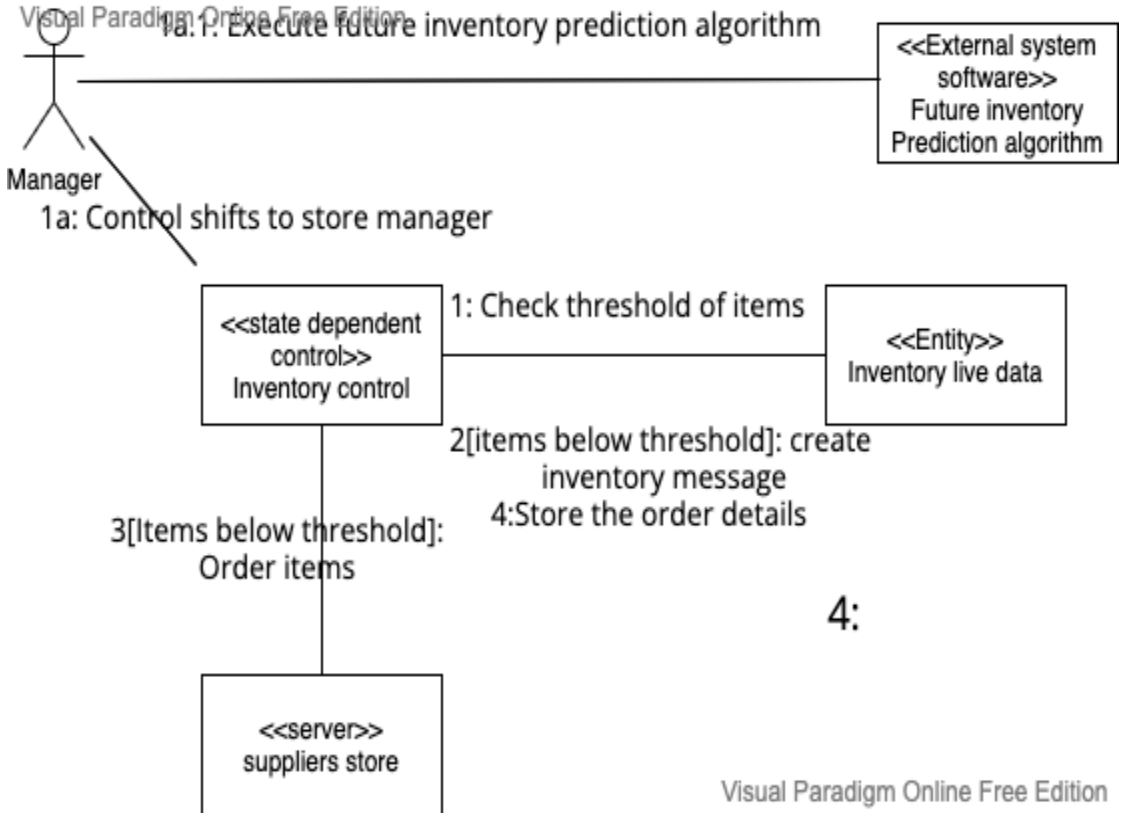
Variant **Cash payment** in self service checkout



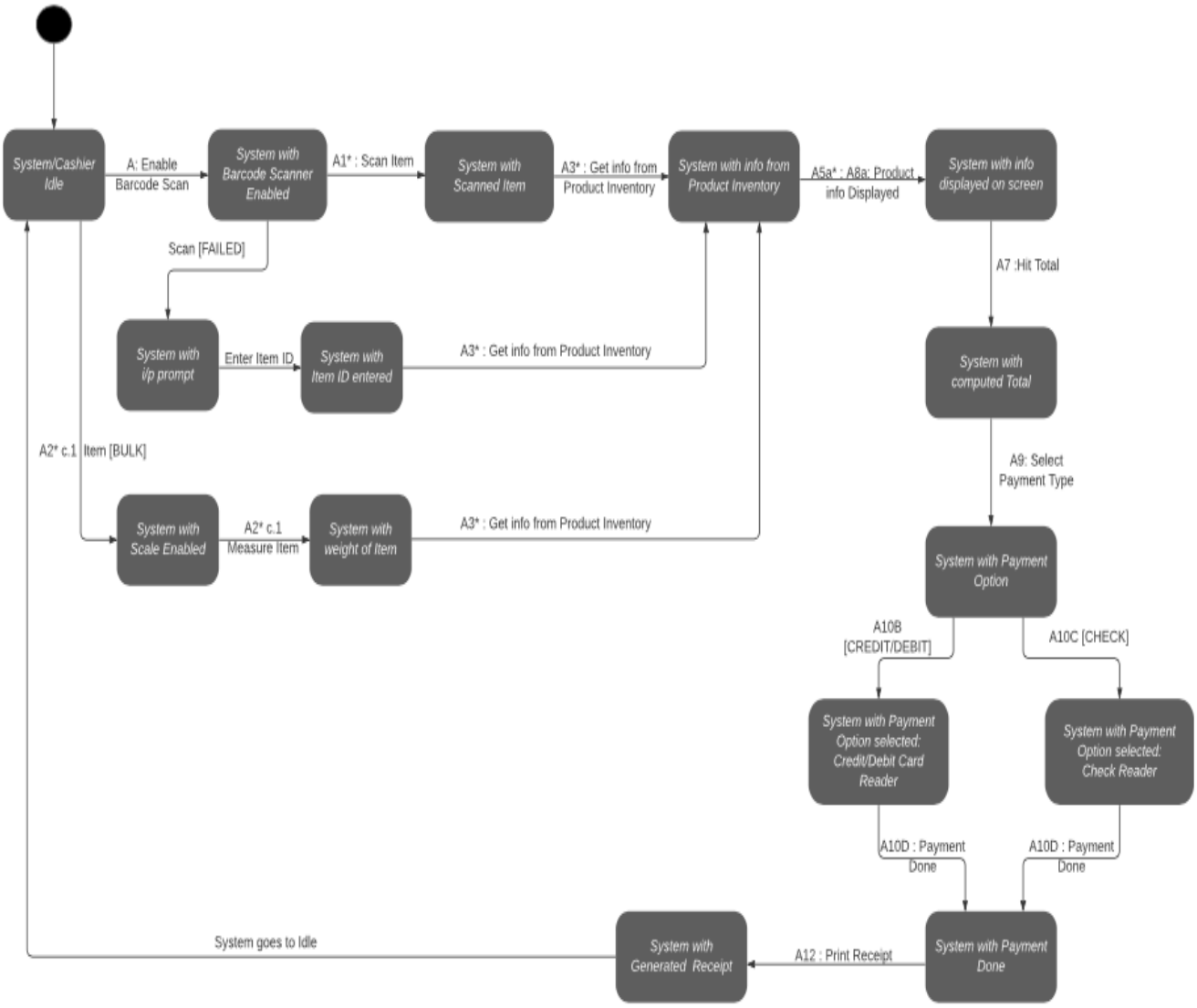
Communication diagram for **discount coupon** in both self service and full service checkout



Communication diagram for **Inventory system**

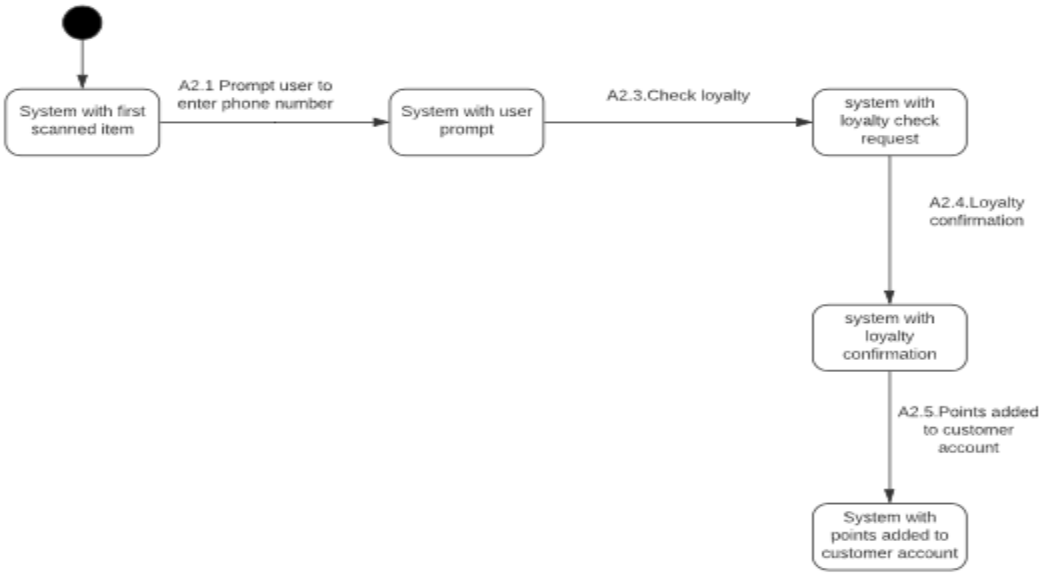


State chart diagram for kernel full service checkout



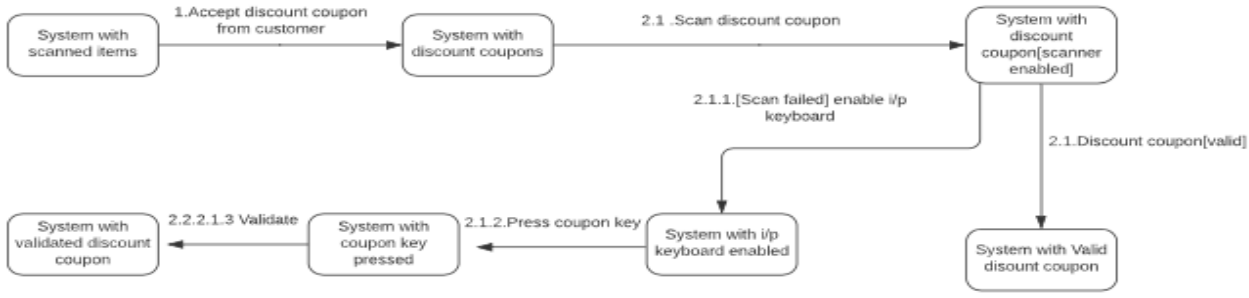
State Chart for Kernel Full Service Check-Out

Statechart for variant Membership in full service checkout



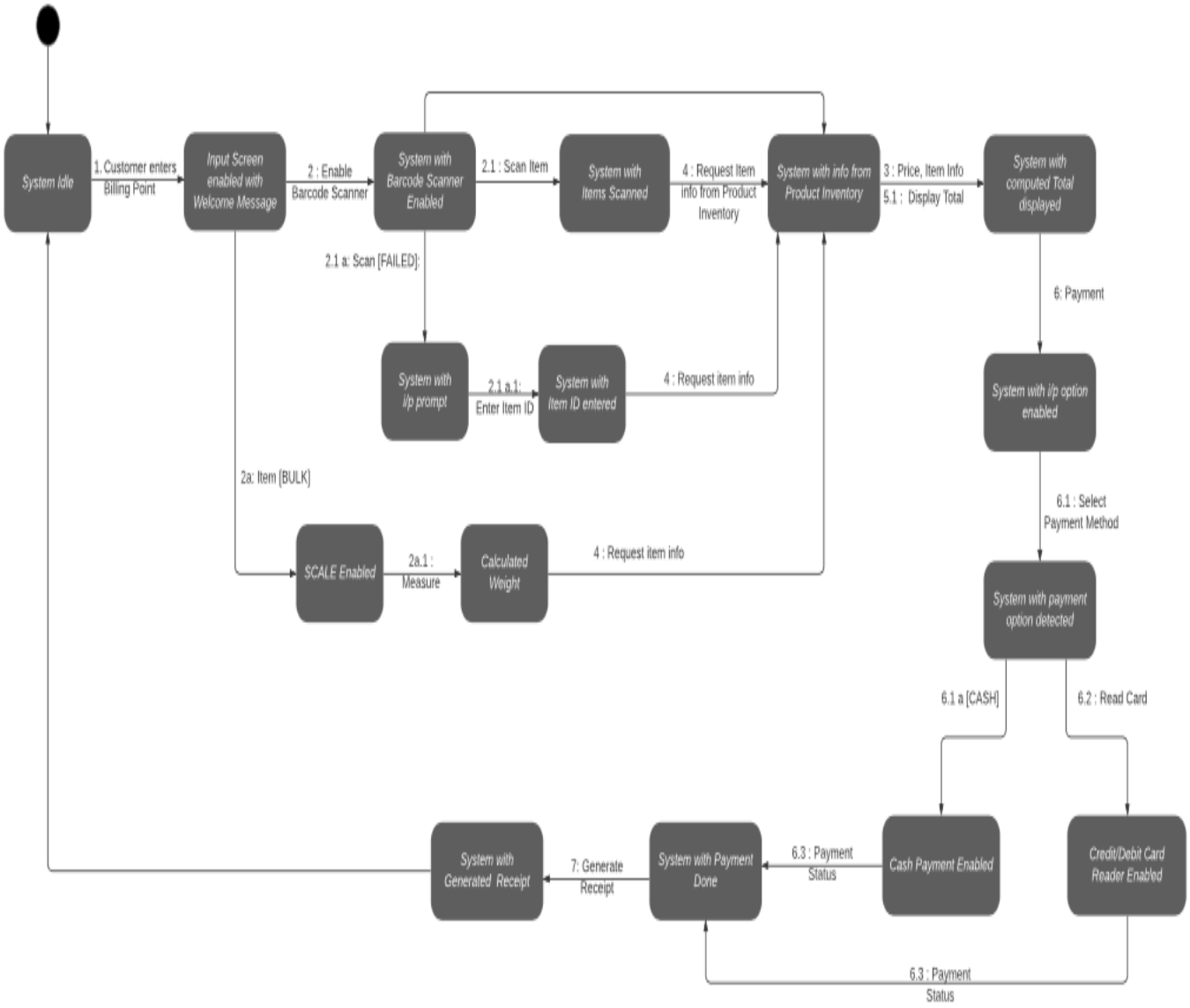
STATE CHART FOR VARIANT MEMBERSHIPS FOR FULL SERVICE CHECKOUT

State chart for Discount coupon for both self service and full service checkouts



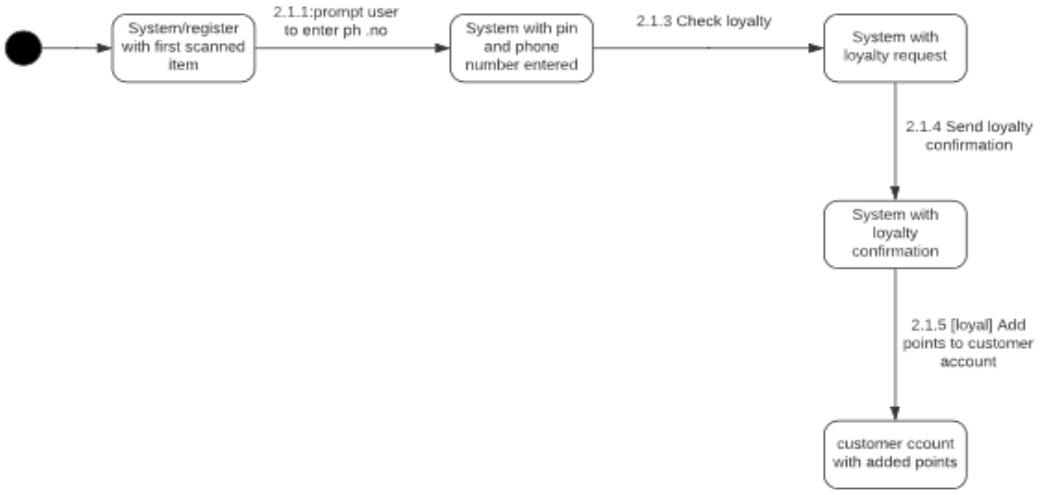
STATE CHART FOR DISCOUNT COUPON IN BOTH SELF SERVICE AND FULL SERVICE

Statechart for self service checkout



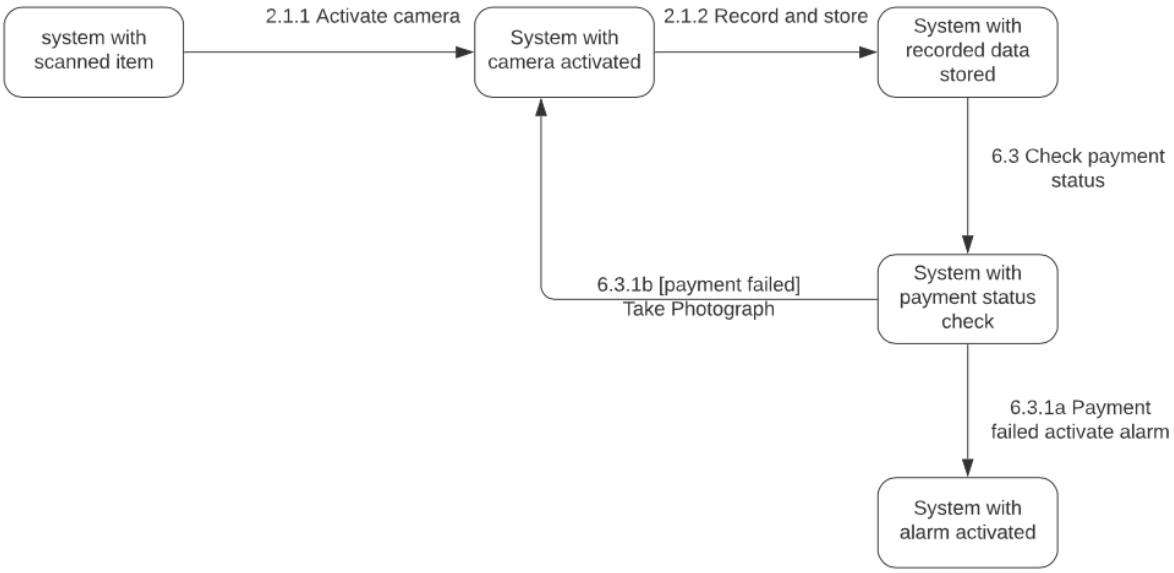
State Chart for Optional Self Service Check-Out (Kernel Flow)

Statechart for variant **memberships** in self service checkout



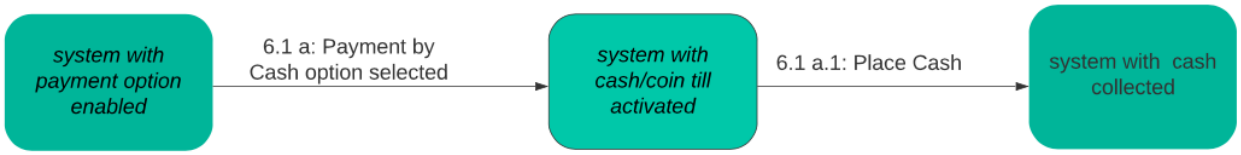
STATE CHART FOR VARIANT MEMBERSHIPS IN SELF SERVICE CHECKOUT

Statechart for variant **security (Camera and alarm)** in self service checkout



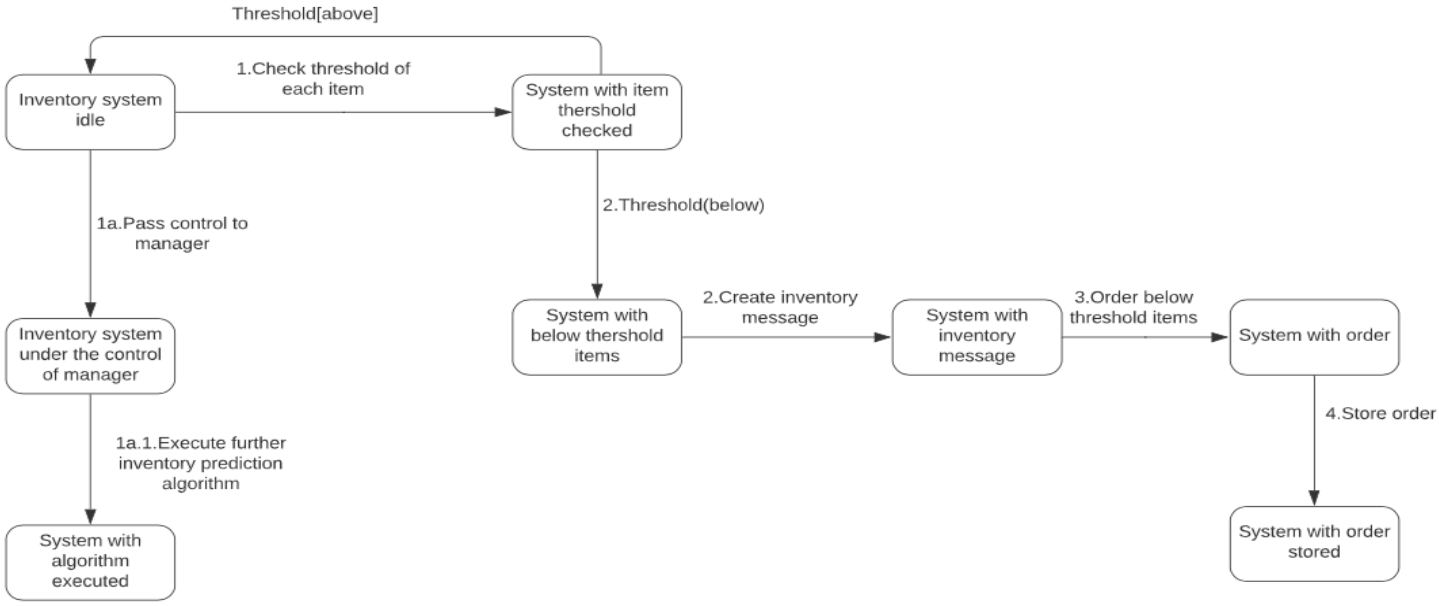
state chart for variant security cameras and alarm in self service checkout

Statechart for variant cash payment in **self service checkout**



State chart for variant cash payment in self service checkout

State chart for variant **inventory system**



STATE CHART FOR INVENTORY STSTEM

Feature/Class dependency model

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
Full Service checkout System	Common	Full service checkout control class	Kernel	
		Barcode Scanner Interface class	Kernel-param-vp	
		Credit Card reader interface class	Kernel-param-vp	
		I/p keypad interface	Kernel-abstract-vp	
		Product Inventory	Kernel-param-vp	
		CustomerDisplay	Kernel-abstract-vp	
		Check reader	Kernel-param-vp	
		SCALE	Kernel-param-vp	
		Cashier Display	Kernel-abstract-vp	
English	default	English display prompts	default	
French	alternative	French display prompts	Variant	
Spanish	alternative	Spanish display prompts	Variant	
German	alternative	German display prompts	Variant	
Loyalty Program/Membership	Optional	Membership interface	Optional	
Store coupon	Optional	Coupon interface	Optional	
Self service checkout System	Optional	Self Service Control class	kernel	
		Camera interface	Optional	
		Alarm interface	Optional	
		Cash/coin till interface	Kernel-param-vp	
		CustomerDisplay	Kernel-abstract-vp	
		Credit Card reader interface class	Kernel-param-vp	
		Coupon interface	Optional	
Extra discount	Optional	Discount Coupon	Optional	

		class		
Inventory update	Optional	Inventory class	optional	
Future inventory Prediction algorithm	Optional	Inventory class	optional	

Phase 3 - Design Modeling

The supermarket checkout software product line is designed as a component based software architecture based on a centralized control pattern. The control component provides the overall control of the system.

First, a kernel system is designed which contains the kernel and defaut components. Next the message communication between components is designed. With the evolutionary design approach the process is repeated for optional and variant components

The entire supermarket checkout system is divided into subsystems and each subsystems is divided into subcomponents

There are 3 main components each divided into sub components

1. Input component
2. Output component
3. Control component

I/P Components

1. Barcode Scanner component
2. SCALE component
3. Product Inventory component
4. i/p display/keypad component
5. Camera component
6. Credit/debit card reader component
7. Check Reader component
8. Cash/Coin till component

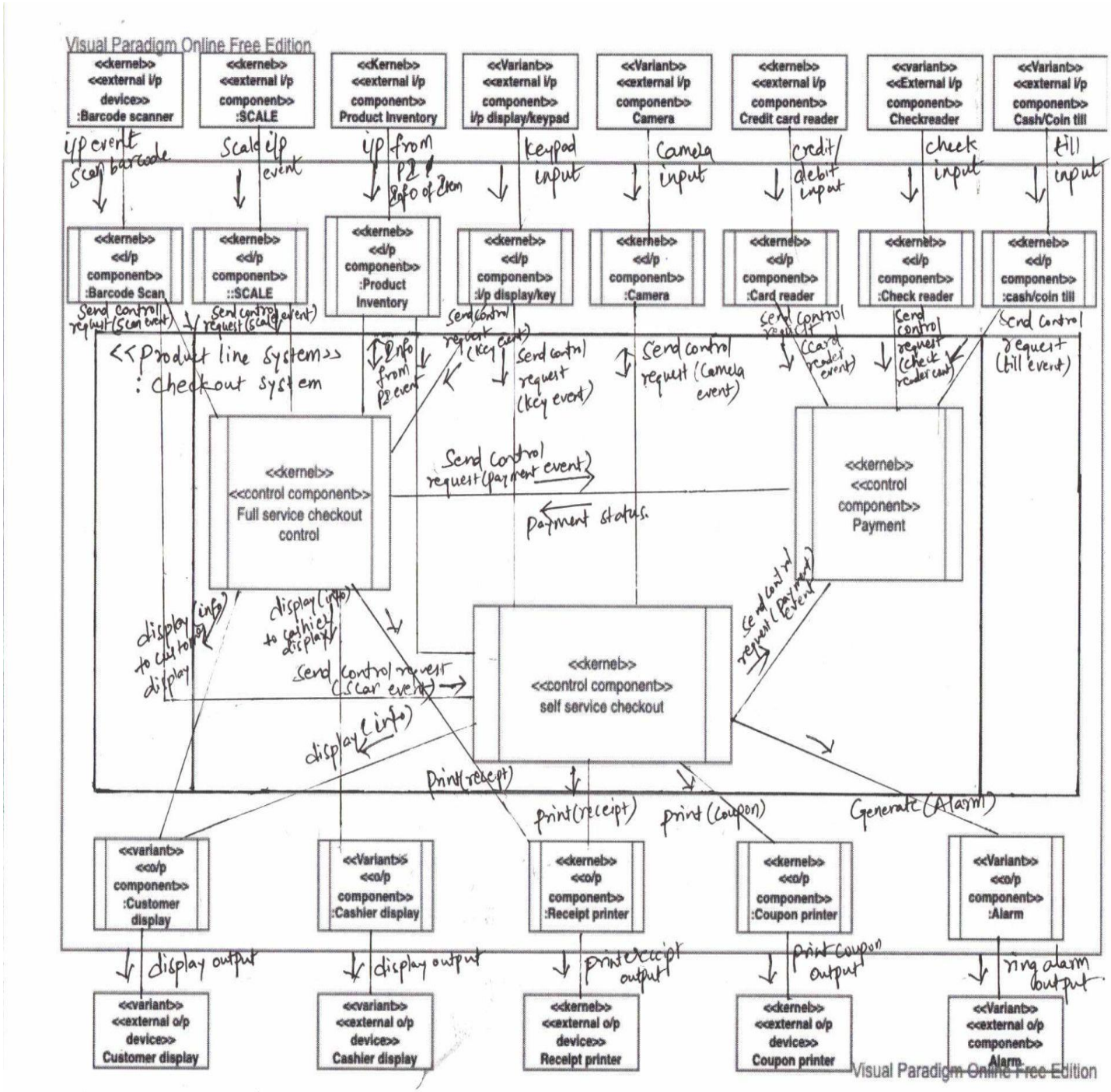
O/P Components

1. Customer Display
2. Cashier Display
3. Receipt Printer
4. Store Coupon Printer
5. Alarm

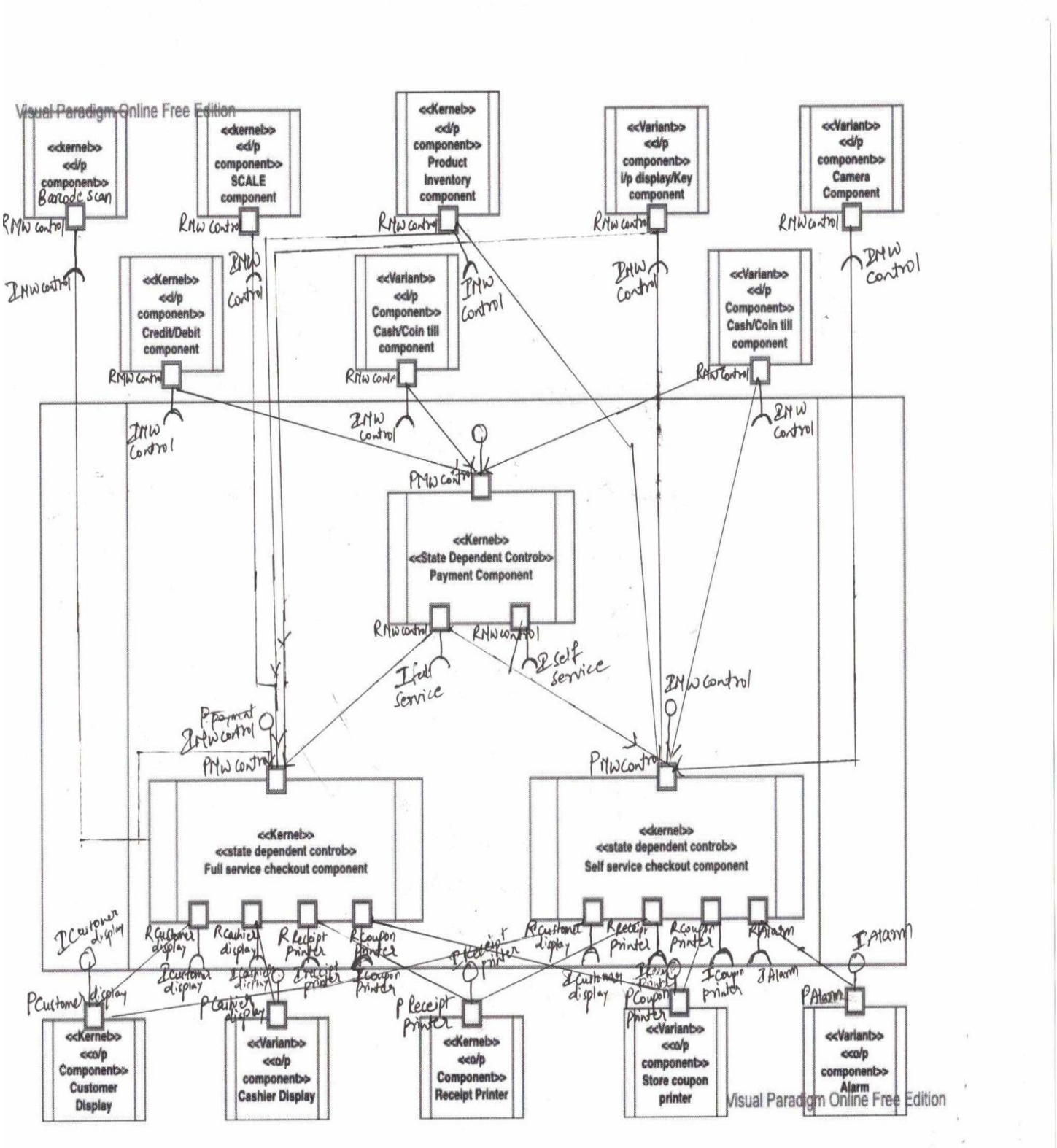
Control Components

1. Full Service Checkout
2. Self Service Checkout
3. Payment Component

Distributed software architecture for the supermarket checkout system



Component based software architecture showing the components and connectors including ports and interfaces



Code

```
package Checkout;
import java.util.*;
public class CheckoutRegister {
	public static void main( String args[])
	{
		int option=0;
		@SuppressWarnings("resource")
		Scanner sc=new Scanner(System.in);
		System.out.println("Please enter 1 for Full service or 2 for self service checkout
option");
		System.out.println("If no option selected, default checkout (Full service
checkout) will be selected");
		option=sc.nextInt();
		if(option==2)
		{
		  SelfServiceCheckout ssc=new SelfServiceCheckout();
		}
		else
		{
		  FullServiceCheckout fsc=new FullServiceCheckout();
		}
	}
}

package Checkout;
import java.util.*;
public class FullServiceCheckout {
	FullServiceCheckout()
	{
		double total=0;
		int i=0;
		@SuppressWarnings("resource")
		Scanner sc=new Scanner(System.in);
		String item="0";
		BarCodeScanner bs=new BarCodeScanner();
		@SuppressWarnings("unused")
		CashierDisplay cd1=new CashierDisplay("Scan/Enter Items one by one and hit 0
when done with last item");
		System.out.println("\n");
		item=sc.nextLine();
		while(true)
		{
			double cost=bs.scanItem(item,i);
			@SuppressWarnings("unused")
```

```
                    CashierDisplay cd2=new CashierDisplay("Scanned");
                    @SuppressWarnings("unused")
                    CustomerDisplay c1=new CustomerDisplay("\t\tScanned\n");
                    total=(double)total+cost;
                    i++;
                    System.out.println("Scan next item or hit 0 if done");
                    item=sc.nextLine();
                    if(item.equals("0")) break;
                }
                @SuppressWarnings("unused")
                CashierDisplay cd3=new
CashierDisplay("CashierDisplay\t\t\t\tCustomerDisplay\n\nTotal cost ="+total);
                @SuppressWarnings("unused")
                CustomerDisplay c3=new CustomerDisplay("Total cost ="+total);
                System.out.println("Does customer have any discount coupon");
                String answer=sc.nextLine();
                if(answer.equals("NO")) {
                Payment p=new Payment();
                EntityData ed=new EntityData();
                ed.getTotal(total,i);
                @SuppressWarnings("unused")
                ReceiptPrinter rp=new ReceiptPrinter();
                }
                else
                {
                        System.out.println("Enter Discount coupon code");
                        int coupon=sc.nextInt();
                        DiscountCoupon dc=new DiscountCoupon();
                        double newtotal=dc.Discountcoupon(coupon, total);
                        System.out.println("Total after applying 10% discount"+newtotal);
                        Payment p=new Payment();
                        EntityData ed=new EntityData();
                        ed.getTotal(newtotal,i);
                        ReceiptPrinter rp=new ReceiptPrinter();
                }
                }
        }

package Checkout;
import java.util.*;
public class BarCodeScanner {
        ProductInventory pi=new ProductInventory();
        Scanner sc=new Scanner(System.in);
        public double scanItem(String item,int i)
        {
                double cost=0.0;
```

```
                int id=0;
                if(item.equals("Bear"))
                {
                        @SuppressWarnings("unused")
                        CashierDisplay cd1=new CashierDisplay("Scan did not work..Try entering
the Item ID!!!");
                        id=sc.nextInt();
                        @SuppressWarnings("unused")
                        CashierDisplay cd2=new CashierDisplay("Getting information from
ProductInventory...\n\n");
                        cost=pi.getItemCost(id);
                        @SuppressWarnings("unused")
                        CashierDisplay cd3=new CashierDisplay(id);
                        @SuppressWarnings("unused")
                        CashierDisplay cd4=new CashierDisplay(cost);
                }
                else
                {
                        @SuppressWarnings("unused")
                        CashierDisplay cd7=new CashierDisplay("Getting information from
ProductInventory...\n\n");
                        id=pi.getItemId(item);
                        cost=pi.getItemCost(id);
                        System.out.print("Cashier Display\t");
                        System.out.println("\t\t\tCustomer Display");
                        @SuppressWarnings("unused")
                        CashierDisplay cd6=new CashierDisplay(id);
                        @SuppressWarnings("unused")
                        CashierDisplay cd5=new CashierDisplay(cost);
                        @SuppressWarnings("unused")
                        CustomerDisplay cd8=new CustomerDisplay(id);
                        @SuppressWarnings("unused")
                        CustomerDisplay cd9=new CustomerDisplay(cost);
                }
                EntityData ed=new EntityData();
                ed.getItem(item, i);
                ed.getCost(cost, i);
                ed.getID(id, i);
                return cost;
        }
        public double scanItem(String item,int i,String s)
        {
                double cost=0.0;
                int id=0;
                if(item.equals("Bear"))
                {
```

```
                System.out.println("Scan did not work..Try entering the Item ID!!!");
                id=sc.nextInt();
                System.out.println("Getting information from ProductInventory...\n\n");
                cost=pi.getItemCost(id);
                System.out.print("ID="+id);
                System.out.println("Cost="+cost);
            }
            else
            {
                System.out.println("Getting information from ProductInventory...\n\n");
                id=pi.getItemId(item);
                cost=pi.getItemCost(id);
                System.out.print("ID="+id);
                System.out.println("\t\tCost="+cost);
            }
            EntityData ed=new EntityData();
            ed.getItem(item, i);
            ed.getCost(cost, i);
            ed.getID(id, i);
            return cost;
        }
}

package Checkout;
public class ProductInventory {
        public double getItemCost(int id)
        {
            double cost=0.0;
            if(id==1234)
            {
                cost=3.25;
            }
            if(id==5678)
            {
                cost=1.25;
            }
            if(id==91011)
            {
                cost=5.75;
            }
            if(id==121314)
            {
                cost=9;
            }
            if(id==151617)
            {
```

```
                cost=11.75;
        }
        if(id==181920)
        {
                cost=2.50;
        }
        if(id==9876)
        {
                cost=8.25;
        }
        return cost;
}
public int getItemId(String s)
{
        int id=0;
        if(s.equals("Apple"))
        {
                id=1234;
        }
        if(s.equals("Banana"))
        {
                id=5678;
        }
        if(s.equals("Orange"))
        {
                id=91011;
        }
        if(s.equals("Bread"))
        {
                id=121314;
                System.out.println("This item is eligible for store coupon");
                StoreCouponPrinter scp=new StoreCouponPrinter();
        }
        if(s.equals("Milk"))
        {
                id=151617;
        }
        if(s.equals("Yogurt"))
        {
                id=182920;
        }
        if(s.equals("Bear"))
        {
                id=9876;
        }
        return id;
```

```
	}
}

package Checkout;
import java.util.Scanner;
public class Payment {
	Payment()
	{
		@SuppressWarnings("resource")
		Scanner sc=new Scanner(System.in);
		@SuppressWarnings("unused")
		CustomerDisplay c1=new CustomerDisplay("Select Payment method \n\t\t\t1:
Credit/Debit Card\t 2: Check");
		int option=sc.nextInt();
		if(option==1)
		{
			@SuppressWarnings("unused")
			CreditCardReader ccr=new CreditCardReader();
		}
		else if(option==2)
		{
			@SuppressWarnings("unused")
			CheckReader cr=new CheckReader();
		}
	}
	Payment(String s,double total)
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Select Payment method \n1: Credit/Debit Card\t 2: Cash");
		int option=sc.nextInt();
		if(option==1)
		{
			CreditCardReader ccr=new CreditCardReader("SelfserviceCheckout");
		}
		else if(option==2)
		{
			Cash ch=new Cash(s,total);
		}
	}
}

package Checkout;
public class EntityData {
	static String items[]=new String[10];
	static int ids[]=new int[10];
	static double costs[]=new double[10];
```

```
	static double total;
	static int count;
	void getCost(double cost,int i)
	{
		EntityData.costs[i]=cost;
	}
	void getID(int id, int i)
	{
		EntityData.ids[i]=id;
	}
	void getItem(String item,int i)
	{
		EntityData.items[i]=item;
	}
	void getTotal(double total,int count)
	{
		EntityData.count=count;
	}
}

package Checkout;
public class CustomerDisplay {
	CustomerDisplay(String message)
	{
		System.out.println("\t\t\t"+message);
	}
	CustomerDisplay(int message)
	{
		System.out.print("\t\t\t id="+message);
	}
	CustomerDisplay(double message)
	{
		System.out.println("\tcost="+message);
	}
}

package Checkout;
public class CashierDisplay {
	CashierDisplay(String message)
	{
		System.out.print("\n"+message);
	}
	CashierDisplay(int message)
	{
		System.out.print("ID="+message);
	}
```

```
	CashierDisplay(double message)
	{
		System.out.print("\tCost="+message);
	}
}

package Checkout;
import java.util.*;
public class Cash {
	Cash(String s,double total)
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Please insert "+total+"$ cash in the till");
		double cashpaid=sc.nextDouble();
		if(cashpaid==0.0 && s.equals("YES")) System.out.println("Alarm Activated");
		double cashbalance=cashpaid-total;
		while(cashbalance!=0.0)
		{
		 if(cashbalance>0.0)
		 {
			System.out.println("Please collect change "+cashbalance+"$");
			cashbalance=0.0;
		 }
		 else if (cashbalance<0.0)
		 {
			System.out.println("Please insert "+cashbalance*(-1)+" to complete the
transaction");
			double cash=sc.nextDouble();
			cashbalance=cashbalance+cash;
		 }
		}
		System.out.println("Thankyou");
	}
}

package Checkout;
import java.util.*;
public class CreditCardReader {

	CreditCardReader()
	{
		@SuppressWarnings("resource")
		Scanner sc=new Scanner(System.in);
		@SuppressWarnings("unused")
		CustomerDisplay c1=new CustomerDisplay("Please insert/enter credit card");
		String card=sc.next();
```

```
                @SuppressWarnings("unused")
                CustomerDisplay c2=new CustomerDisplay("Enter Expiry date");
                String expiry=sc.next();
                BankServer bs=new BankServer();
                int status=bs.Bankserver(card, expiry);
                if(status==1)
                {
                        CashierDisplay cd1=new CashierDisplay("Transaction authorized !!");
                        CustomerDisplay c3=new CustomerDisplay("Transaction authorized !!");
                }
                else
                {
                        CashierDisplay cd2=new CashierDisplay("Transaction not
approved..Order cancelled!!");
                        CustomerDisplay c4=new CustomerDisplay("Transaction not
approved..Order cancelled!!");
                }
        }
        CreditCardReader(String s)
        {
                @SuppressWarnings("resource")
                Scanner sc=new Scanner(System.in);
                @SuppressWarnings("unused")
                CustomerDisplay c1=new CustomerDisplay("Please insert/enter credit card");
                String card=sc.next();
                @SuppressWarnings("unused")
                CustomerDisplay c2=new CustomerDisplay("Enter Expiry date");
                String expiry=sc.next();
                BankServer bs=new BankServer();
                int status=bs.Bankserver(card, expiry);
                if(status==1)
                {
                        CustomerDisplay c3=new CustomerDisplay("Transaction authorized !!");
                }
                else
                {
                        CustomerDisplay c4=new CustomerDisplay("Transaction not
approved..Order cancelled!!");
                        System.out.println("Alarm activated");
                }
        }
}

package Checkout;
import java.util.*;
public class CheckReader {
```

```
        String checks[] = {"abc","def"};
        CheckReader()
        {
                Scanner sc=new Scanner(System.in);
                CustomerDisplay cd1=new CustomerDisplay("Enter check");
                String check=sc.next();
                for(int i=0;i<2;i++)
                {
                        if(check==this.checks[i])
                        {
                                CashierDisplay c1=new CashierDisplay("Check verified");
                                CustomerDisplay cd2=new CustomerDisplay("Check verified");
                        }

                }
                CashierDisplay c2=new CashierDisplay("Check not valid..Order cancelled!!!");
                CustomerDisplay cd3=new CustomerDisplay("Check not valid..Order
cancelled!!!");
        }
}

package Checkout;
public class BankServer {
        String cards[]= {"123456789","987654321"};
        String expiry[]= {"1024","1025"};
        public int Bankserver(String card,String expiry)
        {
                for(int i=0;i<2;i++)
                {
                        if(card.equals(this.cards[i]) && expiry.equals(this.expiry[i]))
                        {
                                return 1;
                        }
                }
                return 0;
        }
}

package Checkout;
import java.util.*;
public class SelfServiceCheckout {
        SelfServiceCheckout()
        {
                System.out.println("Do you wish to use security feature in self service
checkout");
                Scanner sc= new Scanner(System.in);
```

```
                String response=sc.nextLine();
                if(response.equals("NO"))
                {
                String item=null;
                double total=0.0;
                int i=0;
                System.out.println("Welcome!!\nPress START button to scan");
                BarCodeScanner bs=new BarCodeScanner();
                System.out.println("Scan/Enter Items one by one and hit 0 when done with last
item");
                item=sc.nextLine();
                while(true)
                {
                        double cost=bs.scanItem(item,i,"selfservicecheckout");
                        System.out.println("Scanned\n");
                        total=(double)total+cost;
                        i++;
                        System.out.println("Scan next item or hit 0 if done");
                        item=sc.nextLine();
                        if(item.equals("0")) break;
                }
                System.out.println("Total cost ="+total);
                System.out.println("Does customer have any discount coupon");
                String answer=sc.nextLine();
                if(answer.equals("NO")) {
                Payment p=new Payment("SelfServiceCheckout",total);
                EntityData ed=new EntityData();
                ed.getTotal(total,i);
                ReceiptPrinter rp=new ReceiptPrinter();
                }
                else
                {
                        System.out.println("Enter Discount coupon code");
                        int coupon=sc.nextInt();
                        DiscountCoupon dc=new DiscountCoupon();
                        double newtotal=dc.Discountcoupon(coupon, total);
                        System.out.println("Total after applying 10% discount"+newtotal);
                        Payment p=new Payment("SelfServiceCheckout",newtotal);
                        EntityData ed=new EntityData();
                        ed.getTotal(newtotal,i);
                        ReceiptPrinter rp=new ReceiptPrinter();
                }
        }
        else
        {
                String item=null;
```

```
                double total=0.0;
                int i=0;
                System.out.println("Welcome!!\nPress START button to scan");
                BarCodeScanner bs=new BarCodeScanner();
                System.out.println("Scan/Enter Items one by one and hit 0 when done with last
item");
                item=sc.nextLine();
                while(true)
                {
                        double cost=bs.scanItem(item,i);
                        Security s=new Security();
                        System.out.println("Scanned\n");
                        total=(double)total+cost;
                        i++;
                        System.out.println("Scan next item or hit 0 if done");
                        item=sc.nextLine();
                        if(item.equals("0")) break;
                }
                System.out.println("Total cost = "+total);
                System.out.println("Does customer have any discount coupon");
                String answer=sc.nextLine();
                if(answer.equals("NO")) {
                Payment p=new Payment(response,total);
                EntityData ed=new EntityData();
                ed.getTotal(total,i);
                ReceiptPrinter rp=new ReceiptPrinter();
                }
                else
                {
                        System.out.println("Enter Discount coupon code");
                        int coupon=sc.nextInt();
                        DiscountCoupon dc=new DiscountCoupon();
                        double newtotal=dc.Discountcoupon(coupon, total);
                        System.out.println("Total after 10% discount"+newtotal);
                        Payment p=new Payment(response,newtotal);
                        EntityData ed=new EntityData();
                        ed.getTotal(newtotal,i);
                        ReceiptPrinter rp=new ReceiptPrinter();
                }
        }
        }
}

package Checkout;

public class Security {
```

```
	Security()
	{
		System.out.println("Camera Activated");
	}

}

package Checkout;

public class StoreCouponPrinter {
	StoreCouponPrinter()
	{
		System.out.println("Store coupon printed..Please collect");
	}
}

package Checkout;

public class DiscountCoupon {
	int coupons[]= {420,840,1260};
	public double Discountcoupon(int coupon,double total)
	{
		int i=0;
		while(i<coupons.length)
		{
			if(coupon==coupons[i]) {
				System.out.println("Discount applied");
				total=total-0.1*total;
				return total;
			}
			else
				System.out.println("Coupon not valid");
			i++;
		}
		return total;
	}
}
```

O/P Screenshots

```
Please enter 1 for Full service or 2 for self service checkout option
If no option selected, default checkout (Full service checkout) will be selec
2
Do you wish to use security feature in self service checkout
YES
Welcome!!
Press START button to scan
Scan/Enter Items one by one and hit 0 when done with last item
Apple

Getting information from ProductInventory...

Cashier Display                          Customer Display
ID=1234 Cost=3.25                         id=1234        cost=3.25
Camera Activated
Scanned

Scan next item or hit 0 if done
Banana

Getting information from ProductInventory...

Cashier Display                          Customer Display
```

```
420
Discount applied
Total after 10% discount4.05
Select Payment method
1: Credit/Debit Card     2: Cash
2
Please insert 4.05$ cash in the till
0.0
Alarm Activated
Please insert 4.05 to complete the transaction
5
Please collect change 0.9500000000000002$
Thankyou

Printing Receipt..

Item    ID      Cost

Apple   1234    3.25
Banana  5678    1.25

Thank you!!
```

```
Please enter 1 for Full service or 2 for self service checkout option
If no option selected, default checkout (Full service checkout) will be selec
```

```
Please enter 1 for Full service or 2 for self service checkout option
If no option selected, default checkout (Full service checkout) will be selec
1

Scan/Enter Items one by one and hit 0 when done with last item

Apple

Getting information from ProductInventory...

Cashier Display                          Customer Display
ID=1234 Cost=3.25                         id=1234        cost=3.25

Scanned                                  Scanned

Scan next item or hit 0 if done
Banana
```

```
Scan next item or hit 0 if done
0

CashierDisplay                           CustomerDisplay

Total cost =4.5                 Total cost =4.5
Does customer have any discount coupon
```

```
Discount applied
Total after applying 10% discount4.05
                        Select Payment method
                        1: Credit/Debit Card     2: Check
1
                        Please insert/enter credit card
123456789
                        Enter Expiry date
1024

Transaction authorized !!                        Transaction authorized !!

Printing Receipt..
```

```
Transaction authorized !!                        Transaction authorized !!

Printing Receipt..

Item    ID      Cost

Apple   1234    3.25
Banana  5678    1.25

Thank you!!
```