Requirements Definition Document

Connected Commerce (ROPO)

– MDNI 2014-2015

Author: Kishor Chockalingam

Date: 08 October 2014

Version: 0.1

# Document Control

**Author**

| Position | Name | Contact no |
| --- | --- | --- |
| Tech Arch Associate Manager | Kishor Chockalingam | 9008780046 |

**Revision history**

| Version | Issue date | Author/editor | Description/Summary of changes |
| --- | --- | --- | --- |
| 0.1 | 08 Oct 2014 | Kishor Chockalingam | Created template |
| 0.2 | 08 Oct 2014 | Kishor Chockalingam | Created requirements FR3.1.1,FR3.1.2,HSR3.3.1,HSR3.3.2,HSR3.3.3,HSR3.3.4,HSR3.3.5,HSR3.3.6,TR3.2.1,TR3.2.2 |
| 0.3 | 13 Oct 2014 | Kishor Chockalingam | Added the Functional representation diagram to introduction section. |
| 0.4 | 16 Oct 2014 | Neeraj Shah | Added functional requirement FR 3.1.6 |
| 0.5 | 17 Oct 2014 | Kishor Chockalingam | Added high level function requirement FR3.1.7 & FR3.1.8 for Google glass. |
| 0.6 | 3rd Nov 2014 | Kishor Chockalingam | Updated FR3.1.2 |

**Reviewed by**

| Version | Issue date | Name | Position | Review date |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Approvals**

Approval refers to the approver’s acceptance of the content and overall intention of this document, including acceptance of any commitments described in order to successfully deliver the initiative. The approver, where relevant, also confirms that this document complies with relevant strategies, policies and regulatory requirements.

| Version | Issue date | Name | Position | Approval date |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

# Table of Contents

[1. Introduction 5](#_Toc401325899)

[2. Current System 5](#_Toc401325900)

[3. Requirements 5](#_Toc401325901)

[3.1 Functional Requirements 6](#_Toc401325902)

[3.2 Technical Requirements 7](#_Toc401325903)

[3.3 System Requirements 8](#_Toc401325904)

[3.3.1 Hardware / Software 8](#_Toc401325905)

[3.4 Non- Functional Requirements 8](#_Toc401325906)

[4. Constraints 8](#_Toc401325907)

[5. Assumptions 8](#_Toc401325908)

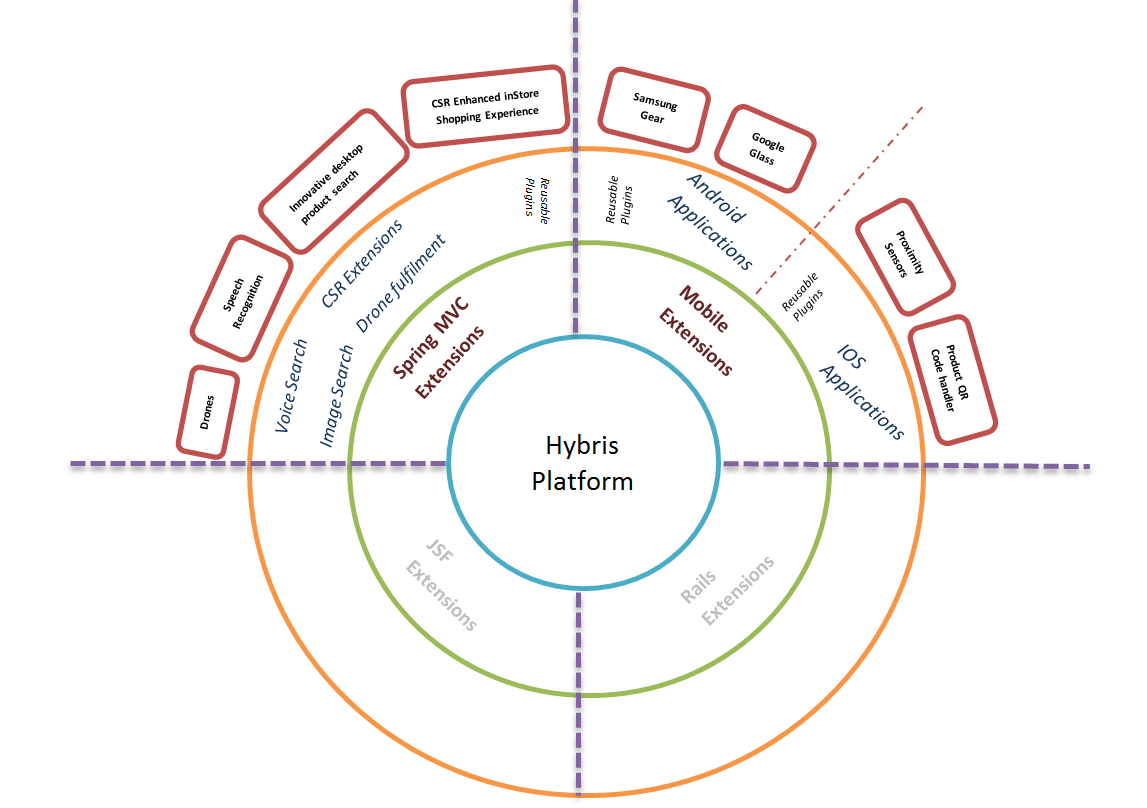
[6. Dependencies 8](#_Toc401325909)

[7. Risks 9](#_Toc401325910)

[8. References 9](#_Toc401325911)

# Introduction

**Connected** **Commerce** is an initiative to create and Architect, reusable framework that would enable e-commerce platforms to interact with connected devices like iBeacon, Galaxy Gear, Nike fuel-band, Google glass etc. This framework shall be re-used by the delivery teams to accelerate their development effort. This document contains detailed use cases that will help in creating independent demonstrable features. These features shall be developed as portable plug-ins, which can be re-used while developing any web/mobile application appropriately.



# Current System

The existing system allows the user to browses a product at home PC or Mobile and show interest in a product but does not buy it. When the same customer visits the store, the mobile application receives command from proximity sensors (iBeacon) to alert a Store representative (CSR) about the preference and choice of that customer. This will enable CSR to make more personalized sales pitch and would also help him close deal faster.

# Requirements

All requirements are defined in point form and are rated either Mandatory (M) or Highly Desirable (HD) or Desirable (D), dependent on business need.

## Functional Requirements

| **ID** | **Requirement** | **Preference** |
| --- | --- | --- |
|  | Get the current system ready for demo. | M |
|  | QR code reader to be integrated with iPhone app:-   1. User scans the QR code off a product from inside the store, using an iPhone (app) 2. The iPhone app, receives the product’s pdp url from the QR code and loads that products PDP within the app 3. Customer using the app, proceeds from PDP page to place the order 4. Once the order is placed:    1. The app allows the user to choose from available shipping methods    2. Customer chooses inStore pickup and places order.    3. The app generates a secret unique customer & order identifier [UCOID] on the fly based on order id, customer id, DOP, etc. and shares it to the customer    4. CSR agent receives the order in to the inStore order queue.    5. CSR agent can re-view order details and customer details.    6. CSR agent packs the product(s) in the order for hand over to the customer. And updates the order status as ready for store pickup.    7. The app notifies the customer to collect the product(s) from a particular counter inside the store. 5. The CSR agent generates invoice, packs the product and updates the order status to ready\_for\_pickup\_over\_the\_counter 6. Customer approaches the counter as per the app notification and asks the CSR agent for the product and provide him with the UCOID 7. The CSR agent keys in the UCOID in the CSR web application, to retrieve the order details, then hands over the correct product(s) to the customer 8. The CSR agent marks the order status to fulfilled\_over\_the\_counter   Fig 3.1.2.1 QR Code Scan Architecture  C:\Git-Repository\PRY-CC\docs\arch\Finished\UC 1 Architecture.png | HD |
|  | 1. Customer does his routine exercise every day 2. Store’s NATIVE IOS / Android (store)application installed on the customer's phone receives info (heartbeat, miles run, calories burnt, etc. ) from the wrist bands.    * assumed that the customer wears the wrist band and have our app installed on his smart phone (which is in his pocket) when doing exercise. 3. Store’s NATIVE IOS / Android application does a real time sync of all health data collected, to the back end ecomm web application (currently Hybris) 4. Based on the health data of the customer,    * The NATIVE IOS / Android app fetches promotional messages (tailored for this customer) and notifies customer on their smart phone. (in the notification area)    * The web application sends promotional content via email. 5. Customer can purchase via native app/on-line by applying the promotions made available to them through email or notifications.   (OR)   1. Customer walks into store,    * NATIVE IOS / Android app senses iBeacons inside the store and pulls customer tailored(health related) (and/or) Store specific promotional messages and notifies the customer (in the notification area) 2. Customer uses the NATIVE IOS / Android app to request directions to the promoted product. 3. the NATIVE IOS / Android app opens up the store map and guides the user from his present location to the product on shelf (using multiple iBeacons located inside the store) 4. Continued in use case FR3.2.1 | HD |
|  | Voice search for products:-   1. Customers who are accessing the store web application over a laptop can give input to the product search text box through voice. 2. Customers who are accessing the store IOS application over a iPhone can give input to the product search text box through voice.   (To be made in detail) |  |
|  | Locate product inside the store:-   1. User views pdp page of a product in a mobile device. 2. The pdp page provides an option to locate the product inside the store 3. Customer chooses that option 4. An store map opens and locates the customer’s position inside the store using proximity sensors (iBeacons) installed inside the stores. 5. The app provides step by step navigation to the user to the product location shelf   (To be made in detail) | D |
|  | Image search for products:-   1. Customers who are accessing the store web application over a laptop can give input to the product search text box by dragging and dropping a product image into it. | D |
|  | Order delivery by Drone:-  The details of the order places by the customer will be supplied to mobile app by Ecommerce. This mobile app will communicate the delivery address of the order (received from ecommerce) to the Drone that will finally deliver the order.   1. Customers accessing the store web application places order using Ecommerce. 2. Ecommerce exposes the order details (Billing/Shipping/location etc.) as web service. 3. Mobile app (IOS/Android) retrieves the order details by communicating with Ecommerce web service. 4. This mobile application will provides interface to fly drone using the GPS & Internet connection present in the mobile. 5. The communication between mobile application and drone takes place using inertial navigation system (INS) of the drone combined and mobile GPS. 6. The drone delivers the order at the delivery address location retrieved from mobile application. |  |
|  | The inbuilt camera in Google glass can be used to take picture of products on shelf (inStore) and sent to native application on an android phone. This native application can process the product image (using image recognition algorithms):   1. Retrieve the product details, 2. product rating & reviews, 3. Price comparisons, 4. Facilitate the purchase process online (or) via the app itself. 5. Share product order details with friends via email or social media   (To be made in detail) |  |
|  | Campaign through push notification (to google glass):   1. As customer walks or cycles by the store 2. Proximity sensors sense the google glass 3. Marketing messages like products on discount or special offers can pop inside the google glass of the wearer.   (To be made in detail) |  |

## Technical Requirements

| **ID** | **Requirement** | **Preference** |
| --- | --- | --- |
|  | Requirement FR2.1.3 shall be implemented using Google tool kit |  |
|  | Requirement FR2.1.3 (part 1) shall be implemented as a plug and use component. Shall be developed in a way that it can be integrated with ATG, Hybris & WCS in future. |  |
|  |  |  |

## System Requirements

### Hardware / Software

| **ID** | **Requirement** | **Preference** |
| --- | --- | --- |
|  | iPhone 5s – 2 nos | M |
|  | Unix server on the cloud with full access | D |
|  | Wifi (dongle) with internet connection | D |
|  | iPads for CSR agent | D |
|  | Laptop – 1 nos (min) | M |
|  | iBeacon instrument – 1 nos | D |
|  | Eclipse | M |
|  | Hybris 5.0 | M |
|  | VLan for developer machines – required for SVN server setup and use. | HD |
|  | Samsung Gear – watch | D |
|  | Google Glass | D |
|  | Ready to fly Drone device & Kit | D |
|  | Mac book or iMac to develop IOS apps | M |

## 

## Non- Functional Requirements

| **Requirement ID** | **Non functional requirement** | **Preference** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# Constraints

Constraints that inhibit the project’s ability to deliver the requirements

| **Requirement ID** |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Assumptions

| **Requirement ID** |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Dependencies

| **Requirement ID** |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Risks

| **Requirement ID** |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# References