New Retail Store Creation and Maintenance

Group-2

Requirements Specification

# Introduction

Scalability and profitability are the two factors that determine the growth of any business. In retail industry, it is very common that a new store gets established very often. This immediately creates the need for the new store to have the necessary software to manage its day to day tasks.

It is a hectic task to create a new web application and provide separate access to each new sub branch that is set up that includes maintenance of its data to track the business related operations.

To overcome this problem, we have come up with a solution that gave us base for developing our project “New Retail Store Creation and Maintenance”. Our main objective behind developing this project is to avoid the redundancy and re-organization of the existing data whenever a new store is created from the existing parent store.

## Project Overview and Statement of Proposal

Our project "New Retail Store Creation and Maintenance" aims to quickly provide software services required to run a new store.

**Statement of Proposal:**

We propose to automate the task of providing a web application service to the newly created store from an existing parent store with just a single click by the new Store Head. When an existing parent store establishes a new sub store irrespective of the location, the system would enable the client to provide all the required services corresponding to the new store. This is an automated process, in which client configures required data and the store is created automatically.

The application allows creation of new stores and defines the departments present in the store. Inventory of various items sold in the stores is maintained and updated accordingly. Store customer will be issued a store credit card upon request. Points are added to the customer credit card based on the purchases made. The system generates bill for the items that have been sold. The application allows special deals to be configured on items sold in the store.

## Scope and Objectives

The various objectives that are in the scope of the project can be broadly classified as follows:

### Creation of New Stores

The application allows creation of new store, configure the store with details such as the various departments the store has and assign department heads to each department. The task of creation of a new store is carried out by Store Head.

### Inventory Management

The inventory keeps track of various items that a store sells. Sales, promotions, customer rewards are interlinked with the inventory. Store head can manipulate items of any department. Department head can manipulate items in his/her department. Sales, returns, promotions, customer rewards also trigger changes/ updates to inventory.

### Promotions and Clearances

It is common for stores to provide special discounts on products during sales. Promotion and Clearances allow such special discounts to be set up on various items sold by a store. Store head can create promotions and clearances over items of any department. The department head can create promotions and clearances over items of his/her department.

### Customer Points System

The application allows customers with the store credit card eligible to earn points upon purchases. These points are linked with the store credit card. Points need to be added upon purchase and deducted accordingly upon returns. Upon accumulation of certain points, the points can be used to claim an extra discount offered by the store.

### Point of Sales

All the activities that happen at a point of sale need to be tracked. Activities include generation of bills, handling returns, updating inventory accordingly, updating reward points if the customer owns a store credit card.

**Out of Scope:**

1. The application does not support interaction with barcode scanners and card readers required for credit or debit cards while billing items.
2. The application does not provide the required analytics to help decide the special deals and promotions to be put on items
3. The application does not support multiple currencies or any kind of international transactions. All transactions are in US Dollars.
4. Departments that a new store can have must be selected from a set pre defined values by the parent store
5. The application is limited in handling store credit cards. The application does not validate whether or not a customer is eligible to be granted a store credit card. Everyone who applies for a store credit card gets it.
6. Currently promotions or clearances or any kind discounts offered need to be in the form of percentage discounts. Other kinds of special deals are not supported.

**Future Enhancements:**

1. Adding support for hardware such as card readers and bar code scanners.
2. Adding support for data analytics to help make managerial decisions such as discounts on items and identifying items that need to be put special promotions.
3. Adding support to multiple currency transactions.
4. Expanding the application's support to store credit cards by help deciding the customers eligible for allocating store credit.
5. Adding support to different types of store credit cards like platinum, gold and silver store credit cards based on the frequency and amount of purchases.
6. Adding support to buy one get one offers apart from discounts on price.

# Non-Functional Requirements

## Security Requirements

### **Authentication requirements**

Department Head, Store Head, POS Executive, Service Desk Operator should be authenticated before they can access the application.

**2.1.2 Authorization requirements**

The role and therefore the functions that a user can access must be determined and restricted to those functions only once authenticated.

## Product Requirements

**2.2.1 Portability requirements**

The application should be able to run on Windows, Linux and Mac OS.

## Organizational Requirements

**2.3.1 Delivery Requirements**

All the functionalities must be completely incorporated before 12/03/2014

**2.3.2 Implementation Requirements**

The server technology architecture required by the application consists of:

- JDK 1.7

- MySQL Community Edition 5.6

- The server machine must have at least 4GB RAM dedicated to the application

- The server machine must have at least 50GB space on Hard disk

The client machines require:

- Web Browsers - Mozilla Firefox, Google Chrome or Internet Explorer

**2.3.4 Standards requirements**

The application code must use intuitive names, camel case convention for variables including comments.

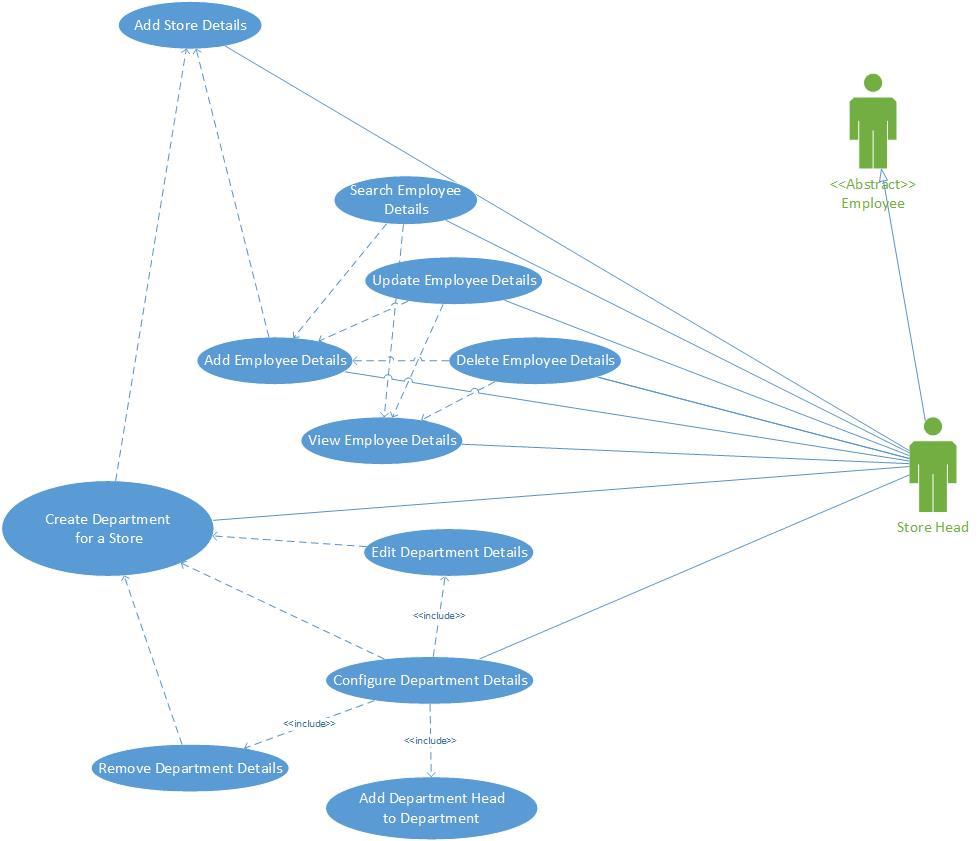
# Functional Requirements

**Actor Description**:

1. **Employee(abstract):** All the actors in the system are a specialization of the Employee actor. Employees actoris any actor employed in a store.
2. **Store Head:** Store Head is a specialized Employee who sets up a new store providing the details of the Department such as Department Name and Department Head. He maintains the records of the Employees of the new store. He can apply promotions/Clearances on the item and apply discount percentages to the promotional items.
3. **Department Head:** Department head is a specialized Employee whose actions affect a department in a store.He is the one who adds, deletes and updates all the items in a particular department.
4. **Service Desk Operator:** He is a specialized employee who checks the points present in customer credit card and provides certain discount on the purchase made by redeeming the points availed, if applicable.
5. **Point of Sales Executive:** He is the person who adds/remove the items for billing and he also issues store credit card to the customers upon the request existing in the system and he also generates receipt.

## Creation of New Store:

**Architecture View:**



**Use case Description:**

* **Use Case Name:** Creation of a new store
* **Participating actors:** Head of the store
* **Objectives of the use case:**

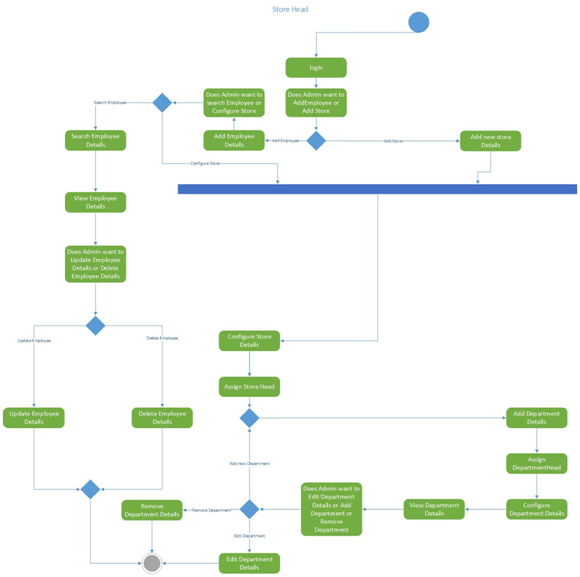
Objective of this use case is to create a new store in which Store Head adds departments to the new store and manages the details of the employees of the new store. He configures details of the department which includes assigning a head to each department. Also configuring Department Details includes editing department and removing Department from the store.

* **Pre-conditions:** Store head should be able login to the application by authenticating his/her credentials
* **Post-conditions:** There should be a new store configured with the department and employee details.
* **Primary Flow of events:**

Store head logs in into the application. He adds Employee Details. After adding the employee details, he can search for the Employee Details to either update or delete the details of the Employee. He also adds the New Store Details. After the store is created and employee details are added, she/he adds Department Name and assigns Department Head to the new store. After Adding Department details, he/she can configure the department details. While configuring the Department details, he/she can either add new Department to the new store and assign department head or edit or remove the already added department details from the store.

* **Formal model:**

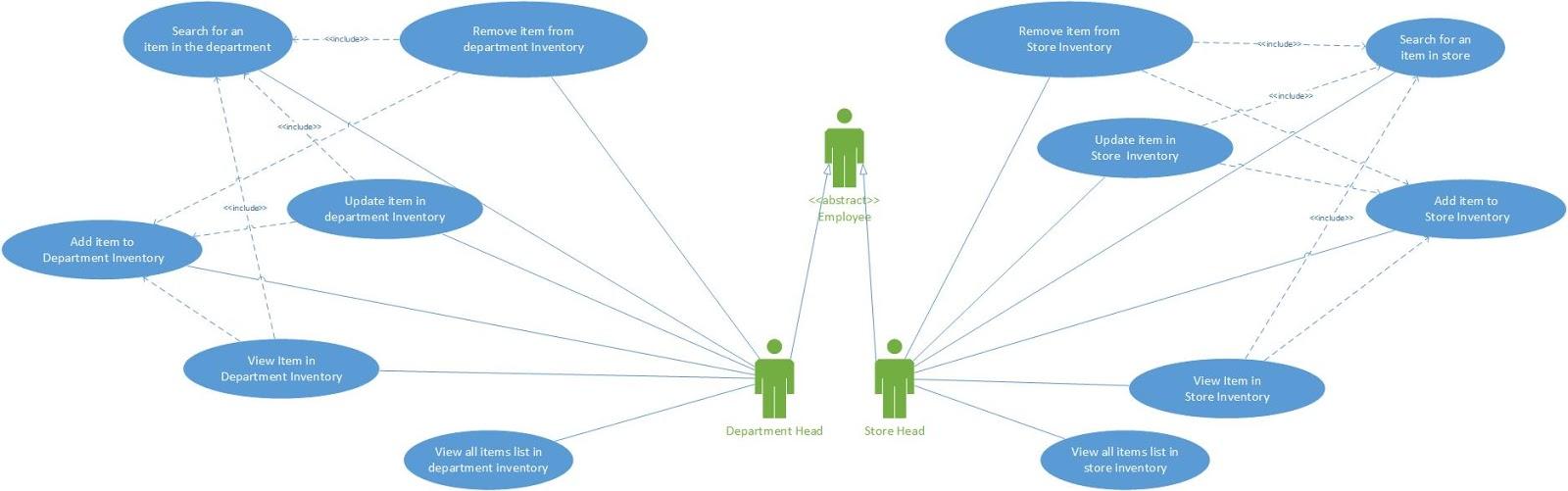
**Activity Diagram**



**3.2. Inventory Management:**

**Inventory Management Use case model:**

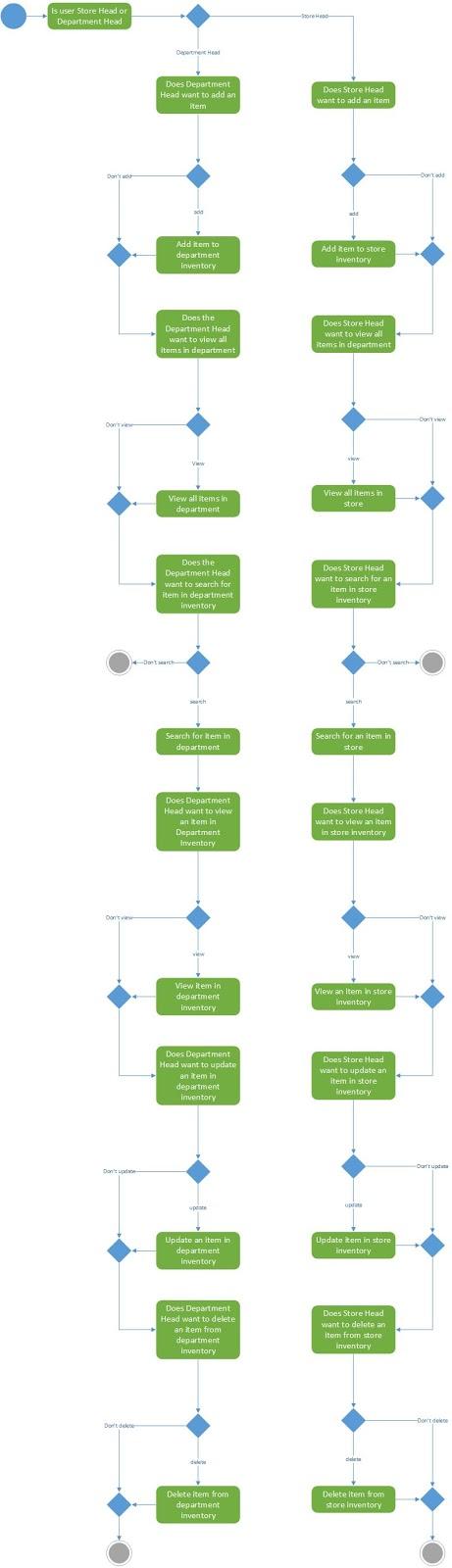
**Use case diagram:**



**Use case Description:**

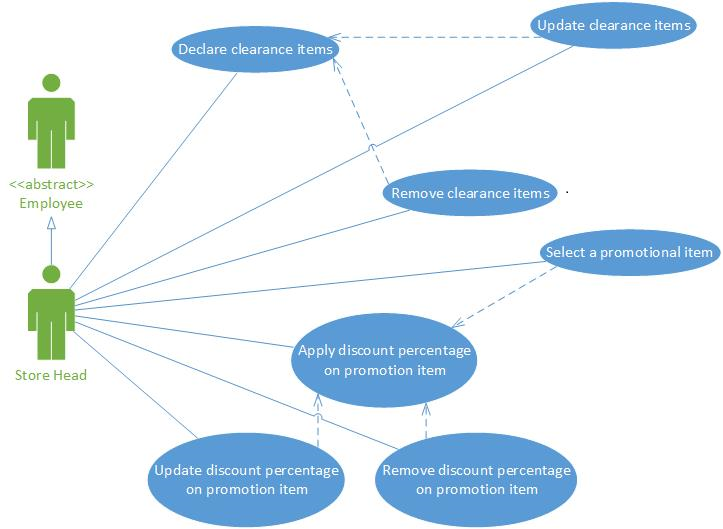
1. **Use case name:** Inventory Management
2. **Participating Actors:** Store Head, Department Head both of which are specializations of Employee
3. **Goals of the use case:** Store head must be able to manipulate items in store inventory. Department Head should be able to manipulate items in store inventory that belong to his/ her department
4. **Pre-Conditions and Post-Conditions:**
   1. **Pre-Conditions:** The Store Head must be logged in to exercise any of his use cases. The Department Head must be logged in to exercise any of his use cases.
   2. **Post-Conditions:** Changes are made to store inventory based on the actions of the Store head or Department Head
5. **Primary and Alternative Flow of Events:** The Store Head can add an item to the store inventory. The Store Head can search an item in the store inventory. The Store Head can search and view an item in the store inventory. The Store Head can search and update an item in the store inventory. The Store Head can search and remove an item in the store inventory. The Store Head can view a list of all items in the inventory. The Department Head can add an item to his department inventory. The Department Head can search an item in his department. The Department Head can search and view an item in his department. The Department Head can search and update an item in his department. The Department Head can search and remove an item in his department. The Department Head can view a list all items in his department.
6. **Formal Model:**

**Activity Diagram:**



**3.3**. **Promotions and Clearance**:

**Architecture View:**



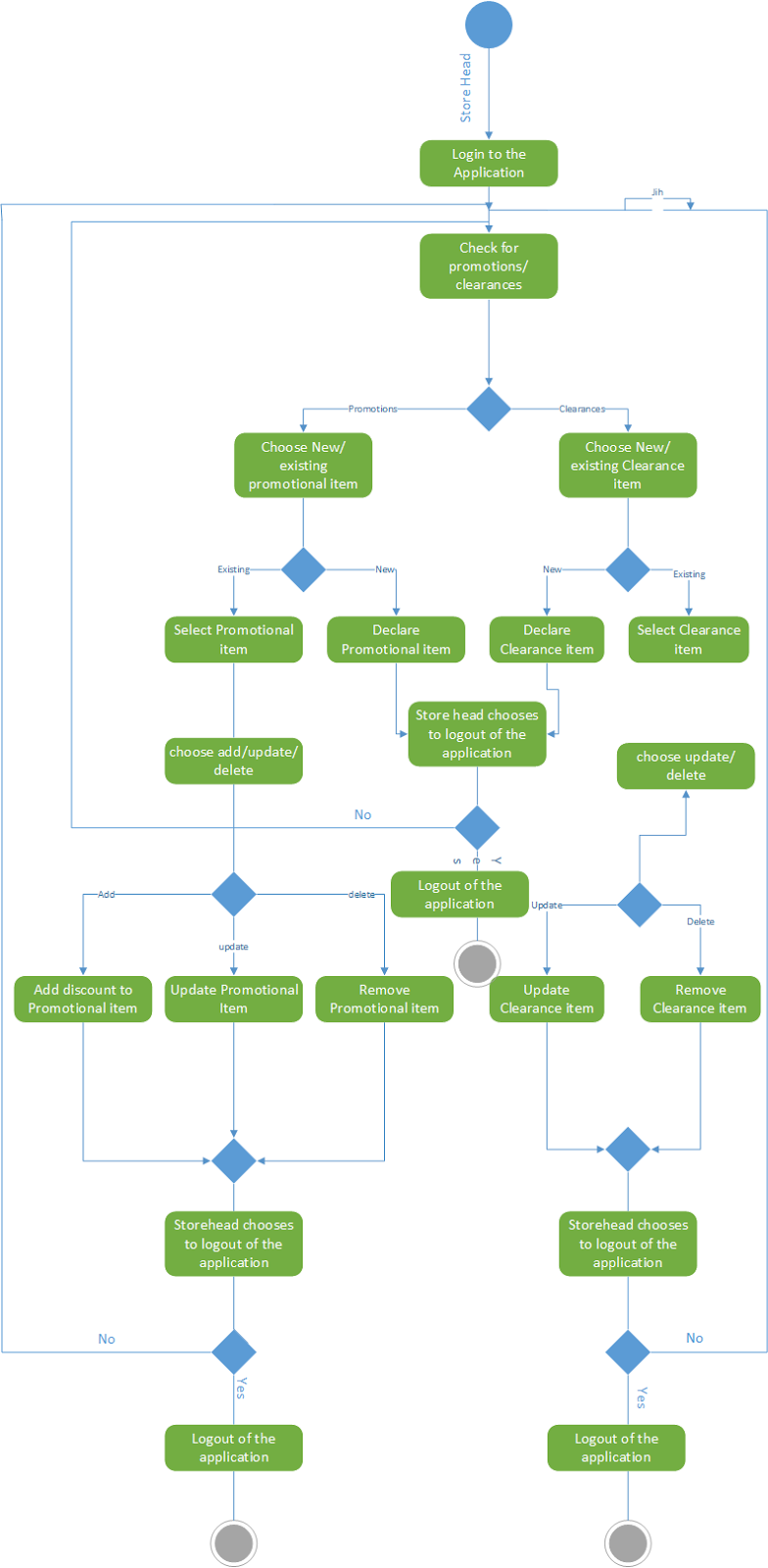
**Use Case description:**

1. **Use Case Name:** Promotion and Clearances.
2. **Participating actors:** Store Head .
3. **Objective of the use case:** The main objective of this use case is to provide store head to apply discount percentage on promotional items and also to declare clearance items.
4. **Pre-conditions:** The store head should be able to login to the application
5. **Post-conditions:** Store head should be able to successfully update discount percentages and clearance details on items.
6. **Primary & Alternative Flow of events:**

Store head logins to the New Retail Store creation and maintenance application and chooses to add items under promotions or clearances. After selecting promotions or clearances store head can declare new promotions and clearances or modify the existing items that have already been declared. Store head can declare new promotional or clearance item after logging in to the application.If he selects an already existing promotional item he can add discount to the item or update or remove the promotional item. If he selects an already existing Clearance item he can update or remove the clearance item.

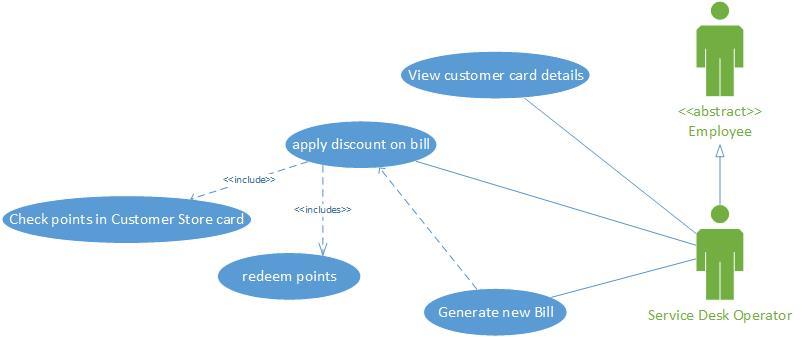
1. **Formal model:**

**Activity Diagram**



## Customer Point System:

**Architecture View:**



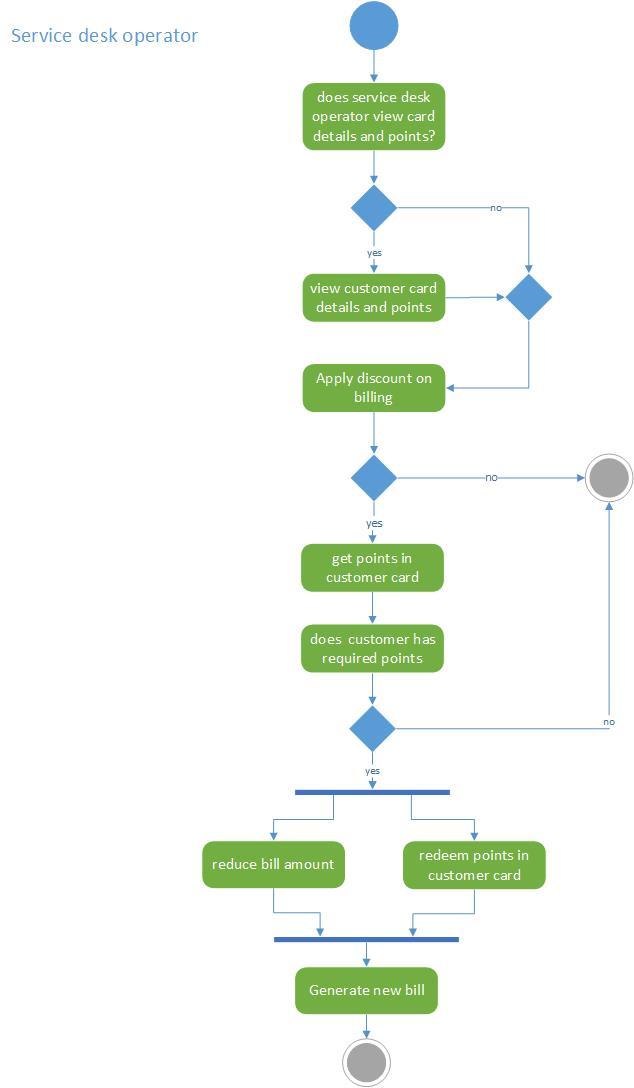
**Use Case description:**

1. **Use Case Name:** Customer Point System.
2. **Participating actors:** Service desk operator.
3. **Objective of the use case:** The objective of this use case is to make frequent customers avail the Customer points system where the service desk operator checks the points present in customer’s store credit card, provide discount on the purchase of the customer if eligible, and generates a new bill receipt.
4. **Pre-conditions:** Customer requesting a store credit card and receiving a card from POS executive and purchasing using the store credit card. Also, service desk operator should be logged into the application.
5. **Post-conditions:** Customer should obtain discount on the billing if satisfactory points are present and points to be redeemed accordingly.
6. **Primary & Alternative Flow of events:**

When a customer visits the service desk, the service desk operator views the customer card details and the points present in the card on the request of customer. If the customer does not request for viewing points, The service desk operator moves to applying discount. If the customer does not request for discount on the billing, the process terminates. If the customer requests for discount, he checks for the points present in the card. Then, If the customer does not have sufficient points to avail discount, the process terminates. If the customer has required points, in parallel the service desk operator reduces the bill amount and points are redeemed in the customer card. After the reduction of both bill amount and the points, he generates a new bill. Then the function of service desk operator terminates.

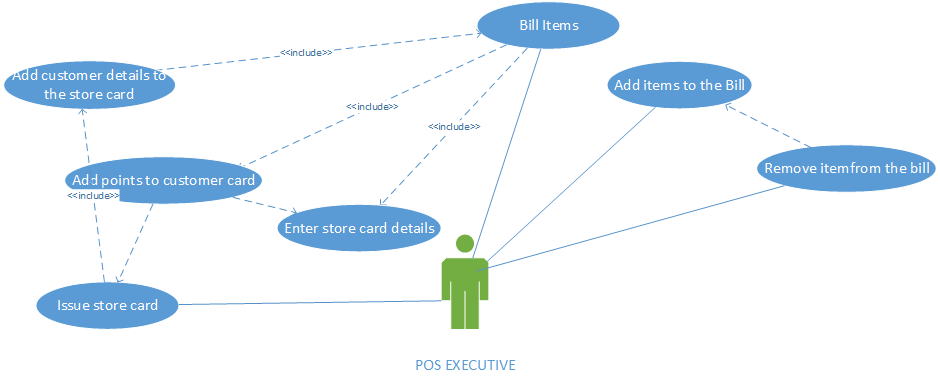
1. **Formal model:**

**Activity Diagram**



* 1. **POINT OF SALES**

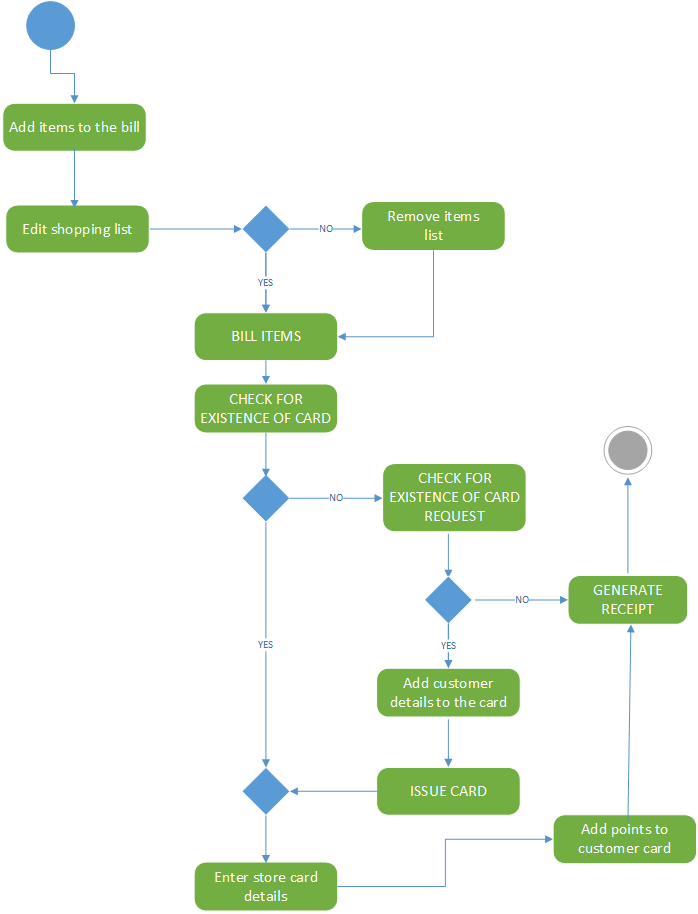
**Architecture View:**



**Use Case description:**

1. **Use Case Name:** Point of Sales.
2. **Participating actors:** Point of Sales Executive
3. **Objective of the use case:** The primary objective of this use case is to enable the POS executive to Bill the items,as a part of it he can add items and he can also remove items for billing and he also adds points to the customer credit cards,if the customer holds the store’s credit card.If the customer do not have any card and he would make decision based upon the existence of request for card in the system,pos executive would issue card and then the points would be added to the card based on the billing amount and generate the reciept.In case, if there is no request for card he would directly generate the reciept
4. **Pre-conditions:** POS Executive should have access to the application.
5. **Post-conditions:** POS Executive should successfully add item and remove items for billing.The store credit card should be issued based on the existence of the request in the system.Post billing, the points will be added to the customer card based on the billing amount.Finally the recieptshould be generated.
6. **Primary & Alternative Flow of events:**POS executive is the person who bills the items.In order to bill the items,he adds the items to the bill,he can edit the shopping list to remove items for billing.After billing those items,he checks for the existence of store credit card.If card exists,then he card details to the system and the points are added to the card and generates the receipt. lf there is no existence of card,he then checks for the existence of credit card request.If the request exists,he then adds customer details to the card and issues the card.After which he enters the card details in the system the points are added to the card and generates the receipt,if there is no card request,he directly generates the receipt.
7. **Formal model:**

**Activity Diagram**

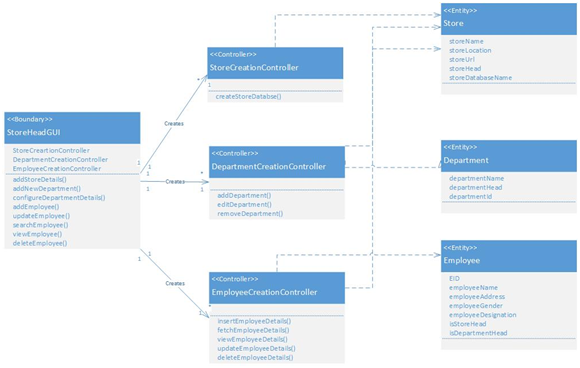


# Requirements Analysis

## Structural Analysis

Present an overall analysis class diagram and briefly describe each analysis class (organized according to type - boundary, control, and entity)

### **Creation of New Store**



#### Analysis Class Diagram

**Description:**

StoreHeadGUI analysis Bounday class contains StoreCreationController, DepartmentCreationController, EmployeeCreationController as the attributes and addStoreDetails(), addNewDepartment(), configureDepartmentDetails(), addEmployee(), updateEmployee(), searchEmployee(), viewEmployee(), deleteEmployee() as the methods.

The StoreHeadGUI analysis class provides an interface for the Store Head to create a new store providing the details of the store. It also gives access to the Store head to add, search, view and update the employee details for the new store. Access for the Creation of the new Departments in the newly created store is provided by the StoreHeadGUI boundary class.

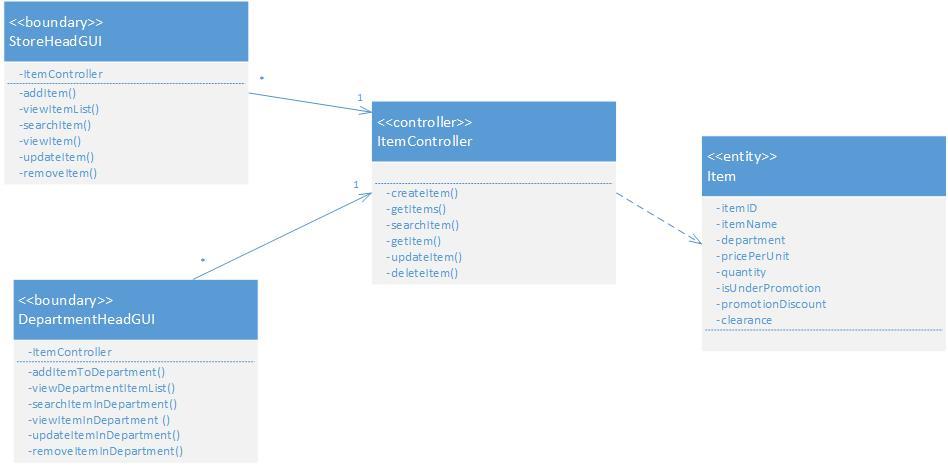
In the StoreCreationController class, createStoreDatabase() method takes Store name, Store Head, Store Location and Store url from StoreHeadGUI Boundary class and creates a new Database for the New Store.

In the DepartmentCreationController class, addDepartment() takes Department Name and Department ID details from the StoreHeadGUI and inserts the Department Details into the Department Table of the newly created Store Database.Simultaneously, Department Head will be mapped to the specific Department. editDepartment()will edit the details of the already added department and updates the department details in the Department table. removeDepartment() will delete the department details from the database.

In the EmployeeCreationController class, insertEmployeeDetails() method takes Employee Name, EID, Employee Address, Employee Gender, Employee Designation, is a Store Head, is a Department Head details from the StoreHeadGUI Boundary class and inserts all these details into the Employee Table. The fetchEmployeeDetails() method takes Employee ID from StoreHeadGUI Boundary class and fetches the information from Employee Table and viewEmployeeDetails() method displays the fetched employee Information on the Store Head GUI. updateEmployeeDetails() method takes Employee Details from StoreHeadGUI Boundary class and updates employee information in the Employee table based on the Employee ID. deleteEmployeeDetails() method takes Employee ID from StoreHeadGUI Boundary class and deletes employee information in the Employee table based on the Employee ID.

### **Inventory Management**

**Inventory Management Analysis Class Diagram:**



1. **Boundary classes:**

**StoreHeadGUI:** The StoreHeadGUI models the GUI interface for Store Head. It has the methods addItem() to add an item to store inventory, viewItemList() to view a list of all items in store inventory, searchItem() to search an item in store inventory, viewItem() to view an item in store inventory, updateItem() to update an item in store inventory, removeItem() to remove an item from store inventory(). StoreHeadGUI has reference to the ItemController.

**DepartmentHeadGUI:** The DepartmentHeadGUI models the GUI interface for Department Head. It has the methods addItemToDepartment() to add an item to department inventory, viewDepartmentItemList() to view a list of all items in department inventory, searchItemInDepartment() to search an item in department inventory, viewItemInDepartment() to view an item in department inventory, updateItemInDepartment() to update an item in department inventory, removeItem() to remove an item from department inventory(). DepartmentHeadGUI has reference to the ItemController.

1. **Controller class:**

**ItemController:** The ItemController has the methods createItem() to create a Item,

getItem() to get an item, getItems() to get a list of items(), updateItem() to update an

Item and deleteItem() to delete an item. The ItemController depends on the Item entity.

1. **Entity class:**

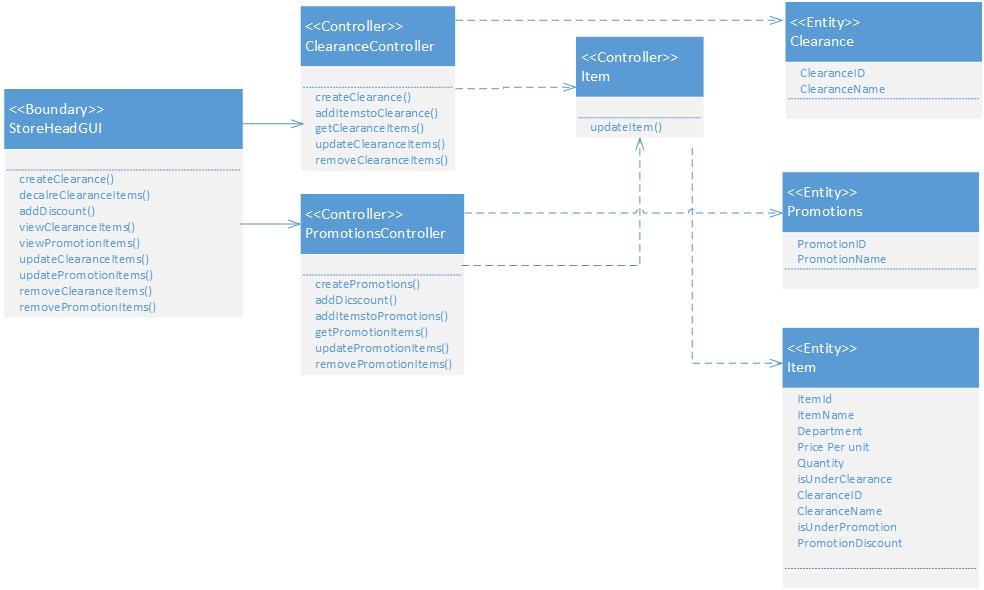
**Item:** Item entity class model the real world Item of a store. It has the attributes itemID

that uniquely identifies an item in store inventory, item name which is the name of the

item and department that species the department the item is classified into.

### **Promotions and Clearances**

#### Analysis Class Diagram



**Description:**

Boundary classes:

StoreHeadGUI: Store Head interactions with UI causes the methods of the Store head GUI to be invoked. createClearance() is used to create and name a Clearance. Other methods related to clearances are declareClearanceItems(), viewClearanceItems(). The class also has methods to manipulate promotions namely,viewPromotionItems(), updatePromotionItems(), removePromotionOnItem() and to addDiscount()

Controller classes:

Clearance Controller: All actions on a Clearance entity are performed through ClearanceController. It has the methods createClearnce(), addItemsToClearance(), getClearanceItems(), updateClearanceItems(), and removeClearanceItems().

Item Controller:

The ClearanceController and PromotionController depends ItemController whenever changes are required to items when a clearance or promotion is added/ updated.

Promotion Controller:

The promotion controller has the methods addPromotion(), removePromotion(), updatePromotion(), viewPromotion(), getPromotion() and addDiscount() to update item to reflect promotions accordingly

Entities:

Clearance:

The clearance entity models a Clearance. It has the attributes clearanceID and clearanceName

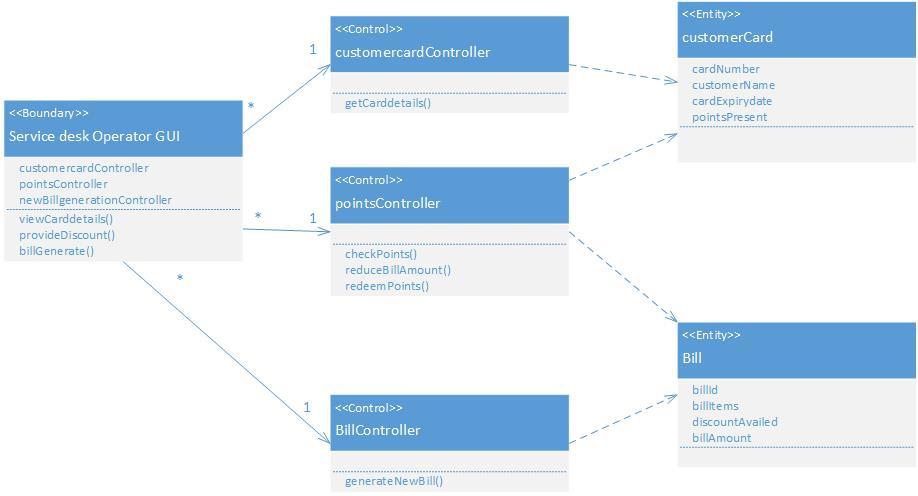
Promotions:

The Promotions entity models a Promotions. It has the attributes promotionID and promotionName

Item:

The item entity models a store item. It has the attributes itemID, itemName, isUnderClearance, isUnderPromotion, clearanceDiscount, promotionDiscount.

### **Customer Point System**



**Description:**

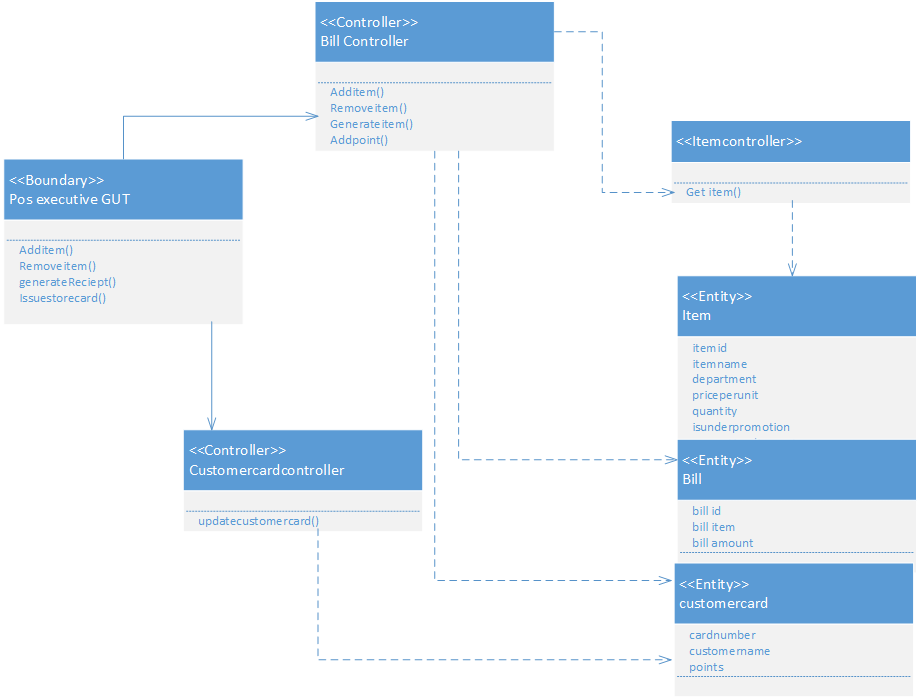
**Boundary Classes:** Service desk operator GUI boundary class has the customercardController, pointsController and BillController as attributes and viewCarddetails(), provideDiscount() and billGenerate() as methods. This boundary class provide an interface to the service desk operator to manage the customer points by redeeming them whenever eligible and requested.

**Controller Classes:** CustomercardController class contains viewCarddetails() as method where the details of the customer such as name, card number can be viewed. PointsController class has checkPoints() method where the points present in the customer card can be checked. reduceBillAmount() method uses the bill entity class and provides discount to the customer bill. redeemPoints() method uses the card entitty class and reduces the points in the card. BillController class has generateBill() method which makes the new bill to be generated that displays the billId, billItems and billAmount.

**Entity classes:** The entity class customerCard models a customer store card and stores the name of the customer, card number, card expiry date and the points present in the card. Bill entity models a bill generated for the purchase and stores the bill id, bill items, bill amount and the discount availed.

### **Point of Sales**

#### Analysis Class Diagram



**Description:**

**Boundary Classes:**

POSExecutiveGUI: POS executive interactions with GUI invoke methods in POSExecutiveGUI. The methods addBillItems(), removeBillItems(), generateReceipt() are used to change and generate bills when a customer wants to check out his items. issueStoreCard() method is invoked when POSExecutive issues a store card to a customer.

**Controller classes:**

ItemController: The item controller is used to manipulate Item entity. It is used by BillContoller to get price of an item by calling the method getPrice()

BillController: The Bill controller is used to manipulate a Bill. It has the methods addItem(), removeItem() and generateReceipt().

CustomerCardContorller: The customer card controller manipulates objects of CustomerCard. It has the method issueCard() to issue a card.

**Entities:**

Item: Item model an item in a store.

Bill: Bill models a bill generated for the items purchased.

CustomerCard: The CustomerCard entity models a customer store card.

## Behavioral Analysis

This section presents a use-case realization – analysis for each use-case in the form of a communication diagram of analysis classes and a supporting narrative.

### **Creation of New Store**

**Communication Diagram:**

In the above communication diagram, the Store Head has the following activities based on the Creation of New Store Use case.

1.AddStoreDetails

2.CreateDepartment

3.configureDepartmentDetail

4.addEmployee

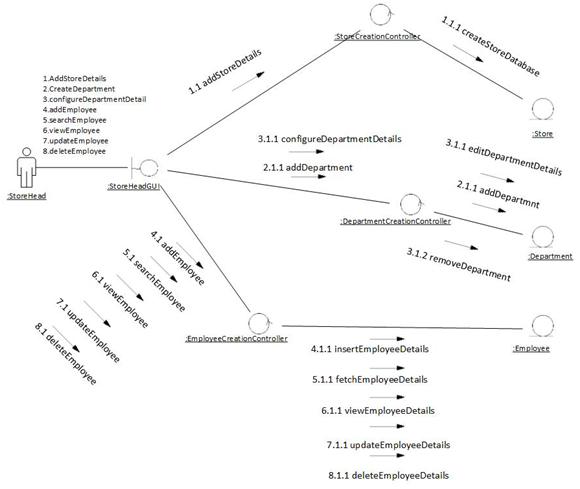
5.searchEmployee

6.viewEmployee

7.updateEmployee

8.deleteEmployee

The diagram clearly depicts how the analysis classes interact to describe the sequence of actions to be performed in order to create a new store, add departments to that new store and adding, updating and deleting of the employee information of the newly created store. Also configure department details method describes how a department head is added to the new department.



### **Inventory Management**

**Inventory Management Communication Diagram:**

**Communication Diagram with StoreHeadGUI:**



**Narrative:**

The Store Head can add an item to store inventory(to any department) by calling addItem() on StoreHeadGUI, which inturn calls createItem() on ItemController. The Store Head can view all items in store inventory by calling viewItemList() on StoreHeadGUI, which inturn calls getItems() on ItemController. The Store Head can search an item in store inventory by calling searchItem() on StoreHeadGUI, which inturn calls searchItem() on ItemController. The Store Head can view an item in store inventory by calling viewItem() on StoreHeadGUI, which inturn calls getItem() on ItemController. The Store Head can update an item in store inventory by calling updateItem() on StoreHeadGUI, which inturn calls updateItem() on ItemController. The Store Head can remove an item to store inventory by calling removeItem() on StoreHeadGUI, which inturn calls deleteItem() on ItemController.

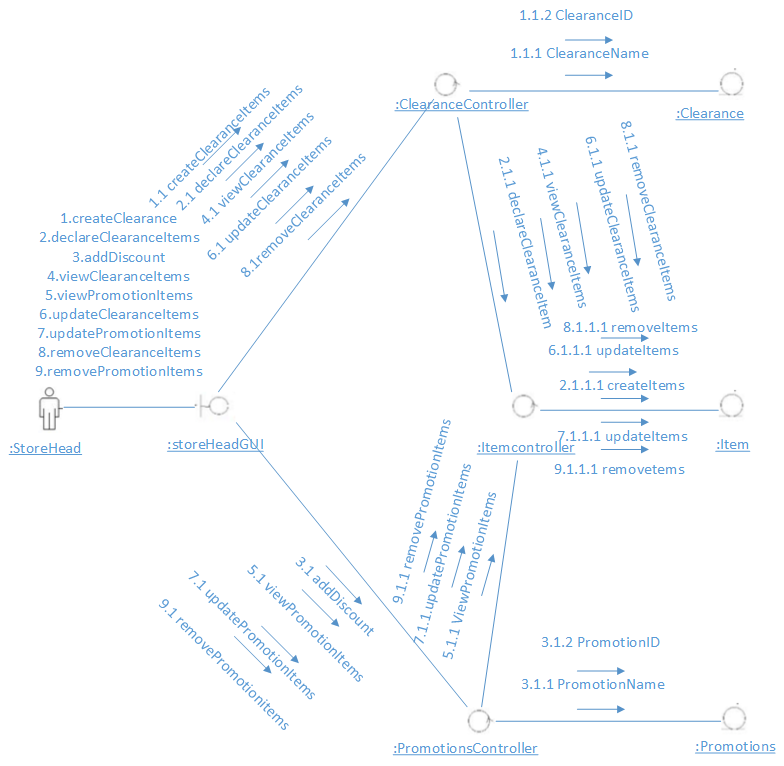
**Communication Diagram with DepartmentHeadGUI:**



**Narrative:**

The Department Head can add an item to his department inventory by calling addItemToDepartment() on DepartmentHeadGUI which inturn calls createItem() on ItemController. The Department Head can view all items in his department by calling viewDepartmentItemList() on DepartmentHeadGUI which inturn calls getItems() on ItemController(). The Department Head can search an item to his department inventory by calling searchItemInDepartment() on DepartmentHeadGUI which inturn calls searchItem() on ItemController(). The Department Head can view an item in his department by calling viewItemInDepartment() on DepartmentHeadGUI which inturn calls getItem() on ItemController(). The Department Head can update an item to his department inventory by calling updateItemInDepartment() on DepartmentHeadGUI which inturn calls updateItem() on by calling removeItemInDepartment() on DepartmentHeadGUI which inturn calls deleteItem() ItemController(). The Department Head can remove an item from his department inventory on ItemController().

### **Promotion and Clearances**



**Narrative:**

The store head can declare clearance and add items to clearance by calling addItemsToClearance() on promotionsandClearanceGUI. The store Head can view all items in PromotionsandClearances by calling viewPromotionItemList() and viewClearanceItemList()on promotionsandClearanceGUI which inturn calls getItems() on ItemController().The store Head can search an promotional or clearance item by calling searchPromotionItem() or searchClearanceItem() on promotionsandClearanceGUI which inturn calls searchItem() on ItemController(). The store Head can update an promotional or clearance item by calling updatePromotionItem() or updateClearanceItem() on promotionsandClearanceGUI which inturn calls updateItem().The store Head can remove an promotional or clearance item by calling removePromotionItem() or removeClearanceItem() on promotionsandClearanceGUI which inturn calls removeItem().

### **Customer Points System**

**Communication Diagram:**

Service desk operator:

### Comunication cps latest.jpg

As shown in the communication diagram, the service desk operator has the following activities.

1. view card details.

2.apply discount.

3. generate new bill.

Here the analysis classes interact to describe the sequence of actions to be performed in order to operate the customer points system. First, The service desk operator GUI invokes the customercardContoller to view card details which in turn invokes the customercard entity class. To apply discount, the GUI invokes the points controller where checkPoints, reduceBillAmount and redeemPoints methods are provided and invokes both the customerCard and bill entity classses. To generate new bill, a billController class is invoked which inturn invokes bill entity class to receive the bill details.

### **Point of Sales**

POS EXECUTIVE:

He does the following activities:

1)Add bill items

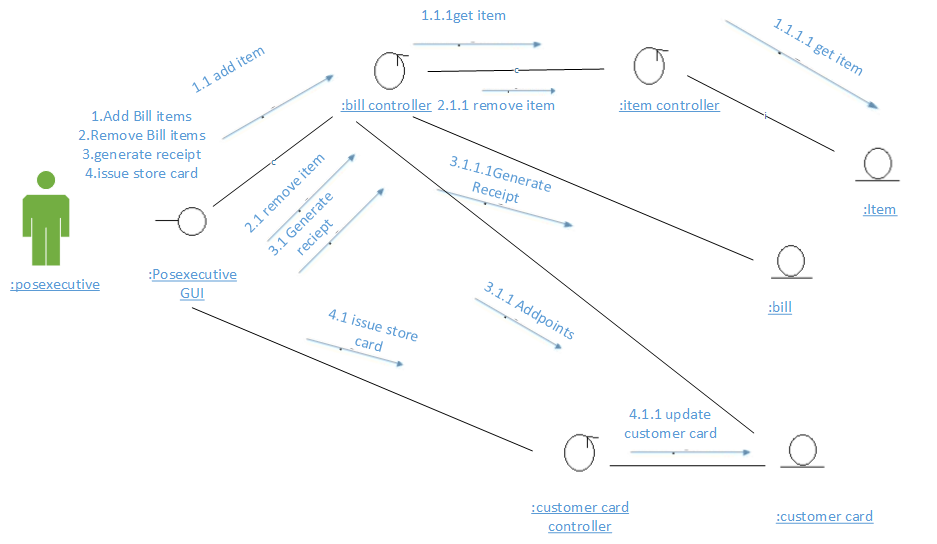
2)Remove bill items

3)Generate receipt

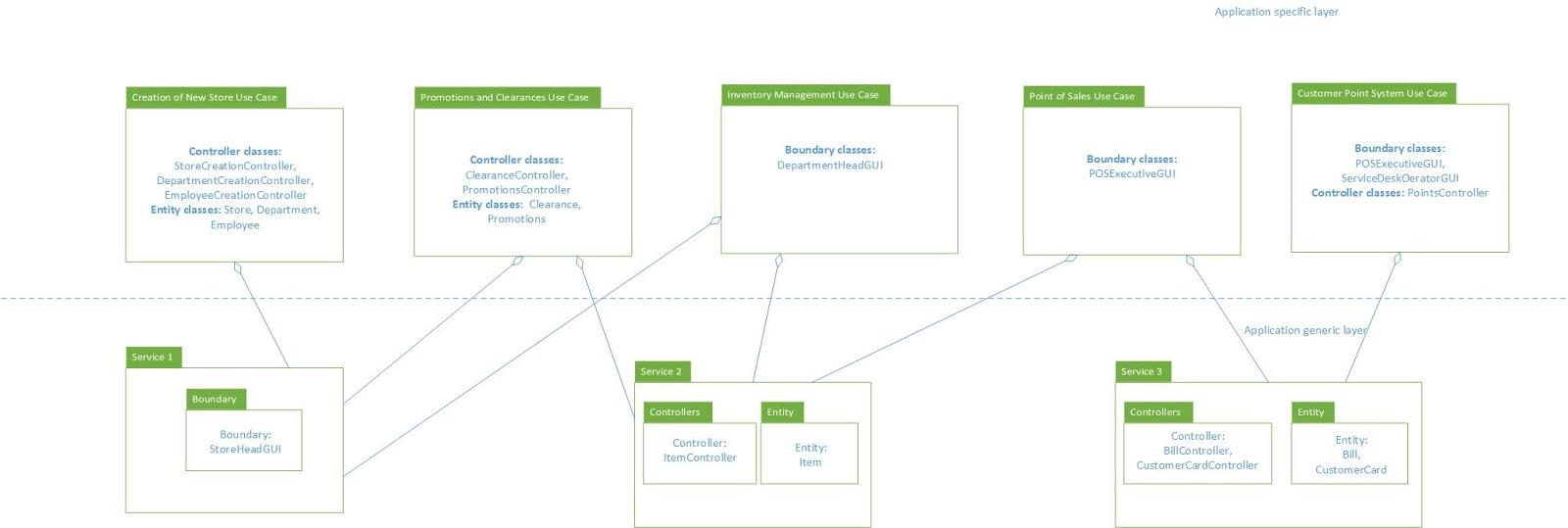
4)Issue store card

In order to add the items for billing,the pos executive GUI invokes bill controller which in turn invokes item controller which fetches item from entity class Item.Remove item can be done at the bill controller using remove item method.In order to generate receipt we need to invoke the Bill controller class

which in turn access the Bill entity class and this Bill controller class also access the customer card entity class for adding points to the cards.To issue store card, pos executive GUI invokes the customer card controller class which inturn invokes the customer card entity class for updating customer card details

.

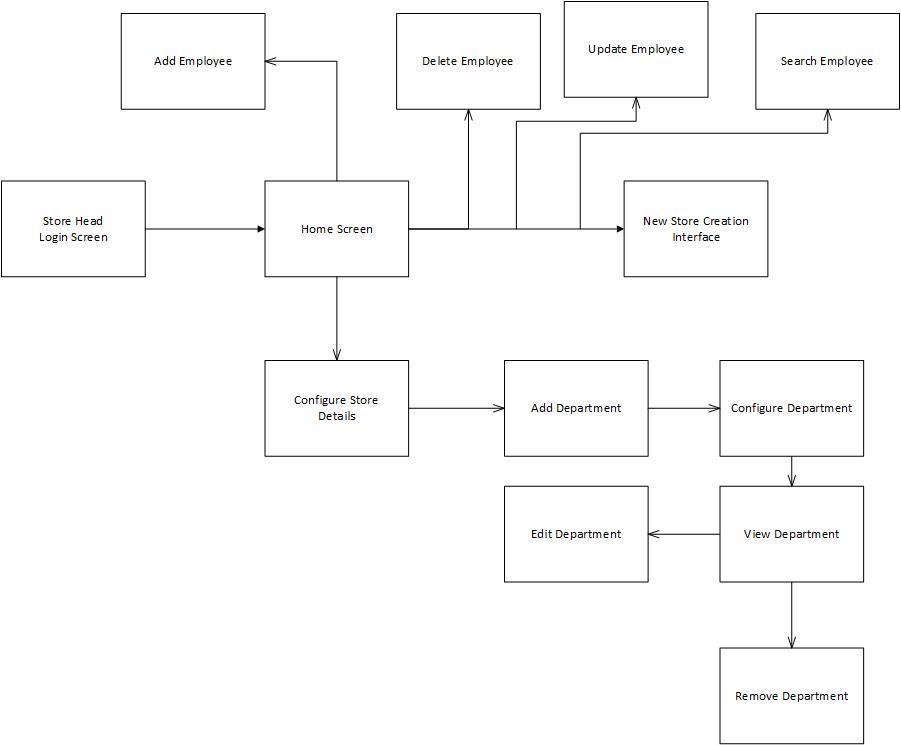
## Analysis Packages



# Use-Case Storyboards

## Use case story board for Store Head:

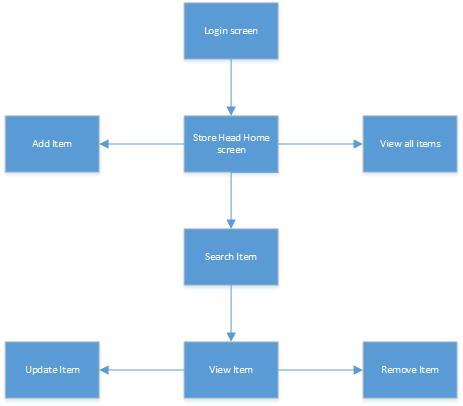
**5.1 Interface flow Diagram For Creation of new store:**



Store head will open the main interfaces and logins and he creates the store using the create a new store interface. As a sequence of steps, he adds the department’s details using the Add department Interface. He configures the department details using the Configure Department Interface and assigns department heads to departments. The store head uses search Employee Interface to search and view the employee information. He uses update Employee interface to update the employee information. He uses the delete employee interface to delete the employee.

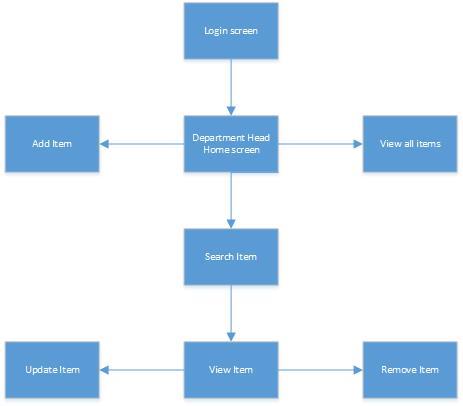
## Interface flow diagram for Inventory Management:

**Inventory Management Store Head Interface Diagram**



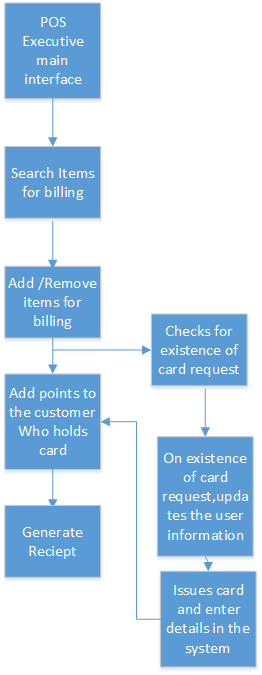
Store Head will login from the login screen which is a prerequisite for the Inventory Management use case model. The Store Head can then view his Home screen. From the Home screen the Store Head can navigate either to Add Item screen to add item to store inventory or View all items screen to view all items in store inventory. The Store Head can also navigate to Search Item screen to search for an item in store inventory. After searching for an item, the user can navigate to View Item screen. From View Item screen Store Head can update or remove item.

**Inventory Management Department Head Interface Diagram**



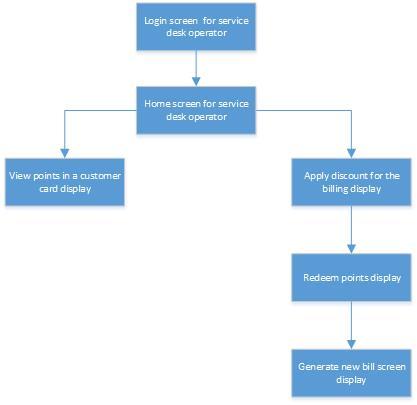
Department Head will login from the login screen which is a prerequisite for the Inventory Management use case model. The Department Head can then view his Home screen. From the Home screen the Department Head can navigate either to Add Item screen to add item to department inventory or View all items screen to view all items in store inventory. The Store Head can also navigate to Search Item screen to search for an item in store inventory. After searching for an item, the user can navigate to View Item screen. From View Item screen Store Head can update or remove item.

## Use case story board for POS Executive



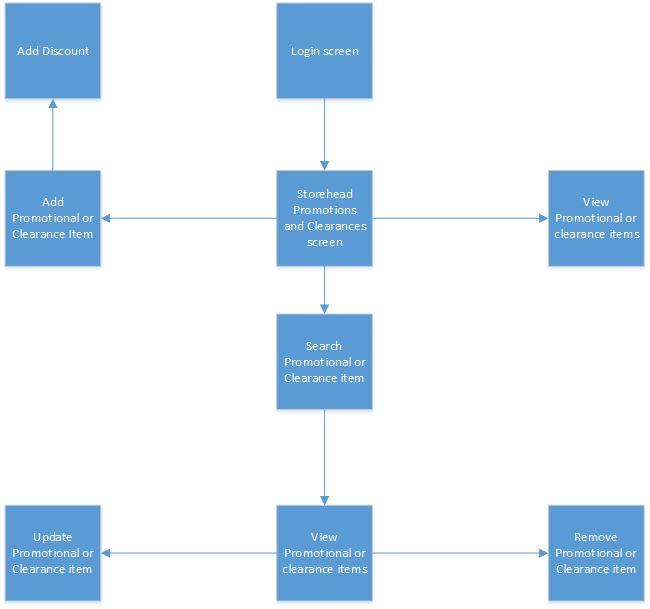
POS executive logins to his main interface and he then adds /removes items for billing .once the items are billed,points are added to the customer credit card based on the bill amount.He also checks for the existence of card request,If the request exist in the system,he adds customer details to the card and issues the card ,post which points could be added to the card and he finally generates the receipt

## Use case story board for Service desk operator



Service desk operator logins into his account and views the customer credit card details and the points present. He can apply discounts to the customer by checking the points available and redeeming them. He then generates a new receipt with the bill amount and points updated.’

**Use case story board for store head in Promotions and Clearances screen**



Store Head logins to Promotional and Clearance screen and add new Promotional or clearance Item or can view the already existing Promotional and Clearance items.If he add new Promotional and Clearance items, he can add discount to the items. He can search for already existing Promotional and Clearance item and update or remove the already existing Promotional or Clearance item

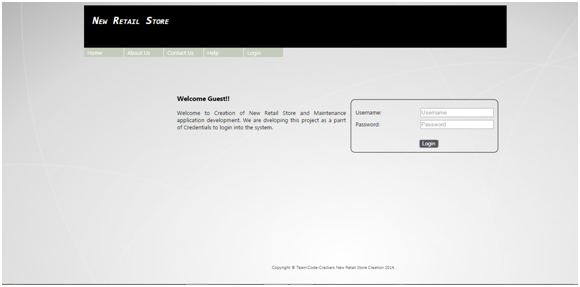
# 

# 6.Prototype User Interfaces

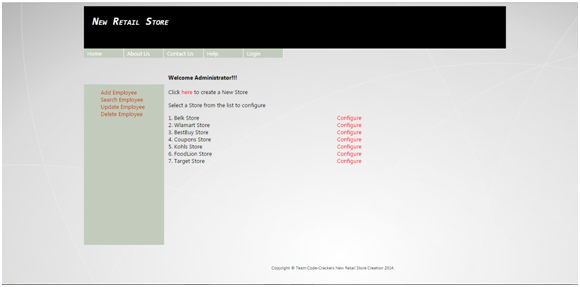
Present mockups of your primary user interfaces.

**Creation of new store:**

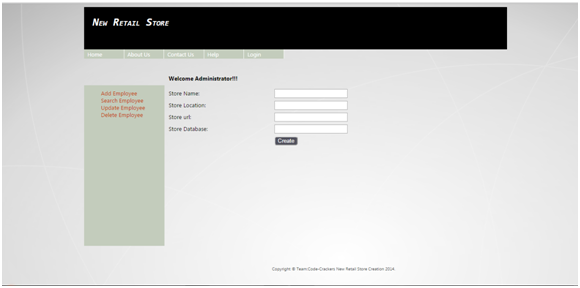
**Login Screen:**



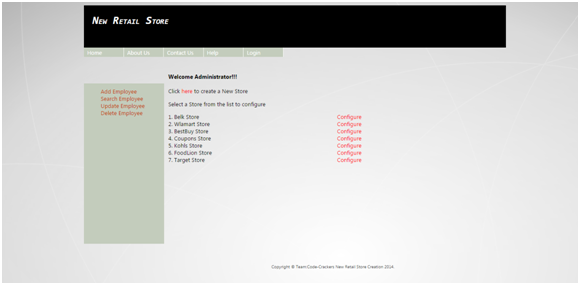
**Home Screen:**



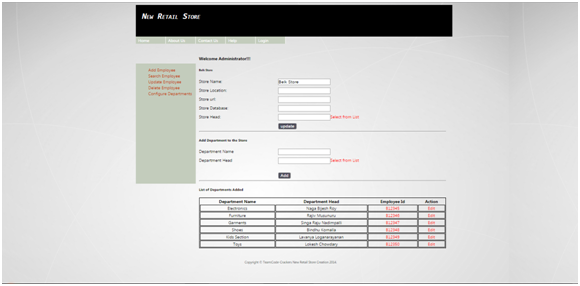
**Adding Store Details screen:**



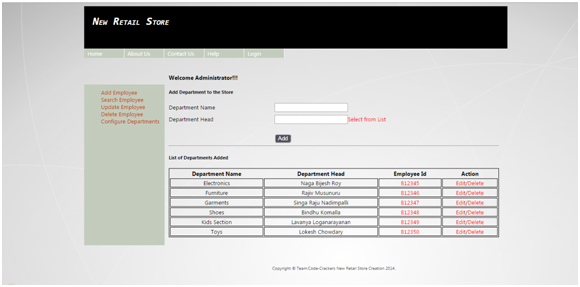
**Home Screen:**



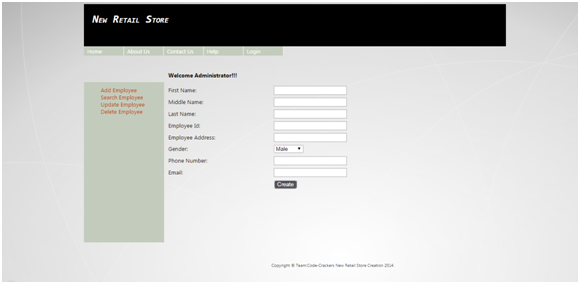
**Configure New Store Screen:**



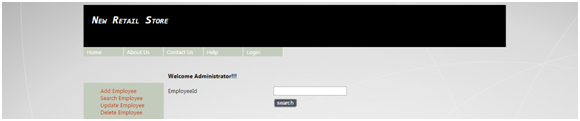
**Configure Department Screen:**



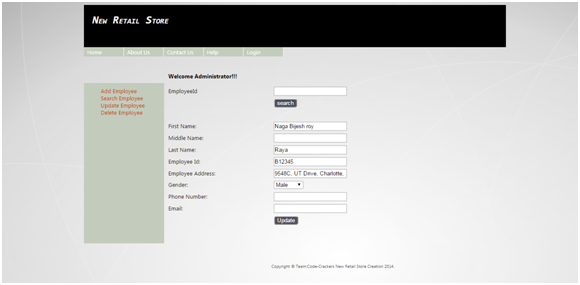
**Add Employee Screen:**



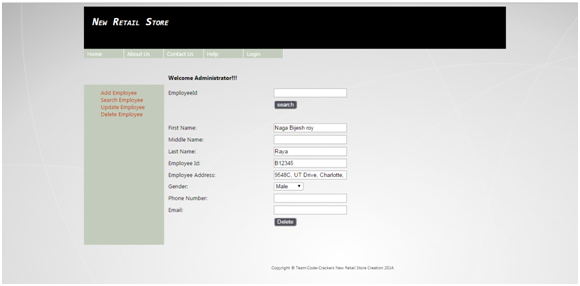
**Search Employee Screen:**



**Update Employee Screen:**



**Delete Employee Scren:**

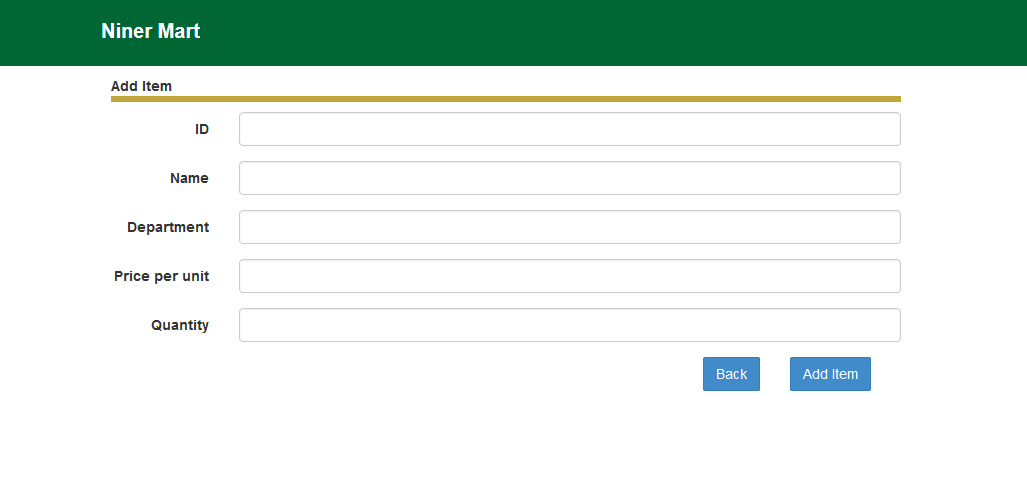


**Inventory Management:**

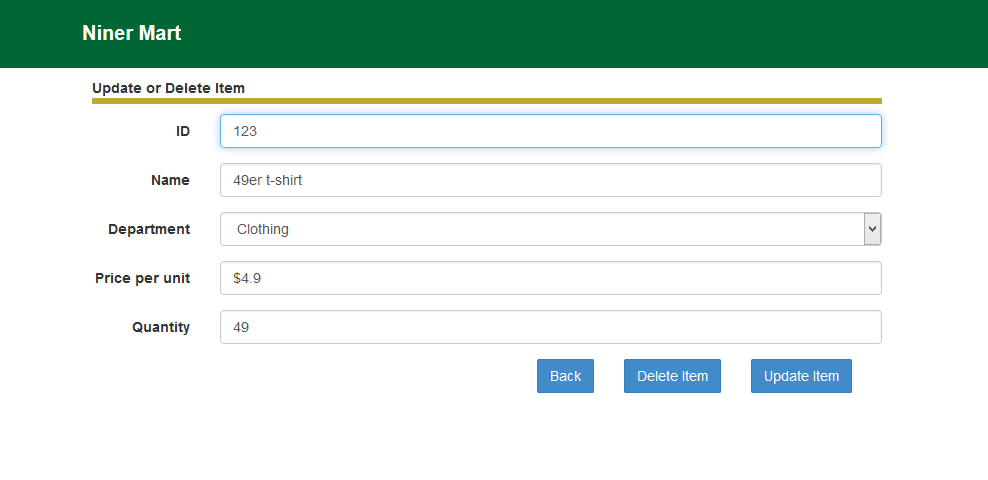
**Home screen:**



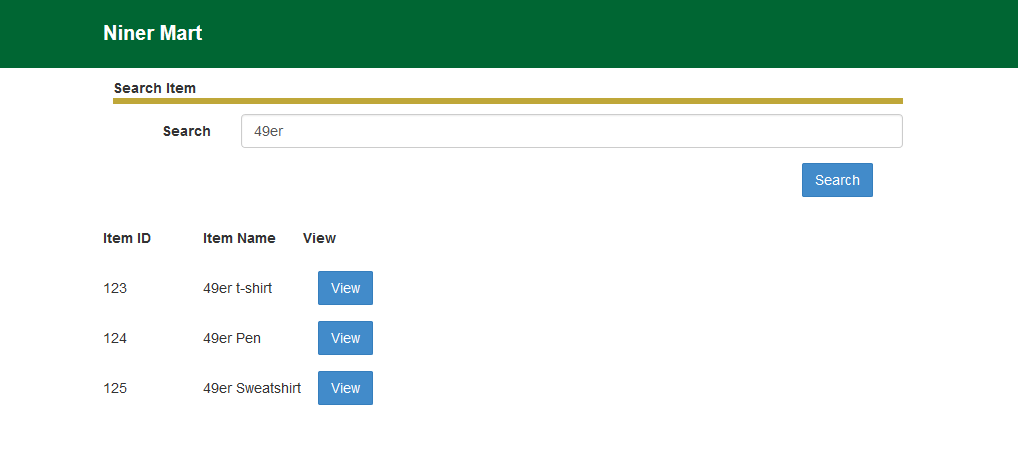
**Add Item prototype:**



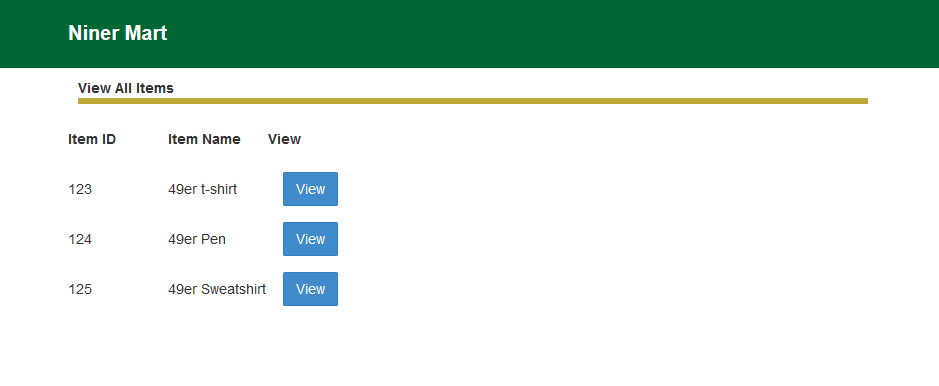
**View - Update - Delete item prototype**



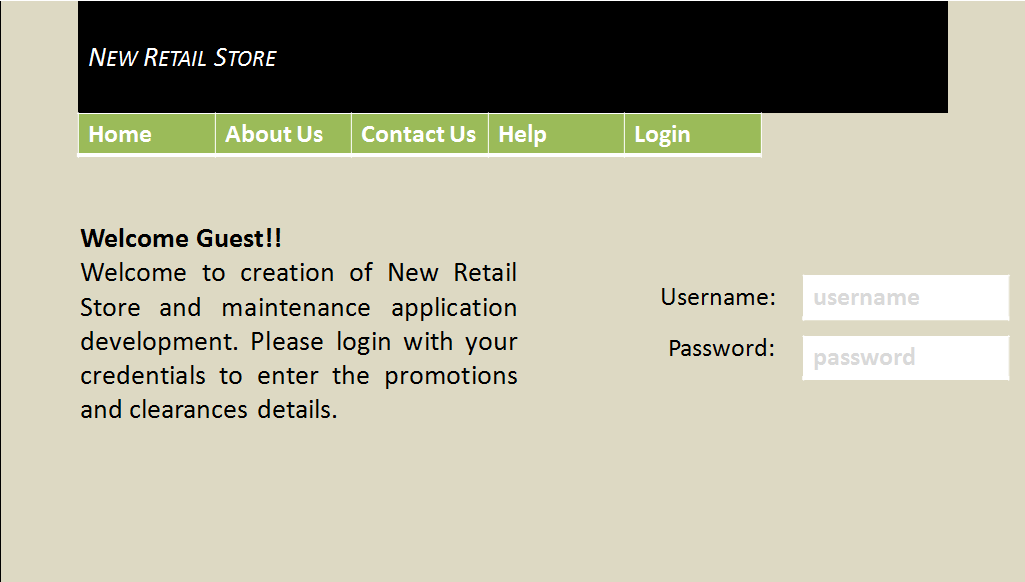
**Search item prototype:**

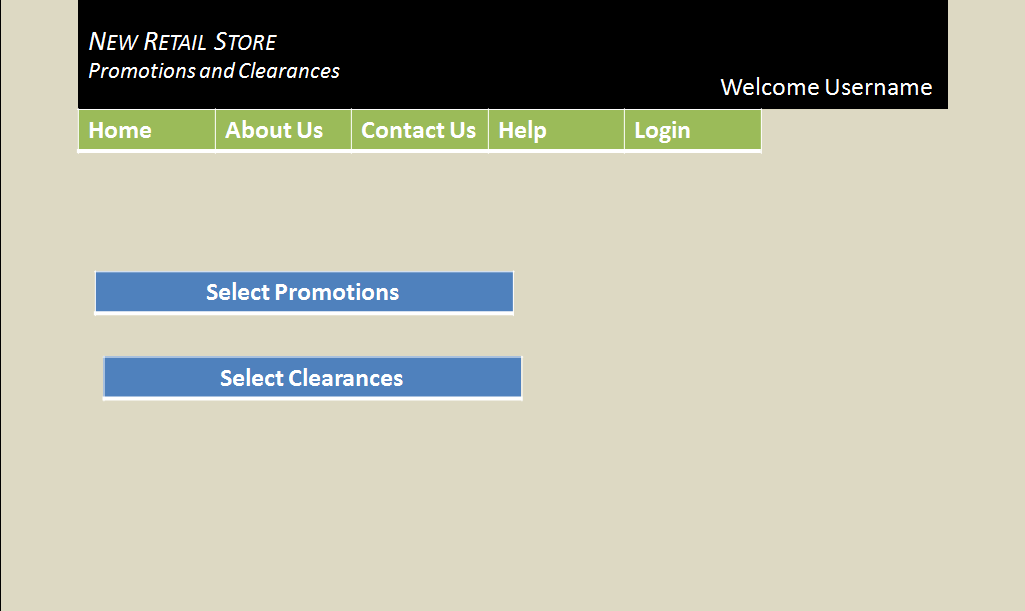


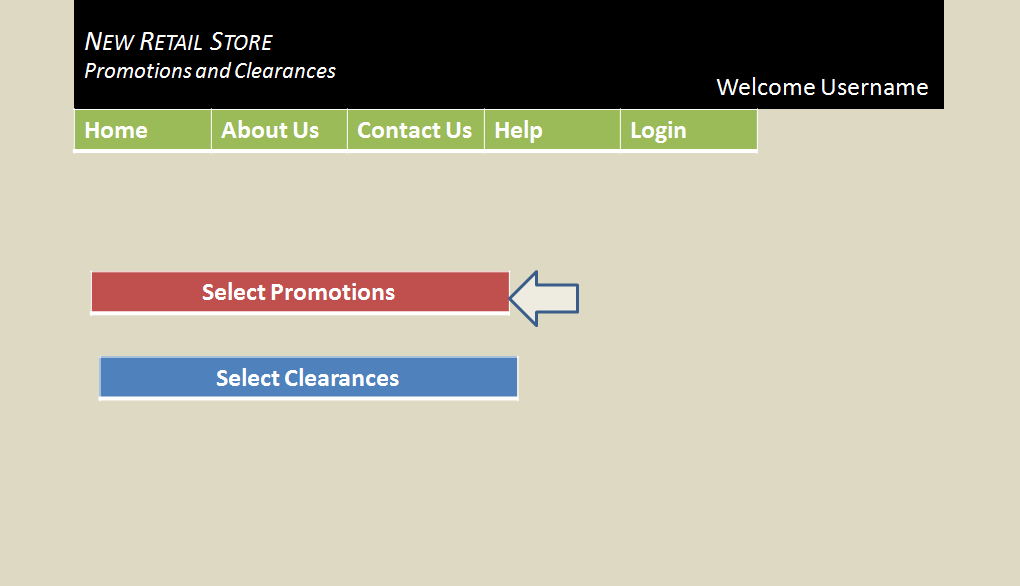
**View all items prototype:**

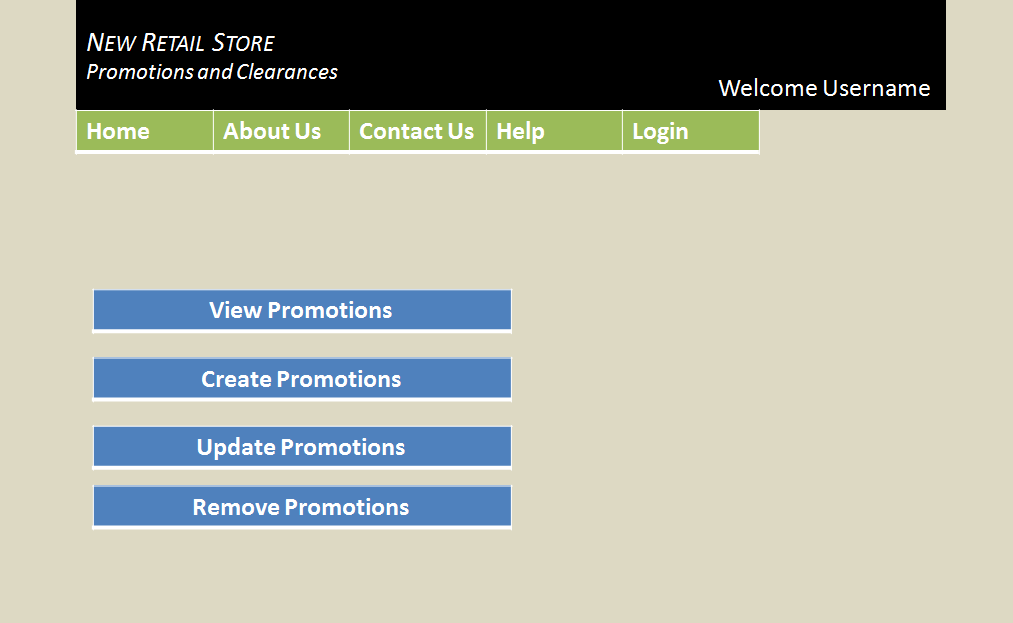


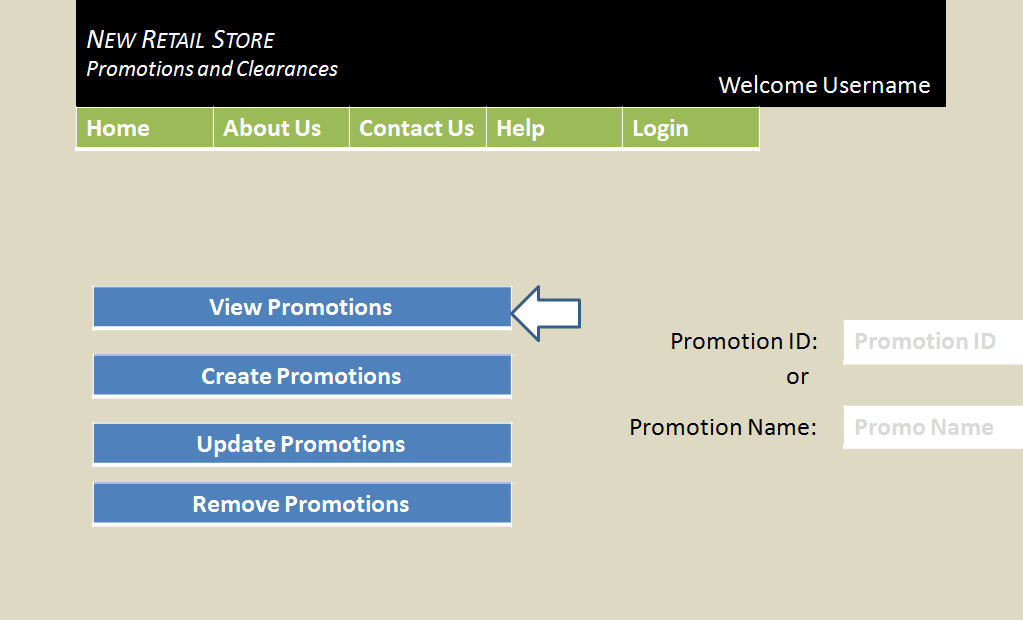
**Promotions and Clearances:**

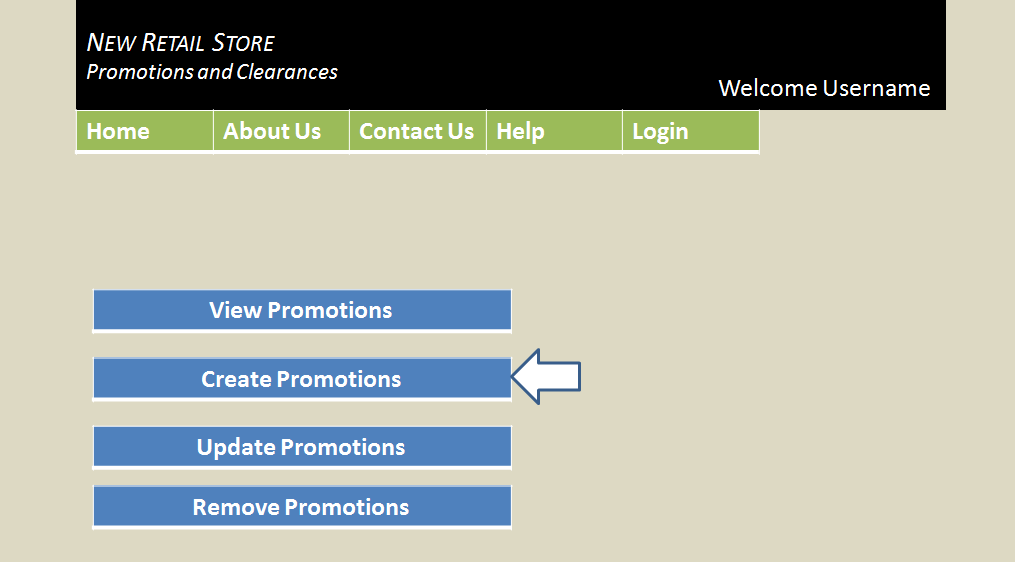


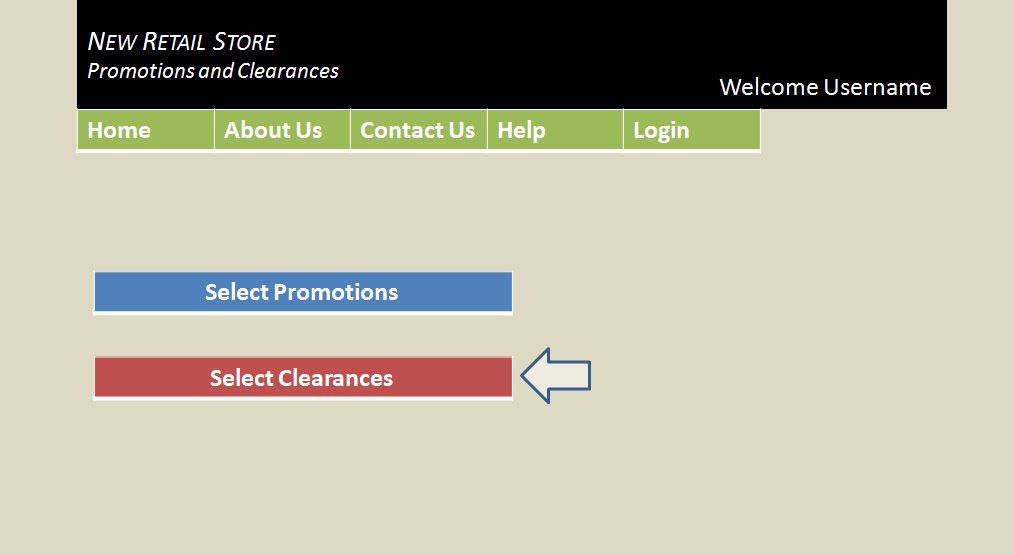


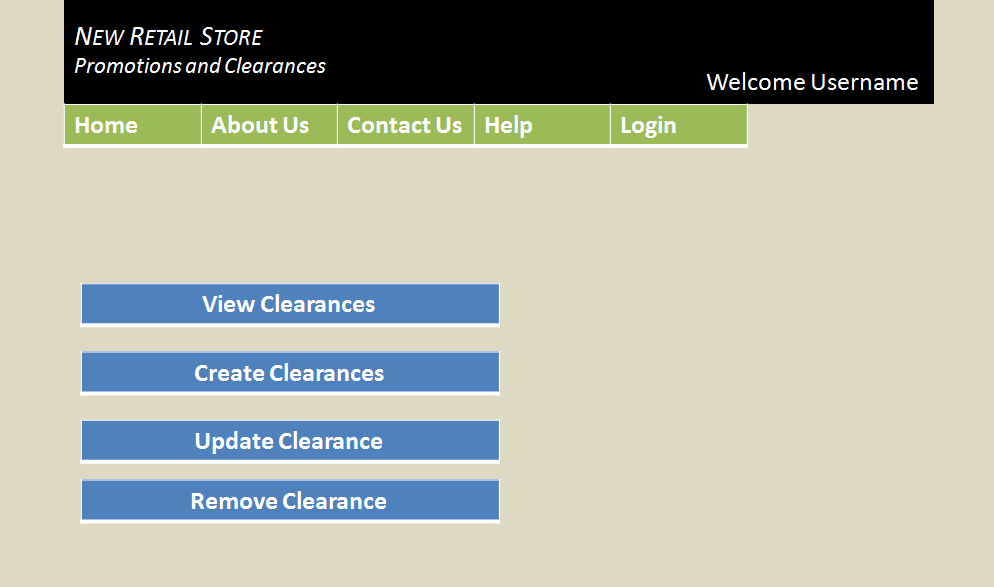






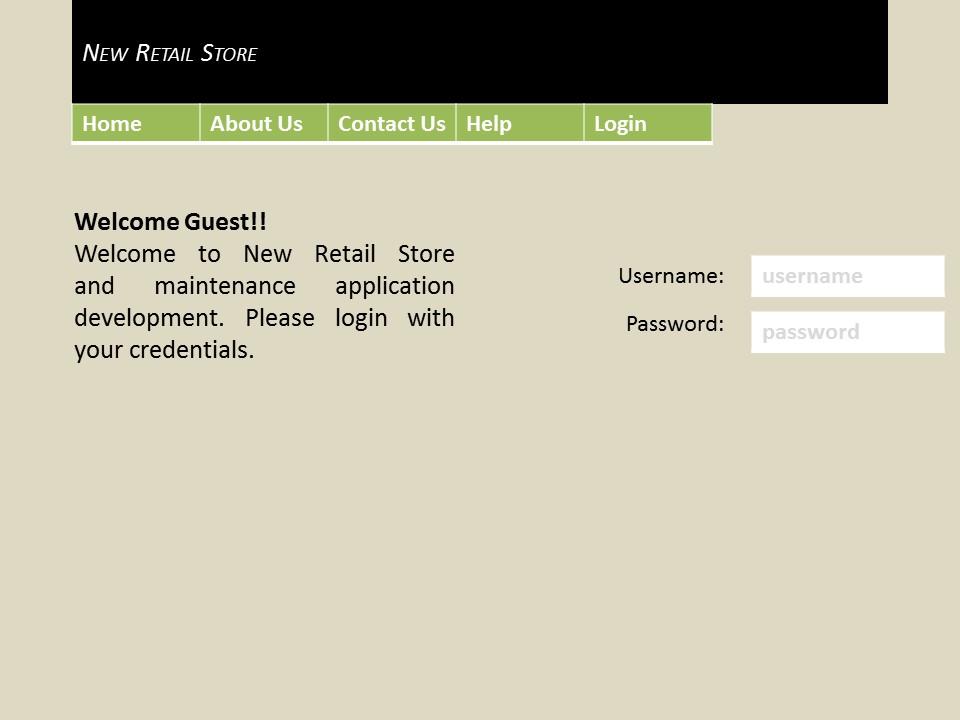




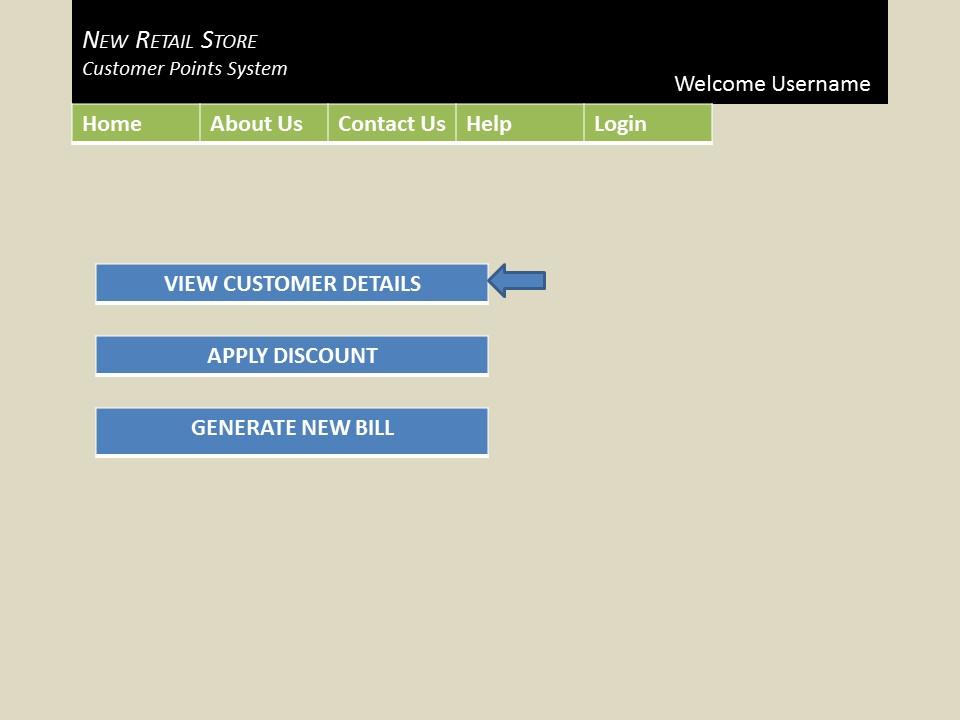


**Customer Point System:**

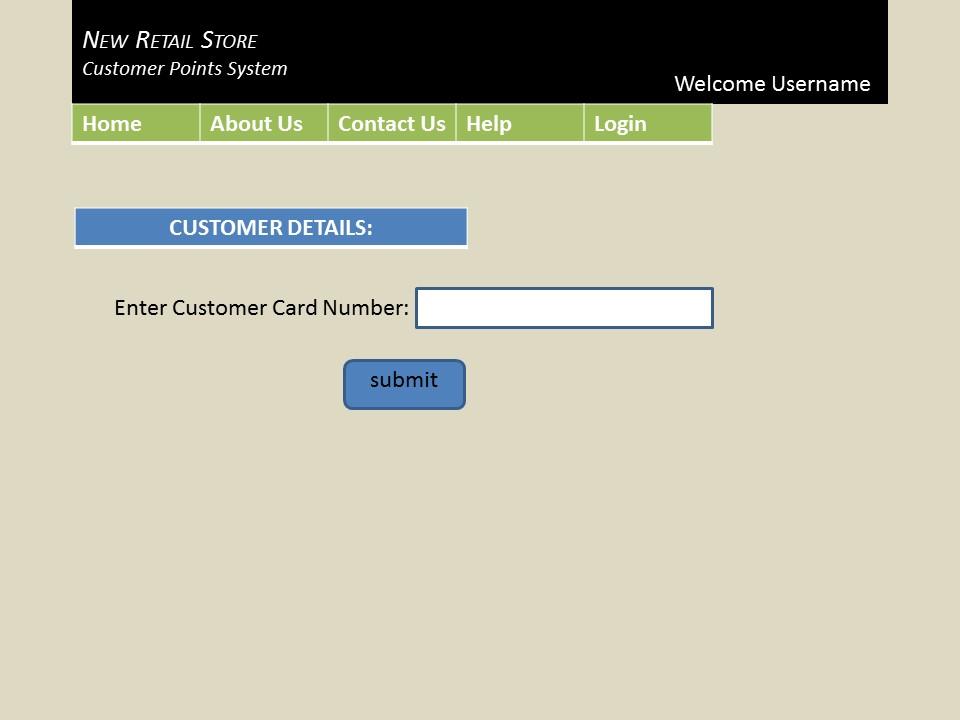
**Login screen for service desk operator:**



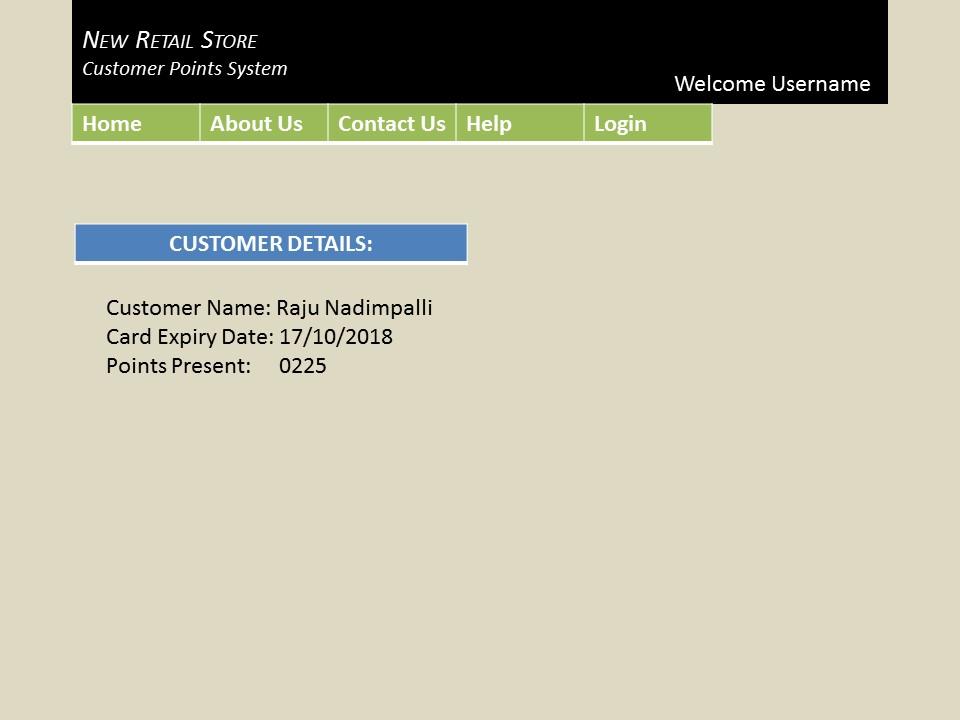
**Home Page for service desk operator:**



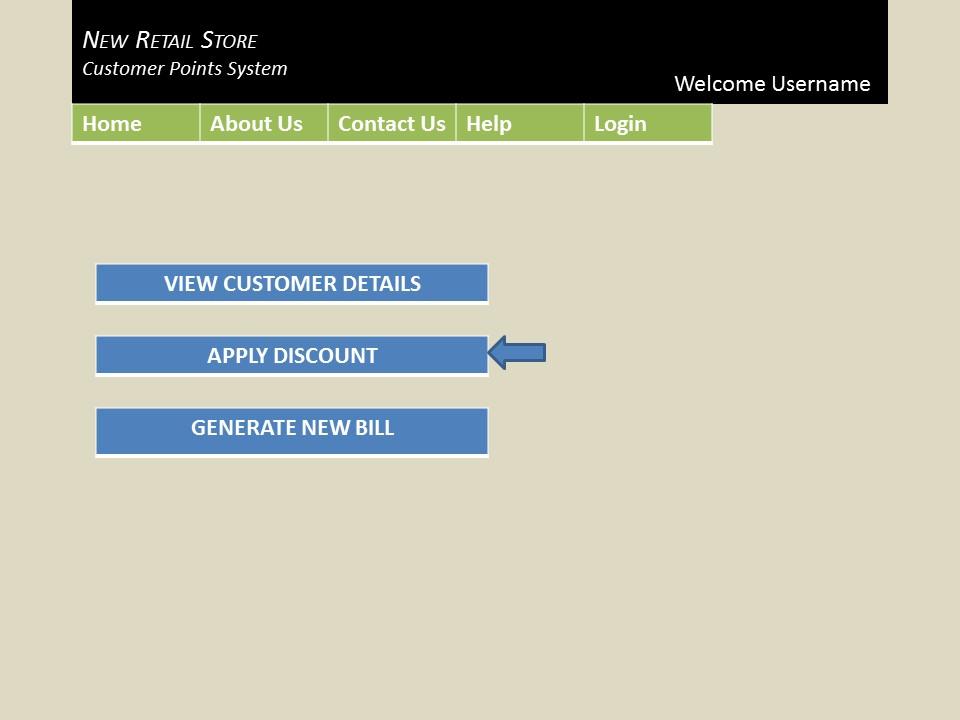
**Enter card number to view customer details:**



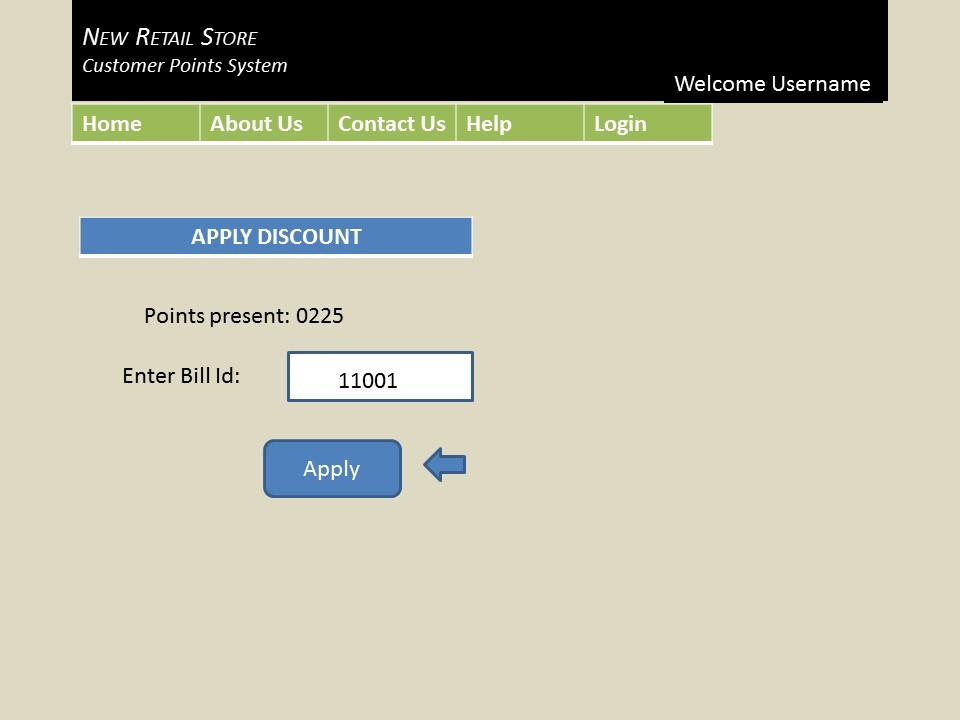
**View customer details:**



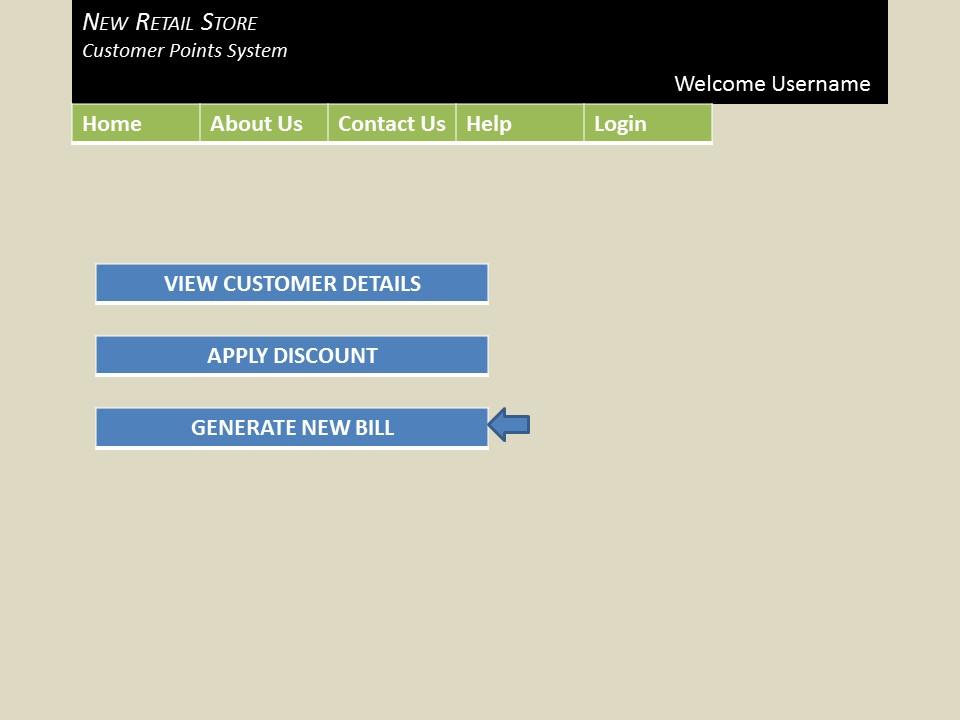
**Select apply discount :**



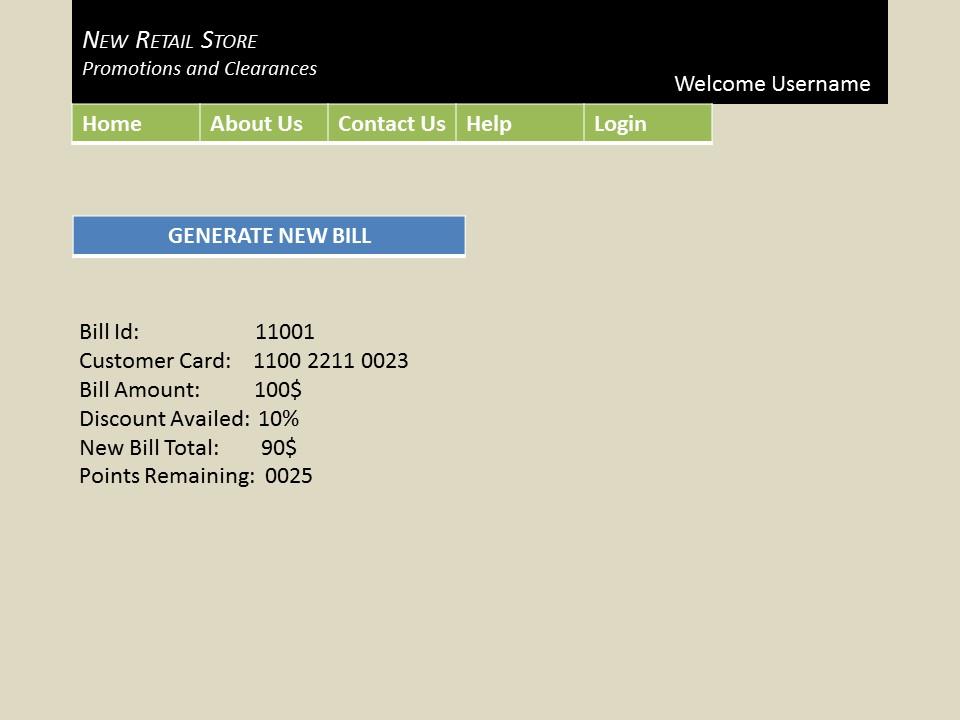
**check points present and enter Bill id to apply discount:**



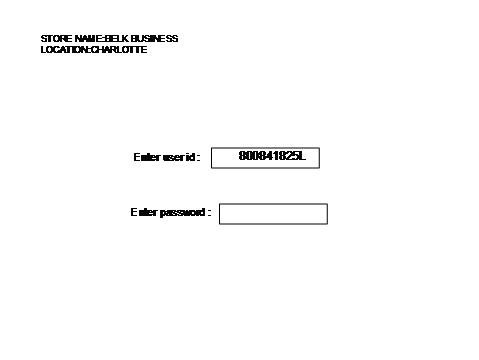
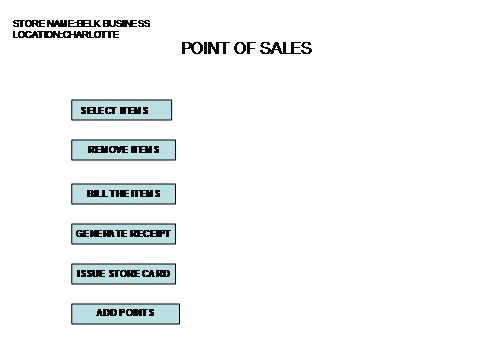
**select “generate new bill”**

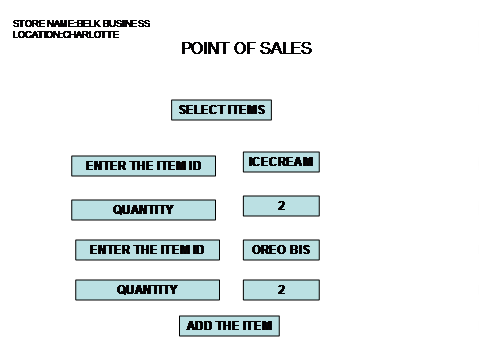


**Display Generated New bill:**

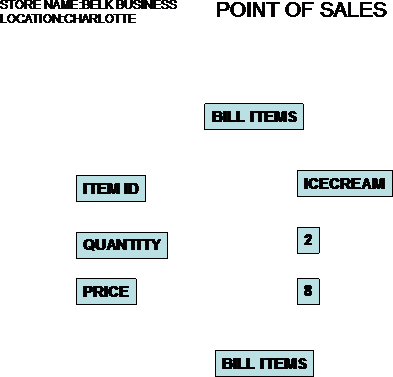


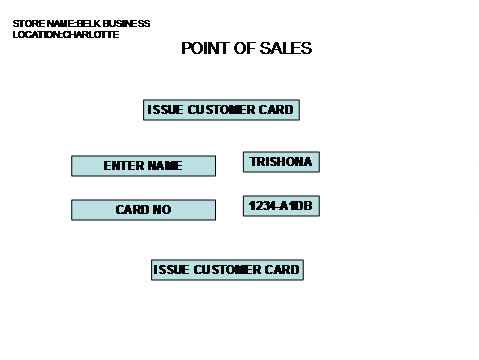
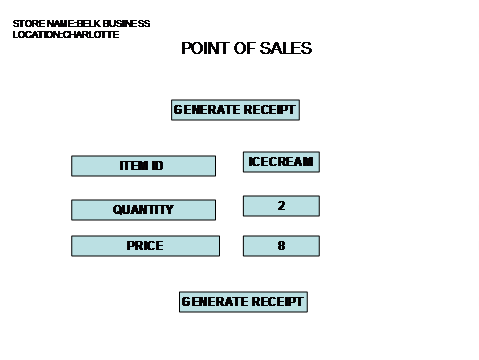
**Point of sales:**





# 





|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## Project Status`

- The project deliverables have so far been timely

- Necessary changes have been made to improve sections of Project proposal and Plan document based on assessment received on 09-10-2014

- Non-functional and functional requirements of the project have been clearly identified

- Current Requirements Specification Document needs to be reviewed after assessment

- Identification of Risks and Risk Management Strategy need improvement

The class diagrams and communication diagrams are drawn and descriptions are provided for all the functionalities.

Mockups are provided for all the functionalities.

Included the screen shot of Gantt chart, updated the task descriptions in the project and proposal.

The use case diagrams, activity diagrams, analysis class diagrams, communication diagrams and the narratives are modified based on the assessment provided.

**Changes made to the document:**

**Creation of New Store Module:**

* Edit Department Details and Remove Department Use cases are are newly included to the Configure Department Details in the Creation of New Store Use case Diagram.
* Activity diagram is changed by removing the Store Database as a user and updating the syntax for nodes. Also the description for the Activity diagrams is modified as per the changes made to the diagrams.
* In the Analysis Class Diagram, editDepartmentDetails() and removeDepartment() methods are added to DepartmentCreationController and the corresponding updations are made to the description of Analysis class Diagram.
* Communication diagram for the same has been changed along with the narrative description.
* Usecase Story board for the Store head is changed along witht the narrative description.
* Prototype/ Mockup screens have been changed. Previous screen are removed and screen shots of the HTML templates are kept.

**Inventory Management:**

* Changes to use case diagram. Added a search use case and view all items use case for Store Head and Department Head. Searching a item is required before viewing, updating and deleting a item.
* Major changes to activity diagram. Used proper syntax, used decision and merge nodes where ever necessary.
* Redrew analysis class diagram from updated use case diagram. Split EmployeeGUI boundary class into StoreHeadGUI and DepartmentHeadGUI
* Changed communication diagram accordingly based on analysis class diagram.
* Redrew analysis package diagram, identified service packages and separated them as application specific and application generic
* Drew separate interface flow diagram for Store Head and Department Head and determined the UI prototype screens to be generated.
* Developed HTML prototypes and included screenshots in the documents
* Included actor descriptions for all actors included in the system before use case diagrams
* Included narrative needed for use case diagrams, analysis class diagram and communication diagram

**Promotions and Clearances module:**

* Promotions and clearances use case has been modified including promotion and clearance details that adds, updates and removes promotional and clearance items.
* Activity diagram has updated.Swim lanes have been removed and decision and merge nodes are included to show the flow.
* Class diagram has been updated by adding entity class for promotions and clearances.
* Communication diagram has been updated according to the class diagram that has been modified.
* Mockup screens are updated for promotions and clearances screen
* Use case story board is updated for store head user in Promotional and Clearances screen

**Customer Point System:**

* The use case diagram is updated, here the POS actor is removed since it is already present in the POS functionality.
* The use case descriptions are updated to match the use case diagram and the activity diagram
* The activity diagram is updated providing that service desk operator is the only actor, removing the swim-lanes and the duplicate actions. Decision node syntaxes are corrected.
* The analysis class diagram and the communication diagram are updated as per the use case. The descriptions are improvised.
* Use case storyboard and its description are updated.
* Mock up screens are improvised.

POINT OF SALES:

1)The use case diagram has been modified(customer has been removed)

2)The use case descriptions are updated

3)The activity diagrams are corrected based on the assessment provided,such as customer is removed from it and swimlanes are removed and node syntax are corrected

4)The description has been updated

5)Analysis class diagram has been updated,such as the entity and the controller classes are linked correctly

6)In communication diagram,the inconsistency are removed such as the sequence of the events are updated

7)Use case story board and its descriptions are improvised

8)Analysis package is modified,the analysis package are based on the actors

9)Mockup screens are done using Powerpoint