SUPERMARKET AUTOMATION SOFTWARE

USE CASE AND UML DIAGRMAS

Vatsalya Chauhan
12CS10053

Kulkarni Bhushan
12CS30016

INDEX

1. USE CASE DIAGRAM

The purpose of this diagram is to demonstrate how objects will interact with SAS and map out the basic functionality of the system. Below is a list of the elements that you will see in the diagram on the next page as well what is included in the use case templates that follow.

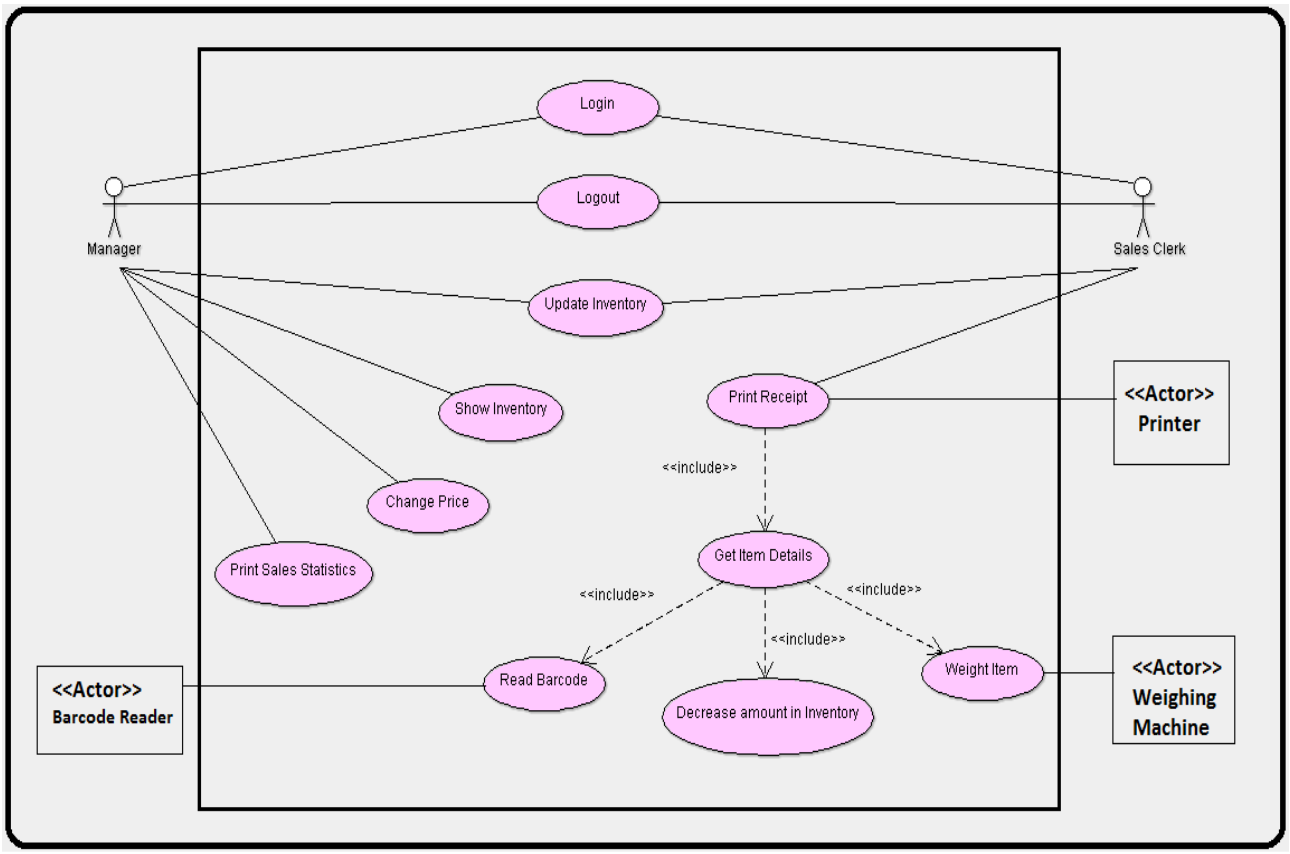
ELEMENTS	DESCRIPTION
ACTORS	Shown in the diagram as stick figures with a name underneath. They represent elements that will be directly interacting with the system.
USE CASES	Oval shapes that have their names in the centre. These represent direct functionality within the system that must be implemented.
INTERACTIONS	Lines that connect the actors with the different Use Cases. These show that there is some form of direct interaction between the actor and that specific functionality.
INCLUDES	Dotted lines labelled “<<include>>” that connect two use cases and have an arrow pointing towards one. This means that the use case without the arrow calls on the functionality of the use case with the arrow.
SYSTEM BOUNDARY	The large rectangle that contains the Use Cases. Everything within the rectangle is what the system is responsible for implementing
TYPE	A field in the use case template that states whether or not the use case is directly interacted with by an actor (Primary) or not (Secondary) as well as whether or not it is essential to having a functioning system.
USE-CASES	A field in the use case templates that state which other use cases must be executed prior to that particular use case.

1.1. Use Case: Login

Actors: Manager, Sales Clerk
Type: Primary and essential
Description: Initiated when a user tries to access his account. The user is then Prompted to enter in their username and password in order to proceed.
Includes: None
Use-Cases: None

1.2. Use Case: Print Receipt

Actors: Printer
Type: Primary and essential
Description: Allows the sales clerk to print the bill
Includes: Get Item details
Use-Cases: The Login use case must be completed.



1.2.1. Use Case: Get Item Details

Actors: Sales Clerk, Manager
Type: Primary and essential
Description: Allows the sale clerk to get the details of the items bought by the customer.
Includes: Read Barcode, Weight Item, Modify Inventory
Use-Cases: The Login use case must be completed.

1.2.2. Use Case: Read Barcode

Actors: Barcode Reader
Type: Primary and essential
Description: Allows the sale clerk to get the details about type of item bought by the customer
Includes: None
Use-Cases: The Login use case must be completed.

1.2.3. Use Case: Weight Item

Actors: Weighing Machine
Type: Primary and essential
Description: Allows the sale clerk to get the details about quantity of item bought by the customer
Includes: None
Use-Cases: The Login use case must be completed.

1.2.4. Use Case: Modify Inventory

Actors: None
Type: Primary and essential
Description: Decrements sold items' quantities from Inventory.
Includes: None
Use-Cases: The Login use case must be completed.

1.3. Use Case: Check Inventory

Actors: Manager
Type: Primary and essential
Description: Allows the manager to view the items present in our database.
Includes: None
Use-Cases: The Login use case must be completed.

1.4. Use Case: Change price

Actors: Manager
Type: Primary
Description: Allows the manager to change the price of a particular item
Includes: None
Use-Cases: The Login use case must be completed.

1.5. Use Case: Show Sales Statistics

Actors: Manager
Type: Primary and essential
Description: Allows the manager to view the sales statistics
Includes: None
Use-Cases: The Login use case must be completed.

1.6. Use Case: Update Inventory

Actors: Manager and Sales clerk
Type: Primary and essential
Description: To change Details of newly purchased items in Inventory.
Includes: None
Use-Cases: The Login use case must be completed.

1.7. Use Case: Add new Item

Actors: Manager and Sales clerk
Type: Primary and essential
Description: Allows an employee to add new items to the database
Includes: None
Use-Cases: The Login use case must be completed.

1..8. Use Case: Add new Employee

Actors: Manager
Type: Primary and essential
Description: Allows the manager to hire new employee in the supermarket.
Includes: None
Use-Cases: The Login use case must be completed.

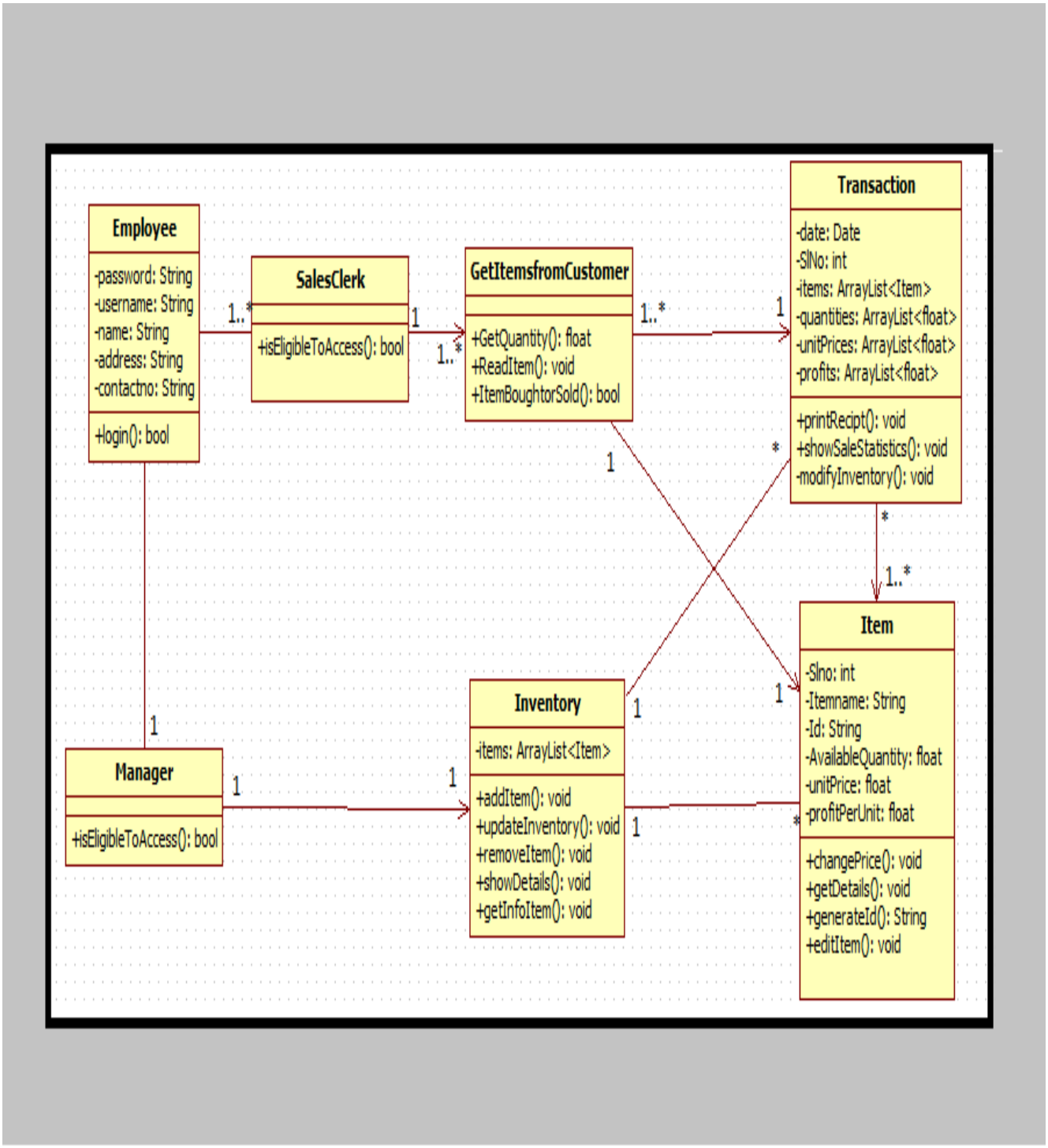
1.9. Use Case: Logout

Actors: Manager, Customer
Type: Primary and essential
Description: The customer should have the option to logout from his account when he is not going to be active for some time.
Includes: None
Use-Cases: The Login use case must be completed.

2. Class Diagrams

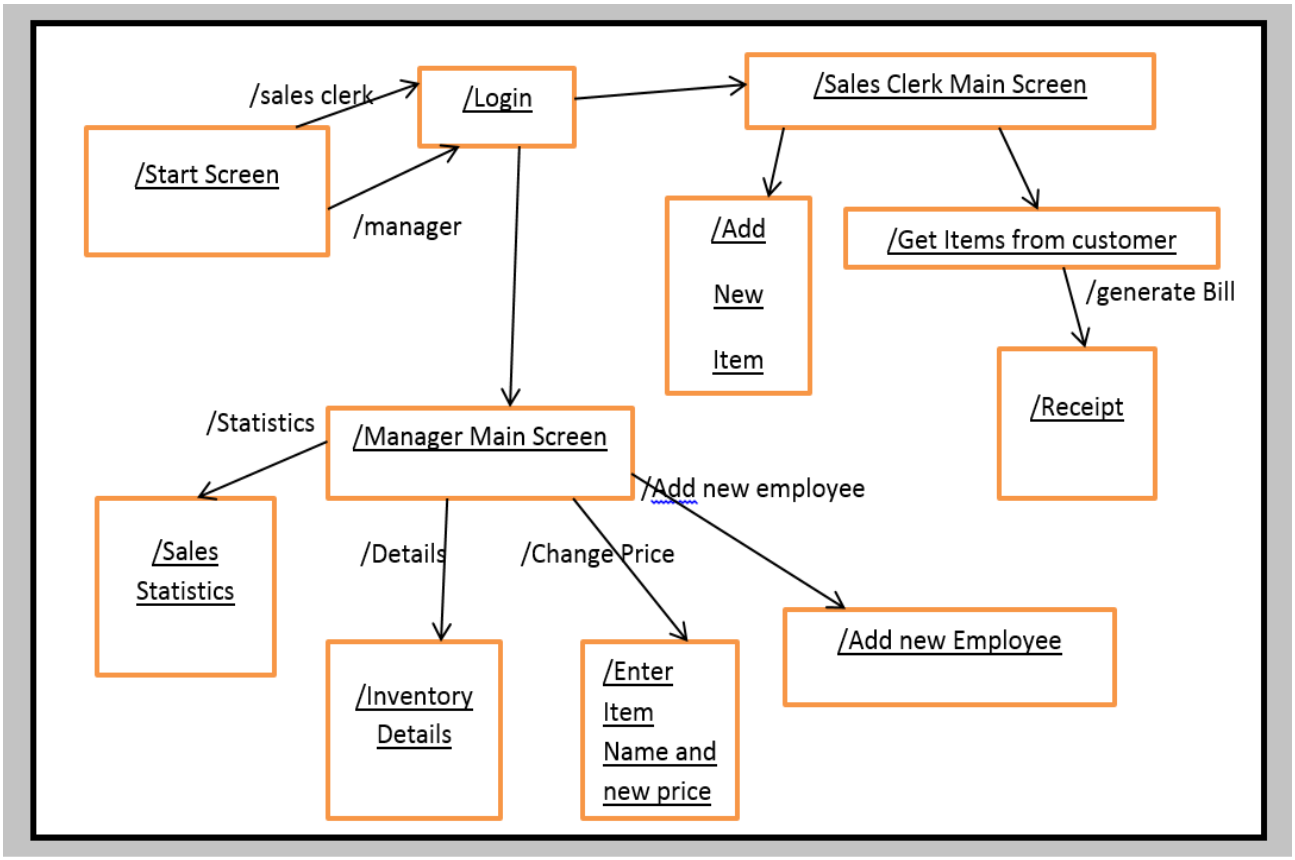
The purpose of this diagram is to show how objects within the SAS system will interact with each other in order to achieve the functionality required by the Use Case diagram. Below is a list of what you will see in the diagram itself as well as the class descriptions that follow.

ELEMENTS	DESCRIPTION
Classes	Rectangles in the diagram that are split into three parts. The top section is the name of the class, the middle section is the list of variables that are stored in the class and the bottom section is the list of functions in the class. These rectangles represent objects within the system.
Variables	These have a name followed by a semicolon and then a type. The type denotes what kind of data can be stored in the variable.
Functions	These have a name followed by a list of any variable that the function receives in-between the parenthesis "()". After that there is a semicolon and any variables that the function may return, if none it will be void.
Generalizations	Shown using a line from one object to the other with an unfilled triangle on one end. The object without the triangle inherits the functionality and variables from the object that has the triangle pointing towards it.
Aggregations	Lines that have an unfilled diamond on one end. This means the object with the diamond contains the object(s) without the diamond. This may have numbers on the ends (multiplicities).
Associations	Lines connecting two classes that can have a name beside it, may point in one direction, and may have numbers at the ends (multiplicities). These designate some relationship between the objects. Arrows are simply there to assist you in recognizing which direction the name of the association is read.


Employee
-password: String
-username: String
-name: String
-address: String
-contactno: String
+login(): bool
SalesClerk
+isEligibleToAccess(): bool
GetItemsfromCustomer
+GetQuantity(): float
+ReadItem(): void
+ItemBoughtorSold(): bool
Transaction
-date: Date
-SlNo: int
-items: ArrayList<Item>
-quantities: ArrayList<float>
-unitPrices: ArrayList<float>
-profits: ArrayList<float>
+printRecipt(): void
+showSaleStatistics(): void
-modifyInventory(): void
Item
-Slno: int
-Itemname: String
-Id: String
-AvailableQuantity: float
-unitPrice: float
-profitPerUnit: float
+changePrice(): void
+getDetails(): void
+generateId(): String
+editItem(): void
Inventory
-items: ArrayList<Item>
+addItem(): void
+updateInventory(): void
+removeItem(): void
+showDetails(): void
+getInfoItem(): void
Manager
+isEligibleToAccess(): bool
1
1..*
1
1..*
1..*
1
1
*
*
1..*
1
1
1
1
1
*

3. Interface Diagram

In UML modeling, *interfaces* are model elements that define sets of operations that other model elements, such as classes, or components must implement. An implementing model element realizes an interface by overriding each of the operations that the interface declares. Interfaces support the hiding of information and protect client code by publicly declaring certain behavior or services.

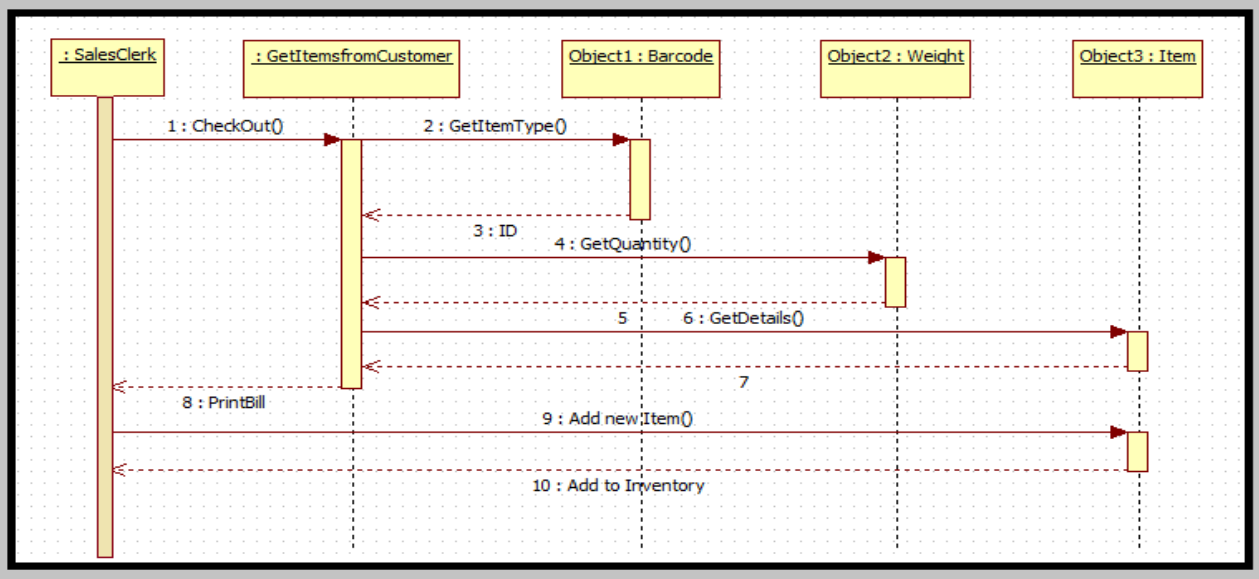


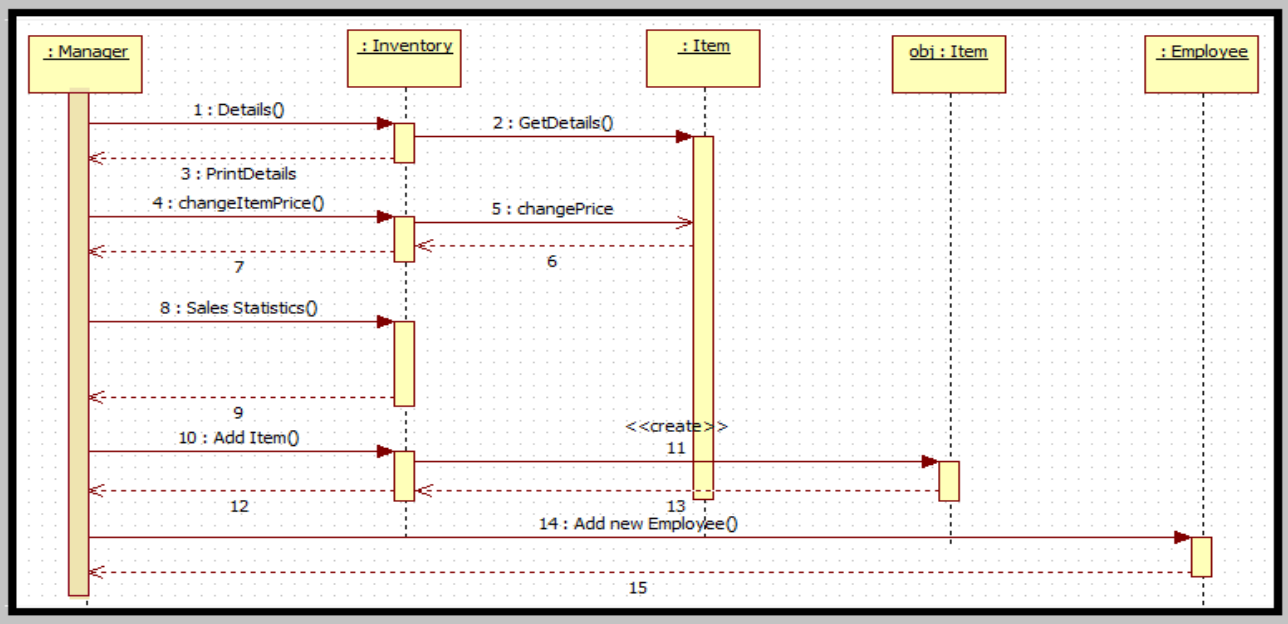
4. Sequence Diagrams

The sequence diagrams use the class diagram and demonstrate specific sequences of actions in the system. The purpose is to ensure that the SAS system runs in an expected way and that the class structure is sufficient to accomplish the tasks needed. Below is a list of the items that you will see in the diagram and their definitions.

ELEMENTS	DESCRIPTION
Axis's	The X-axis identifies movement between objects and the Y-axis identifies time.
Instances	Solid boxes along the top that have dotted lines that stretch vertically below them. These are specific instances of an object. The first part of the title is the name of that specific instance and the object it is an instance of follows it. (Special Note: If there are multiple instances that have the same title then they are actually the same instance and are only there to diagram calls onto themselves.)

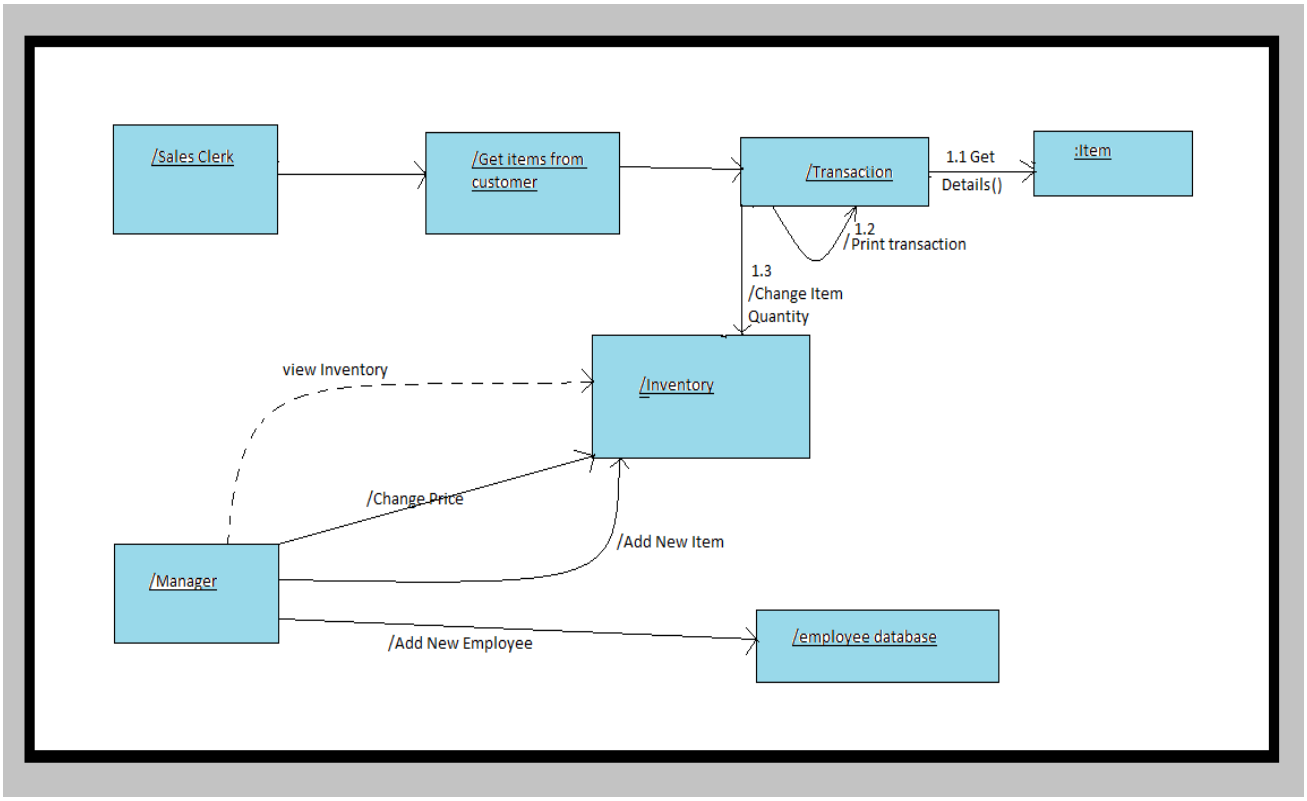
Calls	Lines that have filled triangles at the ends. These are transitions from one instance to another and have a label above them that is a function call, a variable being set, or both. It can also have a guard statement that precedes it.
Object Execution Time	This is shown with the solid white boxes that run vertically along the dotted lines. These simply represent the execution time for the objects. (Special Note: the first object has a solid box all the way down this is a special case and should not be there and is due to the application used to create the diagram).





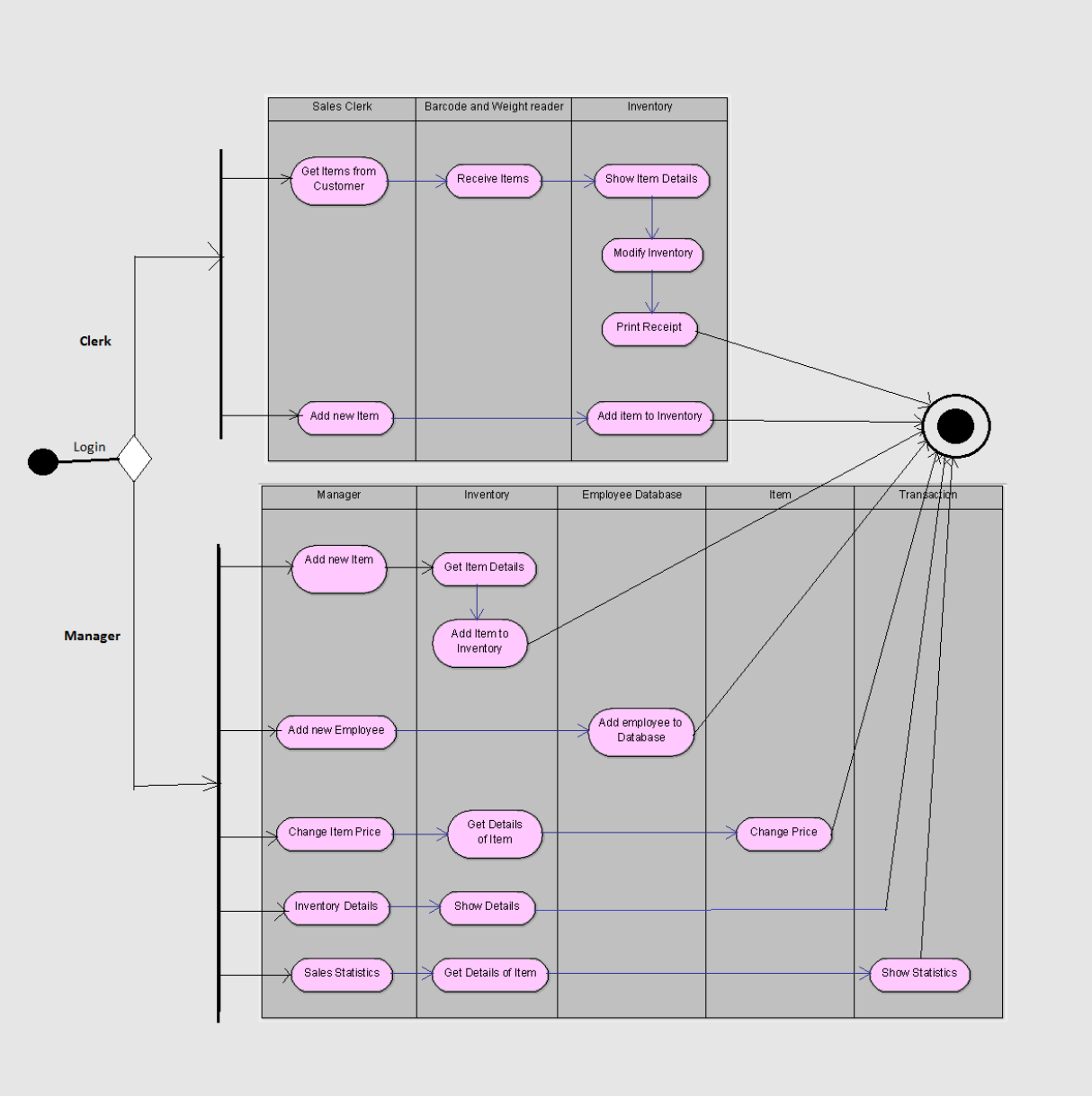
5. Collaboration Diagrams

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.



6. Activity Diagram

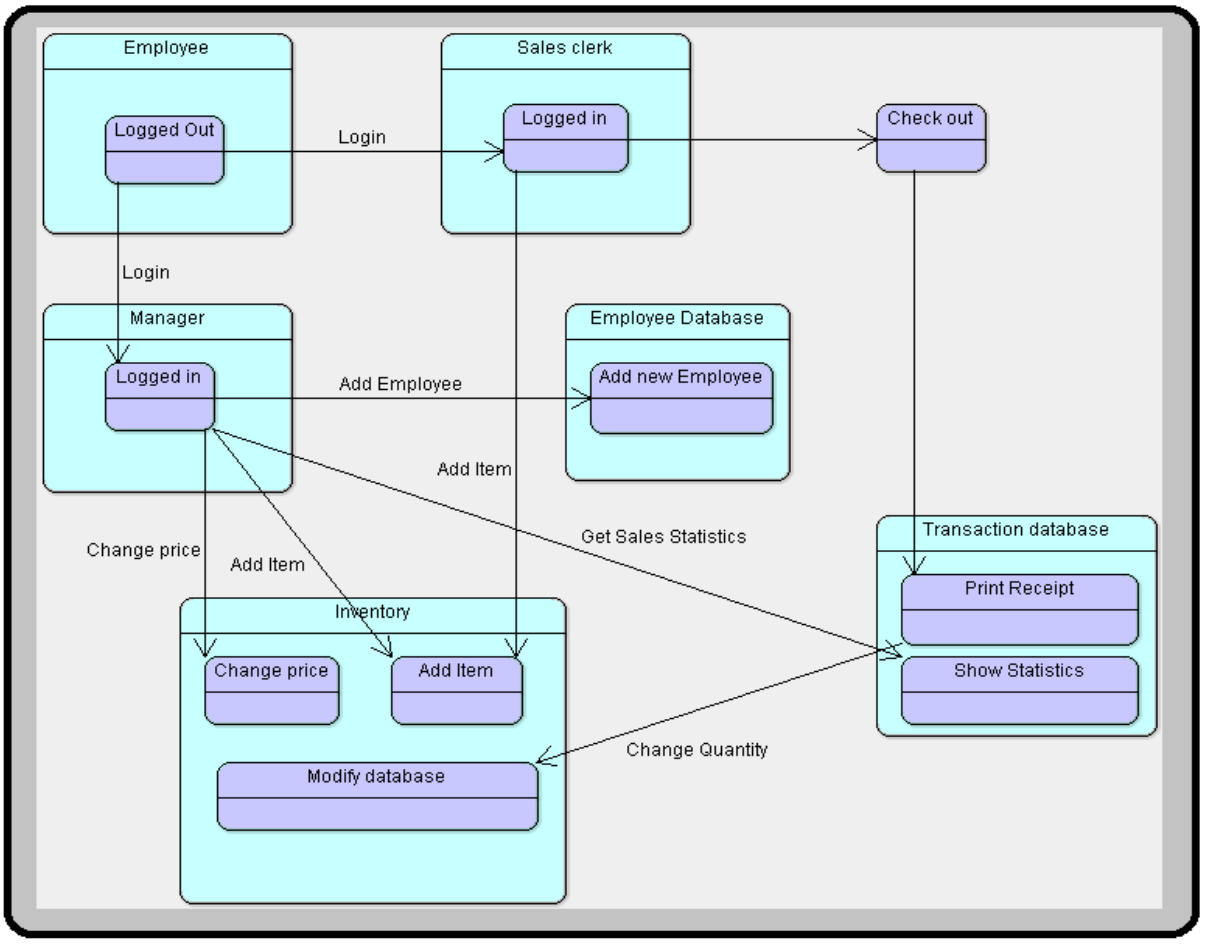
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.



7. Statechart Diagrams

The state diagrams take all of the functionality found in the previous diagrams and combines them together to demonstrate the possible changes in the state within SAS. These state diagrams contain all the possible scenarios shown in the sequence diagrams.

ELEMENTS	DESCRIPTION
Starting Point	This is where the system starts at and it is represented with a filled circle that has an arrow that point towards the starting state.
States	These are represented in the diagram by the boxes that have the rounded edges. They are separated into two half's the top half is the title of the state and the bottom half can have additional states encapsulated within but these state diagrams do not contain any encapsulated states within states.
Transitions	The arrows that connect the states to each other and have a label of the event that occurs which triggers the transition. These triggers are normally functions but they can also be a new call that is the event of creating a new object; this is represented simply with the word "new". Some transitions do not have a label, these are returns from the current state to the previous state.
Objects	The larger boxes that can contain several states and transitions. They are classes within the state diagram. The text at the top of the box is the name of the class.


Employee
Logged Out
Login
Sales clerk
Logged in
Check out
Login
Manager
Logged in
Add Employee
Employee Database
Add new Employee
Add Item
Change price
Add Item
Get Sales Statistics
Transaction database
Print Receipt
Show Statistics
Inventory
Change price
Add Item
Change Quantity
Modify database

Thank you…