**Grocery Shopping Application**



|  |
| --- |
| **Team Members** |
| 1. **Satyam Kumar** |
| 1. **Manish Jawage** |
| 1. **Sonali Deshmukh** |
| 1. **Rannvijay Kumar** |

1. Introduction

This document outlines a case study for Sprint 2 project. The project is to develop a Grocery Shopping Application as integration of all independent micro-services. This document contains the work flow of the system and gives guidelines on how to build the functionality gradually in each of the course modules.

2. Overview

This project is aimed at developing an online Grocery Shopping Application (GSA)for customers and administrator. GSA can be used to search for products based on search condition, add products, modify an existing products details like price or availability status and display all products’ menu. Customers can view and search for products and add the selected products to cart. After adding it to the cart he can view his cart and modify it if he wants and then proceed for billing.

The model followed is an agile model. Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations.

The project had to go under 2 sprints. First sprint had design implementation by core java and collections implementations. The assumed data was made static by using collections to enhance and have a practical knowledge on the collection’s framework. In second sprint we had to convert our monolithic architecture to micro-service architecture. Implementation of sprint 2 is with spring rest services along with Logger implementation and h2/MySQL database, the front end of this project is designed in AngularJS. Some of DevOps tools are also used such as Jenkins and SonarQube. Herein the entire application is ready to serve as a Grocery Shopping Application with all major functionalities.

Following is a list of functionalities of the system

There are two types of users who would access the system viz. Warehouse Administrator(Retailer) and Customers. Each one of them would have some exclusive privileges as follows.

1. Administrator

A set of administrators are assigned for managing the system. An admin has been assigned a set of privileges to manage the system. An admin can perform the following functionalities:

* + Login to the system using his/her credentials.
  + Add individual product details by providing all the field values and inject the values into database table if data are valid else display an appropriate error messages.
  + Modify product details according to rise or fall in in prices or stock of a particular product.
  + Display the products menu.

1. Customers
   * Login to the system using his/her credentials.
   * Search a product based on any of the fields – Category or Name.
   * Add products to cart
   * Modify his personal cart
   * Navigate to the billing page

3. User and Epic Stories

In a sense, stories and epics in agile are similar to stories and epics in film or literature. A story is one simple narrative; a series of related and interdependent stories makes up an epic. The same is true for your work management, where the completion of related stories leads to the completion of an epic. The stories tell the arc of the work completed while the epic shares a high-level view of the unifying objective.

* [**Epics**](https://www.atlassian.com/agile/project-management/epics) are large bodies of work that can be broken down into a number of smaller tasks (called stories).
* [**Stories**](https://www.atlassian.com/agile/project-management/user-stories), also called “user stories,” are short requirements or requests written from the perspective of an end user.
* Table 1: User and Epic Stories of Employee Maintenance System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| User | Login | Admin / Customer | Validate the entered username and password | Check whether the entered password and username is correct or not |
|  | Register | Customer | Register new users | The next time users can directly log in. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Products Module | Add Products | Admin | Enter Product details | A new product can be added to the database |
| Search Products | Admin / Employee | Search products by various fields and wild card searches(elastic search) | A particular product can be viewed from Products database |
| Delete Products | Admin | Delete a Product by Product Id or Delete all Products at once | Product/s will be deleted from Products database |
| Update Products | Admin | Modify Products details | Product details of selected Products will be modified |
| View all the Products | Admin / Customer | View all the Products from database | All the updated Products can be viewed from Product database |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Cart Module | Apply for Leave | Employee | Apply leave to the respective manager. | A new leave application entry will be created in the database. |
| Check leave status | Employee | Check the personal leave status and personal leave history. | A particular status of leave can be checked by Employee. |
| Approve / Reject leave | Employee(Sr.) | Approve / Reject the leave if having the authority of manager | Leaves can be granted if applicable. |
|  |  |  |  |  |

4. Use Case and UML Diagrams

1. **Pre-requisites**

User must be logged in as an Admin to perform add, update, view all, search or delete Products.

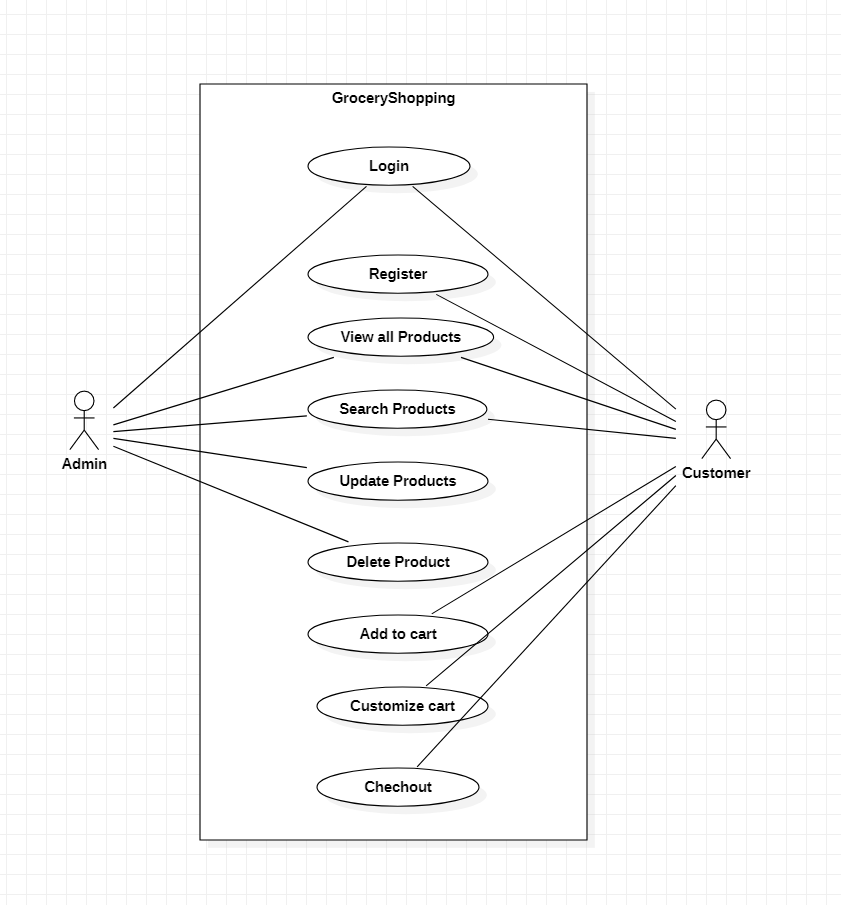
Customer must be logged in to perform the elastic search for products and add products to cart, view and modify status of his cart, and then check out.

1. **Non-functional requirement**

Proper form validations are maintained. To make it more user-friendly proper alert messages are shown wherever required. Any changes in database will only be performed if user gives a confirmation to those pop-up warnings.

* 1. **Employee Maintenance System Diagram:**
* Use case diagram

Following is the use case diagram of Employee Maintenance System.

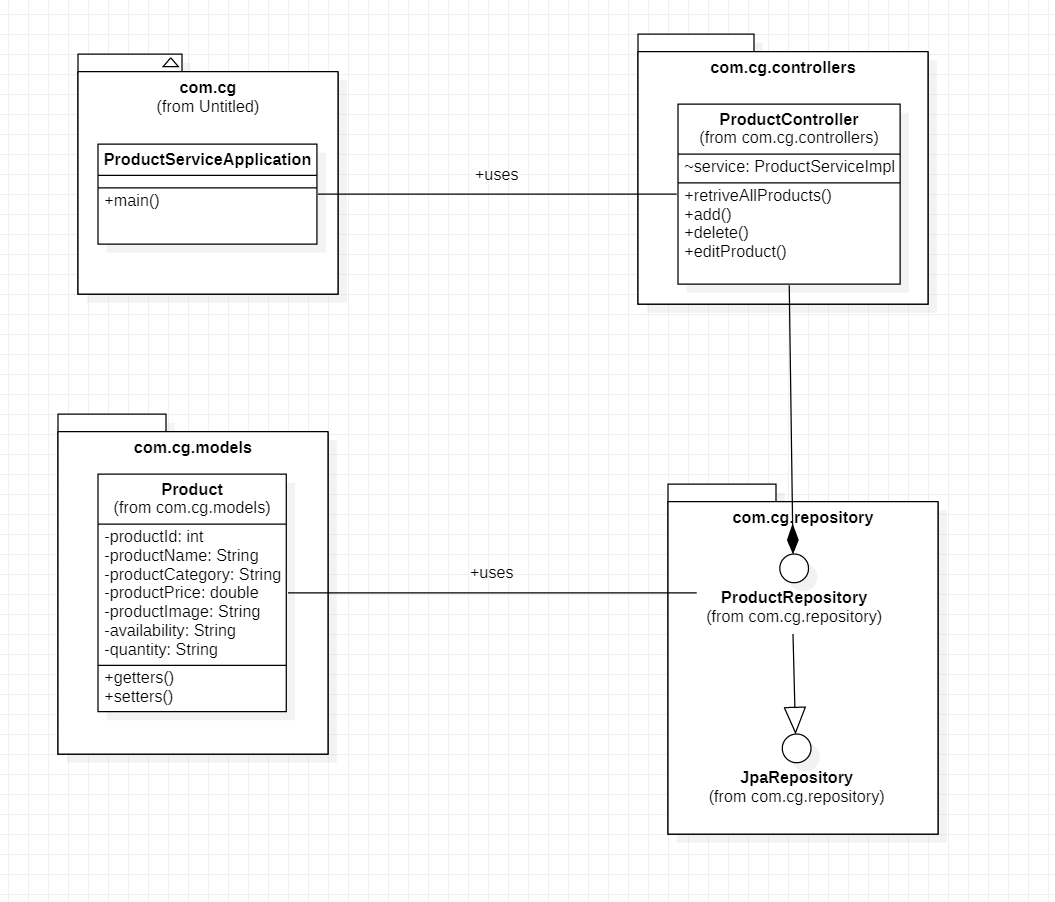


**Figure 4.1 Use Case Diagram for Employee Maintenance System**

**4.4 Product Module:**

Class Diagram:

Following is the class diagram of Product Module:



**Figure 4.4 Class Diagram for Product Module**

* **Product Module:**

This micro-service is used to perform all the CRUD operations related to the Products which will be managed by the Admin. The respective operations will store the corresponding data in the Product database.

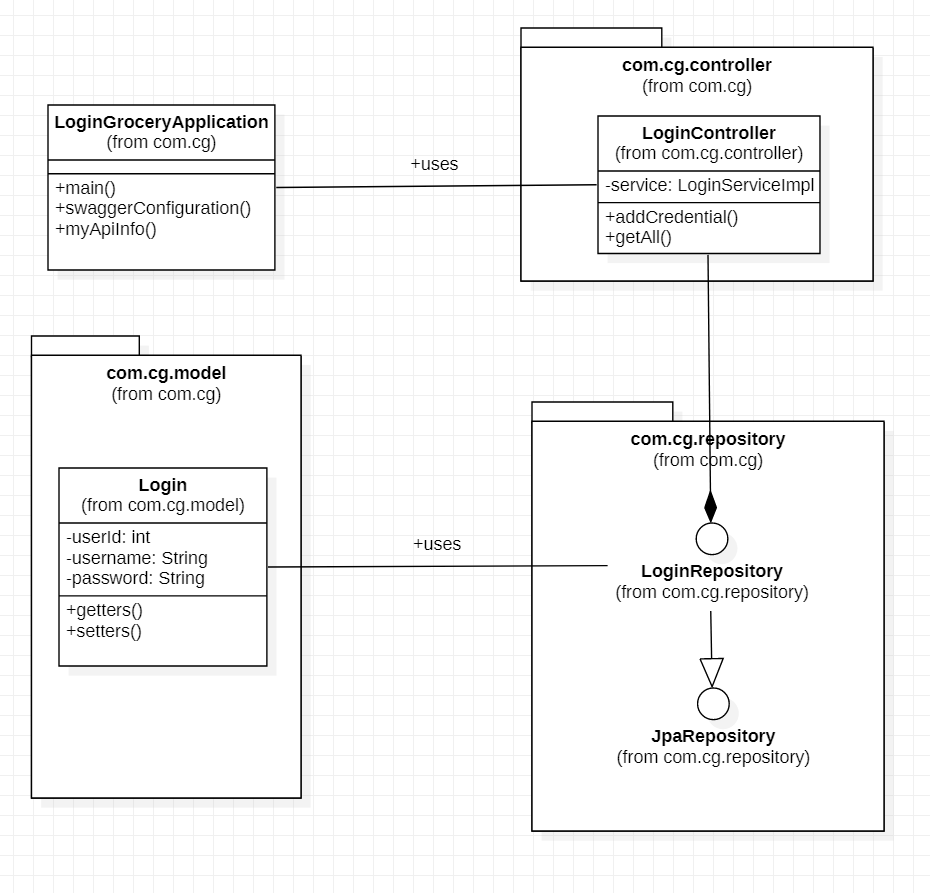
Following is the list of all the functionalities in this module:

1. **Add Product:** This functionality has been designed to add a Product in the database. This can be only accessed by Admin. The form should be designed with proper validation with all fields required.
2. **Delete Product:** Admin needs to click a button to delete a Product from the database. A confirmation warning will pop up. On clicking on the confirm message the Product will be deleted.
3. **Update Product:** There should always be a scope to update the data which has been once entered about a product as its availability and price are really dynamic attributes. Thus, it is our admin’s requirement to introduce an update option. Hence, we have implemented a user-friendly update option for our Grocery Shopping Application. Just on clicking an update button, the update form will appear and the Admin can easily update the stock and/or price, or maybe other attributes too, of the product.
4. **Search Product:** This functionality has been designed to search a particular Product on the basis of category or elastic name specified. The required Product will be fetched from the Product database.
5. **View all Product:** This functionality has been designed so the Admin will be able to view all the Products stored in the database.

**4.5 Login Module**

* Class diagram

Following is the class diagram of Login Module:

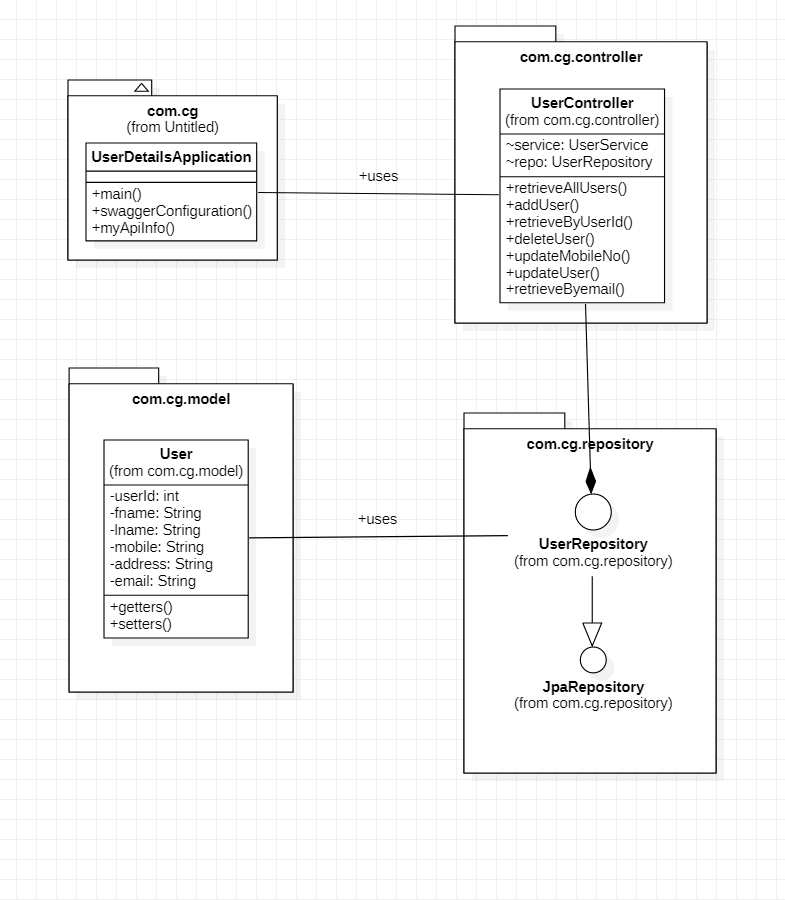


**Figure 4.6 Class Diagram for Login Module**

**4.6 User Module**

* Class diagram

Following is the class diagram of User Module:

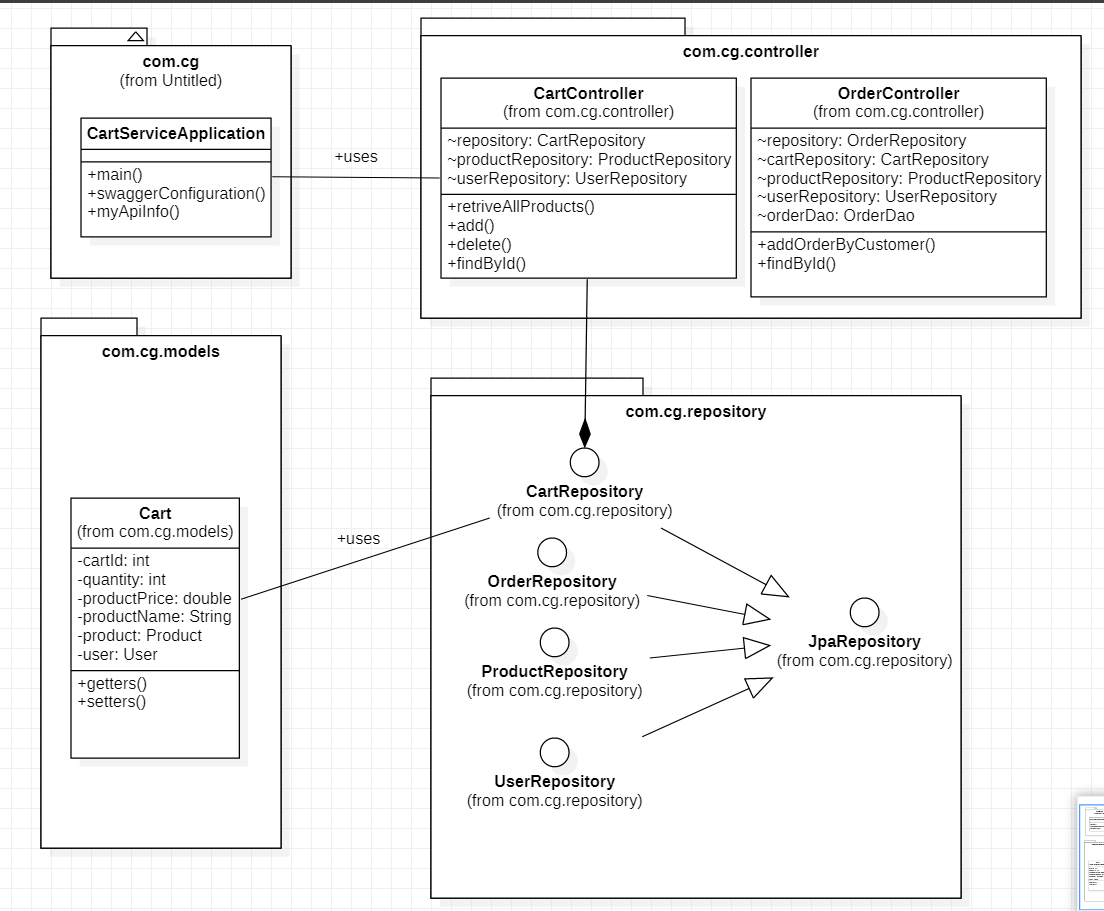


**Figure 4.8 Use Case Diagram for User Module**

**4.7 Cart Module:**

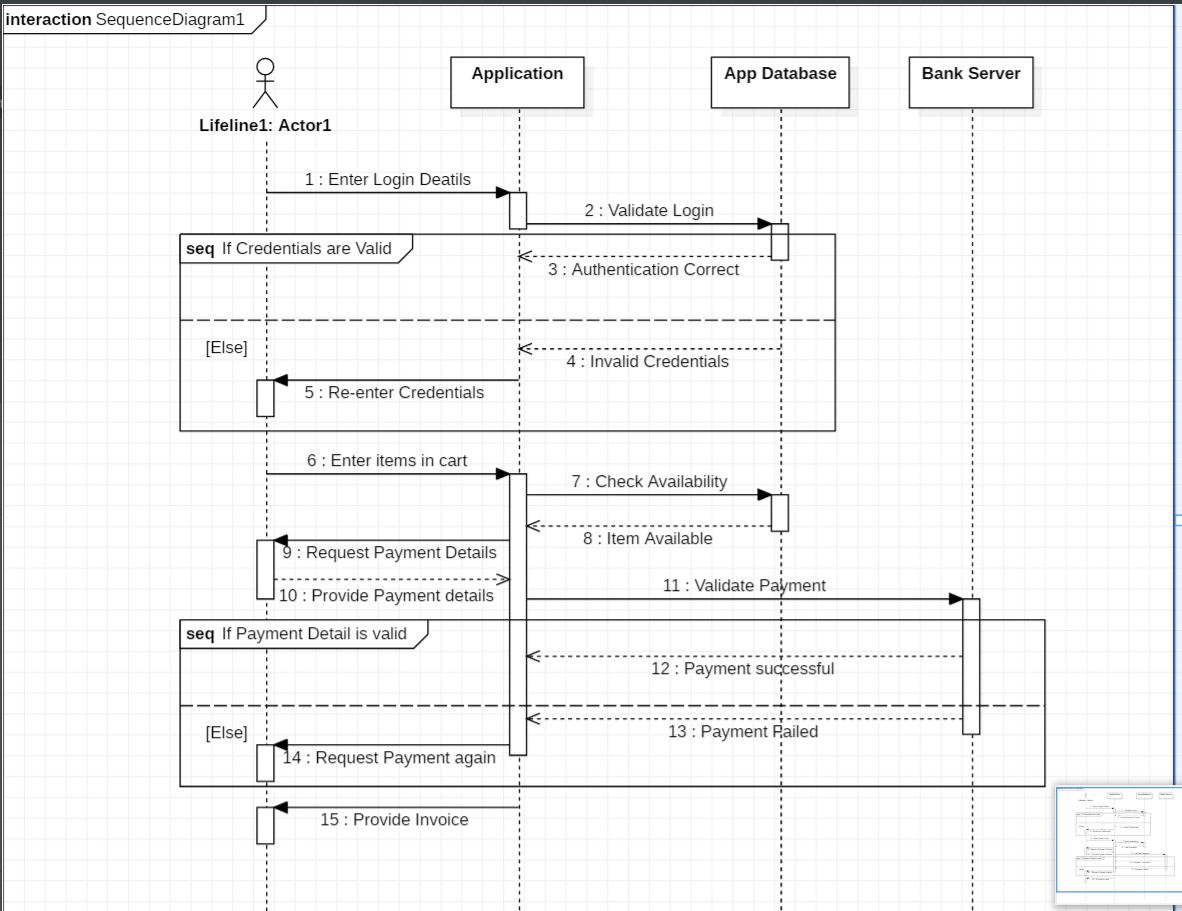
* Class diagram

Following is the class case diagram of Cart module.

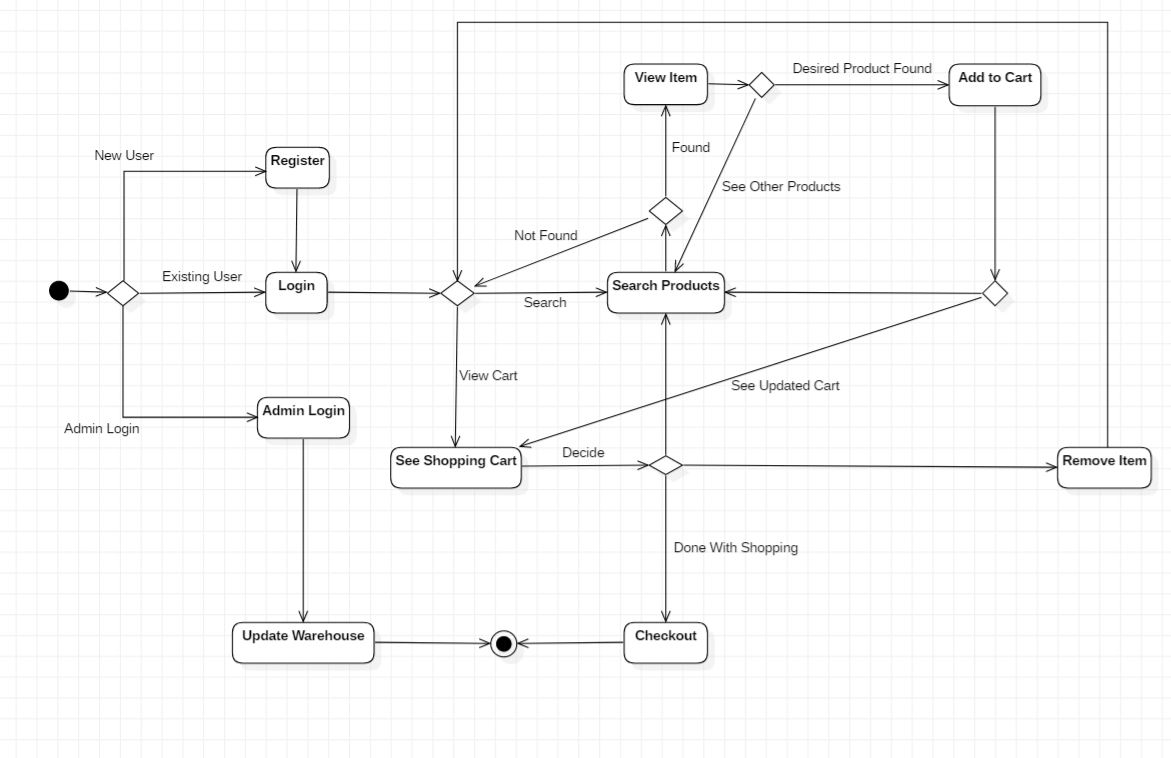


**Figure 4.8 Use Case Diagram for Cart Module**

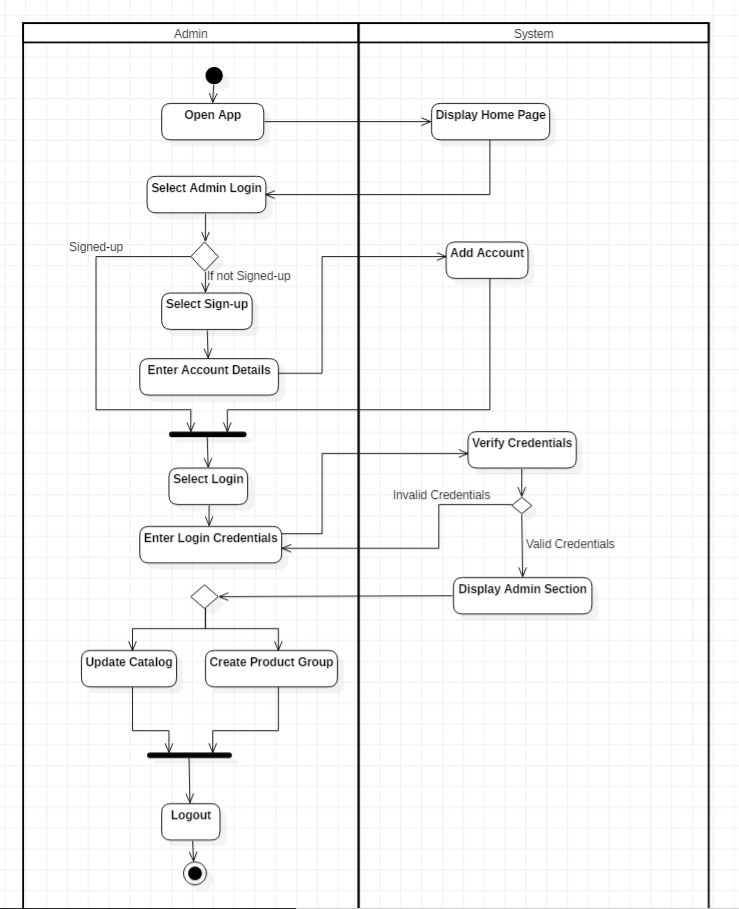
**Sequence Diagram**



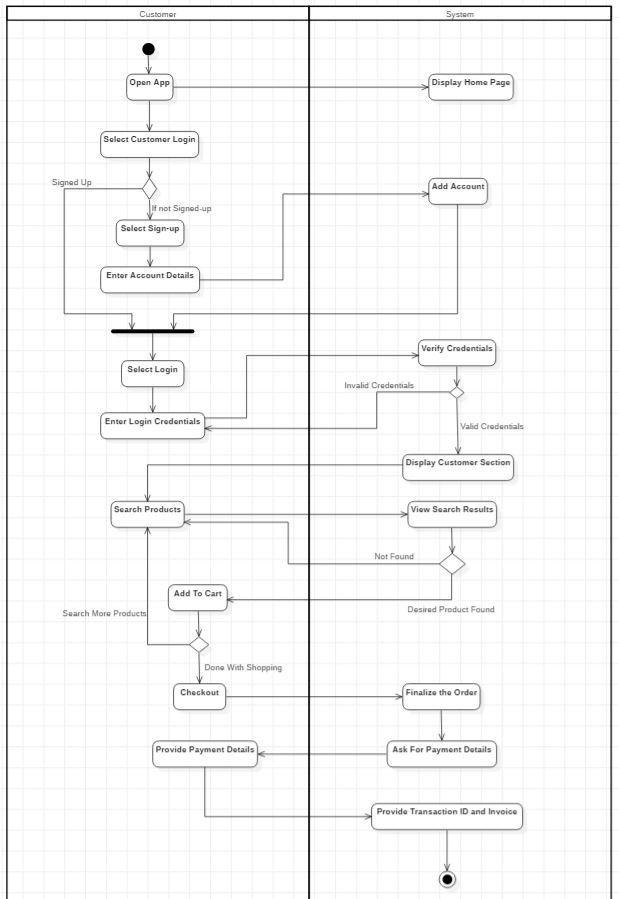
**State Diagram**



**Activity Diagram (Admin)**



**Activity Diagram (Customer)**



1. System Requirements

Below is a list of the minimum Hardware and Software requirements to access Grocery Shopping Application.

**Operating System:**

* Windows 7 and above.
* Mac OSX 10.8, 10.9, 10.10 or 10.11
* Android 3 and onwards.

**Hardware:**

* Processor (CPU) with 2 gigahertz (GHz) frequency or above
* A minimum of 4 GB of RAM
* Monitor Resolution 1024 X 768 or higher (For better view)
* A minimum of 5 GB of available space on the hard disk
* Internet Connection Broadband (high-speed) Internet connection with a speed of 2 Mbps or higher
* Keyboard and a Mouse or some other compatible pointing device

**Browsers:**

* Chrome\* 58+
* Microsoft Edge\* 20+
* Mozilla Firefox 40+
* Internet Explorer 11+ (Windows only)

*\**Google Chrome version 42+ and Microsoft Edge do not support NPAPI-type plug-ins, including Java plug-ins and many media browser plug-in.

*Users using unsupported browsers may experience issues submitting forms, placing orders, purchasing, updating details and transaction management threads.*