Software Architecture Document

Shopping Cart System

OpenCart



**By :**

**Khaled J.Awadallah**

**Abdallah Azmi Abudaqqa**

Table of Contents

Contents

[1. Introduction 4](#_Toc387140791)

[1.1. Purpose 4](#_Toc387140792)

[1.1. Scope 5](#_Toc387140793)

[1.2. Definitions, Acronyms and Abbreviations 5](#_Toc387140794)

[1.3. References 5](#_Toc387140795)

[1.4. Overview 5](#_Toc387140796)

[2. Architectural Representation 6](#_Toc387140797)

[Logical view 6](#_Toc387140798)

[Process view 6](#_Toc387140799)

[Implementation view 6](#_Toc387140800)

[Deployment view 6](#_Toc387140801)

[Use Case view 6](#_Toc387140802)

[Data view 6](#_Toc387140803)

[3. Architectural Goals and Constraints 7](#_Toc387140804)

[3.1. Technical Platform 7](#_Toc387140805)

[3.2. Transaction 7](#_Toc387140806)

[3.3. Security 7](#_Toc387140807)

[3.4. Persistence 7](#_Toc387140808)

[3.5. Reliability/Availability (failover) 7](#_Toc387140809)

[3.6. Performance 8](#_Toc387140810)

[3.7. Internationalization (i18n) 8](#_Toc387140811)

[3.8. Integration with third party services 8](#_Toc387140812)

[4. Components of Online Shopping Cart System 8](#_Toc387140813)

[5. Views 8](#_Toc387140814)

[6. Use-Cases 9](#_Toc387140815)

[6.1. Buyer View – Order placement 9](#_Toc387140816)

[6.2. Merchant View – Inventory modification 10](#_Toc387140817)

[6.2 Use-Case Realizations 10](#_Toc387140818)

[7. Logical View 10](#_Toc387140819)

[7.1. Overview 10](#_Toc387140820)

[7.2. Architecturally Significant Design Packages 11](#_Toc387140821)

[7.3. Delivery Service 13](#_Toc387140822)

[7.4. Payment Service 13](#_Toc387140823)

[8. Process View 13](#_Toc387140824)

[9. Deployment View 14](#_Toc387140825)

[9.1. Global Overview 14](#_Toc387140826)

[External Desktop PC 14](#_Toc387140827)

[Web Server 15](#_Toc387140828)

[Database Server 15](#_Toc387140829)

[Detailed deployment model with clustering 15](#_Toc387140830)

[10. Implementation View 15](#_Toc387140831)

[10.1. Overview 15](#_Toc387140832)

[11. Data View 16](#_Toc387140833)

[12. Size and Performance 16](#_Toc387140834)

[13. Quality 17](#_Toc387140835)

[Scalability: 17](#_Toc387140836)

[Reliability, Availability: 17](#_Toc387140837)

[Portability: 17](#_Toc387140838)

[Security: 17](#_Toc387140839)

[Availability on various browsers 17](#_Toc387140840)

[Generic Requirements 17](#_Toc387140841)

# Introduction

This document explains the software architecture of OpenCart – an online shopping cart system. OpenCart is based on MVC architecture and provides a professional and reliable foundation from which to build a successful online store.

The OpenCart Web App has been installed at : <http://www.safka.ps/opencart> for documentation process .

OpenCart’s technology stack includes following prerequisite –

* Web Server (preferably Apache)
* PHP (at least 5.2)
* MySQL

To ensure the modularity and extensibility of cart system, OpenCart makes extensive use of caching and has vQmod plugin which lets user install new modules without changing any of the original PHP source code files.

A RUP Software Architect will typically perform eight major steps in order to define a global architecture.

|  |  |
| --- | --- |
| Architectural activities | Software Architecture Document |
| Step 1 - Identify and prioritize significant Use-Cases | Section 6 |
| Step 2 - Define the candidate architecture | Section 3, 5.1, 10, 11 |
| Step 3 - Define the initial Deployment Model | Section 9 |
| Step 4 - Identify key abstractions | Section 11 |
| Step 5 - Create an Analysis Model | Section 7 |
| Step 6 - Create the Design Model | Section 7 |
| Step 7 - Document concurrency mechanisms | Section 8,9 |
| Step 8 - Create the Implementation Model | Section 11 |

## Purpose

This document provides a comprehensive architectural overview of the online shopping cart system provided by OpenCart Limited, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the “4+1” model view of architecture [KRU41].



The “4+1” View Model allows various stakeholders to find what they need in the software architecture.

## Scope

The scope of this Software Architecture Document (SAD) is to depict architecture of the online shopping cart system provided by OpenCart Limited.

## Definitions, Acronyms and Abbreviations

RUP: Rational Unified Process

UML: Unified Modeling Language

SAD: Software Architecture Document

MVC: Model View Controller

QoS: Quality of Service

## References

[KRU41]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf>

[RSA]: IBM Rational Software Architect

<http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html>

[RUP]: The IBM Rational Unified Process : <http://www306.ibm.com/software/awdtools/rup/index.html>

[MVC] Model-View-Controller Design <https://faculty.washington.edu/markk/sites/default/files/CSS555_Model-View-Controller_Design.pdf>

Web-Application Development Using the Model View Controller Design Pattern

<http://professor.unisinos.br/wp/crespo/files/2011/07/00950428.pdf>

## Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural constraints of the system

Section 4: describes the functional requirements with a significant impact on the architecture

Section 5: describes the most important use-case realization. Will contain the Analysis Model and the Design Model

Section 6: describes design’s concurrency aspects

Section 7: describes how the system will be deployed. Will contain the Deployment Model

Section 8: describes the layers and subsystems of the application

Section 9: describes any significant persistent element. Will contain the Data Model

Section 10: describes any performance issues and constraints

Section 11: describes any aspects related to the quality of service (QoS) attributes

# Architectural Representation

This document details the architecture using the views defined in the “4+1” model [KRU41], but using the RUP naming convention. The views used to document the OpenCart application are:

### Logical view

**Audience**: Designers.

**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations.

**Related Artifacts**: Design model

### Process view

**Audience**: Integrators.

**Area**: Non-functional requirements: describes the design's concurrency and synchronization aspects.

**Related Artifacts**: (no specific artifact).

### Implementation view

**Audience**: Programmers.

**Area**: Software components: describes the layers and subsystems of the application.

**Related Artifacts**: Implementation model, components

### Deployment view

**Audience**: Deployment managers.

**Area**: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.

**Related Artifacts**: Deployment model.

### Use Case view

**Audience**: all the stakeholders of the system, including the end-users.

**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.

**Related Artifacts**: Use-Case Model, Use-Case documents

### Data view

**Audience**: Data specialists, Database administrators

**Area**: Persistence: describes the architecturally significant persistent elements in the data model

**Related Artifacts**: Data model.

MVC Architecture model divides the whole software into three layers – **Model**, **View**, and **Controller**.

Each layer has been assigned a well-defined functionality, thus making it modular and easily scalable.

Each view as defined by RUP corresponds to a layer in MVC.

* Data view corresponds to Model layer in MVC architecture
* Use Case and Logical View corresponds to View layer in MVC architecture
* Process view corresponds the Controller layer in MVC architecture.

# Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture

## Technical Platform

The OpenCart, online shopping cart will be deployed onto a Webserver like Apache, nginx and requires PHP for execution.

## Transaction

The OpenCart online application is based on REST services and uses cookies and request data to manage transactions.

## Security

The system must be secured, so that a customer can make online payments.

The application must implement basic security behaviors:

* Authentication: Login using at least a user name and a password
* Authorization: With respect to authorization- online customer should be able to log in and browse through their order history, check credits and coupons, and modify and update the billing and shipping info. Customer should be able to see data pertaining to him  
    
  For internet access, the following requirements are mandatory
* Confidentiality: sensitive data must be encrypted (credit card payments)
* Data integrity : Data sent across the network cannot be modified by a tier
* Auditing: Every sensitive action can be logged
* Non-repudiation : gives evidence a specific action occurred

## Persistence

Data persistence will be addressed using a relational database - MySQL.

**Note:** Database Schema Attached with this document

## Reliability/Availability (failover)

The availability of the system is a key requirement by nature, as it is a selling system. The candidate architecture must ensure failover capabilities. Reliability/Availability will be addressed through the PHP and Web server platform.

Targeted availability is 24/7: 24 hours a day, 7 days a week

Maintenance or updates will be carried out once in a month and takes 15-20 minutes.

## Performance

The payment process (credit card authorization and confirmation) must be under 10 seconds.

Customers should be able to select available inventory items. Merchant should be able to add and delete inventory items within 30 seconds.

Inventory updating should occur in real-time without any major time lag.

## Internationalization (i18n)

The online shopping cart must support various languages as it is used over the world. So the presentation layer must be able to support i18n. Internationalization is taken care of by language files and language configuration settings. Other layers must be generic enough to work with any internationalization context.

## Integration with third party services

The online shopping cart is rarely used as a standalone component. It requires multiple third party services like payment, catalogue and ERP management systems, shipping tracker and store-front or template system. This requires it to be flexible enough to support various types of payment gateways and other systems. Therefore its architecture should accommodate a plug and play system for supporting these various third party services.

# Components of Online Shopping Cart System

Any shopping cart has few integral component to make it work. These generic components are –

1. **Search** – Lets buyer search for the desired item
2. **Ordering System** – Lets buyer select item, add it to cart and navigate till payment gateway
3. **Billing System** – Computes the total payable amount after applying all the discounts, coupons and tax.
4. **Payment Gateway Integration system** – this system is responsible for communicating all the details correctly to payment gateway from cart and from gateway back to cart.
5. **Merchant Admin** – Lets merchant update, add and delete inventory. Show report of the transactions
6. **Buyer Registration System (Optional)** – Lets buyer store their billing, shipping info and order history on the server/with cart system.

# Views

The online shopping cart has two types of views –

1. **Buyer View** - It lets buyer select items, log on or register, order, make payment and check invoices.
2. **Merchant View or Admin View** – It lets merchant configure the store, integrate with other third party plugins or modules, update and add inventory, update store front etc. Also merchant view should have a reporting view that let merchant find out the number of orders and total billing etc.

Both views are important for overall success. However Buyer View is very critical to success as it needs 24/7 availability with very little possibility of downtime. Also integration with payment gateways make this view further critical from security point of view.

# Use-Cases

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system.

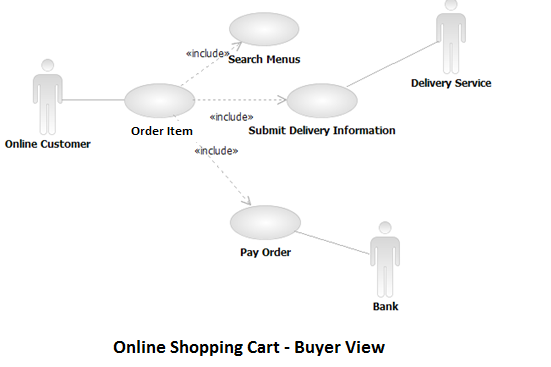
Important Use Cases –

1. Customer selects item, does checkout and makes payment and receives invoice of the order.
2. Customer selects item, registers, performs checkout and makes payment. All transaction should be stored in his/her account in persistence DB
3. Customer shops items, logs in, all information prepopulated, performs checkout.
4. Tax and Discount in cart are correctly shown in bill and correct payment amount is passed to payment gateways
5. Merchant should get correct information about orders placed and other details like shipping information, contact number.
6. Merchant should be able to add, update and delete inventory items.

**Of all the above mentioned use cases critical ones are – 1), 4), 5) and 6).**

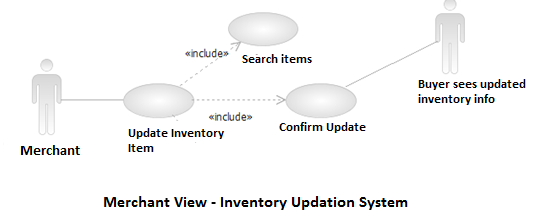
## Buyer View – Order placement

A customer accesses the online shopping cart and search for the available items. The customer chooses from a list of menus and select what she/he wants to order. Then, the customer performs an online payment (credit card). Once the payment has been validated, the customer confirms the order, enters her/his delivery information (name, address, phone number, etc..) and all the relevant information is sent to the Merchant’s configured delivery service.



## Merchant View – Inventory modification

As soon as merchant receives update about any inventory item, he will change it to reflect on store front. So he will log in to merchant view and search for the desired item and enter the updated information. Confirm the update and submit it. Once confirmed, any new buyer will see now updated information for that item.



## Use-Case Realizations

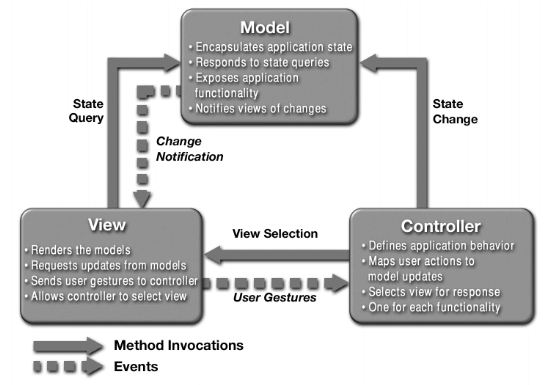
Refers to section 8.2 to see how design elements provide the functionalities identified in the significant use-cases.

# Logical View

# Overview

The online catering application is divided into layers based on the MVC architecture. The layering model of the online cart application is based on a responsibility layering strategy that associates each layer with a particular responsibility. This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.

Architecture Layer Dependencies



MVC Architecture has three layers as stated above – Model, View and Controller.

* Model stores the persistent as well as transactional data.
* View is responsible for displaying the information as requested by user.
* Controller acts as a coordinator between Model and View.

So when a user of MVC application presses next button. View sends all the data it has in current view to model and sends the request to display next page to controller. Controller runs process and business logic for the request and user and determines the next state of application. After determination of next state of application, it determines view and renders view. Now new view determines what all data it needs and requests Model to send the data.

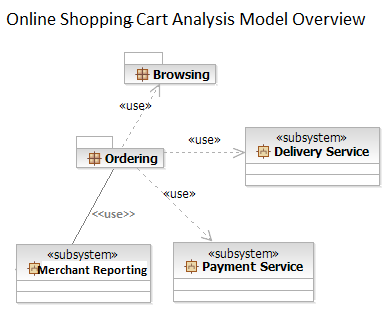
So now controller doesn’t need to find out data needed by each view. Once it determines the view, it’s the responsibility of view to show the data.

Each layer has specific responsibilities.

* Controller’s responsibility is to run business and process logic based on request from view and data stored in model.
* View’s responsibility is to render and display correct info to users.
* Model’s responsibility is to maintain data in correct state always.

The initial version of OpenCart is quite simple and only contains three basic features, one for the submit orders and the other allowing a customer to browse the online catalogue and last one allowing merchant to access information regarding order placed.

External services are used for the payment and the delivery functionalities.

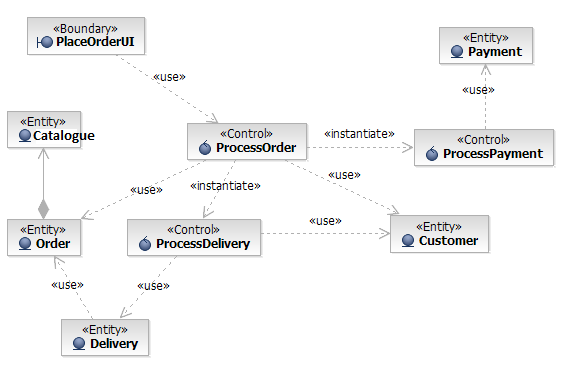


## Architecturally Significant Design Packages

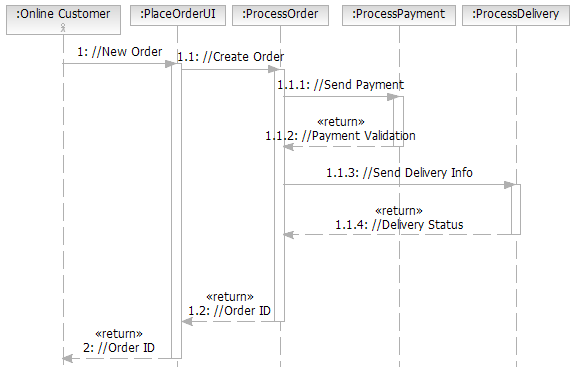
This package is responsible for all the logic related to the online orders. It provides ordering features and the necessary components to access the external services (Payment and Delivery)

Analysis Model

**Participants:**

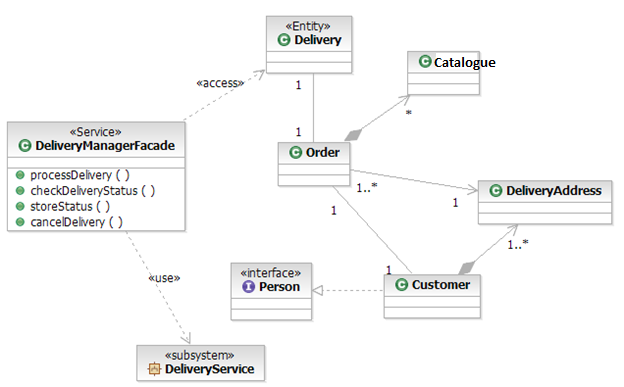


**Basic Flow:**



Design Model

**Process Delivery**

****

## Delivery Service

Contains all the logic related to the Merchant’s configured delivery service.

The Delivery Service is an external subsystem documented in its own Software Architecture Document

## Payment Service

Contains all the logic related to the online payment and credit card validation.

The Payment Service is an external subsystem documented in its own Software Architecture Document.

# Process View

There are three critical processes –

* Store Front display - This will be initiated as soon as a customer request online store.
* Order Placement process – Initiated by pressing add to cart button
* Payment – On confirming the order placement, payment process will be initiated

<<process>>

Browsing through Catalogue

<<process>>

Place Order

<<process>>

Payment Processing

StoreFront

🡪showCatalogue()

🡪showSearchForm()

Order Controller

🡪add selected item to cart()

🡪new<<Order>>

🡪calculateBilling()

🡪getShippingInfo()

Payment Controller

🡪passBillAmountToGateway()

🡪getResponseCode()

<<process>>

Close Order Process

Close Order Controller

🡪isCheckoutSuccessful()

🡪closeOrder()

<<process>>

Delivery Info Processing

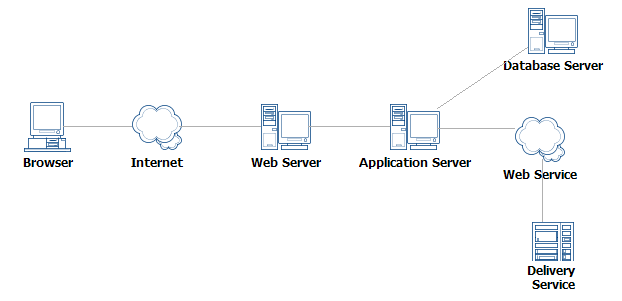
DeliveryInfoController

🡪getDeliveryInfo()

🡪emailDeliveryInfo()

# Deployment View

## Global Overview



### External Desktop PC

Buyer places order using an external PC which is connected to Apache Web Server via internet or mobile data connection.

Merchant also updates inventory and modifies store settings using an External Desktop PC which is connected to web server via internet connection.

### Web Server

It is the main Unix server that deploys any web server compatible with PHP like Apache or Nginx. It processes all kinds of requests coming from buyers and merchant and directs to appropriate process or program for further processing.

### Database Server

To store all the persistent data related to customer, store and orders. As OpenCart is only compatible with MySQL, so we’ll be using MySQL only. MySQL can be installed on same server hardware on which webserver or it can be installed on a separate dedicated hardware.

### Detailed deployment model with clustering

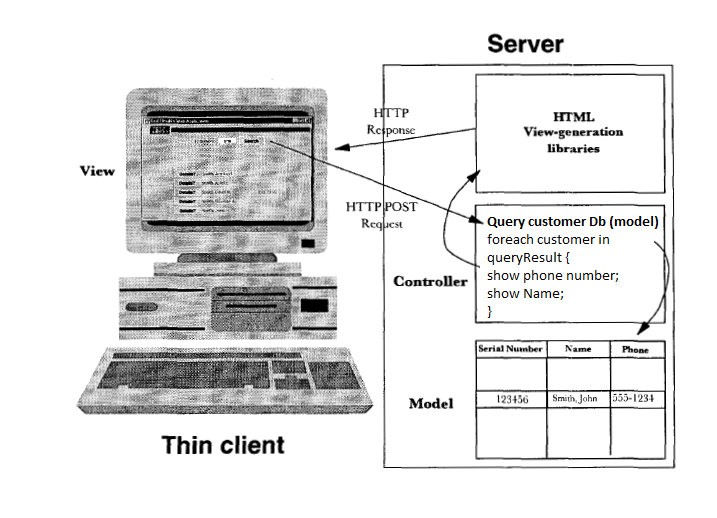
* One Apache HTTP Server will dispatch requests to Php process and php process will request data from database server as and when needed.
* Apache server can be preceded by a load balancing server which could direct request to any of the Apache server in cluster.
* A MySQL Database stores all the information related to online orders.

# Implementation View

## Overview

The Implementation view depicts the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files).

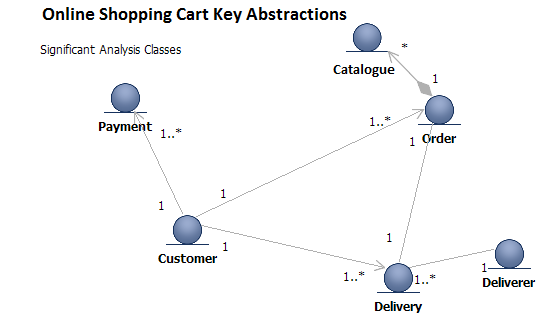
Usually, the layers of the Implementation view do fit the layering defined in the Logical view



# Data View

The key data elements related to the online shopping cart system are:

1. Inventory/Catalogue Data
2. Customer Specific Data
3. Order Data
4. Payment information
5. Delivery data as in shipping status etc.



# Size and Performance

The chosen software architecture supports the key sizing and timing requirements, as stipulated in the Supplementary Specification [15]:

* The system shall support up to 2000 simultaneous users against the central database at any given time, and up to 500 simultaneous users against the local servers at any one time.
* The system shall provide access to the item catalog database with no more than a 10 second latency.
* The system must be able to complete 80% of all transactions within 2 minutes.
* The client portion or buyer interface shall require less than 20 MB disk space and 512 MB RAM.

The selected architecture supports the sizing and timing requirements through the implementation of a client-server architecture. The buyer interface is implemented on HTML and Javascript which can run on any browser. The components have been designed to ensure that minimal disk and memory requirements are needed on the buyer interface.

# Quality

As far as the online shopping cart application is concerned, the following quality goals have been identified:

### Scalability:

* **Description** : System’s reaction when user demands increase
* **Solution**: Web application servers support several workload management techniques, also more than one front-end server can be deployed to manage high loads. MySQL also supports multiple database servers and takes care of all replication and concurrency issues.

### Reliability, Availability:

* **Description**: Transparent failover mechanism, mean-time-between-failure.
* **Solution:** Web application server supports load balancing through clusters.

### Portability:

* **Description** : Ability to be reused in another environment
* **Solution:** The system needs PHP, MySQL and curl. So it can work on any \*nix web server which has these three software installed.

### Security:

* **Description** : Authentication and authorization mechanisms
* **Solution:** Webserver and PHP native security mechanisms will be reused. To ensure security on client side, CSRF token will be used and appended to every response and validated by every request.

### Availability on various browsers

* **Description**: Availability on various browsers – Internet Explorer, Firefox and Chrome etc..
* **Solution:** OpenCart makes use of HTML template with very little javascript compatible on all browsers. However third party store templates might use various JS libraries incompatible with older versions of IE in that case its clearly mentioned and is left to discretion of merchant.

### Generic Requirements

* The user interface of the online shopping cart shall be designed for ease-of-use and shall be appropriate for a computer-literate user community with no additional training on the System.
* Each feature of the online shopping cart shall have built-in online help for the user. Online Help shall include step by step instructions on using the System. Online Help shall include definitions for terms and acronyms.
* The online shopping cart system shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time. Mean Time between Failures shall exceed 300 hours.