pratian
connect to infinity

SkillAssure

Project Specification

SHOP-ON

V 1.0

Revision History

Date	Version	Title	Author	Reviewer
07-30-2020	1.0	First Version	Giri B	

Contents

1. Introduction

Shop-On is enterprise open source e-commerce software for retailers who want flexibility, speed and control of their commerce platform. Shop-On is a software solution that gives organizations the ultimate flexibility to take an experience-first approach to commerce, with simple powerful APIs and built in stores models.

Shop-On e-commerce system provides the following functionality:

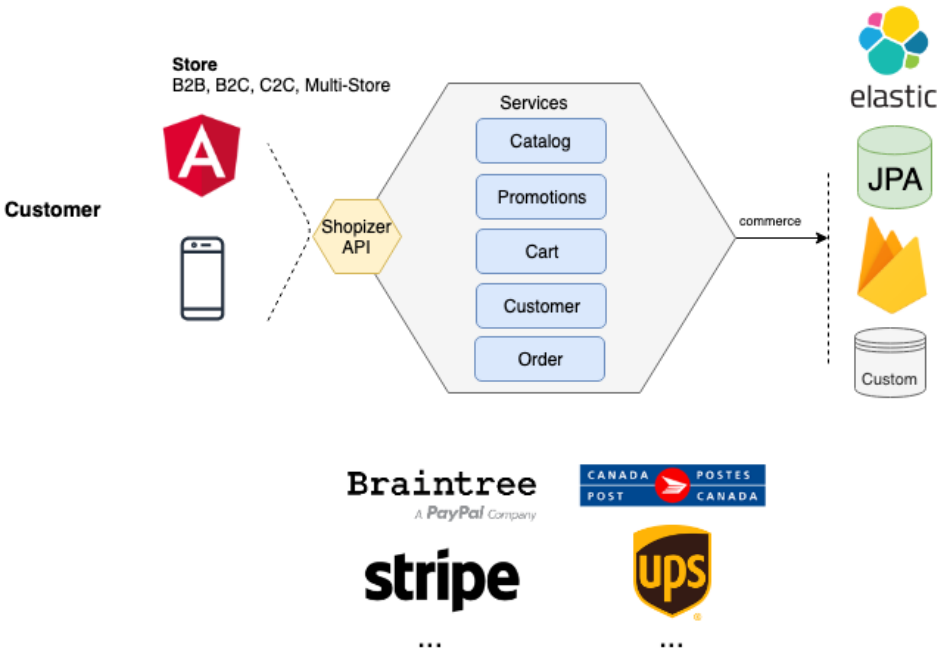
- Catalog management
- User management
- Customer management
- Content management
- Order management
- Pricing management
- Shopping cart management
- Configuration management
- Shipping
- Payment
- Promotions
- Search

1.1 Users

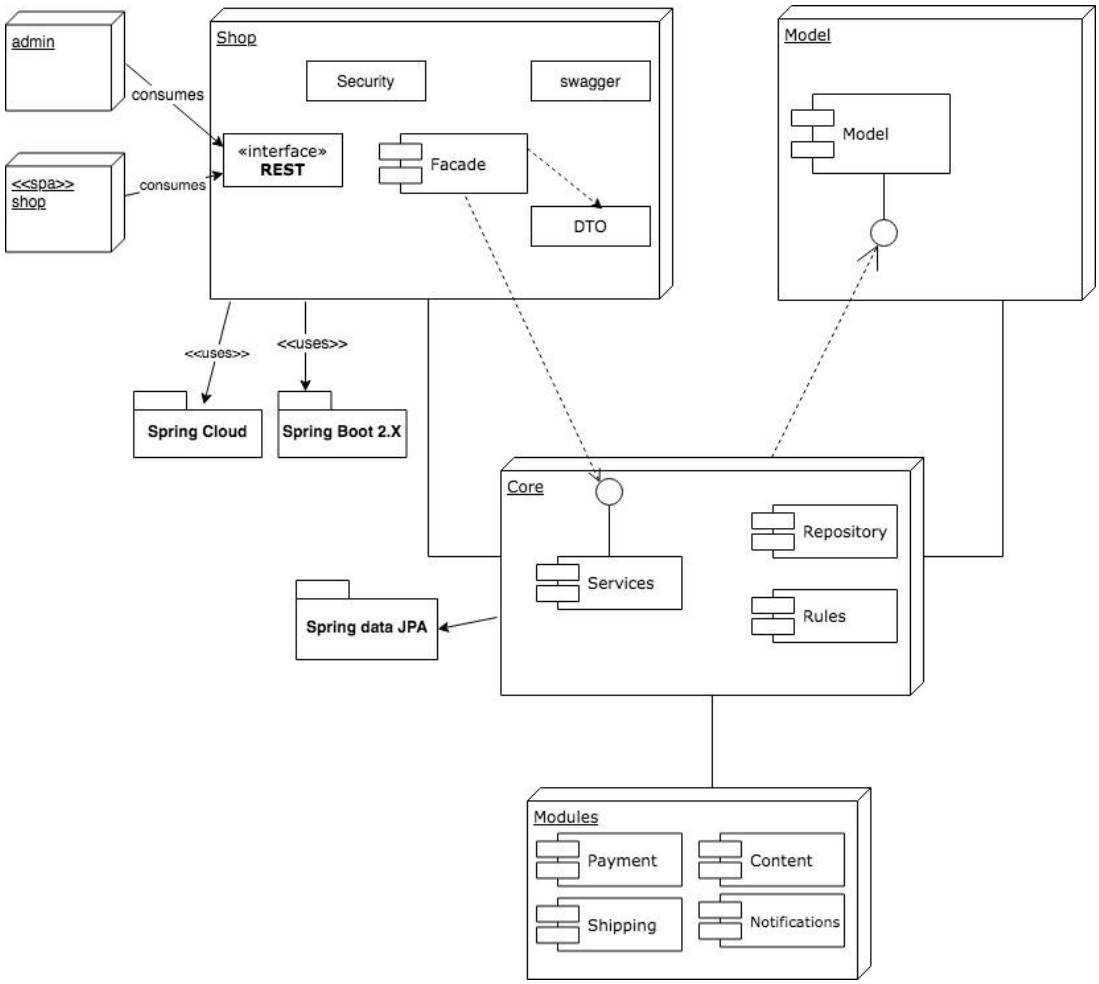
Shop-On has the following users:

Role	Description
superadmin	Can create, modify any user of any stores
admin	Can create, modify any user its pertaining store
admin_store	Can create, modify any user its pertaining store
admin_retail	Can only modify their profile, Deletion is not supported

2. High Level Architecture



3. Components



4. Specification

4.1 Functional Specifications

4.1.1 Customer:

1. Home Page
2. Register Page
3. Login Page
4. Product Details Page
5. Review Page
6. Checkout Page and Summary Page
7. Order Success Page

#	EPIC	FEATURE
1	User Login	Login
2	User Login	Remember Me
3	User Login	Forgot Password
4	User Login	Register
5	Homepage	Offers
6	Homepage	Language Preference
7	Homepage	Check Cart option
9	Homepage	Search for a Product
10	Homepage	Lists of Products
11	Product Detail Page	Details for a product
12	Product Detail Page	Add to cart
13	Product Detail Page	Sort by Option
14	Product Detail Page	Collection option
15	All Pages	Scroll to beginning of the page
16	All Pages	Contact with Admin (Address, Contacts, Social Media links etc..)
17	Personal Info Page	Check your account details
18	Personal Info Page	Change Password
19	Personal Info Page	Check Item Recently purchased
20	Personal Info Page	Billing and Shipping information
21	Personal Info Page	Forgot Password

4.1.2 Admin:

1. Login Screen
2. Home Page
3. Store Page
4. Profile
5. Catalogue
6. Manage Content
7. Shipping
8. Payment
9. Customers
10. Orders
11. Manage taxes
12. Cache Management
13. Configurations

#	EPIC	FEATURE
1	User Page	Login In
2	User Page	Remember Me
3	User Page	Forgot Password
4	Home Page	Store Information
5	Home Page	Recent Orders
6	Home Page	Detail of a Product ordered
7	Store Page	Modify the store details
8	Store Page	Modify the store branding
9	Store Page	Modify the store home page
10	Profile Page	User Details
11	Profile Page	Create a Group for security
12	Profile Page	Change the Account Type
13	Catalogue	Product Catalogue
14	Catalogue	Categories
15	Catalogue	Options Catalogue
16	Catalogue	Product Group
17	Catalogue	List of Manufacturers
18	Manage Content	Manage Pages
19	Manage Content	Manage Boxes
20	Manage Content	Manage Images
21	Manage Content	Manage Files
22	Shipping	Shipping Configuration
23	Shipping	Shipping Method
24	Shipping	Shipping origin

4.2 API

HTTP Request	API	Description
	Catalog management resource (Catalog Management Api)	Catalog Api
POST	POST/api/v1/private/catalog	Create catalog
GET	GET/api/v1/private/catalog/{id}	Get catalog
POST	POST/api/v1/private/catalog/{id}	Add catalog entry to catalog
DELETE	DELETE/api/v1/private/catalog/{id}	Deletes a catalog
PATCH	PATCH/api/v1/private/catalog/{id}	Update catalog
GET	GET/api/v1/private/catalog/{id}/entry	Get catalog entry by catalog
DELETE	DELETE/api/v1/private/catalog/{id}/entry/{entryId}	Remove catalog entry from catalog
GET	GET/api/v1/private/catalog/unique	Check if catalog code already exists
GET	GET/api/v1/private/catalogs	Get catalogs by merchant
	Category management resource (Category Management Api)	Category Api
GET	GET/api/v1/category	Get category hierarchy from root. Supports filtering FEATURED_CATEGORIES and VISIBLE ONLY by adding ?filter=[featured] or ?filter=[visible] or ? filter=[featured,visible
GET	GET/api/v1/category/{id}	Get category list for an given Category id
POST	POST/api/v1/private/category	create
PUT	PUT/api/v1/private/category/{id}	update
DELETE	DELETE/api/v1/private/category/{id}	delete
PUT	PUT/api/v1/private/category/{id}/move/{parent}	Move a category under another category
PATCH	PATCH/api/v1/private/category/{id}/visible	updateVisible
GET	GET/api/v1/private/category/unique	Check if category code already exists
	Content management resource (Content Management Api)	Content Api
GET	GET/api/v1/content/boxes	Get pages summary created for a given MerchantStore
GET	GET/api/v1/content/boxes/{code}	Get box content by code for a code and a given MerchantStore
GET	GET/api/v1/content/folder	folder
GET	GET/api/v1/content/images	Get store content images
GET	GET/api/v1/content/pages	Get page names created for a given MerchantStore
GET	GET/api/v1/content/pages/{code}	Get page content by code for a given MerchantStore
GET	GET/api/v1/content/pages/name/{name}	Get page content by code for a given MerchantStore

GET	GET/api/v1/content/summary	Get pages summary created for a given MerchantStore. Content summary is a content bux having code summary.
POST	POST/api/v1/private/content	Create content (page or box)
DELETE	DELETE/api/v1/private/content/	Deletes a file from CMS
PUT	PUT/api/v1/private/content/{id}	Update content page
DELETE	DELETE/api/v1/private/content/{id}	Deletes a content from CMS
GET	GET/api/v1/private/content/any/{code}	Get page content by code for a given MerchantStore
GET	GET/api/v1/private/contents/any	Get contents (page and boc) for a given MerchantStore
POST	POST/api/v1/private/file	upload
POST	POST/api/v1/private/files	uploadMultipleFiles
	Customer authentication resource (Customer Authentication Api)	Authenticate Customer Api
GET	GET/api/v1/auth/customer/refresh	refreshToken
POST	POST/api/v1/customer/login	Authenticates a customer to the application
PUT	PUT/api/v1/customer/password	Sends a request to reset password
POST	POST/api/v1/customer/password/reset	Change customer password
POST	POST/api/v1/customer/register	Registers a customer to the application
	Customer management resource (Customer Management Api)	Customer Api
PUT	PUT/api/v1/auth/customer/{id}	Updates a loged in customer profile
PATCH	PATCH/api/v1/auth/customer/address	Updates a loged in customer address
POST	POST/api/v1/private/customer	Creates a customer
GET	GET/api/v1/private/customer/{id}	get
PUT	PUT/api/v1/private/customer/{id}	Updates a customer
DELETE	DELETE/api/v1/private/customer/{id}	Deletes a customer
PATCH	PATCH/api/v1/private/customer/{id}/address	Updates a customer
GET	GET/api/v1/private/customer/profile	getAuthUser
GET	GET/api/v1/private/customers	getFilteredCustomers
	Manufacturer / Brand management resource (Manufacturer / Brand Management Api)	Manufacturer Api
GET	GET/api/v1/category/{id}/manufacturers	Get all manufacturers for all items in a given category
DELETE	DELETE/api/v1/manufacturer/{id}	delete
GET	GET/api/v1/manufacturers/	List manufacturers by store
GET	GET/api/v1/manufacturers/{id}	get
POST	POST/api/v1/private/manufacturer	create
PUT	PUT/api/v1/private/manufacturer/{id}	update
GET	GET/api/v1/private/manufacturer/unique	Check if manufacturer code already exists

	Merchant and store management resource (Merchant - Store Management Api)	Merchant Store Api
GET	GET/api/v1/private/merchant/{code}/children	Get child stores
GET	GET/api/v1/private/merchant/{code}/stores	Get retailer child stores
POST	POST/api/v1/private/store	Creates a new store
PUT	PUT/api/v1/private/store/{code}	Updates a store
DELETE	DELETE/api/v1/private/store/{code}	Deletes a store
GET	GET/api/v1/private/store/{code}/marketing	Get store branding and marketing details
POST	POST/api/v1/private/store/{code}/marketing	Create or save store branding and marketing details
POST	POST/api/v1/private/store/{code}/marketing/logo	Add store logo
DELETE	DELETE/api/v1/private/store/{code}/marketing/logo	Delete store logo
GET	GET/api/v1/private/store/unique	Check if store code already exists
GET	GET/api/v1/private/stores	Get list of stores. Returns all retailers and stores
GET	GET/api/v1/store/{code}	Get merchant store
GET	GET/api/v1/stores	Get list of store names. Returns all retailers and stores
	Product attributes /options / options values management resource (Product Option Management Api)	Product Option Api
POST	POST/api/v1/private/product/{id}/attribute	createAttribute
GET	GET/api/v1/private/product/{id}/attribute/{attributeId}	Get product attributes
PUT	PUT/api/v1/private/product/{id}/attribute/{attributeId}	updateAttribute
DELETE	DELETE/api/v1/private/product/{id}/attribute/{attributeId}	deleteAttribute
GET	GET/api/v1/private/product/{id}/attributes	Get product attributes
POST	POST/api/v1/private/product/option	createOption
GET	GET/api/v1/private/product/option/{id}	getOption
PUT	PUT/api/v1/private/product/option/{optionId}	updateOption
DELETE	DELETE/api/v1/private/product/option/{optionId}	deleteOption
GET	GET/api/v1/private/product/option/unique	Check if option code already exists
POST	POST/api/v1/private/product/option/value	createOptionValue
GET	GET/api/v1/private/product/option/value/{id}	getOptionValue
PUT	PUT/api/v1/private/product/option/value/{id}	updateOptionValue
DELETE	DELETE/api/v1/private/product/option/value/{id}	deleteOptionValue
POST	POST/api/v1/private/product/option/value/{id}/image	addOptionValueImage
DELETE	DELETE/api/v1/private/product/option/value/{id}/image	removeOptionValueImage
GET	GET/api/v1/private/product/option/value/unique	Check if option value code already exists
GET	GET/api/v1/private/product/options	options

GET	GET/api/v1/private/product/options/values	optionsValues
	Product groups management resource (Product Groups Management Api)	Product Group Api
POST	POST/api/v1/private/products/{productId}/group/{code}	addProductToGroup
DELETE	DELETE/api/v1/private/products/{productId}/group/{code}	removeProductFromGroup
POST	POST/api/v1/private/products/group	Create product group
PATCH	PATCH/api/v1/private/products/group/{code}	Update product group visible flag
GET	GET/api/v1/private/products/groups	Get products groups for a given merchant
GET	GET/api/v1/products/group/{code}	Get products by group code
DELETE	DELETE/api/v1/products/group/{code}	Delete product group by group code
	Product inventory resource (Product Inventory Api)	Product Inventory Api
GET	GET/api/v1/private/product/{id}/inventory	get
GET	GET/api/v1/private/product/{id}/inventory/{inventoryId}	get
GET	GET/api/v1/private/product/{id}/inventory/store/{code}	get
PUT	PUT/api/v1/private/product/{productId}/inventory/{id}	update
POST	POST/api/v1/private/product/inventory	create
DELETE	DELETE/api/v1/private/product/inventory/{id}	delete
	Product management resource (Product Management Api)	Product Api
PUT	PUT/api/v1/auth/product/{id}	Update product
DELETE	DELETE/api/v1/auth/product/{id}	delete
POST	POST/api/v1/auth/product/{productId}/category/{categoryId}	addProductToCategory
DELETE	DELETE/api/v1/auth/product/{productId}/category/{categoryId}	removeProductFromCategory
POST	POST/api/v1/auth/products	create
POST	POST/api/v1/private/product	create
PUT	PUT/api/v1/private/product/{id}	Update product
DELETE	DELETE/api/v1/private/product/{id}	delete
PATCH	PATCH/api/v1/private/product/{id}	Update product inventory
POST	POST/api/v1/private/product/{productId}/category/{categoryId}	addProductToCategory
DELETE	DELETE/api/v1/private/product/{productId}/category/{categoryId}	removeProductFromCategory
GET	GET/api/v1/private/product/unique	Check if product code already exists
GET	GET/api/v1/products	getFiltered
GET	GET/api/v1/products/{id}	get
	Product type resource (Product Type Api)	Product Type Api
GET	GET/api/v1/products/types	Get product types list

	Search products and search word/sentence completion functionality (Search Api)	Search Api
POST	POST/api/v1/search	search
POST	POST/api/v1/search/autocomplete	autocomplete
	Shipping Quotes and Calculation resource (Shipping Api)	Order Shipping Api
GET	GET/api/v1/auth/cart/{code}/shipping	shipping
POST	POST/api/v1/cart/{code}/shipping	shipping
	User management resource (User Management Api)	User Api
POST	POST/api/v1/private/user/	Creates a new user
PUT	PUT/api/v1/private/user/{id}	Updates a user
DELETE	DELETE/api/v1/private/user/{id}	Deletes a user
PATCH	PATCH/api/v1/private/user/{id}/password	Updates a user password
GET	GET/api/v1/private/user/profile	getAuthUser
POST	POST/api/v1/private/user/unique	Check if username already exists
GET	GET/api/v1/private/users	Get list of user
GET	GET/api/v1/private/users/{id}	Get a specific user profile by user id
	authenticate-user-api	Authenticate User Api
GET	GET/api/v1/auth/refresh	refreshAndGetAuthenticationToken
POST	POST/api/v1/private/login	authenticate
cache-api	cache-api	Cache Api
DELETE	DELETE/api/v1/auth/cache/store/{storeId}/clear	clearCache
contact-api	contact-api	Contact Api
POST	POST/api/v1/contact	Sends an email contact us to store owner
	customer-newsletter-api	Customer Newsletter Api
POST	POST/api/v1/newsletter	Creates a newsletter optin
PUT	PUT/api/v1/newsletter/{email}	Updates a customer
DELETE	DELETE/api/v1/newsletter/{email}	Deletes a customer
	customer-review-api	Customer Review Api
GET	GET/api/v1/customers/{id}/reviews	getAll
POST	POST/api/v1/private/customers/{id}/reviews	create
PUT	PUT/api/v1/private/customers/{id}/reviews/{reviewid}	update
DELETE	DELETE/api/v1/private/customers/{id}/reviews/{reviewId}	delete
	market-place-api	Market Place Api
GET	GET/api/v1/private/marketplace/{store}	Get market place meta-data
	optin-api	Optin Api
POST	POST/api/v1/private/optin	Creates an optin event type definition
	order-api	Order Api
POST	POST/api/v1/auth/cart/{code}/checkout	checkout

GET	GET/api/v1/auth/orders	list
GET	GET/api/v1/auth/orders/{id}	getOrder
POST	POST/api/v1/cart/{code}/checkout	checkout
GET	GET/api/v1/private/orders	getOrders
GET	GET/api/v1/private/orders/customers/{id}	list
	order-payment-api	Order Payment Api
POST	POST/api/v1/auth/cart/{code}/payment/init	init
POST	POST/api/v1/cart/{code}/payment/init	init
POST	POST/api/v1/private/orders/{id}/capture	caprurePayment
GET	GET/api/v1/private/orders/payment/capturable	listCapturableOrders
	order-total-api	Order Total Api
GET	GET/api/v1/auth/cart/{id}/total	payment
GET	GET/api/v1/cart/{id}/total	calculatePayment
	product-image-api	Product Image Api
POST	POST/api/v1/auth/products/{id}/images	uploadImages
DELETE	DELETE/api/v1/auth/products/images/{id}	deleteImage
POST	POST/api/v1/private/products/{id}/images	uploadImages
DELETE	DELETE/api/v1/private/products/images/{id}	deleteImage
	product-relationship-api	Product Relationship Api
GET	GET/api/v1/products/{id}/related	Get product related items. This is used for doing cross-sell and up-sell functionality on a product details page
	product-review-api	Product Review Api
POST	POST/api/v1/auth/products/{id}/reviews	create
PUT	PUT/api/v1/auth/products/{id}/reviews/{reviewid}	update
DELETE	DELETE/api/v1/auth/products/{id}/reviews/{reviewid}	delete
POST	POST/api/v1/private/products/{id}/reviews	create
PUT	PUT/api/v1/private/products/{id}/reviews/{reviewid}	update
DELETE	DELETE/api/v1/private/products/{id}/reviews/{reviewid}	delete
GET	GET/api/v1/products/{id}/reviews	getAll
	product-variant-api	Product Variant Api
GET	GET/api/v1/category/{id}/variants	Get all variation for all items in a given category
POST	POST/api/v1/products/{id}/variant	Get product price variation based on selected options
	public-configs-api	Public Configs Api
GET	GET/api/v1/config	Get public configuration for a given merchant store
	references-api	References Api
GET	GET/api/v1/country	getCountry
GET	GET/api/v1/currency	getCurrency
GET	GET/api/v1/languages	getLanguages

GET	GET/api/v1/measures	measures
GET	GET/api/v1/zones	getZones
	security-api	Security Api
GET	GET/api/v1/sec/private/{group}/permissions	Get permissions by group
GET	GET/api/v1/sec/private/groups	groups
GET	GET/api/v1/sec/private/permissions	permissions
	shopping-cart-api	Shopping Cart Api
POST	POST/api/v1/cart	Add product to shopping cart when no cart exists, this will create a new cart id
GET	GET/api/v1/cart/{code}	Get a chopping cart by code
PUT	PUT/api/v1/cart/{code}	Add to an existing shopping cart or modify an item quantity
DELETE	DELETE/api/v1/cart/{code}/product/{id}	Remove a product from a specific cart
GET	GET/api/v1/customers/{id}/cart	Get a chopping cart by id
POST	POST/api/v1/customers/{id}/cart	Add product to a specific customer shopping cart

4.3 Non-Functional Requirement

The Key Security and Compliance features of the Shop-On system is mentioned below. Periodically additional security measures and features will be incorporated into the system raising the bar for security for all applications in the system. Subsequently document would be updated of the same.

4.1 Performance Testing

As part of performance test for Shop-On application features are currently limited to understand the system behaviour with respect to load, with 200 concurrent users trying to access the application. Shop-On team will perform the load test using JMeter tool.

1) Objective

The main objective of the performance testing is to:

- Evaluate the application scalability as intended
- Identify performance bottlenecks, and provide recommendations
- Evaluate the capability of the underlying infrastructure to host the given application

2) Approach

- Design the test scenarios and test cases
- Build scripts for the identified test cases using JMeter
- Configure test scenarios using JMeter, with all test settings
- Execute the designed test scenarios
- Monitor and collect metrics
- Communicate any issues encountered during the run to all groups involved (If there are major issues noticed during the test, abort or stop test execution on mutual consent and reschedule the test).
- Consolidate the results from the JMeter
- Analyze the logs to identify bottlenecks
- Discuss with the application team for complete analysis and identification of bottlenecks.
- Log defects in TFS wherever applicable.
- Share the results with all stakeholders

4.2 Security Testing

Security Testing is defined as a type of Software Testing that ensures software systems and applications are free from any vulnerabilities, threats, risks that may cause a big loss.

1) Objective

System should ensure the following after successful completion of security testing:

- OWASP top 10 2017
- Baseline
- CWE/SANS top 25

2) Approach

- Security Testing team will use the ZAP for security testing
- Team will do dynamic comprehensive security scanning to increase visibility and better understanding of enterprise risks.
- Team will be performing Spider Checking and Vulnerability scanning of the Application. Security team will share the report and analyse along with the team.

4.3 Compatibility Testing

Compatibility Testing is defined as a type of Software Testing that ensures software systems and applications will work on 2 browsers, latest 2 versions in Chrome and IE.

1) Objective

The main objective of the compatibility testing is to:

- Evaluate the application is compatible in 2 browsers.
- Identify the issues and provide recommendations.
- Evaluate the capability of the application to work on the 2 browsers

2) Approach

- Compatibility Testing team will use the latest 2 versions of the browsers.
- Team will share the report and analyse along with the team members.

5. Deliverables

5.1 Test Plan

The document should consist of the following points:

1. Introduction
2. Testing Objective
3. Testing scope
4. Test Strategy
5. Testing tools
6. Defect Management
7. Test Schedule
8. Risk contingencies (If any)
9. Definitions and Acronyms (if any)

5.2 Functional Test Scenarios and Test Cases

Quality test scenarios and cases need to be defined, test cases to be written in detail.

5.3 APIs Test Scenarios and Test Cases

Quality test scenarios and cases need to be defined, test cases to be written in detail.

5.4 Defect log in Azure Devops

Defects to be logged in the Azure DevOps and linked to the test cases.

5.5 Test Summary Report

The document should consist of the following points:

1. Purpose of the document
2. Testing scope
3. Metrics
4. Types of testing performed
5. Test Environment
6. Definitions, Abbreviation and Acronyms
7. Automation Framework
8. Automation Script

5.6 Automation Test Suite

Automated test scripts to be created for both Functional and API test cases, where implement -

1	OO Design
2	Unit Testable Code
3	Understand the use case.
4	Create Test Scenario and Test Cases.
5	Implement PageObjects framework.
6	Use TestNG Annotation – @DataProvider, @BeforeSuite, @AfterSuite, @BeforeTest, @AfterTest, @BeforeMethod, @AfterMethod, and @Test.
7	Waiting for the next page to load and process its elements using implicit and explicit wait.
8	Prepare an XLS sheet for data-driven testing.
9	Read the data from XLS sheet.
10	Proper code comments across project – Compliance to Standards and Best Practices
11	Performance report of the Test Scripts
12	Resource Utilization report of the Scripts
13	Testing security aspects

6 Conclusion

Shop-On v1.0 document provides all the necessary information regarding scope of Version 1 release. This document would be updated upon every major and minor release to customers. Given the growing needs of the system.