Business Requirements & Design Document

WEBSTORE APPLICATION

Practice Lab Project 1

Table of Contents

1 Overview

1.1 Overview of the application

This document describes the online shopping portal functionality. With this portal user can navigate through a web-site where a new user can register with the web-site and a registered user can do shopping online.

If the user is already registered then he/she can log in to the website, can view his/her profile, can search for products/items ordered by different categories and catalogs, he/she can place the selected items/products in the cart and the user can make an order by providing his details like address to where the ordered items/products to be shipped and he/she can make payment using his credit/debit card.

This application is built using J2EE Technology which makes use of the following technologies.

(i) Struts – As controller

(ii) Hibernate – For managing the persistence

(iii) JTA – To perform transactions

(iv) JNDI – To connect to the database

The backend (database) used is MySQL. Database is present separately in application folder App_data. The project also has a separate database layer class in App_Code in which all the code for opening a connection to the database, inserting the data into the database and returning the data from the database either as a data-set or as a data-table is written which reduces the redundancy of code.

1.2 Users of the application

There can be two types of users who will be accessing this application. They are Registered User and Un-registered user.

Un-registered user can view the home page of the website where he/she can search the items/products by a specific category/catalog which displays all the items/products belonging to that category/catalog along with the photo and all the sub-categories of the specified category will be displayed along with the photo and he/she can browse the

items/products in that category. The products/items will be displayed along with the detailed description (such as price, brand etc) and photo. Generally if the user wants to view the items/products belonging to a category he/she can do so by clicking on the photo of that particular category. But the unregistered user cannot make an order, only he can browse the items/products and categories. Only registered user can make an order.

Registered user is allowed to access all the features that are provided by the application. Once the user is logged in he/she can view his/her profile, can search for products/items, can make an order by adding them in the cart, can either remove/update cart. After he/she made an order he will be provided with a unique order id so that he/she can keep track of his/her status of the order.

2 Functional Requirements

2.1 Scope of Work

Many websites on internet are providing online shopping facilities. The web store online shopping website is generic enough to directly support all types of products/categories. Also features a fully functional cart which simultaneously stores products as well as items.

It Represents the commodities for sale under the below mentioned classifications:

a. Catalog

b. Category

c. Sub-Category

d. Product/Item

In addition it also offers other shopping categories like IPods at reasonable prices and mobile phones of all major brands like Nokia, Samsung, Sony Ericsson and LG are offered online.

2.2 System Features

2.2.1 Requirement / Module / Use-case for Log-in

2.2.1.1 Description

This use-case describes the **Login** page functionality.

REQ-101: The **Login** page has two fields (One is **UserName** and the other is **Password**) and one (**Login**) button.

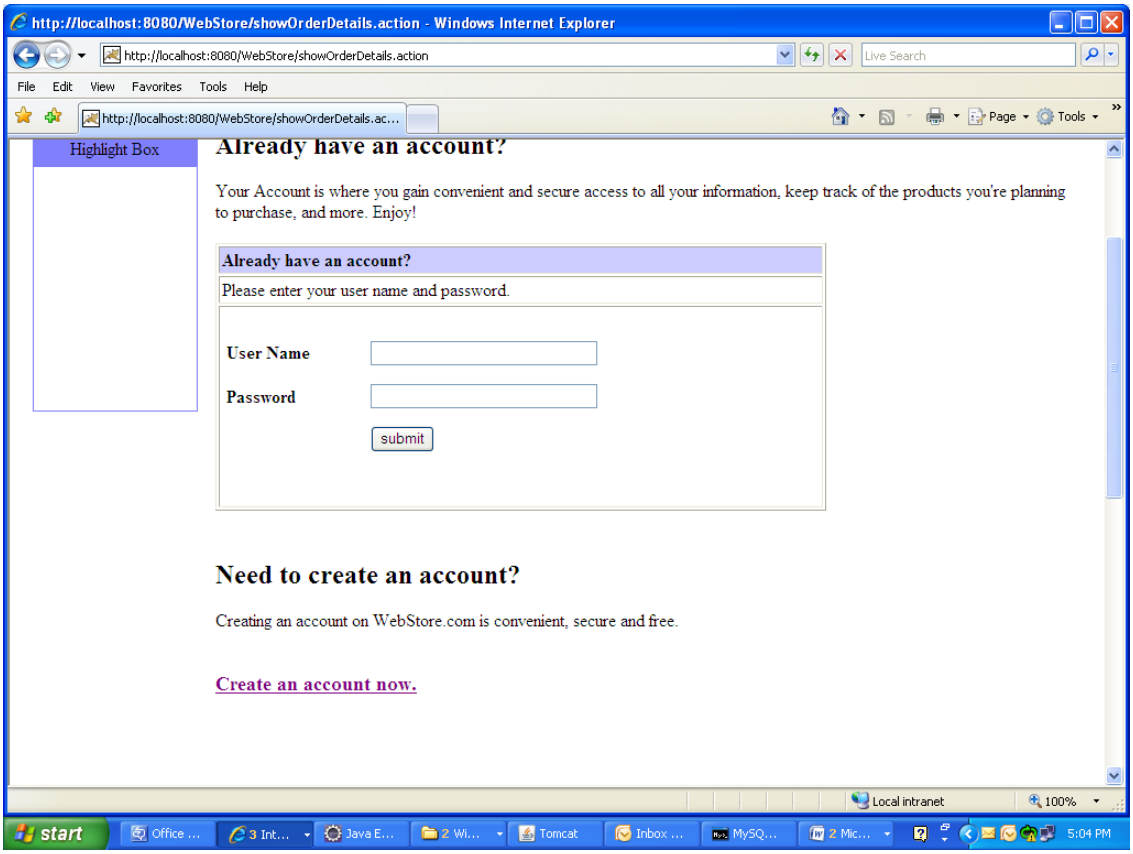
REQ-102: The **UserName** field is marked mandatory and the registered user must enter his/her Username in this field to login.

REQ-103: The **Password** field is marked mandatory and the registered user must enter his/her Password in this field to login.

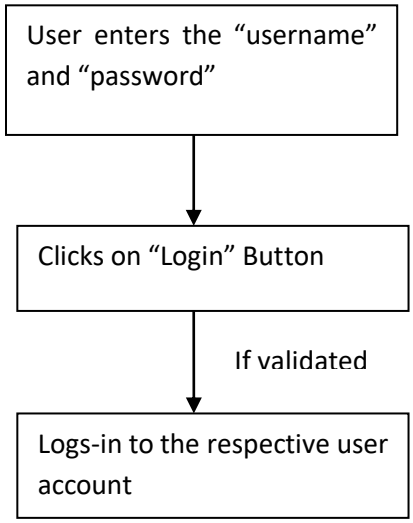
REQ-104: Once the registered user enters valid **UserName** and **Password** and clicks on **Login** button, the user will be logged in to his/her account and the user will be directed to **Home** page displaying the Username of the user on the top of the page.

REQ-105: The login screen provides few links to go to **Home** page, **Cart** page, **Login** page and **Help** page on top of the page.

2.2.1.2 Screen Prototype



2.2.1.3 Screen Flow / Business Flow & User Actions



2.2.1.4 Business Rules & Validations

The business rule to be considered while entering this screen is, the user must be connected to the website and the user must click on “Sign-In” link.

The business rule to be considered while moving to the next page is that the username and password should be entered and both the fields should be valid, If they are valid only, after clicking on the login button the user will be navigated to the next page.

2.2.2 Requirement / Module / Use-case for Registration

2.2.2.1 Description

This use-case describes the **Registration** page functionality.

REQ-201: New user can register by clicking on Create an account now. A registration page will be displayed with 13 fields and one button.

REQ-202: The **UserName** field is marked mandatory and this field will accept letters, digits and any other special symbols. This field accepts maximum of 20 characters only.

REQ-203: The **Password** field is marked mandatory and this should be at-least 6-8 characters long. It accepts alphabets, numerics and special symbols.

REQ-204: The **Confirm Password** field will be marked mandatory and this should match with the **Password**.

REQ-205: The **FirstName** field is marked mandatory and this field will accept only alphabets. This field accepts maximum of 20 characters only.

REQ-206: The **LastName** field is marked mandatory and this field will accept only alphabets. This field accepts maximum of 20 characters only.

REQ-207: The **Date of Birth** field is mandatory and this field accepts digits only.

REQ-208: The **Email-ID** field is mandatory and this field will accept alphabets, digits and two special characters (@ and dot) only.

REQ-209: The **Address** field is marked mandatory and this field accepts alphabets, digits, special symbols.

REQ-210: The **Phone Number** field is marked mandatory and this field will accept only digits excluding alphabets and any other special characters.

REQ-211: The **City** field is marked mandatory and this field will accept only alphabets.

REQ-212: The **State** field is marked mandatory and this field will accept only alphabets.

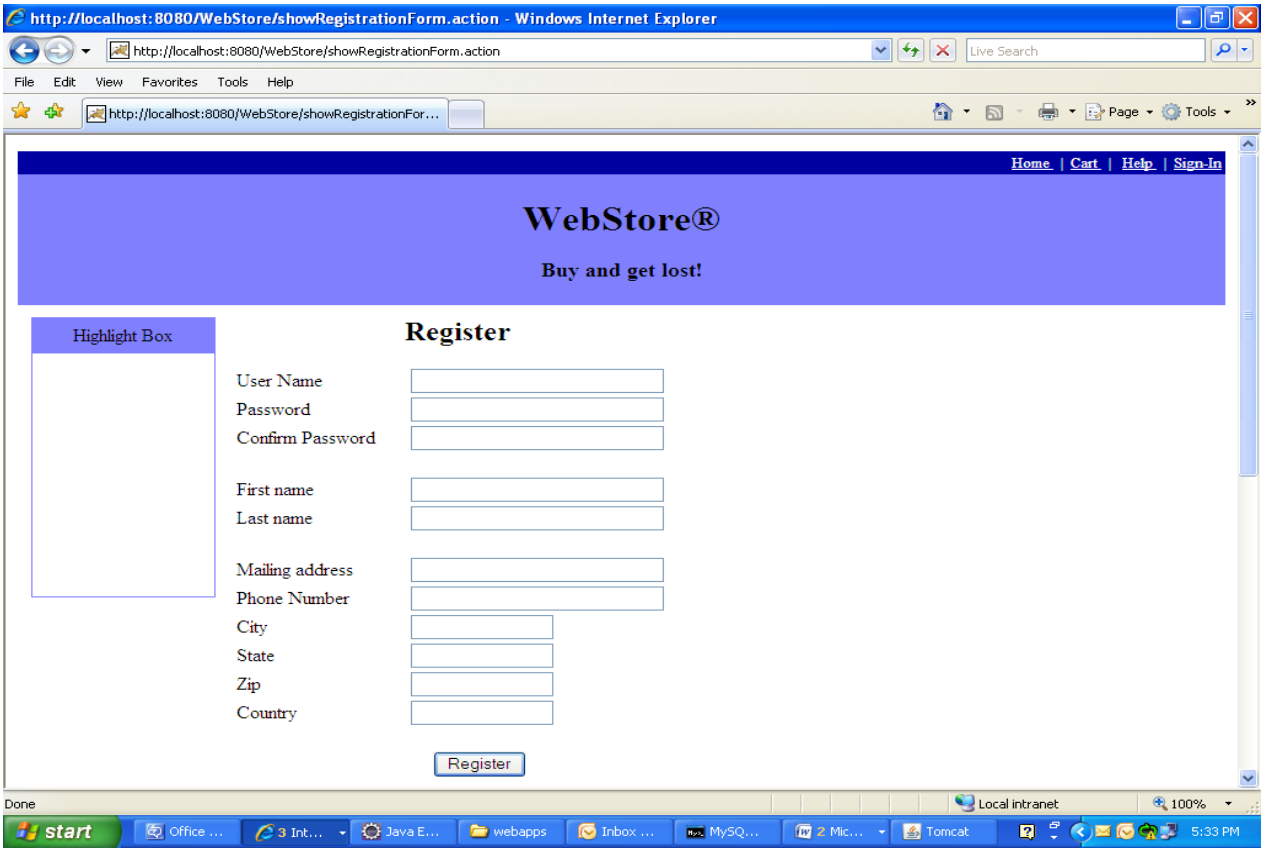
REQ-213: The **Zip** field is mandatory and this field will accept only numbers.

REQ-214: The **Country** field is mandatory and this field will accept only alphabets.

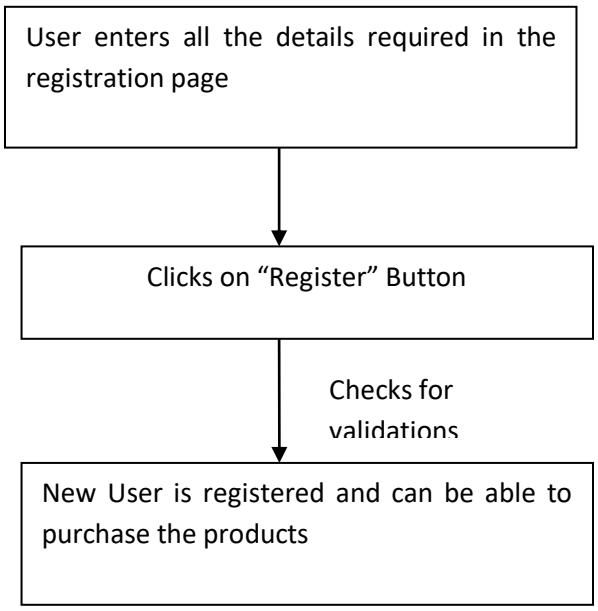
REQ-215: Once the user enters valid data in all the fields and clicks on **Register** button, the user will get registered with WebStore and user will be directed to **Home** page.

REQ-216: The **Registration** page also provides an option to **Login** for registered users using the link **Sign-In** provided at the top of screen and few links to go to **Home** page, **Cart** page and **Help** page etc.

2.2.2.2 Screen Prototype



2.2.2.3 Screen Flow / Business Flow & User Actions



2.2.2.4 Business Rules & Validations

The business rule to be considered while entering this page is the user must be connected to **Webstore** and the user must click on Sign-In link.

The business rule to be considered while moving to next page is that all the mandatory field values must be filled by the user and the data in all the fields should be valid.

2.2.3 Requirement / Module / Use-case for My Account

2.2.3.1 Description

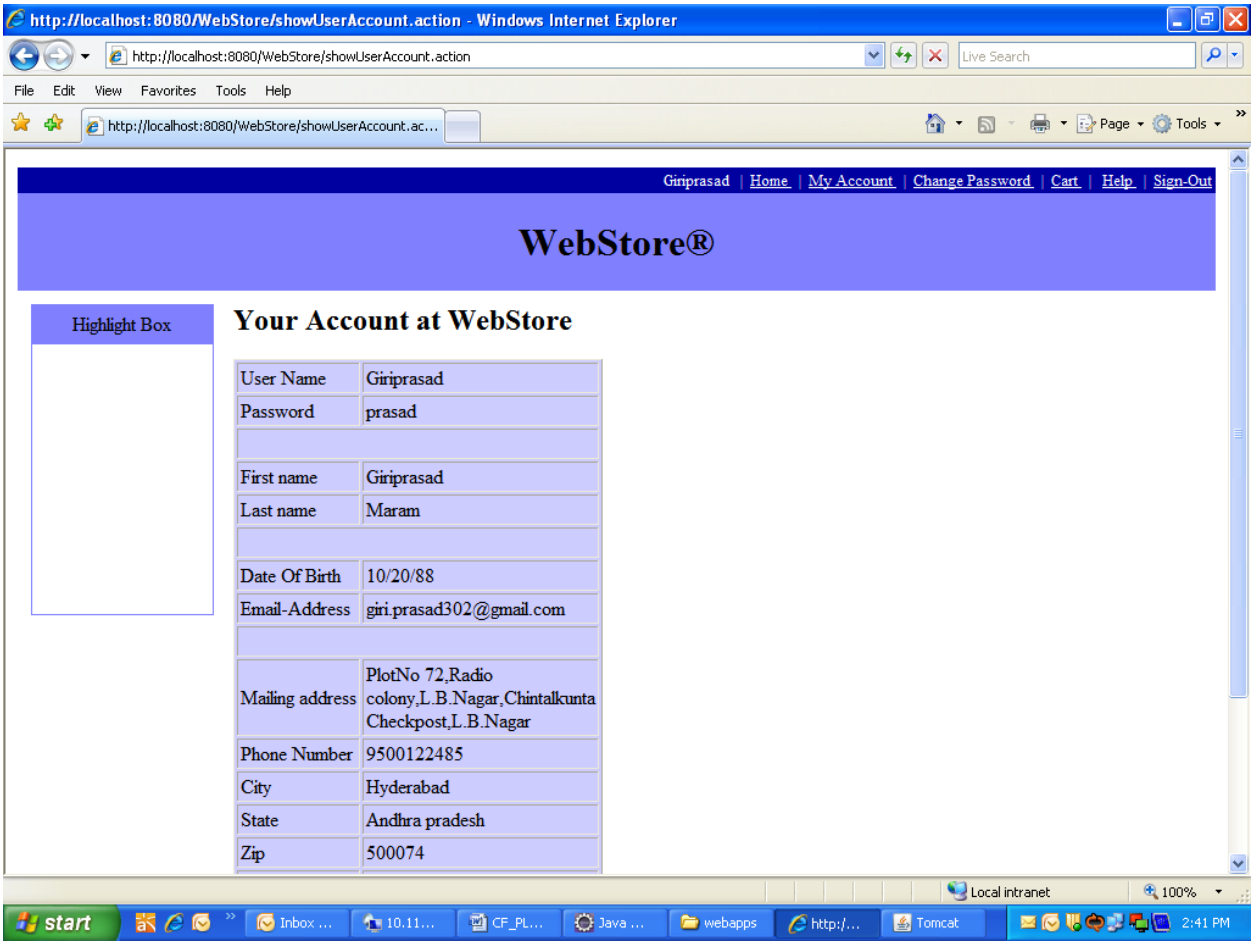
This use-case is describes how to navigate to Logged-in User's profile page.

REQ-301: Once the user Logs-in to the web store, he/she can view his/her profile by clicking on **My Account** link. When User clicks on **My Account** link, the User Information page will be displayed showing all the details of the user which he enters at the time of registration.

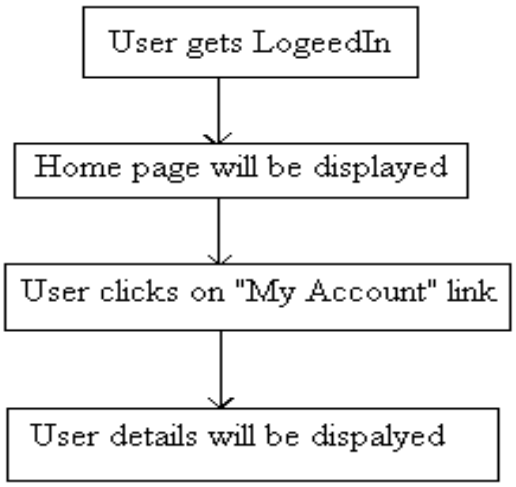
REQ-302: The User Information page provides one **Update** icon at the bottom of the page to change/update his/her profile/details.

REQ-303: This page provides few links to **Sign-out** from his/her account and to go to **Home** page, **Cart** page and **Help** page provided at the top of screen. This page also provides the flexibility for the user to change his/her **Password** through the link **Change Password**.

2.2.3.2 Screen Prototype



2.2.3.3 Screen Flow / Business Flow & User Actions



2.2.3.4 Business Rules & Validations

The business rule to be considered while entering this page is the user must Login. The business rule to be considered while moving to the next page is, to make sure that the user clicked on **My Account** link.

2.2.4 Requirement / Module / Use-case for Home

2.2.4.1 Description

This use case describes about all the available products in the portal.

REQ 401: User clicks on **Home** link and a page will be displayed showing all the available **Catalogs** in the portal along with the images.

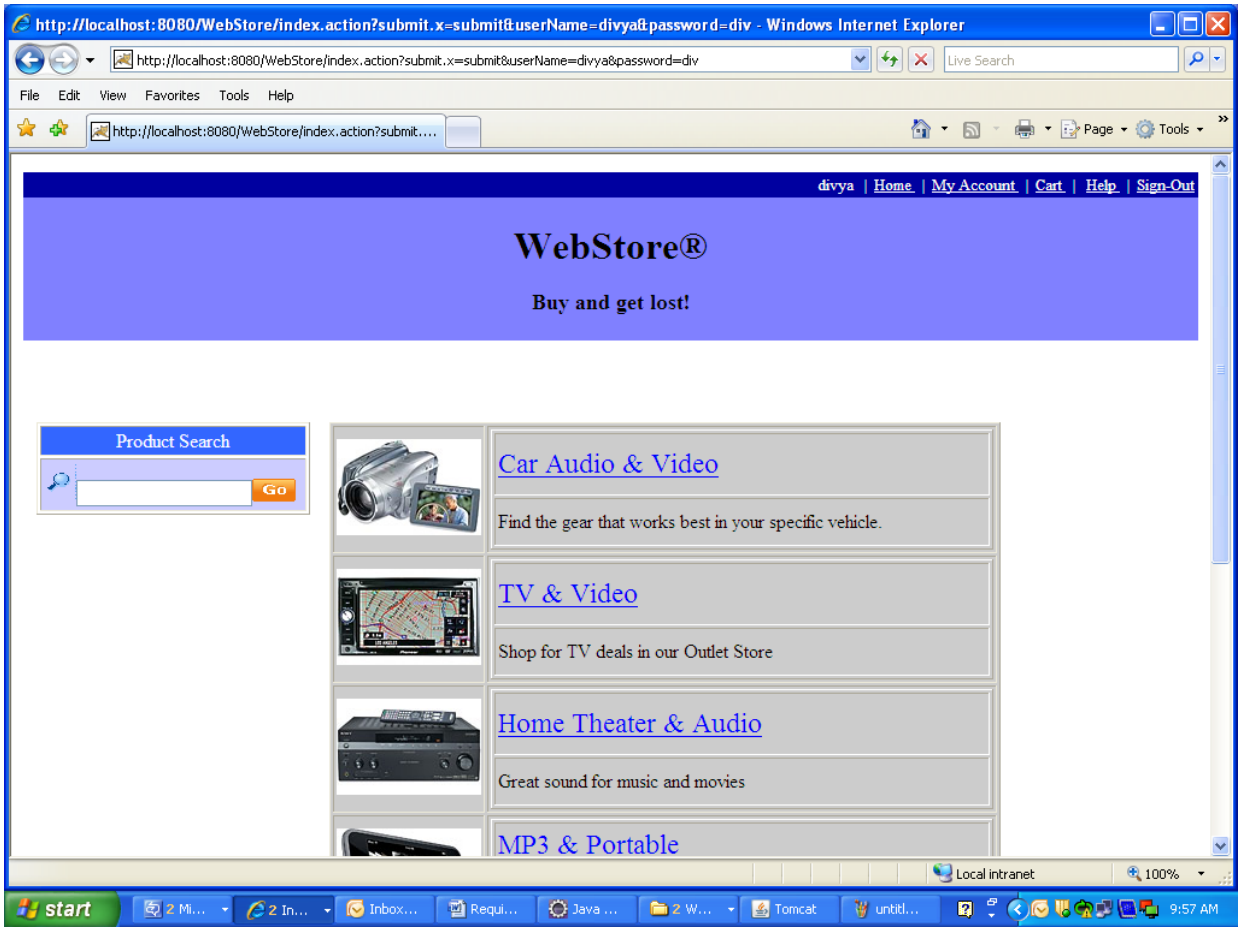
REQ-402: User selects any **Catalog** by clicking on a particular catalog Image/name and all the available **Categories** under the selected catalog will be displayed along with images.

REQ-403: User selects a category from the resulting categories list by Clicking on that particular category image/name and all the **products/items** Under the selected category will be displayed along with the images, if there are any products/items available in that category.

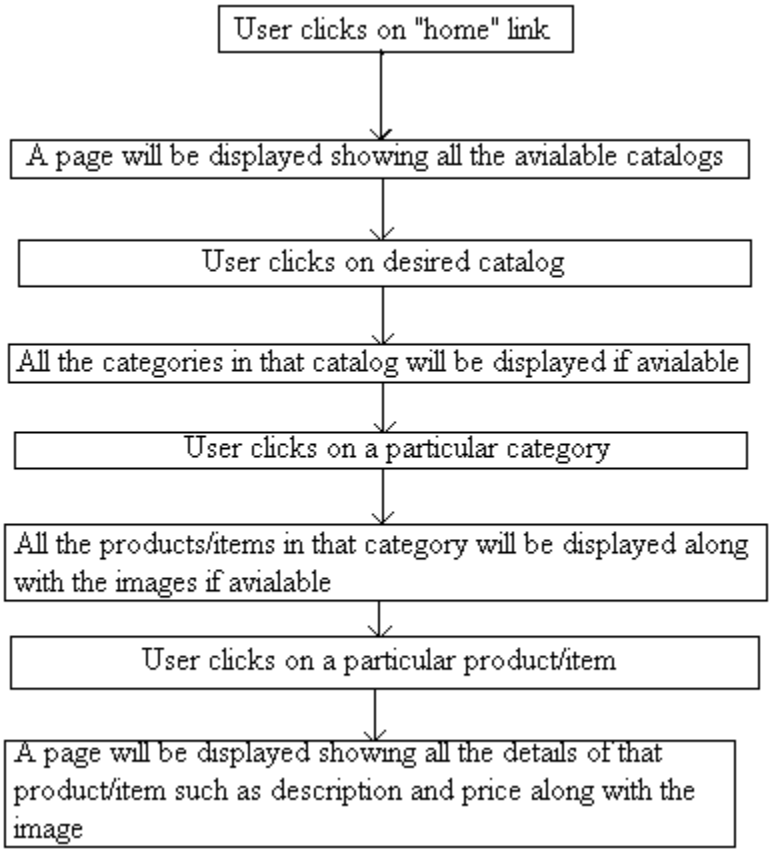
REQ-404: User selects a **product/item** from the resulting products/items List just by clicking on the image/name of that particular product/item. The product/item will be displayed along with the image, description and price

REQ-405: Home page provides few links to navigate to **Home** page, **Cart** page, **Login** page etc.

2.2.4.2 Screen Prototype



2.2.4.3 Screen Flow / Business Flow & User Actions



Business Rules & Validations

The business rule to be considered while entering this screen is the user must click on **Home** link.

The business rule to be considered while navigating to the next page is the user must select at least one of the available catalogs.

2.2.5 Requirement / Module / Use-case for Adding to the cart

2.2.5.1 Description

This use case describes adding a product/item to the cart.

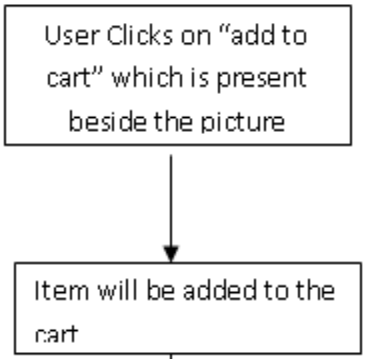
REQ-501: Once the user selects a product/item by clicking on that particular product/item, he/she can add them to the cart by clicking on **Add to cart** icon (which will be present beside that product/item) and after the product is added to cart, the cart page will be displayed showing the details of products/items that are added to cart.

REQ-502: User can get his/her Cart after logging in also, if he adds items to Cart before logging in. But the user cannot get his/her Cart if he logs in after he/she logs out.

REQ-503: User navigates to any page by clicking on any one of the links provided on top of the Cart page and user can navigate to Home page by clicking on **Keep Shopping** icon in cart page.

2.2.5.2 Screen Prototype

2.2.5.3 Screen Flow / Business Flow & User Actions



2.2.5.4 Business Rules & Validations

The business rule to be considered while entering this screen is, the user (registered/unregistered) must select a category.

The business rule to be considered while navigating to next page is, the user (registered/unregistered) must add a product to cart.

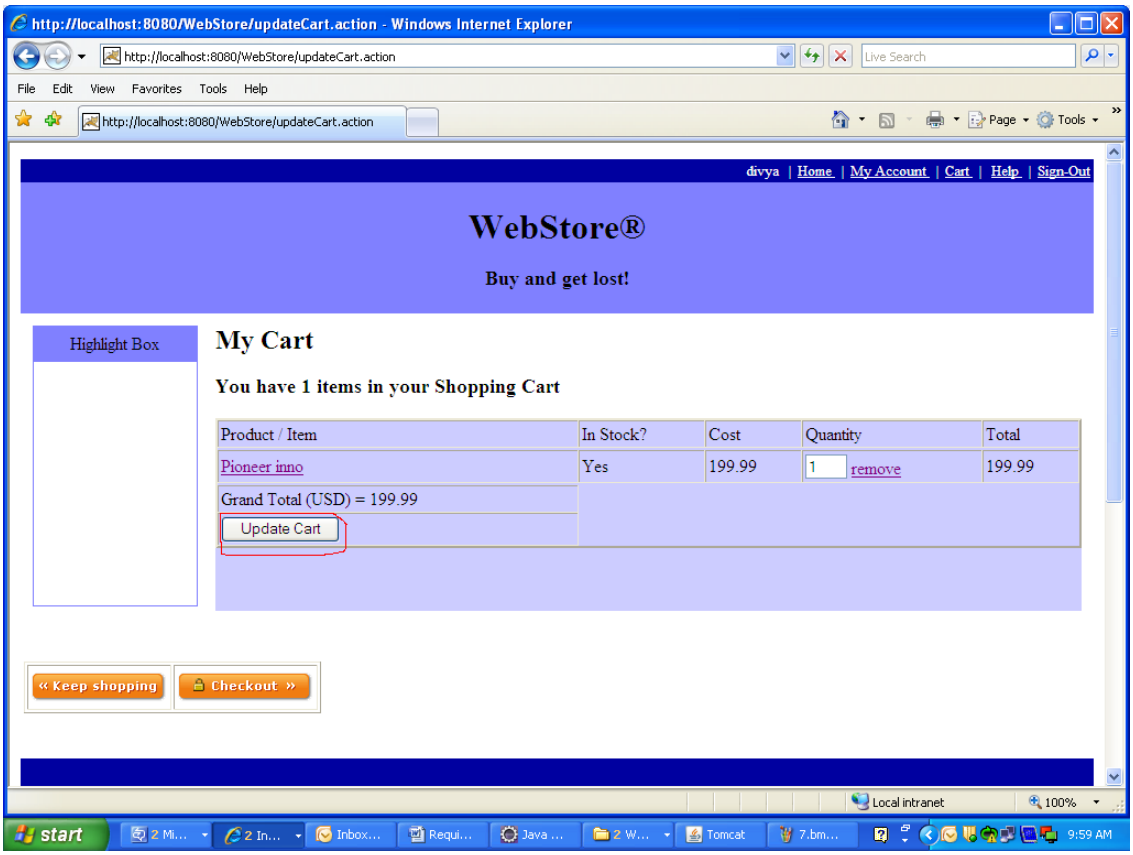
2.2.6 Requirement / Module / Use-case for updating the cart

2.2.6.1 Description

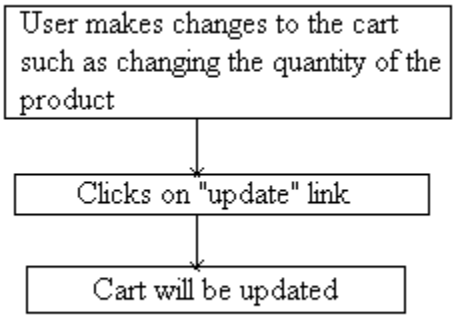
This use case describes updating the existing cart.

REQ-601: Once the user adds the products to cart, he/she can update the cart (if there are any changes made to cart) by clicking on **Update** link. If the user changed the quantity of that particular product/item by changing the value in the Quantity field and if the user clicks on update then the cart will be updated with the new quantity which has been set.

2.2.6.2 Screen Prototype



2.2.6.3 Screen Flow / Business Flow & User Actions



2.2.6.4 Business Rules & Validations

The business rule to be considered while entering this screen is that the user must add at least one item to the cart.

The business rule to be considered when navigating to next page is the user must update the cart.

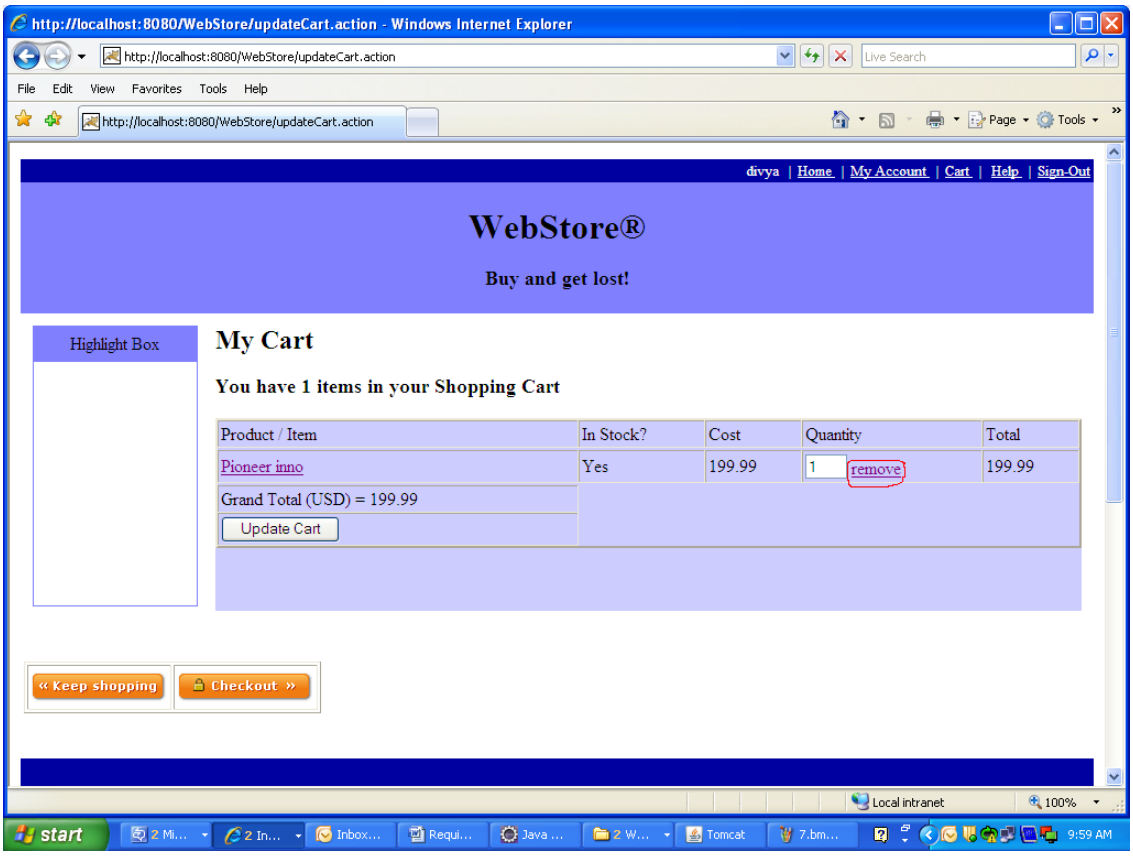
2.2.7 Requirement / Module / Use-case for removing from the cart

2.2.7.1 Description

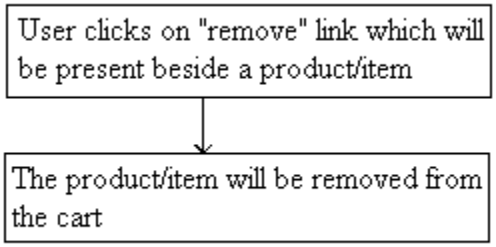
This use case describes removing the existing products/items from the cart.

REQ-701: Once the user adds the products to cart, he/she can remove those products/items from the cart by clicking on **Remove** link.

2.2.7.2 Screen Prototype



2.2.7.3 Screen Flow / Business Flow & User Actions



2.2.7.4 Business Rules & Validations

The business rule to be considered while entering this screen is that the user must add at least one item to cart.

The business rule to be considered when navigating to next page is the user must remove at least one product/item from the cart.

2.2.8 Requirement / Module / Use-case for order

2.2.8.1 Description

This use case describes how to **order/purchase** the products after adding them to cart.

REQ-801: User clicks on **Checkout** icon to order/purchase the products and billing page is displayed with 13 fields and one button.

REQ-802: The first one is the **Shipping Address** field and it is marked mandatory. It accepts alphabets, digits and special symbols.

REQ-803: The next field is the **Phone Number** and it is mandatory. It accepts only digits.

REQ-804: The next is the name of the **City** and it is mandatory. It accepts only alphabets.

REQ-805: The next field is the **State** name and it is mandatory. It accepts only alphabets.

REQ-806: The next is the pin code (**zip**) of the City and it is mandatory. It accepts only digits.

REQ-807: The next field is the name of the **Country** and it is mandatory. It accepts only alphabets.

REQ-808: The next is the **Type** of credit card used to pay the price of the product/item purchased and it is mandatory. It accepts only alphabets.

REQ-809: The next field is **Card Number** and it is mandatory. It accepts only digits.

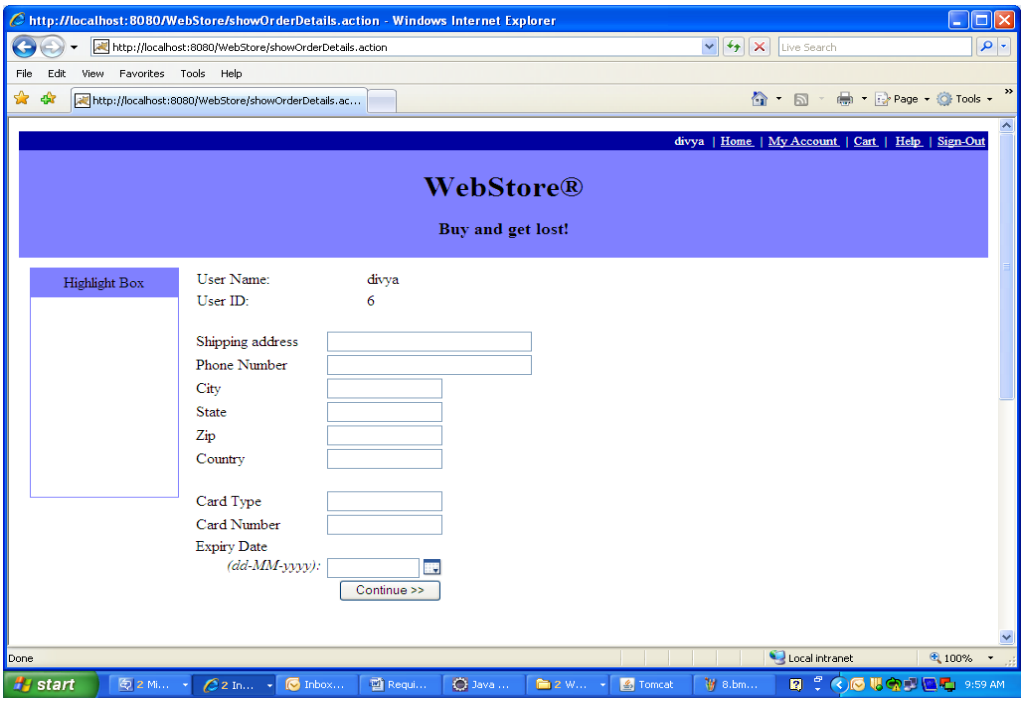
REQ-810: The next field is the **Expiry Date** of the card which the user used to pay the price of the item to be purchased and it is mandatory. It accepts only digits and it does not accept special symbols other than hyphen (-).

REQ-811: Once the user enters valid address, city, state, country, zip, card details and clicks on **Continue** button, a (**Order**) page will be displayed showing all the details about the order and the user will be provided with a unique order id to track his order.

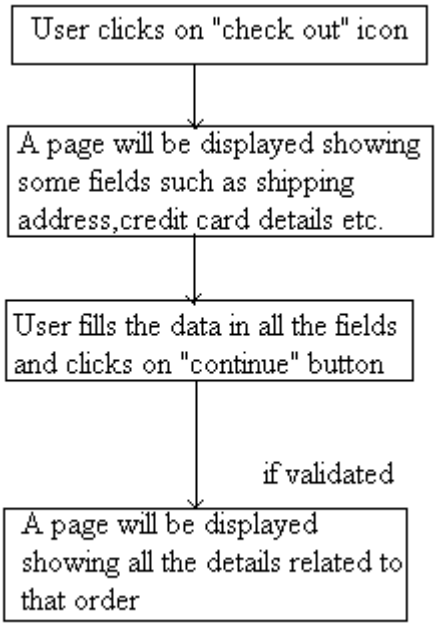
REQ-812: User will be provided with few links to navigate to **Home** page, **Help** page, **Cart** page, Profile page (by clicking on **My Account link)** and two more links to **Sign-Out** from his/her account and to change his/her **Password**.

If user wants to make one more order then he/she can do so by clicking on **Home** icon provided at the bottom of this page.

2.2.8.2 Screen Prototype



2.2.8.3 Screen Flow / Business Flow & User Actions



2.2.8.4 Business Rules & Validations

The business rule to be considered while entering this page is the user must click on **Check out** icon.

The business rule to be considered while navigating to next page is the user must provide valid data in all the fields and the user must click on **Continue** button.

2.2.9 Requirement / Module / Use case for help

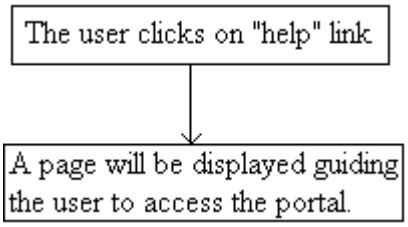
2.2.9.1 Description

This use case is used to guide a new user.

REQ-901: If a new user visits the website then he will be guided using this use case. When the user clicks on **Help** link, a page will be displayed guiding the user how to login to and register with **Web store**.

2.2.9.2 Screen Prototype

2.2.9.3 Screen Flow / Business Flow & User Actions



2.2.9.4 Business Rules & Validations

The business rule to be considered while entering this page is, the user must be able to connect to **Web store** and the user must click on **Help** link.

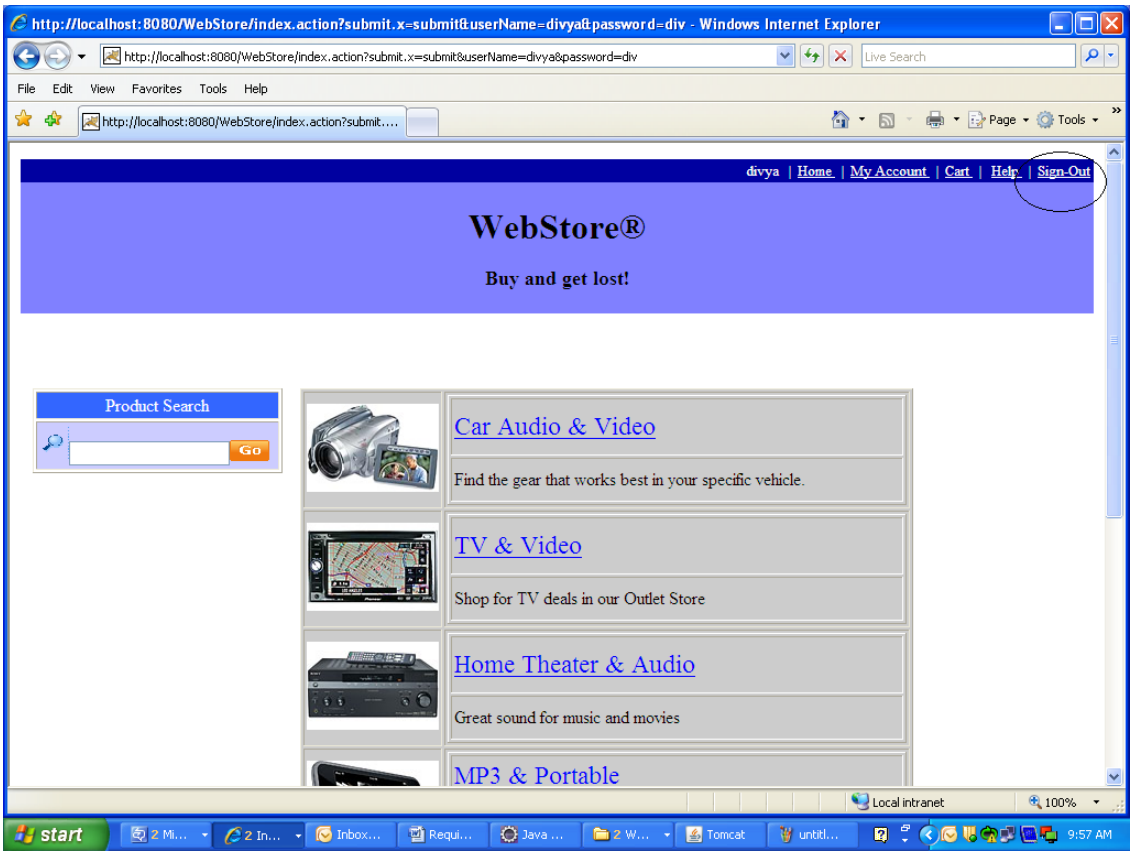
2.2.10 Requirement / Module / Use case for logout

2.2.10.1 Description

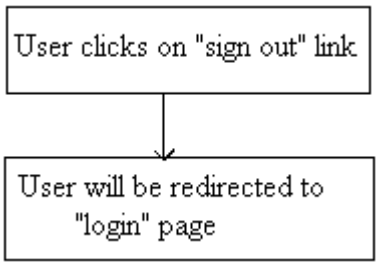
This use case describes the logout action.

REQ-1001: This use case describes about the security reasons so that no other unauthorized user can access the previous user account. After the user clicks on **Sign-Out** link, the user will be directed to **Login** page and the previous user's session will be set to null.

2.2.10.2 Screen Prototype



2.2.10.3 Screen Flow / Business Flow & User Actions



2.2.10.4 Business Rules & Validations

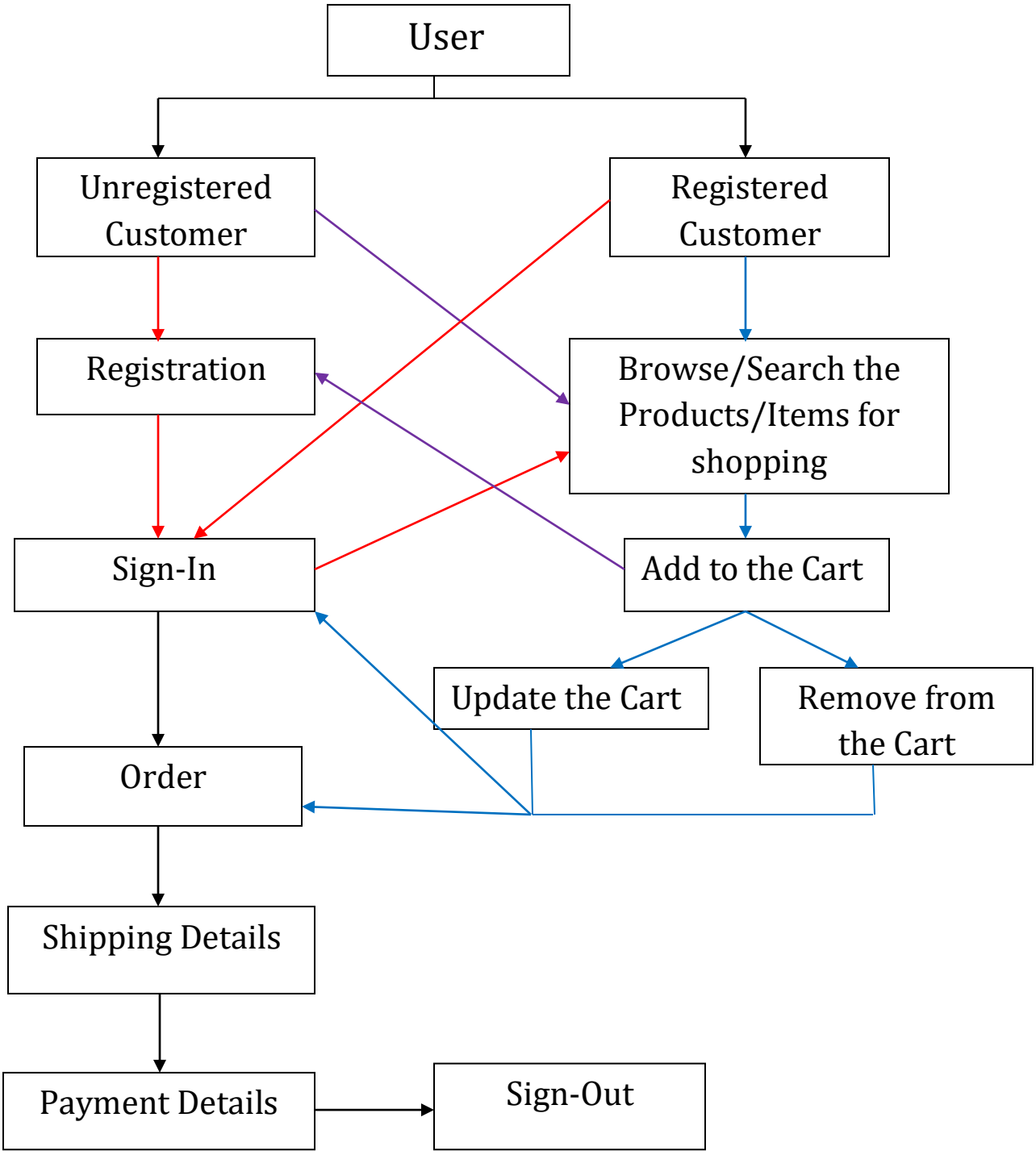
The business rule to be considered while entering this page is the user must be logged in to his account.

2.2.11 Other Requirements

REQ-1101	Webstore provides **24/7** technical support.
REQ-1102	**Product images** are good representations of what Webstore is offering.
REQ-1103	**Home page** provides an overview of key products that Webstore is offering to buy.
REQ-1104	User must have a **bank account and credit card** inorder to purchase products.
REQ-1105	Provides **safety** by hiding user's details at the time of Registration and Login and Paying price for the product.
REQ-1106	System should not **crash** when multiple users are accessing Webstore at a time.
REQ-1107	**MySql data server** is required.
REQ-1108	In order to access the Web store the user must be able to have access to a **computer,** which has an internet connection
REQ-1109	The **response time** of the system is approximately 1-2 seconds.
REQ-1110	Provides the access to the web portal **at all times.**

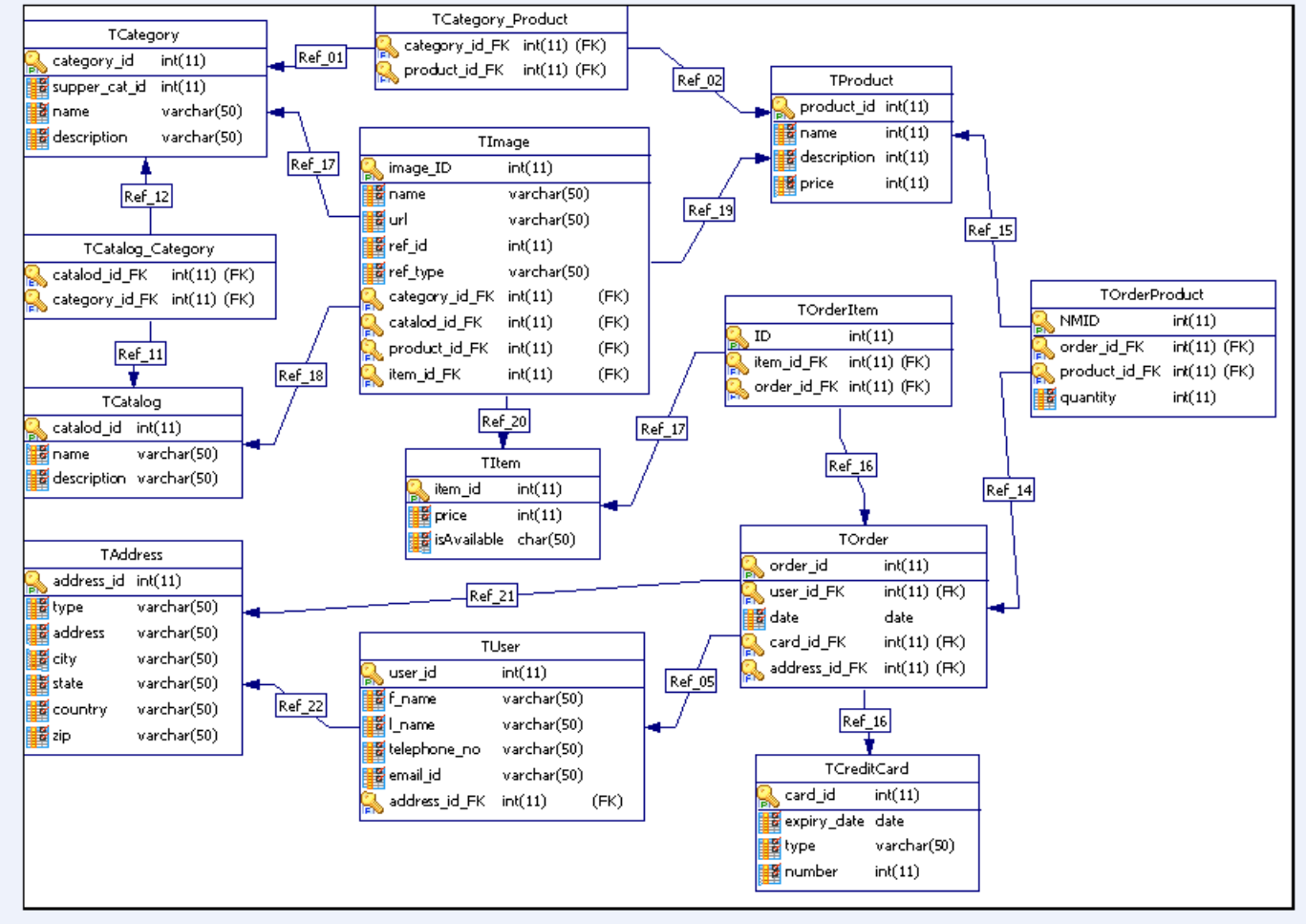
3 Design

3.1 Architecture Diagram

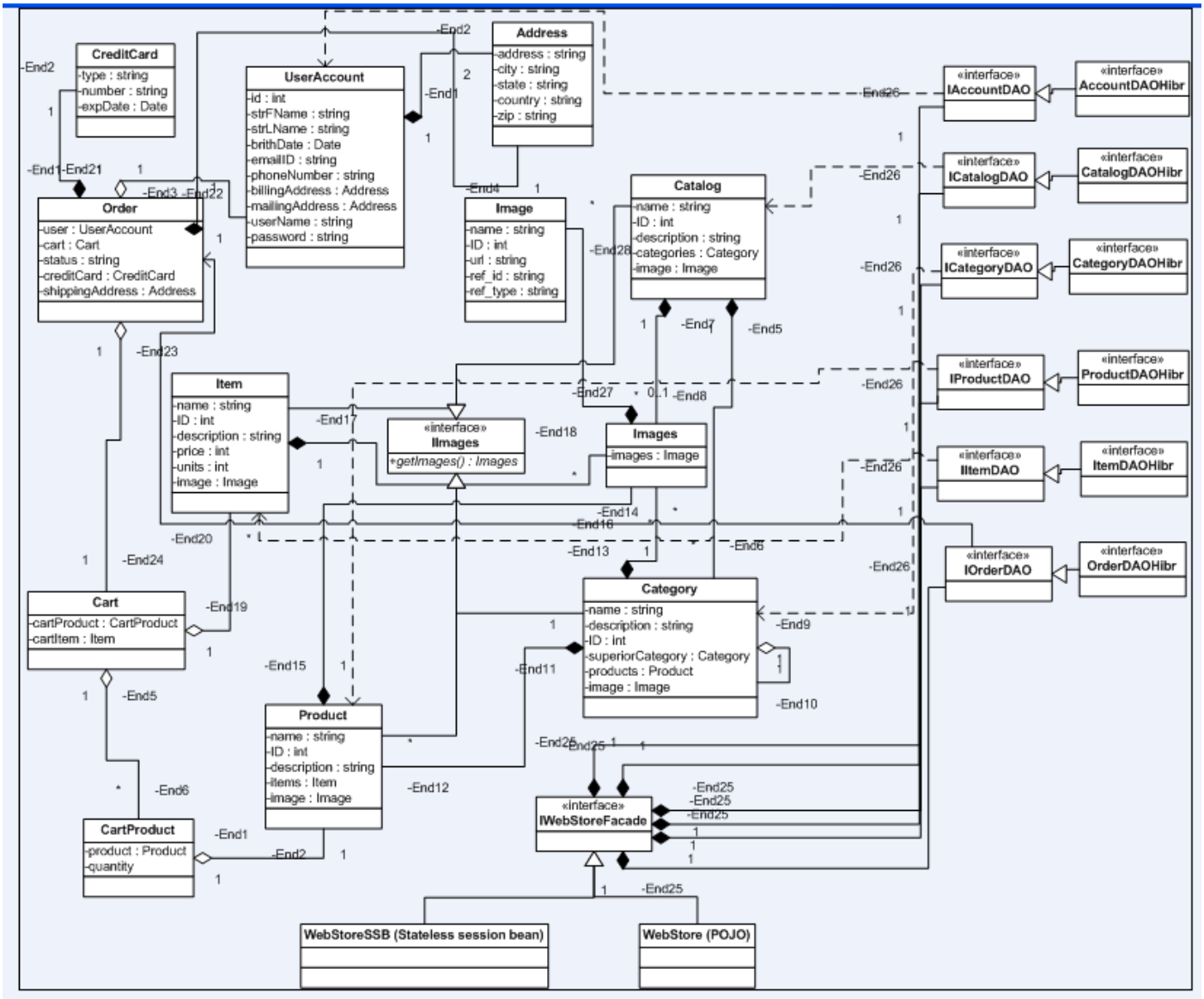


3.2 Database Design

3.2.1 E-R Diagram



3.2.2 Dataflow Diagram



3.2.3 Table Structure

Table Name: Address

Primary Key: ADDRESS_ID

Description: It contains the residence/shipping address of the user.

Column Name	Data Type	Description	Key Type
ADDRESS_ID	bigint(20)	Address ID	Primary key
ADDRESS	varchar(255)	Locality of the user	-
CITY	varchar(255)	City name	-
STATE	varchar(255)	State name	-
COUNTRY	varchar(255)	Country name	-
ZIP	varchar(255)	Pin code	-
TYPE	varchar(255)	Type of the Address	-
PHONE_NUM	varchar(255)	Contact Number	-

Table Name: Catalog

Primary Key: CATALOG_ID

Description: It contains the details of products.

Column Name	Data Type	Description	Key type
CATALOG_ID	Integer	Catalog ID	Primary key
Name	varchar[255]	Catalog Name	-

Description	Integer	Description of the Catalog	-

Table Name: Catalog_Category

Primary Key: CATALOG_ID, CATEGORY_ID

Description: It contains details of categories for each catalog.

Column Name	Data Type	Description	Key type
CATALOG_ID	bigint(20)	ID of catalog	Primary key
CATEGORY_ID	bigint(20)	ID of category	Primary key

Table Name: Category

Primary Key: CATEGORY_ID

Description: It contains the details of categories and the subcategories.

Column Name	Data Type	Description	Key type
CATEGORY_ID	bigint(20)	ID of category	Primary key

Name	Varchar(255)	Name of the category	-
Description	Varchar(255)	Description of the category	-
SUP_ID	bigint(20)	ID of sub category	Foreign key

Table Name: Category_product

Primary Key: CATEGORY_ID,PRODUCT_ID

Description: It contains the details of products under each category.

Column Name	**Data Type**	**Description**	**Key type**
CATEGORY_ID	bigint(20)	ID of category	Primary key
PRODUCT_ID	bigint(20)	ID of product	Primary key

Table Name: Creditcard

Primary Key: CARD_ID

Description: It contains details of credit card used for payment.

Column Name	Data Type	Description	Key type
CARD_ID	bigint(20)	ID of creditcard	Primary key
CARD_NUMBER	Varchar(255)	Credit card number	-
CARD_TYPE	Varchar(255)	Credit card type	-
EXP_DATE	Date	Date of expiry	-

Table Name: Image

Primary Key: IMAGE_ID

Description: It contains THE details of images of catalogs, categories, items & products.

Column Name	Data Type	Description	Key type
IMAGE_ID	bigint(20)	ID of image	Primary key
Name	Varchar(255)	Name of the image	-
PATH	Varchar(255)	Location of image	-
ITEM_ID	bigint(20)	ID of the item	Foreign key

PRODUCT_ID	bigint(20)	ID of the product	Foreign key
CATEGORY_ID	bigint(20)	ID of the category	Foreign key
CATALOG_ID	bigint(20)	ID of catalog	Foreign key

Table Name: Order_item

Primary Key: ORDER_ID, O_I_ID

Description: It contains details of the order, items/products which are ordered.

Column Name	**Data Type**	**Description**	**Key type**
ORDER_ID	bigint(20)	ID of order	Primary key
ITEM_ID	bigint(20)	Item ID	unique
O_I_ID	Integer	Ordered Item ID	Primary key

Table Name: Item

Primary Key: ITEM_ID

Description: It contains the details of items such as description, price etc.

Column Name	Data Type	Description	Key type
ITEM_ID	bigint(20)	ID of Item	Primary key
ITEM_NAME	varchar(255)	Name of the item	-
DESCRIPTION	varchar(255)	Description of the item	-
PRICE	double	Price of the item	-
AVAILABLE	bit(1)	Status of the item (available or not)	-
PRODUCT_ID	bigint(20)	ID of product	Foreign key

Table Name: Order_product

Primary Key: ORDER_ID, O_P_ID

Description: It contains details of products which are ordered

Column Name	Data Type	Description	Key type
ORDER_ID	bigint(20)	ID of order	Primary key

quantity	bigint(20)	Quantity of the product ordered	-
PRODUCT_ID	bigint(20)	ID of product	Foreign key
O_P_ID	Integer	ID of ordered product	Primary key

Table Name: Product

Primary Key: PRODUCT_ID

Description: It contains details of products such as description, price etc.

Column Name	**Data Type**	**Description**	**Key type**
PRODUCT_ID	bigint(20)	ID of subcategory	Primary key
name	varchar(255)	Name of the product	-
description	varchar(255)	description of the product	-
cost	double	Cost of the product	-
units	bigint(20)	No of units	-

Table Name: Torder

Primary Key: ORDER_ID

Description: It contains details of ordered items/products.

Column Name	Data Type	Description	Key type
ORDER_ID	bigint(20)	ID of order	Primary key
orderStatus	varchar(255)	Status of the order	-
date	Date	Date on which the Order is made	-
USER_ID	bigint(20)	ID of user	Foreign key
CREDIT_CARD_ID	bigint(20)	Id of creditcard	unique
ADDRESS_ID	bigint(20)	ID of user address	unique

Table Name: User_account

Primary Key: USER_ID

Description: It contains the details of user .

Column Name	Data Type	Description	Key type
USER_ID	bigint(20)	ID of user	Primary key
FNAME	varchar(255)	FirstName	-
LNAME	varchar(255)	LastName	-
DOB	dateTime	Date of birth	-
EMAIL_ID	varchar(255)	Email id	-
USER_NAME	varchar(255)	Name of the user for logging in	unique
PASSWORD	varchar(255)	Password for logging in	-
ADDRESS_ID	bigint(20)	ID of Address	unique

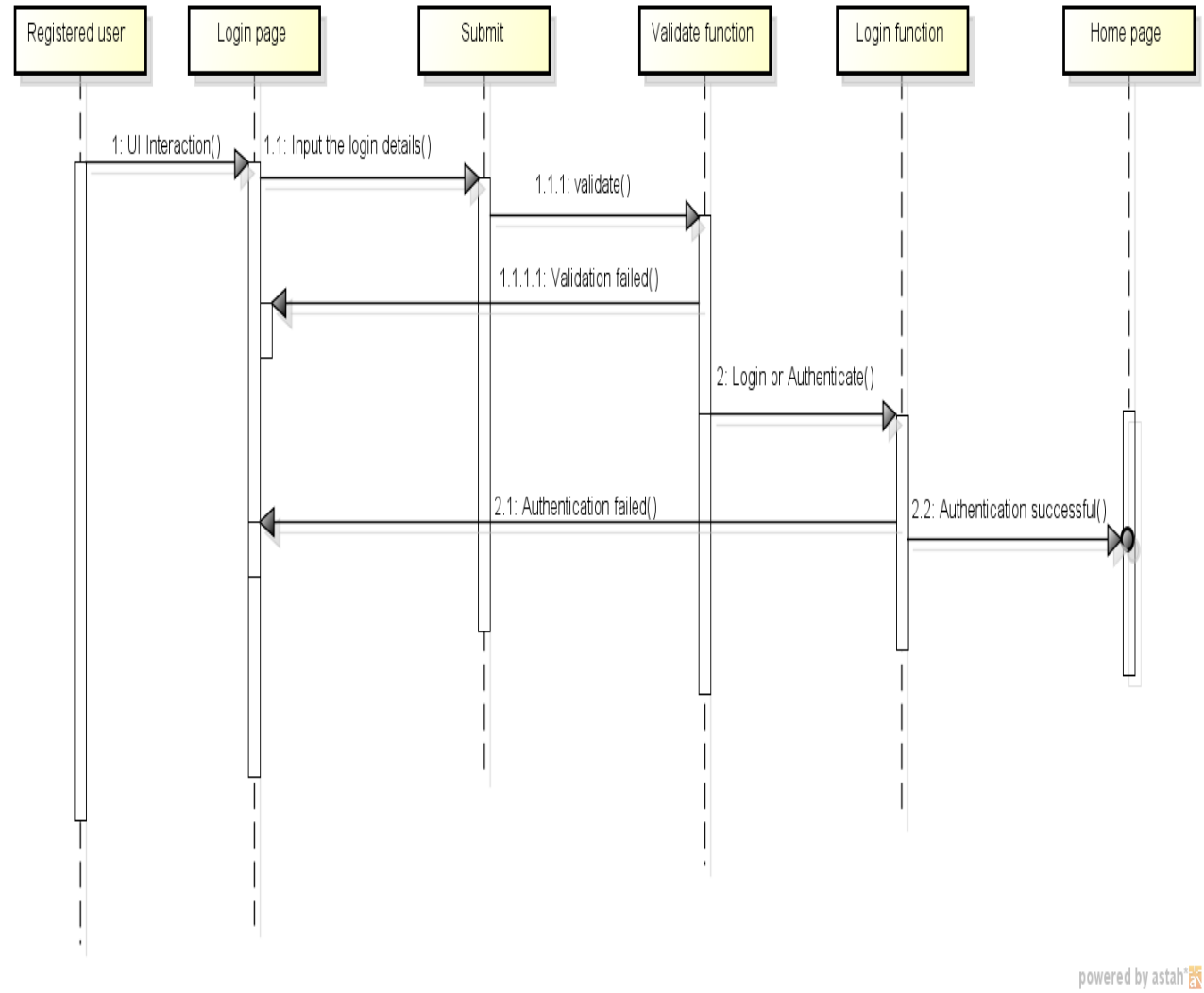
3.3 Detail Design

3.3.1 Requirement / Module / Use-case for Log-in

3.3.1.1 Class Diagram

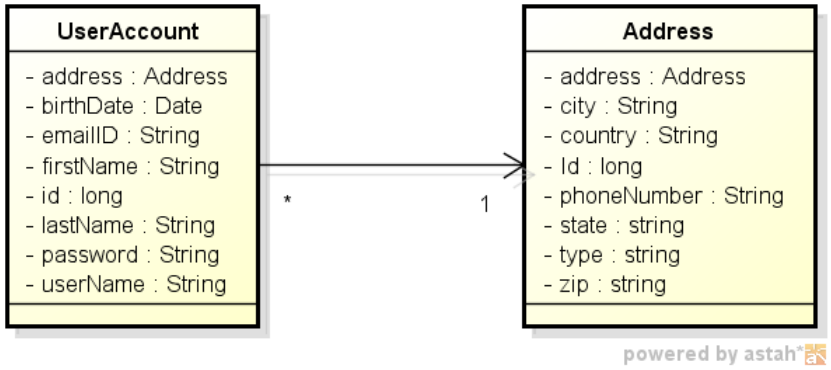
NA

3.3.1.2 Sequence Diagram / Flowchart

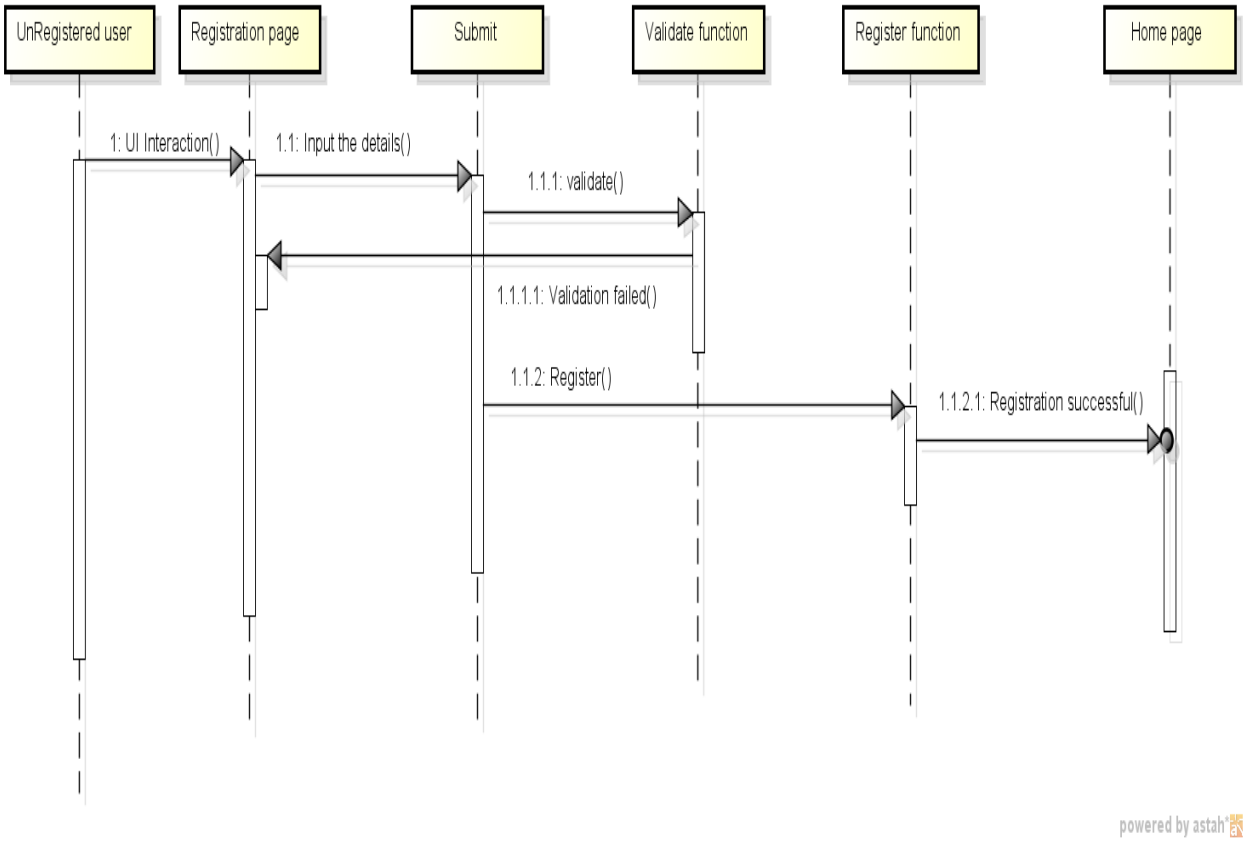


3.3.2 Requirement / Module / Use case for Registration

3.3.2.1 Class Diagram



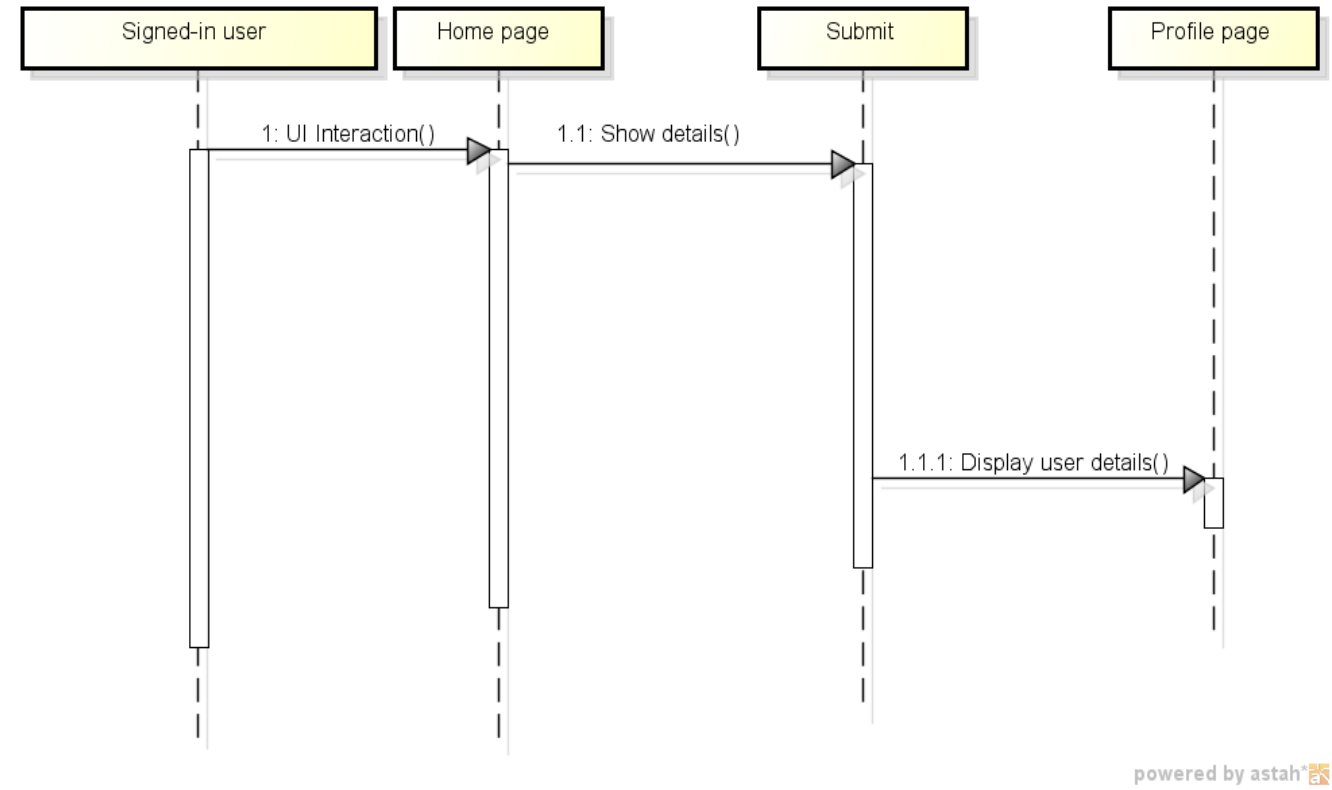
3.3.2.2 Sequence Diagram / Flowchart



3.3.3 Requirement / Module / Use case for My Account

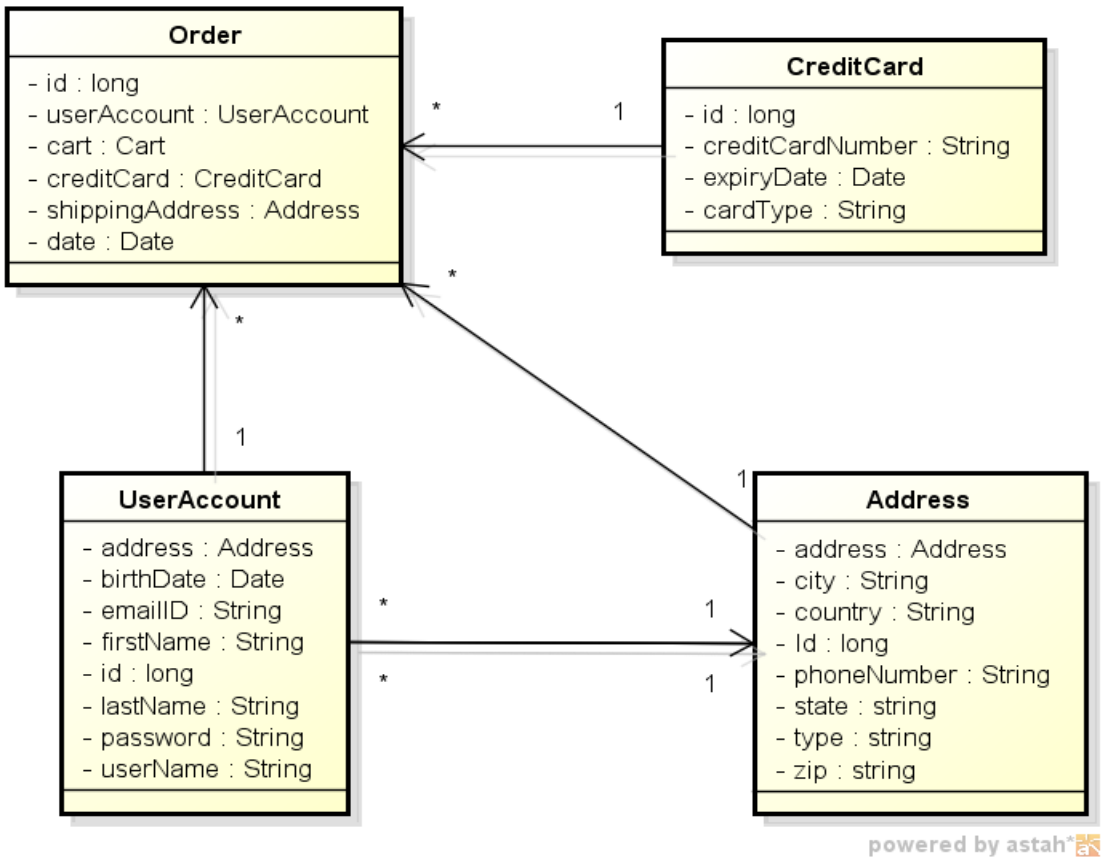
3.3.3.1 Class Diagram

3.3.3.2 Sequence Diagram / Flowchart

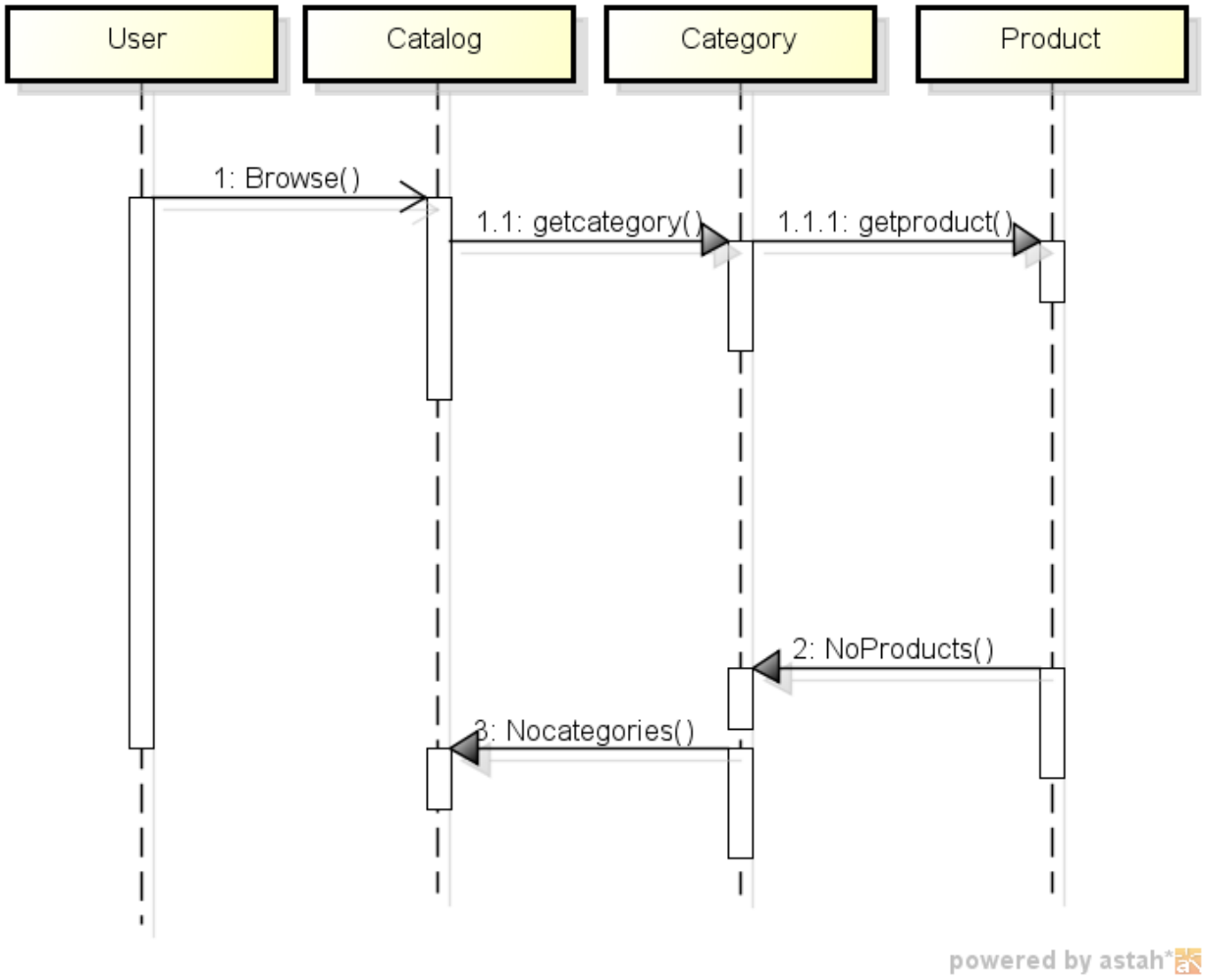


3.3.4 Requirement / Module / Use case for Home

3.3.4.1 Class Diagram



3.3.4.2 Sequence Diagram / Flowchart

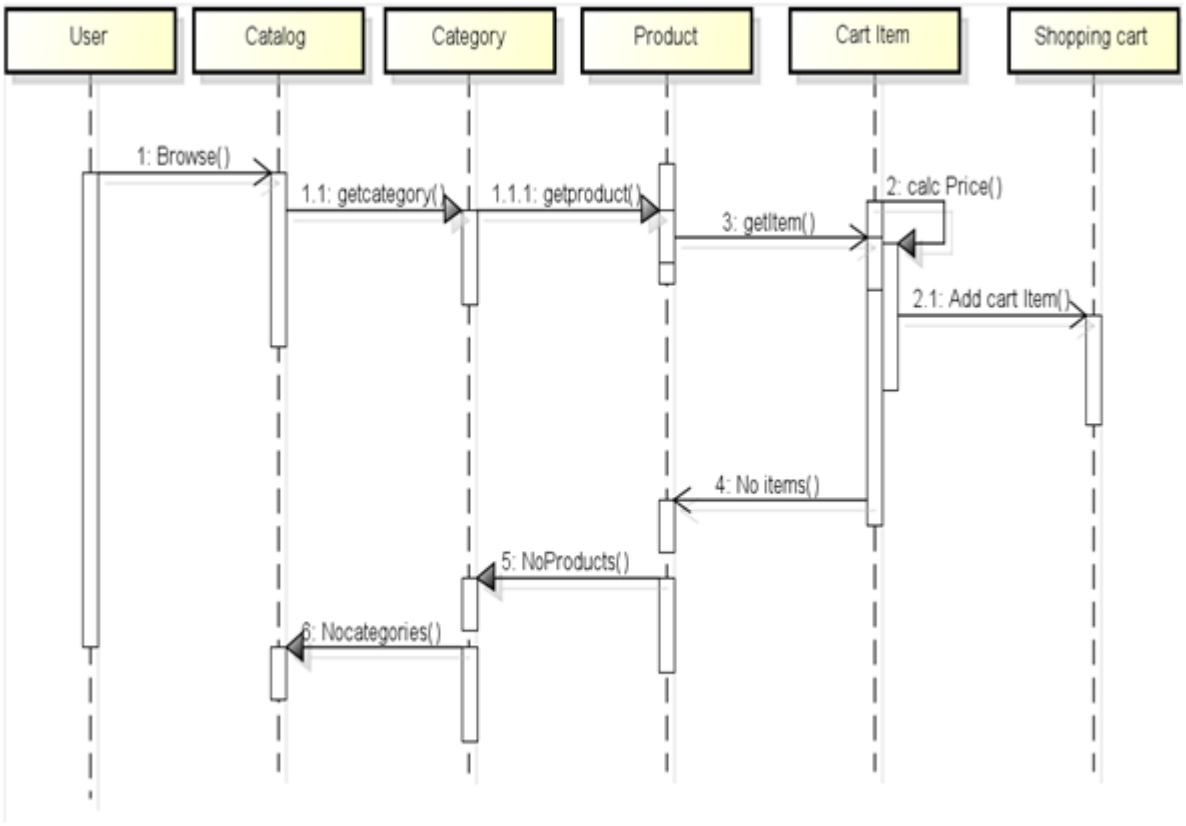


3.3.5 Requirement / Module / Use case for Add to the cart

3.3.5.1 Class Diagram

NA

3.3.5.2 Sequence Diagram / Flowchart

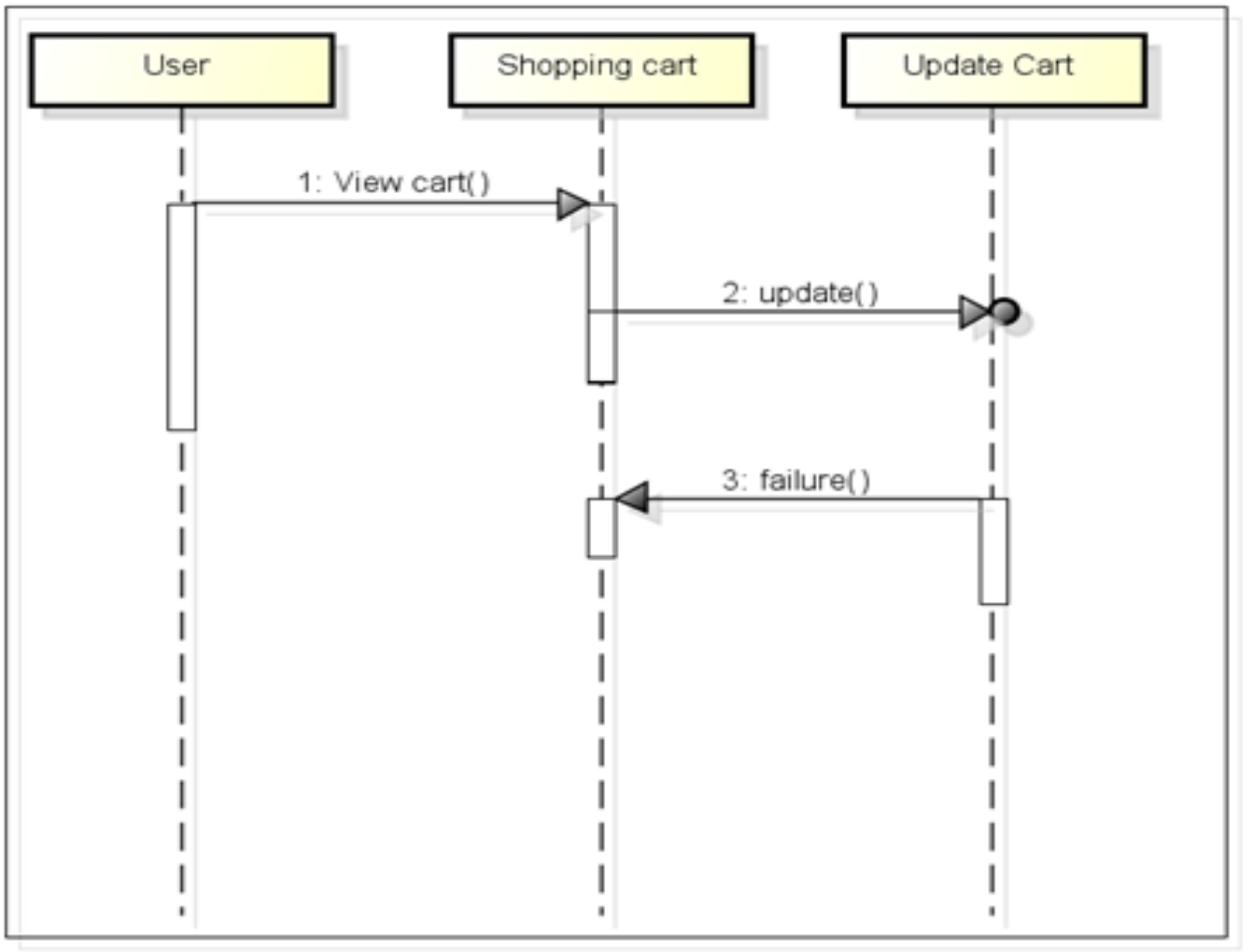


3.3.6 Requirement / Module / Use case for Update the cart

3.3.6.1 Class Diagram

NA

3.3.6.2 Sequence Diagram / Flowchart

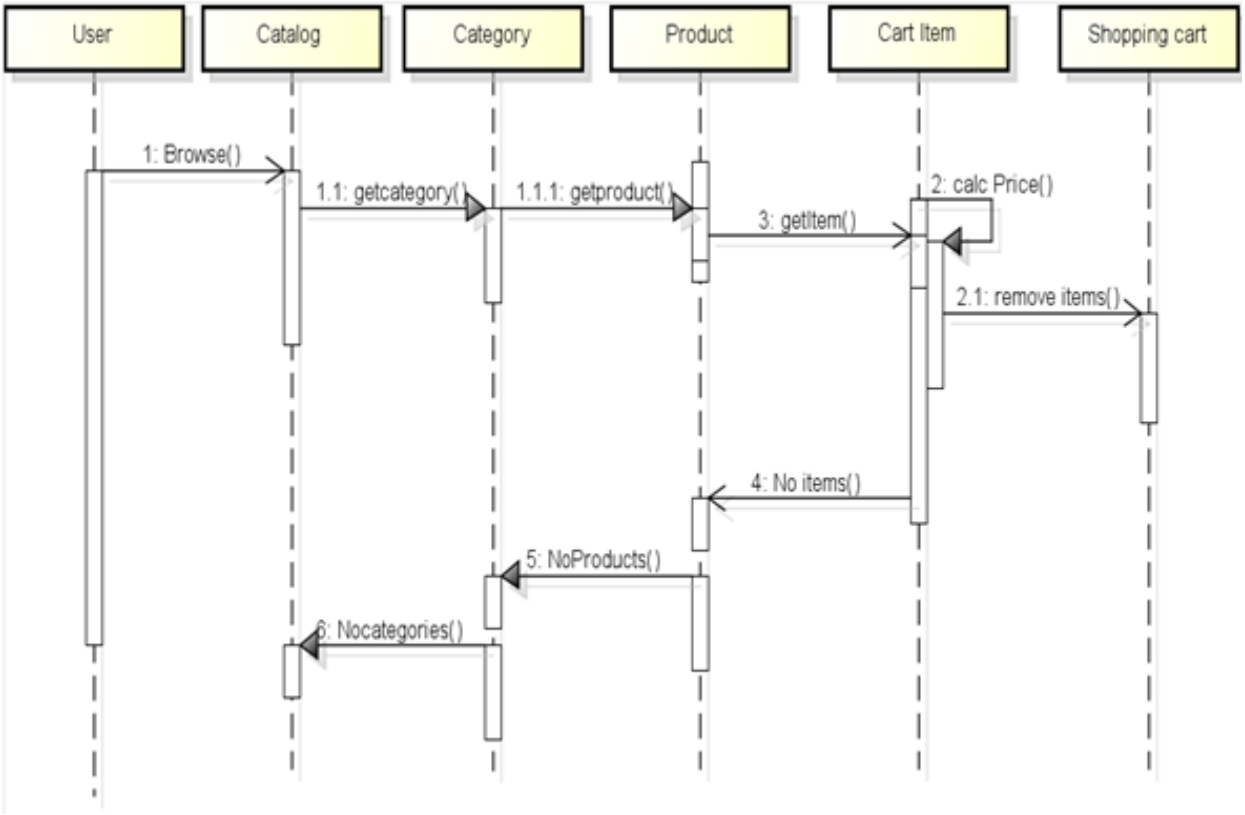


3.3.7 Requirement / Module / Use case for Remove from the cart

3.3.7.1 Class Diagram

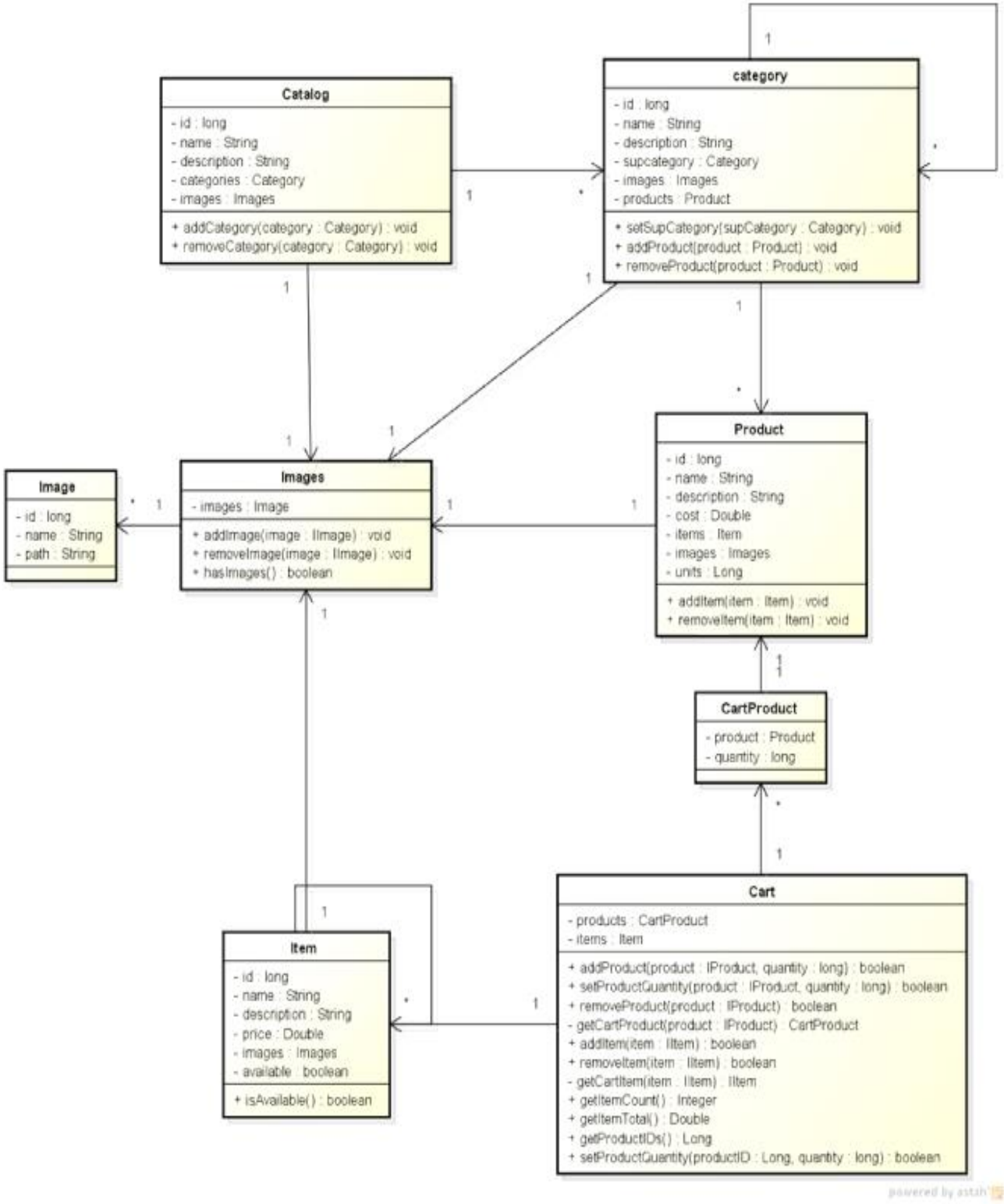
NA

3.3.7.2 Sequence Diagram / Flowchart

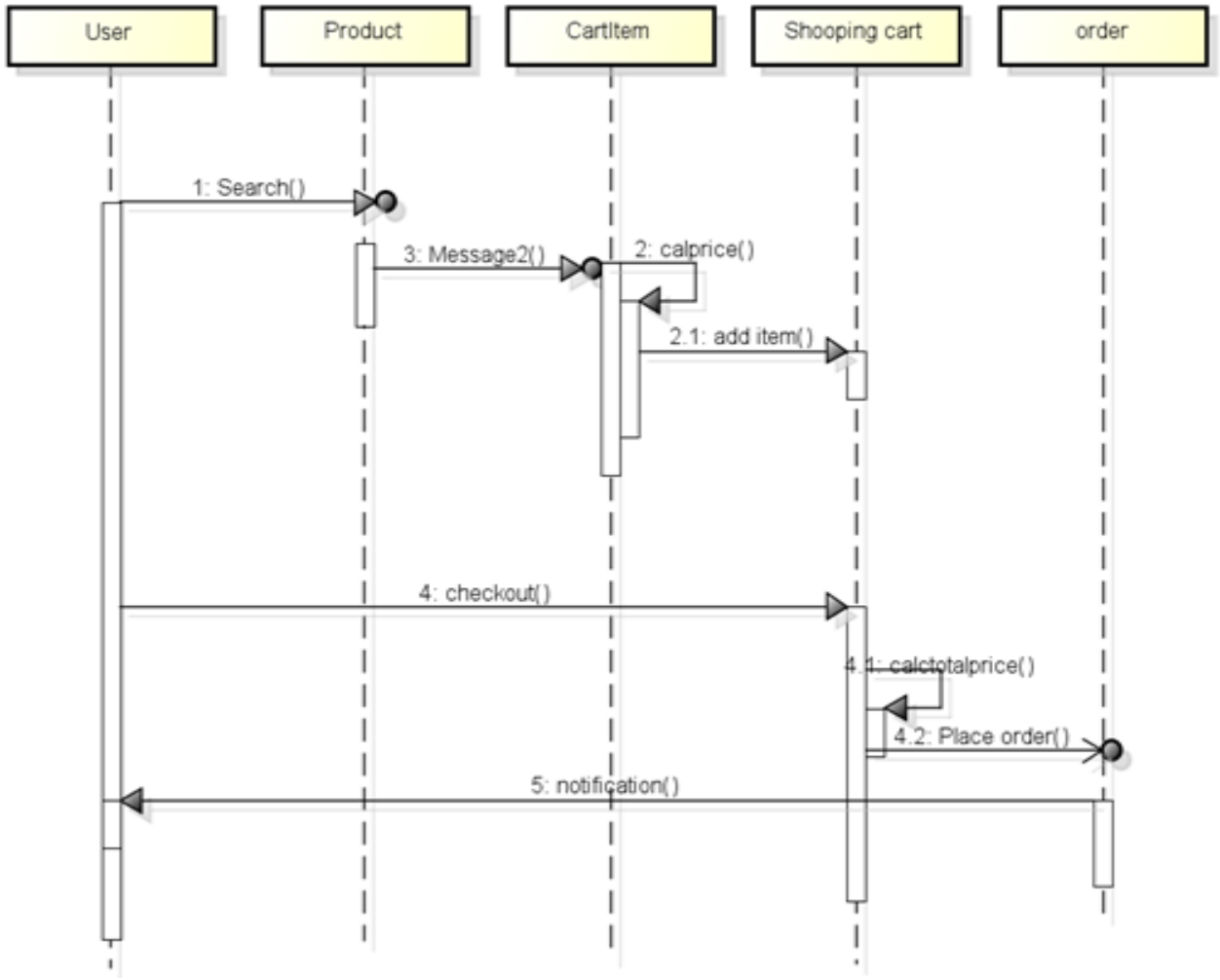


3.3.8 Requirement / Module / Use case for Order

3.3.8.1 Class Diagram



3.3.8.2 Sequence Diagram / Flowchart

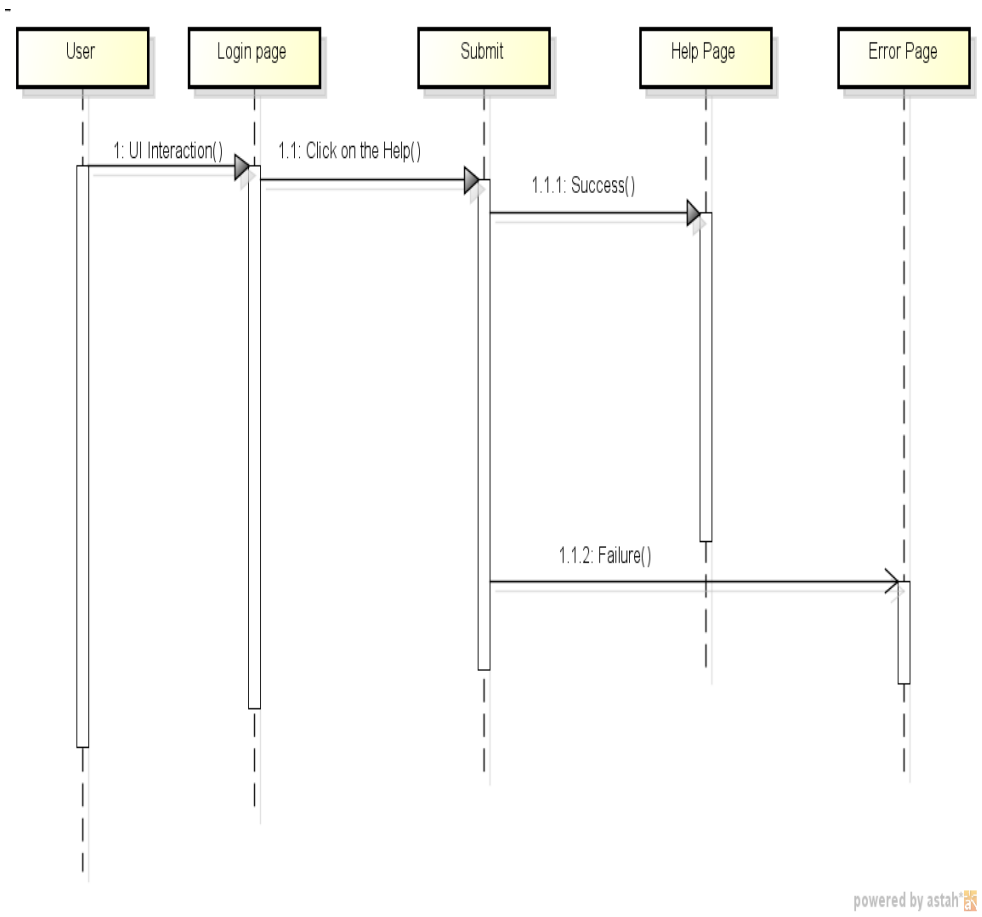


3.3.9 Requirement / Module / Use case for Help

3.3.9.1 Class Diagram

NA

3.3.9.2 Sequence Diagram / Flowchart

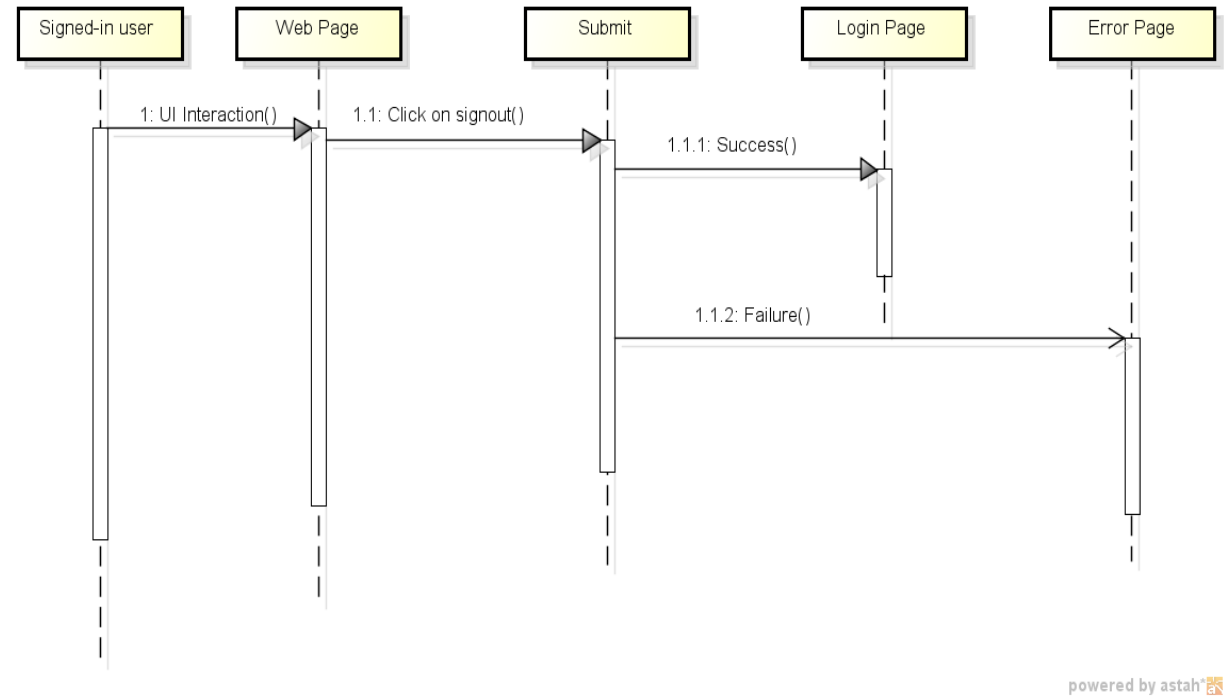


3.3.10 Requirement / Module / Use case for Sign-out

3.3.10.1 Class Diagram

NA

3.3.10.2 Sequence Diagram / Flowchart



3.3.11 Class/File Description

3.3.11.1 Class Name and Type

Address (Public)

Description

This class stores the address of user such as street, phone number, city, state etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	String	address	Null	It stores the street name and house no. / plot no etc.
Private	String	city	Null	It stores the name of the city.

Private	String	state	Null	It stores the name of the state.
Private	String	country	Null	It stores the name of the country.
Private	long	zip	Zero	It stores the pin code of city.
Private	long	phoneNumber	Zero	It stores the phone number.
Private	String	type	Null	It specifies the type of the address i.e. shipping/billing.

3.3.11.2 Class Name and Type

UserAccount (public)

Description

This class stores user details such as first name, last name, user name, password etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	IAddress	Address	Null	It specifies the address of the user.
Private	String	userName	Null	It specifies the User Name of the user.
Private	String	password	Null	It specifies the Password of the user.
Private	String	firstName	Null	It specifies the First Name of the user.
Private	String	lastName	Null	It specifies the Lat Name of the user.
Private	Date	birthDate	01/01/0001	It specifies the Date Of birth of the user.
Private	String	emailID	Null	It specifies the

				Email-ID of the user.

3.3.11.3 Class Name and Type

Credit Card (public)

Description

This class specifies user's credit card details such as expiry date, card number, and card type

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	long	creditCardNumber	Zero	It specifies Credit Card Number.
Private	String	cardType	Null	It specifies Type of the Credit Card.
Private	Date	expirtDate	01/01/0001	It specifies the Expiry Date of Credit Card.

3.3.11.4 Class Name and Type

Order (public)

Description

This class stores user's order details such as User details, Order status etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	IUserAccount	UserAccount	Null	It specifies the details of user.
Private	ICreditCard	CreditCard	Null	It specifies the Credit Card details of user.
Private	ICart	Cart	Null	It specifies the Cart (products/items) which the user ordered.
Private	String	shippingAddress	Null	It specifies the address of the user to which the ordered products have to be shifted.
Private	Date	date	01/01/0001	It specifies the Date on which the user ordered the products.
Private	String	orderStatus	Null	It specifies the Order status i.e. pending/delivered.

3.3.11.5 Class Name and Type

Catalog (public)

Description

This class stores details of Catalog such as name of the Catalog, which categories are present in it and the images describing about the Catalog etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	String	name	Null	It specifies name of the Catalog.
Private	String	description	Null	It specifies description about the Catalog.
Private	Set<Category>	categories	Null	It specifies set of Categories under the Catalog.
Private	Images	images		It specifies Image of the Catalog.

3.3.11.6 Class Name and Type

Category (public)

Description

This class stores details of Category such as name of the Category, which subcategories/products/items are present in it etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	String	name	Null	It specifies name of the Category.
Private	String	description	Null	It specifies description about the Category.
Private	Set<Product>	products	Null	It specifies set of Products under the Category.
Private	Images	images		It specifies Image of the Category.
Private	Category	subCategory	Null	It specifies Image of the Sub Category.

3.3.11.7 Class Name and Type

Images (public)

Description

This class stores Images such as images of products, images of categories etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	Set<Image>	images		It specifies Image of the Category/Catalog/product etc.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add Image** function adds the specified image to the set of Images.
Public	The **Remove Image** function removes the specified image from the set of Images.
Public	The **Has Images** method checks whether the Set has any images or not.

3.3.11.8 Class Name and Type

Image (public)

Description

This class stores details of Image such as Name and Path.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	String	name	Null	It specifies Name of the Image.
Private	String	path	Null	It specifies Path of the Image i.e. the path where the Image resides.

3.3.11.9 Class Name and Type

Product (public)

Description

This class stores details of Product such as Name, Description, Price etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	String	name	Null	It specifies Name of the Image.
Private	String	description	Null	It specifies Description about the Product.
Private	Double	cost	Zero	It specifies the Cost of the Product.
Private	Long	units	Zero	It specifies number of units of

				Products.
Private	Set<Item>	items	Null	It specifies the set of Items which are under the Products.
Private	Images	images		It specifies the images of Products.

3.3.11.10 Class Name and Type

Item (public)

Description

This class stores details of Item such as Name, Description, Price etc.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	String	name	Null	It specifies Name of the Item.
Private	String	description	Null	It specifies Description about the Item.
Private	Double	price	Zero	It specifies the Price of the Item.
Private	Images	images		It specifies the images of Items.
Private	boolean	available	false	It specifies whether the item is available or not.

3.3.11.11 Class Name and Type

CartProduct (public)

Description

This class stores details of Product the User ordered.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	Product	product	Null	It specifies the details of the Product that the user ordered.
Private	long	quantity	Zero	It specifies Quantity of the Product that the user ordered.

3.3.11.12 Class Name and Type

Cart (public)

Description

This class stores details of Product and Items the User ordered.

Data Items:

Protection Level	Data-type	Variable Name	Initial Value	Comments
Private	List<CartProduct>	products	Null	It specifies the details of the Product that the user ordered.
Private	List<Items>	items	Null	It specifies details of the Items that the user ordered.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add Product** method adds the specified Product to Cart with specified Quantity and this method returns **true** if the Product is added to Cart.
Public	The **Set Product Quantity** method sets the specified Product with specified Quantity and this method returns **true** if the Product is set with specified Quantity.
Public	The **Remove Product** method removes the specified Product from Cart and this method returns **true** if the specified Product is removed from Cart.
Public	The **Get Cart Product** method returns the specified Product if the Product is same as the Product which is added to Cart and if not it returns false.
Public	The **Add Item** method adds the specified Item to Cart and it returns **true** if the Item is added to Cart.
Public	The **Remove Item** method removes the specified Item from Cart and this method returns **true** if the Item is removed from Cart.
Public	The **Get Cart Item** method returns the specified Item if the Item is same as the Item

	which is added to Cart and if not it returns false.
Public	The **Get Item Count** method returns the total number of Products and Items added to the Cart.
Public	The **Get Item Total** method returns the total cost of the Products and Items which are added to Cart.
Public	The **Get Product IDs** method returns the list of IDs of all the Products which are added to Cart.

3.3.11.13 Class Name and Type

CatalogDAO (public)

Description

This class contains the business logic for the application that allows the user to perform any operations on the catalog class.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add Catalog** method adds the specified Catalog into **Catalog** table in the database.
Public	The **Add Category** method adds the specified Category into **Category** table in the database under the specified catalog.

Public	The **Get Catalog** method fetches the selected catalog from the **Catalog** table in the database.
Public	The **Get Catalogs** method fetches all the catalogs from the **Catalog** table.
Public	The **Get Categories** method fetches the all the categories present under the specified catalog from **Category** table.
Public	The **Remove Catalog** method removes the specified catalog from the **Catalog** table.
Public	The **Remove Category** method removes the specified category from **Category** table and also from **Catalog** table.
Public	The **Update Catalog** method updates the specified catalog in **Catalog** table in the database.

3.3.11.14 Class Name and Type

CategoryDAO (public)

Description

This class contains the business logic for the application that allows the user to perform any operations on Category class.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add Product** method adds the specified

	product into Product table in the database under the specified category.
Public	The **Add SubCategory** method adds the specified subcategory into Category table in the database under the specified category.
Public	The **Get Category** method fetches the Category with specified id from Category table in the database.
Public	The **Get Products** method fetches set of all the products under the specified Category from Product table in the database.
Public	The **Get Subcategories** method fetches set of all subcategories under the specified Category from Category table in the database.
Public	The **Has Products** method checks for the availability of products in Product table in the database under the specified Category.
Public	The **Has Subcategories** method checks for the availability of subcategories in Category table in the database under the specified Category.
Public	The **Remove Product** method removes the specified product from Product table and also from Category table.
Public	The **Update Category** method updates the specified category in Category table in the database.

3.3.11.15 Class Name and Type

ItemDAO (public)

Description

This class contains the business logic for the application that allows the user to perform any operations on Item class.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Get Item** method fetches the item with specified from Item table in database.
Public	The **Update Item** method updates the specified item in Item table in the database.

3.3.11.16 Class Name and Type

OrderDAO (public)

Description

This class contains the business logic for the application that allows the user to perform any operations on Order class.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add Order** method adds the specified order into Order table in the database.

Public	The **Get Order** method fetches the order with specified from Order table in the database.
Public	The **Get Orders** method fetches the set of orders with specified UserName from Order table in database.

3.3.11.17 Class Name and Type

ProductDAO (public)

Description

This class contains the business logic for the application that allows the user to perform any operations on Product class.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add Item** method adds the specified item into Item table in the database under the specified product.
Public	The **Get Items** method fetches set of all the items under the specified product from Item table in the database.
Public	The **Get Product** method fetches the product with specified id from the Product table
Public	The **Has Displayable Item** method checks for availability of items in Item table under the specified Product.

Public	The **Get Item count** method fetches the number of items present under the specified product in Item table.
Public	The **Remove Item** method removes the specified item from Item table and also from Product table.
Public	The **Update Product** method updates the specified product in Product table in the database.

3.3.11.17 Class Name and Type

UserAccountDAO (public)

Description

This class stores details of User

- updateUserAccount(IUserAccount account)
- getCreditcard(**long** id)

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Add User Account** method adds the specified user's personal details into User_Account table and address details into Address table in the database.
Public	The first **Get User Account** method fetches the details of user with specified id from User_Account table in the database.

Public	The second **Get User Account** method fetches the details of user with specified user name and password from User_Account table in the database.
Public	The third **Get User Account** method fetches the details of user with specified user name from User_Account table in the database.
Public	The **Remove User Account** method removes the specified user's personal details from User_Account table and address details from Address table in the database.
Public	The **Update User Account** method updates the specified user's personal details in User_Account table and address details in Address table in the database.
Public	The **Get Credit Card** method fetches the credit card object/row with specified id from CreditCard table in the database.

3.3.11.18 Class Name and Type

WebStorePOJO (public)

Description

This class includes the business logic to perform various operations on the application. The business logic is implemented by the methods in this class. Most of the methods in this class are implemented by the DAO classes as explained above.

Member Functions:

Protection Level	Function Description/Algorithm
Public	The **Create Address** method creates a new Address object.
Public	The **Create Catalog** method creates a new Catalog object.
Public	The **Create Category** method creates a new Category object.
Public	The **Create Cart** method creates a new Cart object.
Public	The **Create CreditCard** method creates a new CreditCard object.table.
Public	The **Create Item** method creates a new Item object.
Public	The **Create Image** method creates a new Image object.
Public	The **Create Product** method creates a new Product object.
Public	The **Create Order** method creates a new Order object.
Public	The **Create UserAccount** method creates a new UserAccount object.

3.4 Packaging Structure.

The software used to develop this application is organized in different folders. The folder structure is explained as follows.

The **src** folder contains packages which include all the classes. Each package is explained as follows.

(i) com.hcl.cf.webstore.dao.interfaces

This package has six interfaces. They are,

a. ICatalogDAO.java

b. ICAtegoryDAO.java

c. IItemDAO.java

d. IOrderDAO.java

e. IProductDAO.java

f. IUserAccounyDAO.java

(ii) com.hcl.cf.webstore.dao.implementation

This package has six classes. Each class implements the methods defined in the above interfaces.

a. CatalogDAO.java

b. CAtegoryDAO.java

c. ItemDAO.java

d. OrderDAO.java

e. ProductDAO.java

f. UserAccounyDAO.java

(iii) com.hcl.cf.webstore.domain.interfaces

This package has eleven interfaces. They are,

a. IAddress.java

b. ICart.java

c. ICatalog.java

d. ICategory.java

e. ICreditCard.java

f. IImage.java

g. IImages.java

h. IItem.java

i. IOrder.java

j. IProduct.java

k. IUserAccount.java

l. IWebStoreFacade.java

(iv) com.hcl.cf.webstore.domain.entities

This package has twelve classes which implement the methods defined in the above interfaces except for the methods in IWebStoreFacade interface.

a. Address.java

b. Cart.java

c. CartProduct.java

d. Catalog.java

e. Category.java

f. CreditCard.java

g. Image.java

h. Images.java

i. Item.java

j. Order.java

k. Product.java

l. UserAccount.java

(v) com.hcl.cf.webstore.domain.constants

This package has three interfaces. Each interface has two static variables of type String.

a. IAddressType.java

b. ICreditCardType.java

c. IOrderStatus.java

(vi) com.hcl.cf.webstore.domain.validators

This package has one class.

(i) ProductItemCostValidator.java

(vii) com.hcl.cf.webstore.facade

This package has one class which implements all the methods in IWebStoreFacade interface.

a. WebStorePOJO.java

(viii) com.hcl.cf.webstore.hibernate.util

This package has one class which returns the Hibernate session factory.

a. HibernateUtil.java

(ix) com.hcl.cf.webstore.web.struts.actions

This package has 21 classes (struts action classes) which are responsible for calling methods in DAO classes.

a. AddItemToCartAction.java
b. AddProductToCartAction.java
c. ClearCartAndOrderAction.java
d. DisplayItemAction.java
e. DisplayProductAction.java
f. ListCatalogAction.java
g. ListCategoryAction.java
h. ListItemsAction.java
i. ListSubCategoryOrProductAction.java
j. LoginAction.java
k. PlaceOrderAction.java
l. RegisterUserAction.java
m. RemoveItemAction.java
n. RemoveProductAction.java
o. ShowCartAction.java
p. ShowHelpAction.java
q. ShowOrderDetailsAction.java
r. ShowRegistrationFormAction.java
s. ShowUserAccountAction.java
t. SignOutAction.java
u. UpdateCartAction.java

(x) com.hcl.cf.webstore.test.junit

This package has five (Junit test) classes which are used to test the methods defined in DAO classes.

a. AllTests.java

b. TestWebStorePOJO_Add.java

c. TestWebStorePOJO_Remove.java

d. TestWebStorePOJO_Select.java

e. TestWebStorePOJO_Update.java

(xi) com.sush.sample.webstore.store.domain.test.nonJUnit

This package has seven (Non-Junit test) classes.

a. TestAddress.java

b. TestCatalog.java

c. TestCategory.java

d. TestCreditCard.java

e. TestItem.java

f. TestProduct.java

g. TestUser.java

This folder also contains the configuration files such as struts configuration file (**struts.xml**)**,** hibernate configuration file (**hibernate.cfg.xml**) and one subfolder **(hibernate)** which contains **hibernate mapping** files.

The **WebContent** folder has five sub folders. One is **db**, second is **images,** third is **jsp,** fourth is **WEB-INF** and the last is **META-INF**. The db folder has all the jpg images and images folder contains all gif images. The jsp folder includes all the web (jsp) pages that are used in this application.

The **WEB_INF** has one folder named as **lib** which contains all the library (jar) files that are required for this application and one configuration file named as **web.xml**.

The **dbscripts** folder has database (SQL) script to create all the tables used in this application and the script to insert values into the tables.

3.5 Configuration Changes

NA

3.6 Deployment Guidelines

Step 1: In project folder open bin folder consists of three files namely Buil.xml, Build.bat, and Deploy.bat.

Step 2: Modify the local machine class path for ApacheAnt, Tomcat and Java_Home which has server class path information in all the three files.

Step 3: Open Command prompt type “build war” it will compile.

Step 4: Then type deploy.bat it will copy the files into the project folder.

Step 5: Start Tomcat and execute the project in the browser by giving

http://localhost:8080//WebStore/jsp/Header.jsp.