**EXPERIMENT2: Date:**

**CASE STUDY: POINT OF SALE TERMINAL**

1. Identify and analyze events

**Procedure**:

The Flow of events of a usecase contains the most important information derived from usecase modeling work. It should describe the usecases flow of events clearly enough for an outsider to easily understand it. Remember flow of events should present what the system does, not how the system is design to perform the required behavior.

Guidelines for the contents of the flow of events are,

1) Describe how the usecase starts and ends.

2) Describe what data is exchanged between the actor and the usecase

3) Do not describe the details of the user interface, unless it is necessary to understand the behavior of the system.

For example, it is often good to use a limited set of web specific terminology when it is known beforehand that the application is going to be web based otherwise your run the risk that the usecase text is being perceived as too abstract.words to include in your terminology could be “navigate”,”browser”,”hyperlink”,”page”,”submit”,and ,”browser”. However it is not as visible to include references to frames or “webpages” in such a way that you are making assumptions about the boundaries between them this is a critical design decision.

🡪Describe the flow of events, not only the functionality to enforce this, start every action with “when the actor”.

🡪Describe only the events that belong to the usecase and not what happens in other usecases or outside of the system.

🡪Avoid vague terminology such as “for example”,” information”.

🡪Detail the flow of events all what should be answered .remember that test designers are to be use this text to identify test cases.

If you have used certain terms in other use cases, be sure to use the exact same terms in this usecase, and that their intented meaning is the same. To manage common terms, put them in a glossary.

**b)Aim: Identify Usecases**

**Procedure:**

1) Buy Product

2) Barcode scanning

3) Pay Bill

4) Process Sale

5) Close Sale

6) Update Inventory

7) Tax Calculator

**1) Buy product:**

Among the multiple products we will select which product we should buy

**2) Barcode Scanning:**

Here the barcode is scanned for the product which we are selected. The barcode will be ready predefined on each and every product and according to that the price of the product will be displayed.

**3) Pay bill:**

Pay bill is a bill in which the price of the product is printed, according to that price the customers will play

**4) Process sale:**

After receiving the bill the products which we have selected are processed to the cashier counter and they will cross check our products according to our bill.

**5) Close sale:**

In this close sale the customer will pay the bill and cashier place a stamp on that bill, that is represents the bill is closed.

**6) Update inventory:**

In this the cashier will update the database according to the products in that store.

**7) Tax calculation:**

The cashier will calculate the tax in various forms such as transportation cost, maintenance cost.

**Graphical representations:**

****

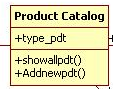
**c) Aim: Develop Event table**

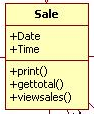
**Event table:**

|  |  |
| --- | --- |
| **USECASE** | **DESCRIPTION** |
| Buy product | Among the multiple products we will select product we should buy |
| Barcode scanning | Here the barcode is scanned for the product which we are selected |
| Pay bill | In this price of the product is printed |
| Process sale | The bill, products are processed to the cashier counter |
| Close sale | The customer will pay the bill and cashier place a stamp on that bill |
| Update inventory | The cashier will update the database according to the products in that store |
| Tax Calculation | The cashier will calculate the tax in various forms such as transportation cost, maintainence cost etc. |

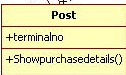
**d) Aim: Identify and analyze domain classes**

**Domain classes:** A Domain class is a class its description of set of objects that share the same attribute , operations, relationships. A class implements one or more interfaces







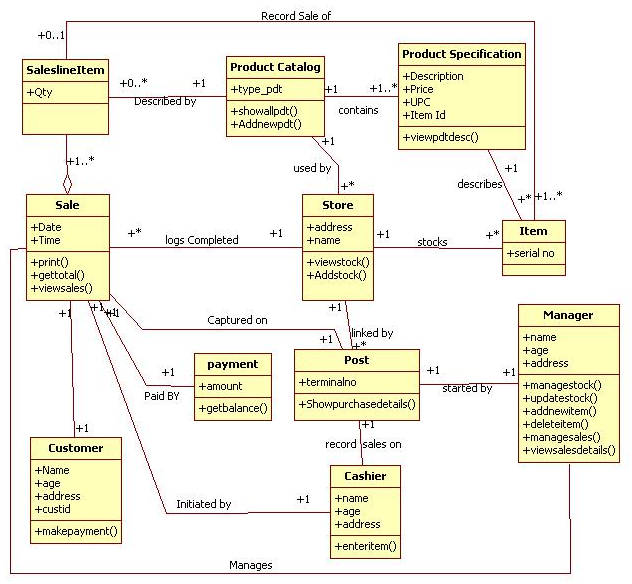


**e)Aim:** Represent usecases and domain class diagram using rational rose

**->Usecase Diagram:**



**>Class diagram:**



**f)Aim:Develop CRUD matrix to represent relationships between usecases and problem domain classes**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity/function** | **Customer** | **cashier** | **Product catalog** | **Product specification** | **Sale** | **Store** | **Payment** | **Post** | **manager** |
| Buy product | C,D | R | C,R,U | R | U | U |  | R | R |
| Barcode scanning |  | R |  |  | R |  | C |  |  |
| Paybill | R | C |  |  |  |  | R |  |  |
| Process sale |  |  | R |  | C |  |  |  |  |
| Close sale |  |  | C,R | R | R | U |  | R | R |
| Update inventory |  |  | C,R | U | U | U |  | R | U |
| Tax calculation |  | C |  |  |  |  | R | R | C,R |

**Result:** The Design was successfully completed.