

Project Report

MVC FrameWork

Watch Store

Object Oriented Analysis & Design

Group Members:

1. Asad Tariq Sheikh 11355
2. Muhammad Nomir 11330

Submitted To:

Sir Sohail Imran

Date: May 19, 2022

ACKNOWLEDGEMENT:

"This Report Have Been Prepared Through The Help Of Online Consultation Of Different Webpages For Datasheets & Also In This Report We Explain Our Project Thoroughly. This Project Report Have Been Created Within The Given Time Spectrum. All Of The Given Requirement Are In This Report With Correct Format."

ABSTRACT:

In this whole report you will see all our research and hard-work put into this report. We created a project with **MVC Framework** and the website we made with the framework was named **WatchStore** we will explains' the functionalities of the website and also show the different UML diagrams that were developed with this MVC framework project.

SCOPE:

The scope of this project is not just to develop a working MVC framework website but also understand a website is developed in a proper manner and how in the future and becoming a project analyst we will have to develop UML diagrams so that a team of developers can easily understand every piece of information given in the diagrams.

OPERATING ENVIRONMENT:

1. Visual Studio 2019 (with proper & additional installations)
2. MVC Framework 5
3. SQL Server Management Studio 2019
4. Azure Hosting
5. Support of HTML, CSS, Bootstrap and jQuery

LINKS:

GitHub:

<https://github.com/Nomir-01/OOAD-Project>

Azure (Web Hosting):

<https://watchstoreapp.azurewebsites.net>

Email -----> asad@gmail.com

Password -----> asad123

Admin Panel:

<https://watchstoreapp.azurewebsites.net/Auth/SignIn>

Username -----> asad

Password -----> asad123

WATCH STORE (WEBSITE):

Our website is simple but highly functional, the basic functions are that a customer can browse the website add the products to cart login to account, register to account, checkout and while checking out increase or decrease the quantity of a product or also remove an item, customer can also cancel their order at any given point in time.

The admin can add a new brand of watches in the website and also edit, delete or add any products in the website of the brands, the admin can also see order details, edit their status or even delete the orders.

The system works on managing the website stocks and orders such that if a customer decreases the quantity or increases the system is to manage the stock of the products accordingly, or if the admin edits a product or adds a new brand category, the system is to manage that as well.

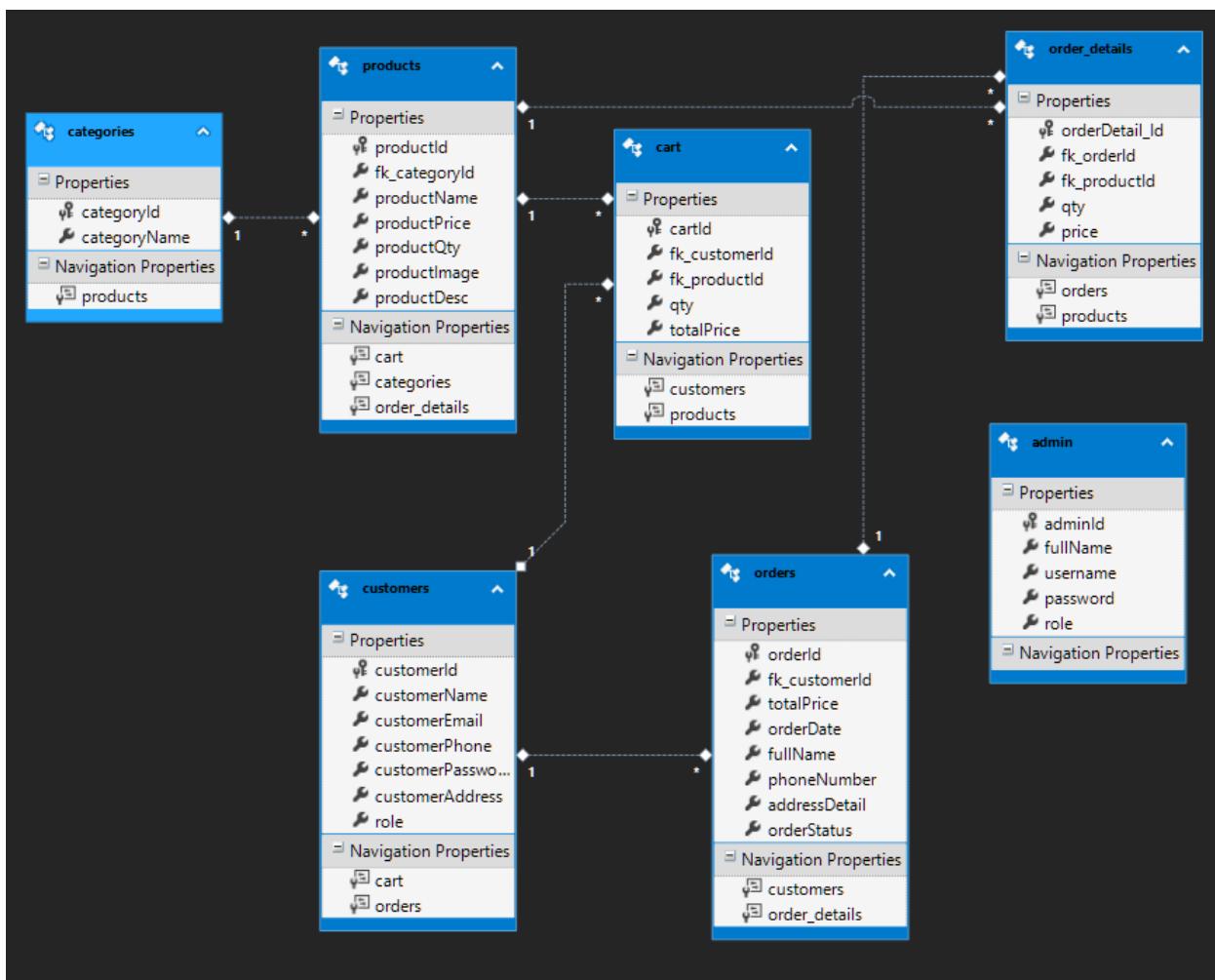
IMPORTANCE OF ADMIN PANEL:

The admin panel is an important functionality that just doesn't help in managing the orders but also the website. The admin panels helps' in increasing our brand categories without accessing the code, if we want to increase two three or even four categories the admin panel allows us to do that and also add products to that brand new category without having to access the code, we can add as many products as we want and edit it as many times as we want.

The admin panel also helps us in managing the orders either seeing the details or editing the order status accordingly or even deleting the order if it is not needed anymore.

DATABASE:

We chose SQL Server as our database platform, we choose this because we had the most hands-on on this software, also connecting a SQL server database to a MVC framework is easier and it allows us to create the Model Class without any complexities and ease and also our hosting platform azure supports SQL server database so it was an obvious choice to use this software.



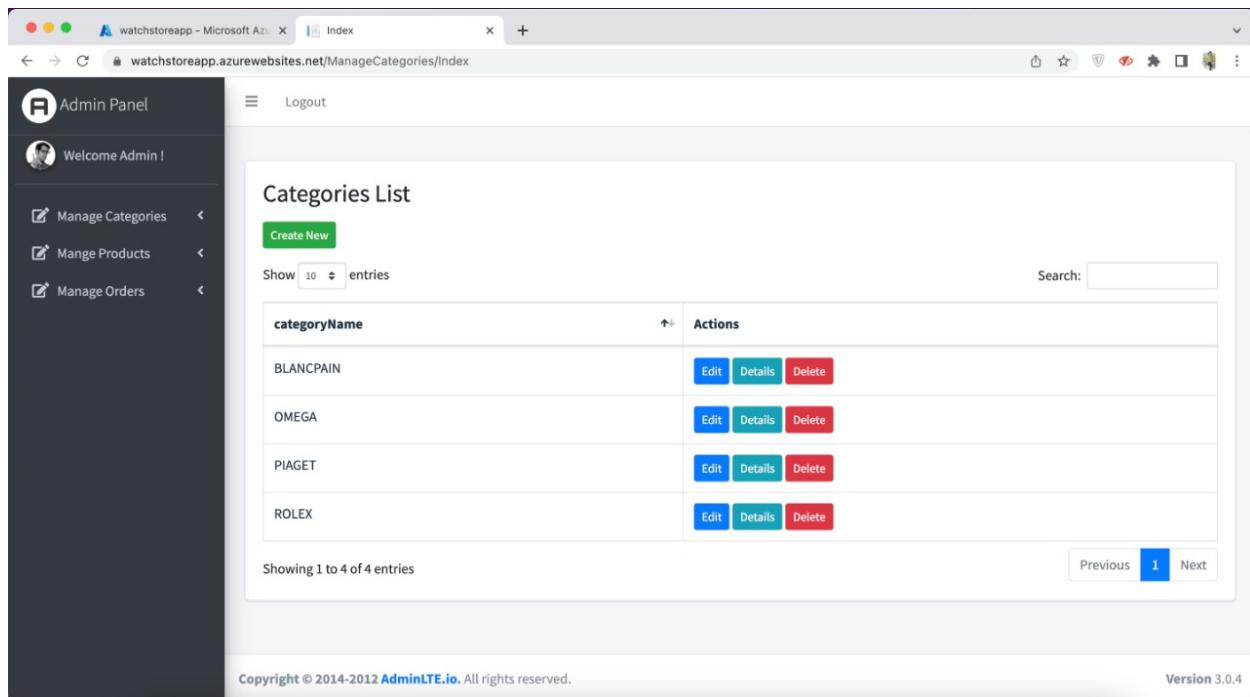
WEB HOSTING:

We chose azure as our hosting platform as it provides a free 12-month service, which for the time spectrum of our project was more feasible and also affordable. Azure also helps us in connecting our database directly to the MVC framework so this is the second reason we choose azure as our hosting platform.

The screenshot shows the Microsoft Azure portal interface for the 'watchstoreapp' App Service. The top navigation bar includes links for Home, watchstoreapp, Microsoft Azure, and a search bar. The main content area is titled 'watchstoreapp' and shows the 'App Service' type. The 'Overview' tab is selected, displaying basic information such as Resource group (WatchStoreResourceGroup), Status (Running), Location (Central US), Subscription (Azure subscription 1), and Subscription ID (f1c07b67-5d35-46a7-bec2-828aa5ce6d6). It also lists URL (https://watchstoreapp.azurewebsites.net), Health Check (Not Configured), App Service Plan (WatchStore20220514170437Plan (\$1: 1)), FTP/deployment username (No FTP/deployment user set), FTP hostname (ftp://waws-prod-dm1-247.ftp.azurewebsites.windows.net/sites/...), and FTPS hostname (ftps://waws-prod-dm1-247.ftp.azurewebsites.windows.net/sites/...). A 'Tags' section allows adding tags, with a note to click here to add tags. Below this, there are three cards: 'Diagnose and solve problems' (Our self-service diagnostic and troubleshooting experience helps you identify and resolve issues with your web app.), 'Application Insights' (Application Insights helps you detect and diagnose quality issues in your apps, and helps you understand what your users actually do with it.), and 'App Service Advisor' (App Service Advisor provides insights for improving app experience on the App Service platform. Recommendations are sorted by freshness, priority and impact to your app.). At the bottom, there are three charts: 'Http 5xx' (with values 90, 80, 70, 60), 'Data In' (with values 1.4kB, 1.2kB, 1kB), and 'Data Out' (with values 1.6kB, 1.4kB, 1.2kB, 1kB).

WEBSITE GUI:

We chose a basic and simple design for our website so that a customer can easily browse, search or buy products. For this we used basic HTML and CSS, we also used templates of JQuery and Bootstrap in our website. Our website is also responsive and the minimal design allows it to be used conveniently on different size of devices.

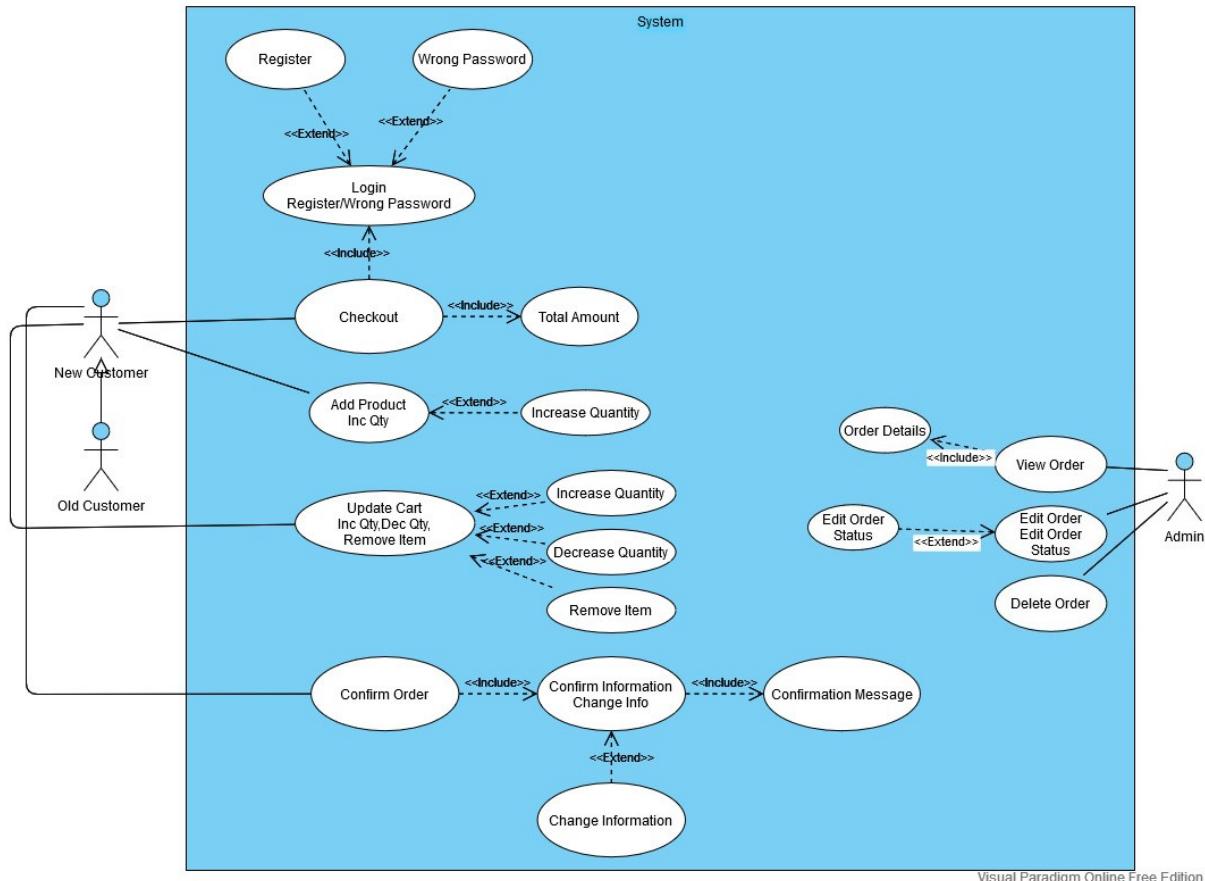


The screenshot shows a web browser window titled "watchstoreapp - Microsoft Azure" with the URL "watchstoreapp.azurewebsites.net/ManageCategories/index". The page is part of an "Admin Panel" and displays a "Categories List". On the left, there is a sidebar with navigation links: "Welcome Admin!", "Manage Categories", "Manage Products", and "Manage Orders". The main content area has a heading "Categories List" with a "Create New" button. It includes a search bar and a table showing four entries: BLANCPAIN, OMEGA, PIAGET, and ROLEX. Each entry has "Actions" buttons for "Edit", "Details", and "Delete". At the bottom, it says "Showing 1 to 4 of 4 entries" and has "Previous" and "Next" buttons. The footer contains copyright information: "Copyright © 2014-2012 AdminLTE.io. All rights reserved." and "Version 3.0.4".

categoryName	Actions
BLANCPAIN	Edit Details Delete
OMEGA	Edit Details Delete
PIAGET	Edit Details Delete
ROLEX	Edit Details Delete

USE CASE DIAGRAM:

Visual Paradigm Online Free Edition

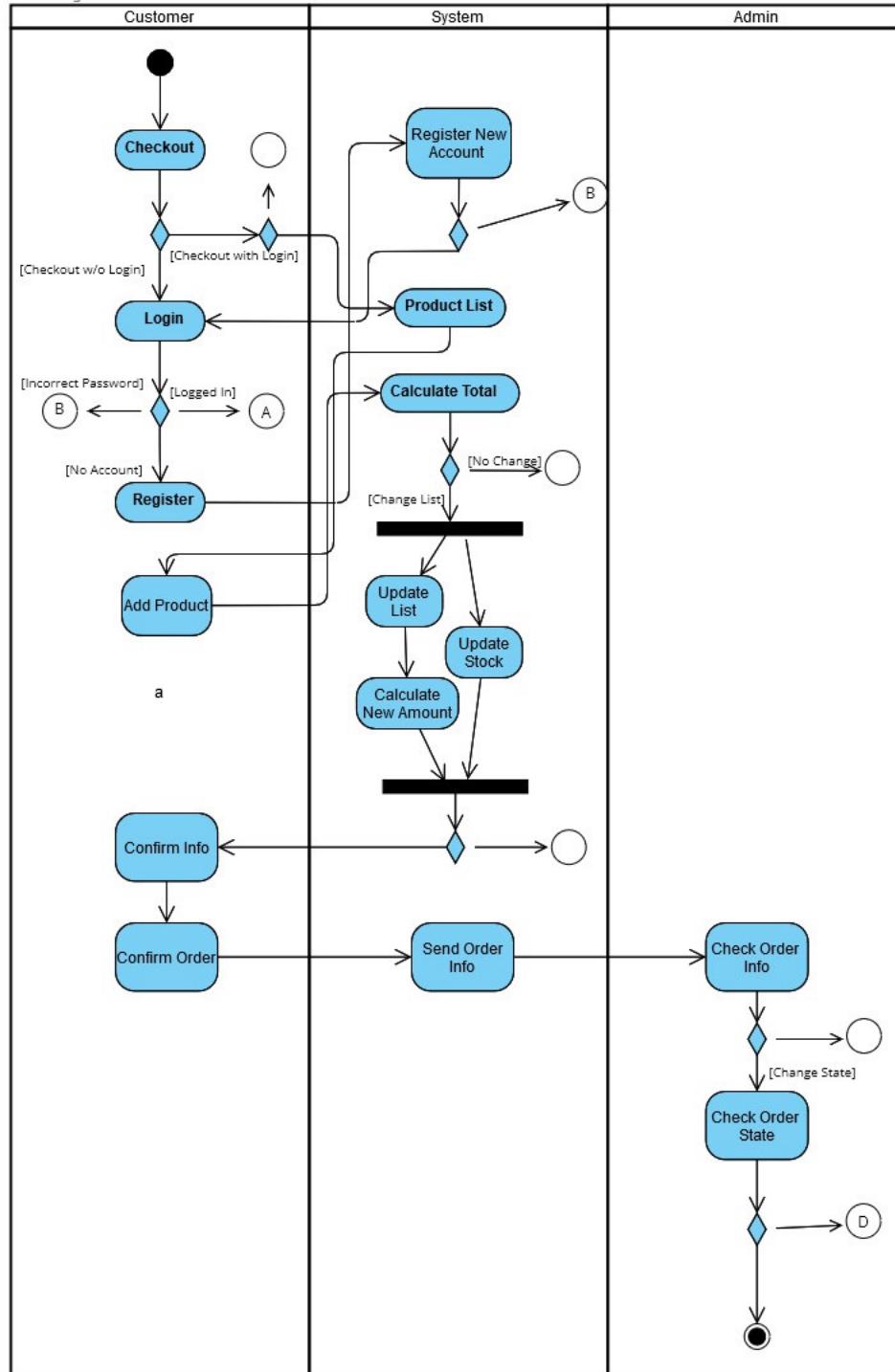


Visual Paradigm Online Free Edition

The use case diagram explains the general use case of our transaction form. It explains how a user can checkout without or with login and which conditions will be triggered when checking out without login. It also shows that user can update their cart to their liking and also they have to confirm their information when checking out. It also shows the admin side uses on how admin can view, edit or delete the order.

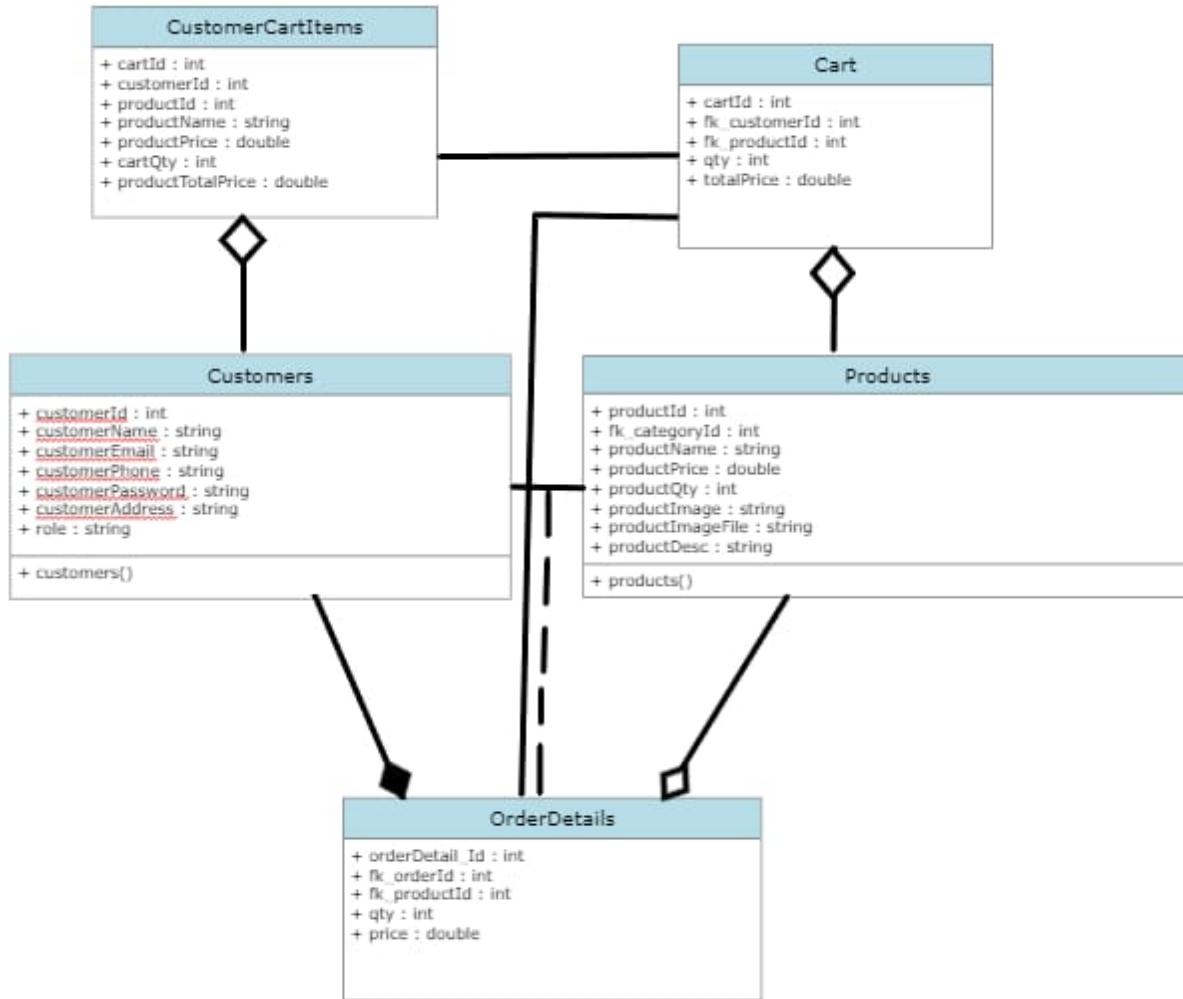
ACTIVITY DIAGRAM:

UML Paradigm Online Free Edition



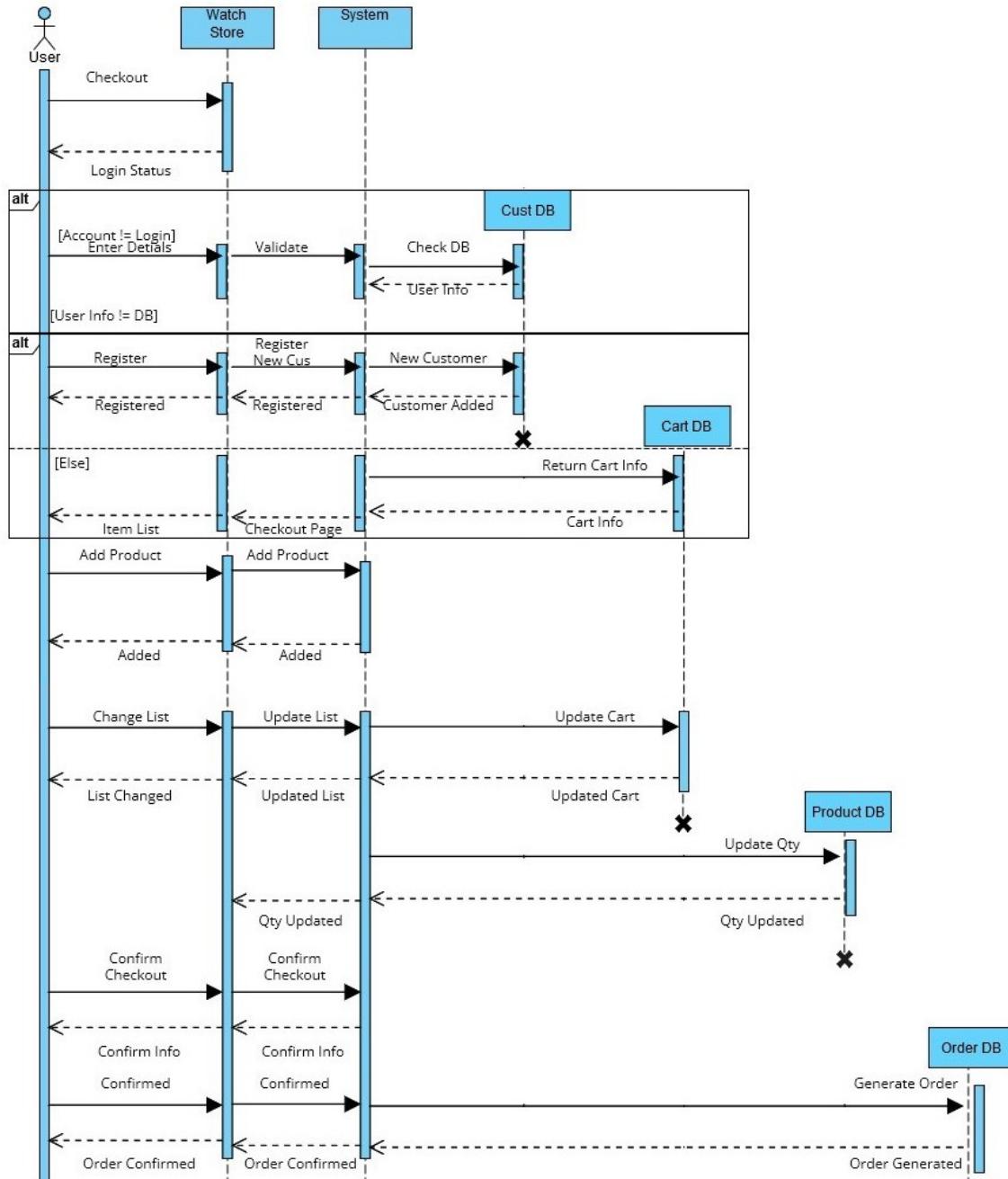
The activity diagram explains the activities of our transaction form. It shows the different conditions and activities happening in the project. It shows the parallel happening activities.

CLASS DIAGRAM:



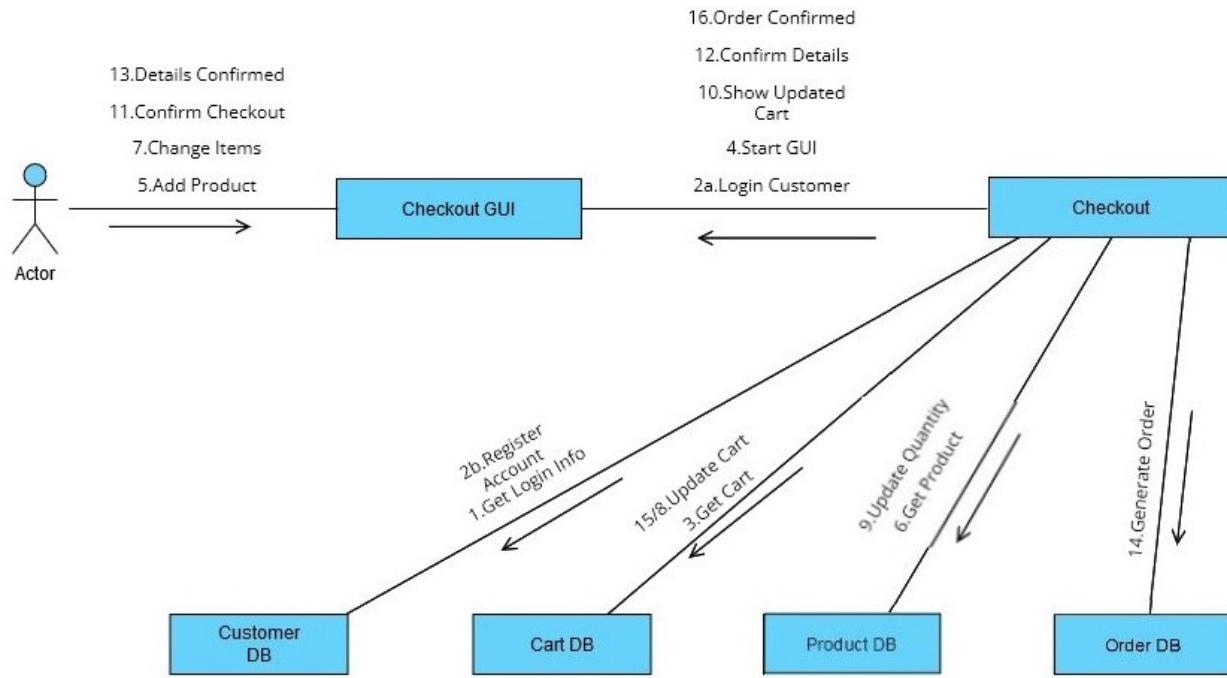
The class diagram explains the classes working in our transaction form, this diagram also tells us association, aggregation & composition between classes.

SEQUENCE DIAGRAM:



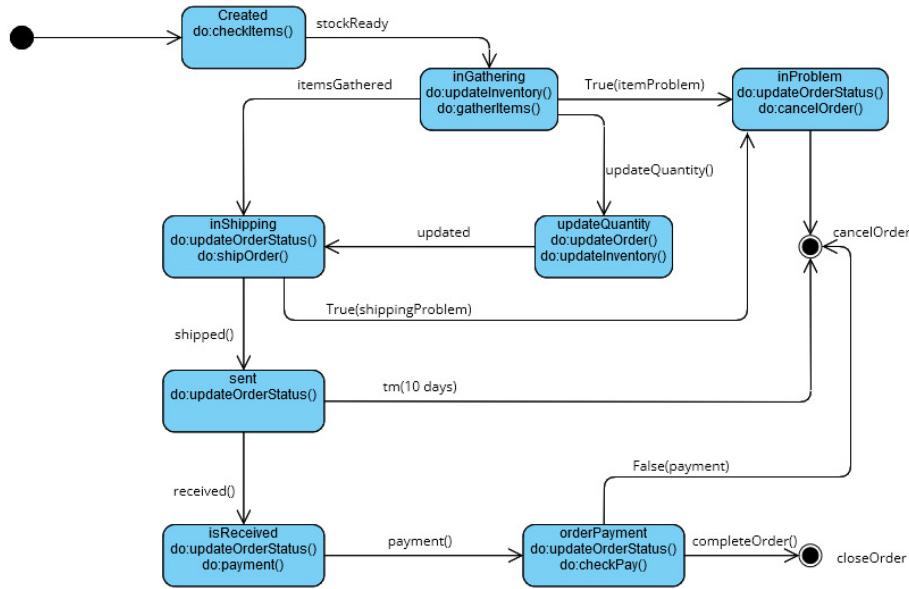
The sequence diagram explains the general sequence of our transaction form. It explains how a user can checkout and which conditions will be triggered on specific actions and how the database will react accordingly. It also shows that user can update cart. The user also has to confirm their information and this confirmed information will generate order from database.

COLLABORATION DIAGRAM:



The collaboration diagram explains the relationship between **View Class**, **Model Class & Controller Class** of our **transaction form**. This diagram specifies all the major events that occur in between the MVC classes, and also creates an ease for the developer to name the methods accordingly.

STATE DIAGRAM:



The state diagram explains the different states of our object (**Order**), how many states an object can retain and achieve before reaching the final state at which the transaction form comes to an end. This diagram allows the developer to make the form according to the given states of order and achieve the most amount of states of an order.