**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW OF MAGENTO 2.0 FRAMEWORK**

Magento Community Edition 2.0 provides online merchants with unparalleled flexibility and control over the look, content, and functionality of their online stores. Magento’s intuitive Admin features powerful marketing, search engine optimization, and product management tools that give you the power to create sites that are tailored to your unique business needs. Robust and scalable, Magento 2.0 offers you a stable, secure, and customizable solution for your growing business.

It is packed full of great features and allows for almost unlimited flexibility and customization; however, the Magento code base is vast and there is very little current documentation on developing custom extensions. Prior to this book many developers come into the Magento world and immediately find themselves puzzled by a shortage of documentation. The process that many developers take at the beginning usually begins by Google search which almost always turns up a stale blog post on how to implement a specific customization to Magento. Upon further searching developers quickly realize that there is a lack of good guides on how to get started creating a new extension.

**1.2 OVERVIEW OF IONIC FRAMEWORK**

Ionic is a front-end HTML framework built on top of AngularJS and Cordova. As per their official document, the definition of this Ionic Open Source Framework is as follows:

Ionic is an HTML5 Mobile App Development Framework targeted at building hybrid mobile apps. Think of Ionic as the front-end UI framework that handles all the look and feel and UI interactions your app needs to be compelling. Kind of like "Bootstrap for Native", but with the support for a broad range of common native mobile components, slick animations and a beautiful design.

Following are the most important features of Ionic:

* AngularJS: Ionic is using AngularJS MVC architecture for building rich single page applications optimized for mobile devices.
* CSS Components: With the native look and feel, these components offer almost all elements that a mobile application needs. The components’ default styling can be easily overridden to accommodate your own designs.
* JavaScript Components: These components are extending CSS components with JavaScript functionalities to cover all mobile elements that cannot be done only with HTML and CSS.
* Cordova Plugins: Apache Cordova plugins offer API needed for using native device functions with JavaScript code.
* Ionic CLI: This is NodeJS utility powered with commands for starting, building, running and emulating Ionic applications.
* Ionic View: Very useful platform for uploading, sharing and testing your application on native devices.
* License: Ionic is released under MIT license.

**1.3 OBJECTIVE**

The Objective of the proposed system is in recent years, mobile e-commerce is developing steadily. Its development has been making new open doors for all types of businesses to derive full benefit from. According to the Forrester Research report, “By 2017, mobile will represent 9 percent of ecommerce sales.” Moreover, Adobe recommends that shoppers will in all probability buy through mobile apps.

**1.4 MOTIVATION**

Mobile apps make an energizing market opportunity for online organizations. As it has potential to build a seamless shopping experience that will improve customer engagement. So, being an online store owner, you have to increase your market share by embracing a mobile strategy that not only satisfies your customers’ needs but also checking the endeavors of your rivals.

**1.5 THESIS ORGANIZATION**

Including this introductory chapter, this thesis contains eight chapters. The report is structured as follows; Chapter 2 presents environment setup for Mageionic. Chapter 3 presents the api roles on magento and ionic frameworks. Chapter 4 explains the proposed system and overview of the system design. Chapter 5 describes the detailed description of the implementation. Chapter 6 describes about testing activity. Chapter 7 presents the results and discussion of the project. Chapter 8 summarizes the contribution of this thesis. It also provides a number of useful directions for future research on the presented work.

**CHAPTER 2**

**PREPARING MAGEIONIC ENVIRONMENT FOR DEVELOPMENT**

**2.1 INSTALLING MAGENTO FRAMEWORK**

Magento is an ecommerce platform built on open source technology which provides online merchants with a flexible shopping cart system, as well as control over the look, content and functionality of their online store. Magento offers powerful marketing, search engine optimization, and catalog-management tools.

**2.1.1 System Requirements**

**Table 2.1. Magento System Requirements**

|  |  |
| --- | --- |
| **Environment** | |
| **Operating System** | Linux x86-64 or Windows x86-64 or Mac |
| Composer | Composer is required for developers who want to contribute to code  Base, or develop extensions. |
| Web Server | **Apache 2.2 or 2.4** The apache mod\_rewrite module must be enabled. To learn more, see: Apache.  **Nginx 1.8.x** (or latest stable version) |
| PHP | PHP 7.0.2 (Magento 2.0.1 and later only)  PHP 5.6.x (or latest stable version)  PHP 5.5.x (or latest stable version) |
| Database | MySQL 5.6.x (Oracle or Percona)  Magento Enterprise Edition 2.0 can use three master databases to Provide scalability for the different functional areas of checkout, orders, and product data. |
| SSL | A valid security certificate is required for HTTPS. Self-signed certificates are not supported. |
| Mail Server | Mail Transfer Agent (MTA) or SMTP server |

**2.1.2** **Installation**

The Setup Wizard is a multi-page wizard that enables you to go back and forward one page at a time. You cannot skip pages, and you must enter all required information on every single page before you can proceed to the next page.

* Enter the following URL in your browser address bar:

<http://www.example.com/setup>

* On the initial page, click **Agree and Set Up Magento**.

Continue with the following topics in the order presented to complete the installation.

**Step 1: Readiness Check**

Click **Start Readiness Check**. If any errors are displayed, you must resolve them before you continue. Click more detail if available to see more information about each check.

Click **Next**

**Step 2: Add a Database**

Fill database information then click next

**Step 3: Web Configuration**

Enter the following information:

Your Store Address: [http://www.example.com](http://www.example.com/)

Magento Admin Address: Enter the relative URL by which to access the Magento Admin. e.g. secret, backend

Then click next

**Step 4: Customize Your Store**

* From the Store Default Time Zone list, click the name of your store’s time zone.
* From the Store Default Currency list, click the default currency to use in your store.
* From the Store Default Language list, click the default language to use in your store.
* Expand Advanced Modules Configuration to optionally enable or disable modules before you install the Magento software.

### Step 5: Create Admin Account

Now enter admin information such as

* New Username
* New E-Mail
* New Password
* Confirm Password
* Then click Next

### Step 6: Install

After completing all previous steps in the Setup Wizard, click Install Now.

Installation Success The message Success will be displayed to indicate a successful installation.

Now go to the frontend and backend to see the result

**Frontend**

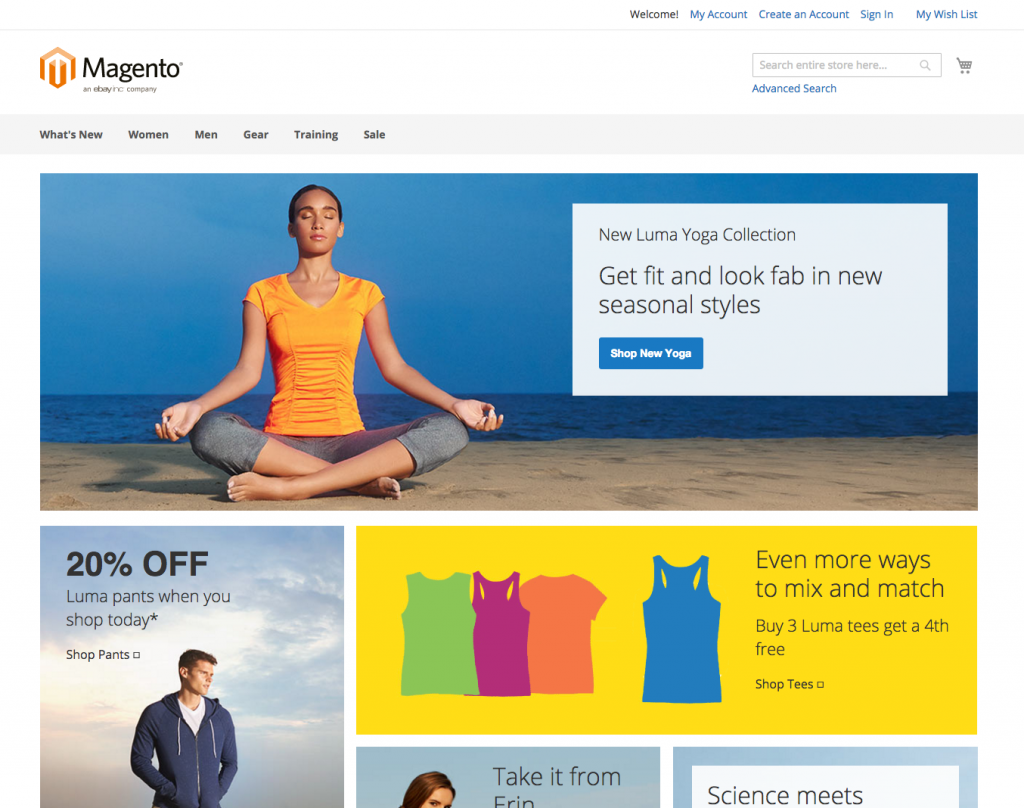


Figure 2.1. Magento Landing Page

**Backend**

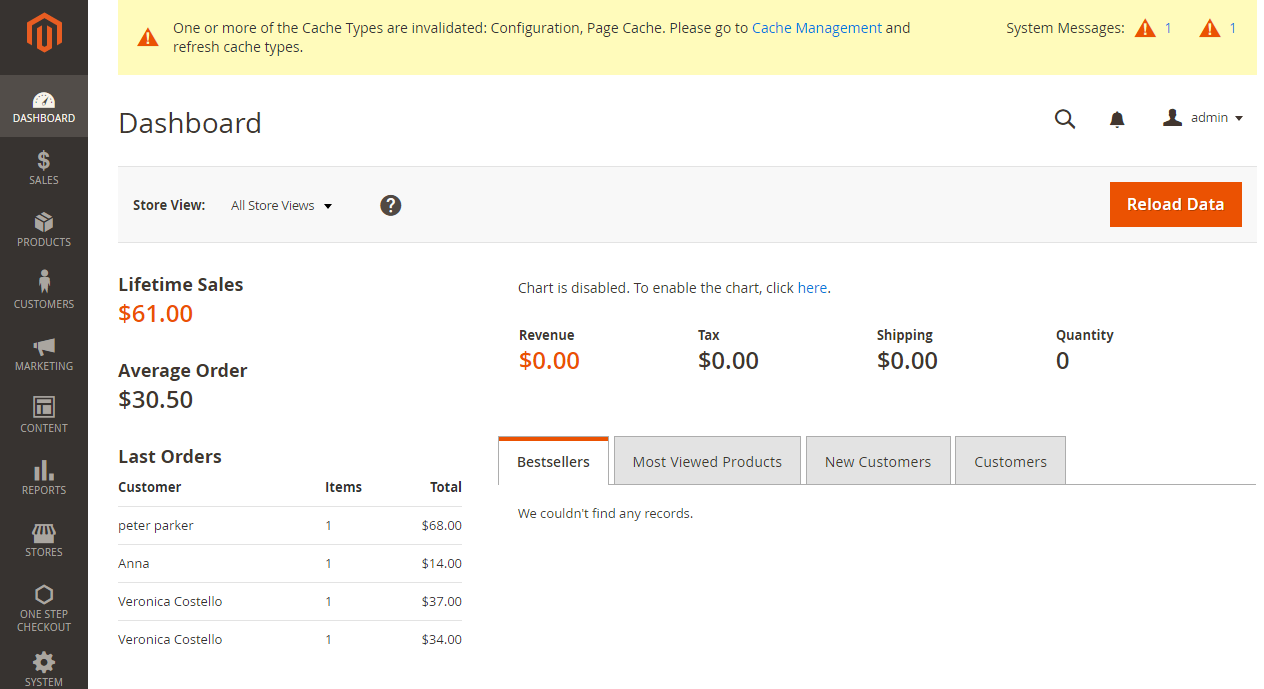


Figure 2.2. Magento Admin Page

**2.2 INSTALLING IONIC FRAMEWORK**

Ionic apps are created and developed primarily through the Ionic command line utility (the “CLI”), and use Cordova to build/deploy as a native app. This means we need to install a few utilities to get developing.

**2.2.1 System Requirements**

**Table 2.2 Ionic System Requirements**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Software** | **Description** |
| 1 | Node Js | This is the base platform needed to create Mobile Apps using Ionic. You can find detail on the NodeJS installation in our NodeJS Environment Setup. Make sure you also install npm while installing NodeJS. |
| 2 | Android SDK | If you are going to work on a Windows platform and are developing your apps for the Android platform, then you should have Android SDK setup on your machine. The following link has detailed information on the Android Environment Setup. |
| 3 | XCode | If you are going to work on the Mac platform and are developing your apps for the iOS platform, then you should have XCode setup on your machine. The following link has detailed information on  the iOS Environment Setup. |
| 4 | Cordova & Ionic | These are the main SDKs which is needed to start working with Ionic. This chapter explains how to setup Ionic in simple  step assuming you already have the required setup as explained in the table above. |

**2.2.2 Ionic Framework Advantages**

Following are some of the most commonly known Ionic Framework Advantages:

* Ionic is used for Hybrid App Development. This means that you can package your applications for IOS, Android, Windows Phone and Firefox OS, which can save you a lot of working time.
* Starting your app is very easy since Ionic provides useful pre-generated app setup with simple layouts.
* The apps are built in a very clean and modular way, so it is very maintainable and easy to update.
* Ionic Developers Team have a very good relationship with the Google Developers Team and they are working together to improve the framework. The updates are coming out regularly and Ionic support group is always willing to help when needed.

**2.2.3 Ionic Framework Limitations**

Following are some of the most important Ionic Framework Limitations:

* Testing can be tricky since the browser does not always give you the right information about the phone environment. There are so many different devices as well as platforms and you usually need to cover most of them.
* It can be hard to combine different native functionalities. There will be many instances where you would run into plugin compatibility issues, which leads to build errors that are hard to debug.
* Hybrid apps tend to be slower than the native ones. However, since the mobile technologies are improving fast this will not be an issue in the future.

In the next chapter, we will understand the environment setup of the Ionic Open Source Framework.

**CHAPTER 3**

**API ROLES IN MAGEIONIC**

**3.1** **MAGENTO WEB API’S**

The Magento web API framework provides integrators and developers the means to use web services that communicate with the Magento system. Key features includes:

* Support for both [REST](https://devdocs.magento.com/guides/v2.2/rest/bk-rest.html) (Representational State Transfer) and [SOAP](https://devdocs.magento.com/guides/v2.2/soap/bk-soap.html) (Simple Object Access Protocol). In Magento 2, the web API coverage is the same for both REST and SOAP.
* Three types of [authentication](https://devdocs.magento.com/guides/v2.2/get-started/authentication/gs-authentication.html):

1. Third-party applications authenticate with [OAuth 1.0a](https://devdocs.magento.com/guides/v2.2/get-started/authentication/gs-authentication-oauth.html).
2. Mobile applications authenticate using [tokens](https://devdocs.magento.com/guides/v2.2/get-started/authentication/gs-authentication-token.html).
3. Administrators and customers are authenticated with  [login credentials](https://devdocs.magento.com/guides/v2.2/get-started/authentication/gs-authentication-token.html).

**3.2** **REST API**

You must register a web service on Magento Admin. Use the following general steps to set up Magento to enable web services.

1. If you are using token-based authentication, create a web services user on Magento Admin by selecting  **System > All Users > Add New User**. (If you are using session-based or OAuth authentication, you do not need to create the new user in the Admin.)
2. Create a new integration on Magento Admin. To create an integration, click **System** > **Integratio**n > **Add New Integration**. Be sure to restrict which resources the integration can access.
3. Use a REST or SOAP client to configure authentication.

In MageIonic, API’s are written in php, in order to provide compatibility with magento backend.

**3.3** **WORKING OF REST API**

The framework used in the proposed system contains following files,

1. config.js,
2. app.js,
3. controller.js,
4. services.js.

Each mentioned files have a specific responsibility and subsequently we shall discuss about the working of REST API here.

**3.3.1** **Config.js**

The config.js provides an advanced configuration mechanism for managing magento site URL, passkey to access that particular site which is defined in the URL constraint. Mageionic fetches this config.js file and sets the URL for the mobile app into the specified URL for fetching contents and products. This config.js file in the framework also takes care about defining constants (i.e.) reference for language list of encoding scheme.

This config.js file includes classes that applies the action on the class name which was declared in app.js file. The main usage scenario when you might prefer to use is the setting up of the default configurations. The config.js file was located in the www directory of project structure. It also provides the option for Enabling and disabling the app featured plugins, themes, Payment plugins, etc. These dependencies are reflected in the configuration data used by the Mageionic (although some dependencies may not be visible as configuration data, but rather be a function of programmatic interactions between ionic and magento).

**3.3.2** **App.js**

Mapping between the files are the important thing to follow the proper flow of Execution of any product. In Mageionic app.js is designed to provide the switching between tabs of the app and mapping to the proper controller pages of the project. In our framework, switching between the tabs and fetching the product from the backend accordingly is the main thing in the perfect flow of app execution:

* The app.js is a JavaScript file which was created as an instance of Angular Module.
* The [angular.module](https://docs.angularjs.org/api/ng/function/angular.module" \t "_blank) is a global place for creating, registering, and retrieving Angular modules. Modules are containers for the various parts of an application, such as controllers and directives.
* It is possible to load the services into the app when you are done the module declaration.
* It is necessary to mention the service names after the module declaration in Mageionic to route the correct controller for correct services .

**3.3.3** **Controller.js**

Handling the control between to different frameworks was done by Controller.js file.

It plays the intermediate role between magento and ionic for transferring request from customer through mobile app and passes that request to the magento backend, which was declared in the app.js file.

It passes the request from the mobile frontend to Services.js to attain the specific requested service.

**3.3.4** **Services.js**

Collecting data from frontend and processing at backend is the core work of services.js file. There are several elements that can be passed, including services, service namespaces, parameters to the services, actions. The Services.js file is the core communication file for the framework and it should be present in the **www** directory of a Mageionic project.

**Major responsibilities of Services.js is follows:**

* Collect the request from the controller.js and frontend of the Mageionic project.
* Process the request and generates the respected result as a json file at the backend server (i.e.) Magento Server.

Converts the json file which was returned by the backend server using **Stringfy** methods of php and placing the results accordingly to the mobile app screen is the responsibility of Controller.js file.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 PROPOSED SYSTEM**

Ionic is an HTML5 mobile app development framework targeted at building hybrid mobile apps. Hybrid apps are essentially small websites running in a browser shell in an app that have access to the native platform layer. Hybrid apps have many benefits over pure native apps, specifically in terms of platform support, speed of development, and access to 3rd party code. Think of Ionic as the front-end UI framework that handles all of the look and feel and UI interactions your app needs in order to be compelling. Kind of like “Bootstrap for Native,” but with support for a broad range of common native mobile components, slick animations, and beautiful design. Unlike a responsive framework, Ionic comes with very native-styled mobile UI elements and layouts that you’d get with a native SDK on iOS or Android but didn’t really exist before on the web. Ionic also gives you some opinionated but powerful ways to build mobile applications that eclipse existing HTML5 development frameworks. Since Ionic is an HTML5 framework, it needs a native wrapper like Cordova or PhoneGap in order to run as a native app. We strongly recommend using Cordova proper for your apps, and the Ionic tools will use Cordova underneath. The diagram for the Proposed System given in Figure 4.1

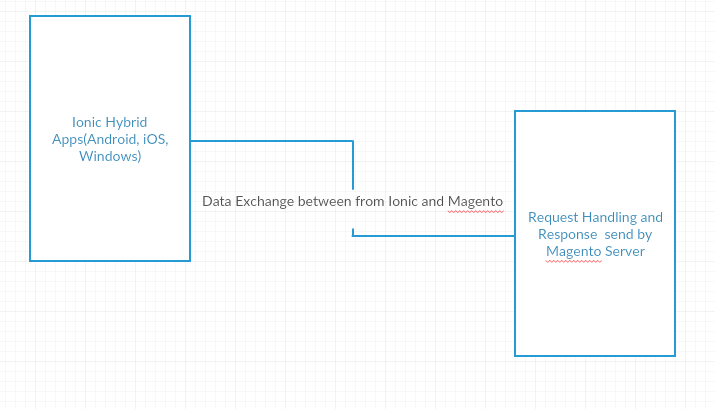
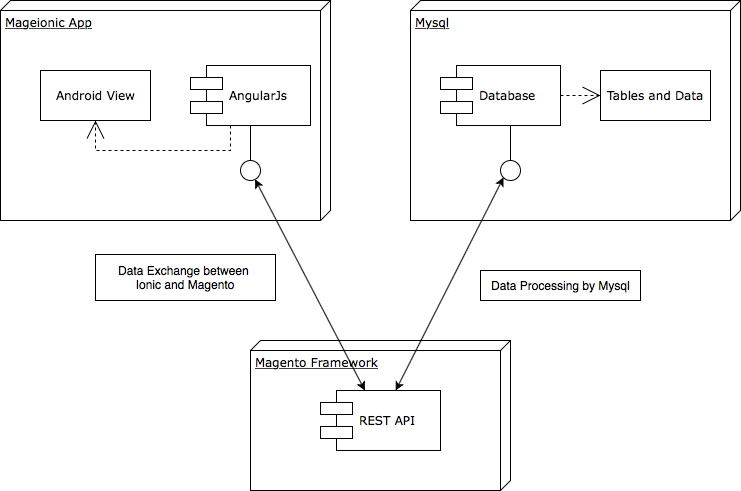


Figure 4.1 Proposed System

**4.2 OVERVIEW OF THE PROJECT**

Mobile apps make an energizing market opportunity for online organizations. As it has potential to build a seamless shopping experience that will improve customer engagement. So, being an online store owner, you have to increase your market share by embracing a mobile strategy that not only satisfies your customers’ needs but also checking the endeavors of your rivals. The architecture diagram for the proposed system is given in Figure 4.2.



**Figure 4.2.1: Architecture of the proposed system.**

The above architecture represents the Role of REST API between mageionic app and magento’s backend. Requests and subsequent response to the mageionic app is passes through AngularJS, also in the magento site it flows till the database operations.

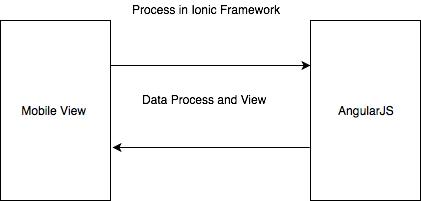
****

Figure 4.2.2. Product view and process

Ionic uses AngularJS for frontend technology that is common for almost all the platforms. So the data from the frontend as request and response from backend are processed by AngularJS and also angular takes responsibility of proper arrangement of components and data in the mobile view.

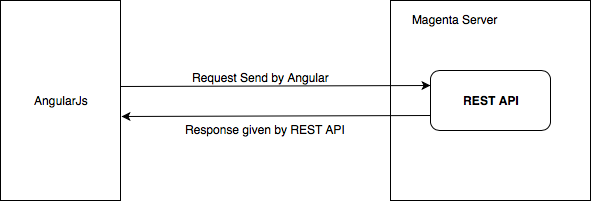
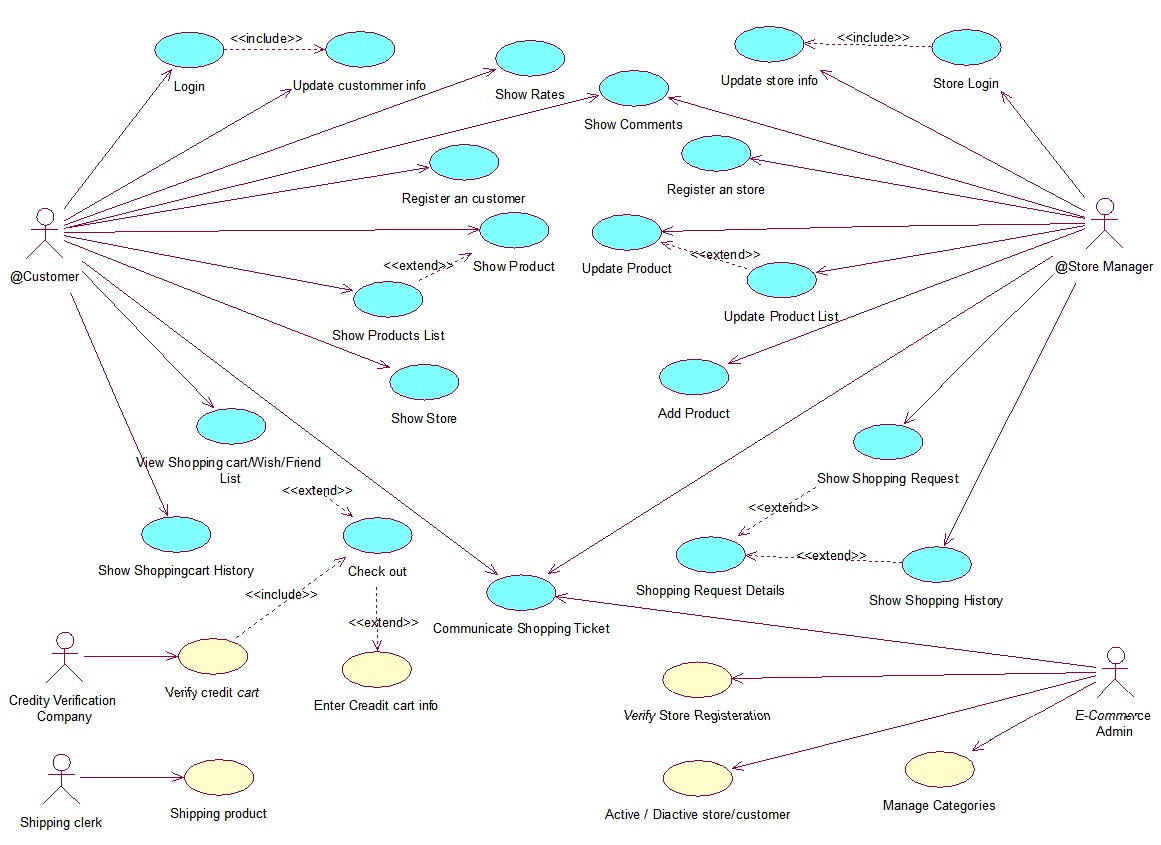
****

Figure 4.2.3. Data processing between AngularJs to REST

**4.3 USE-CASE DIAGRAM**

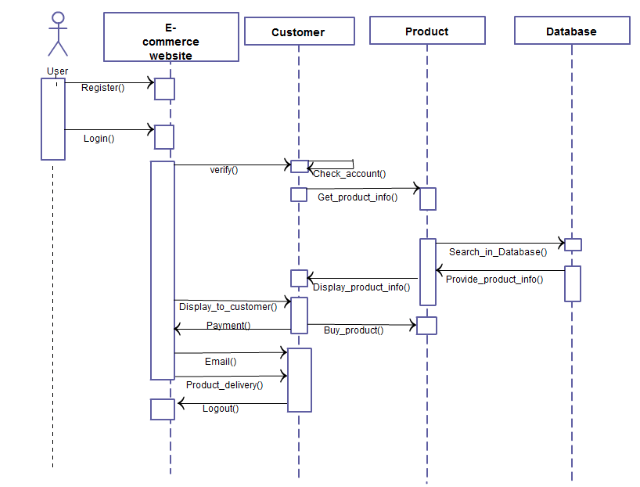
Use cases are used during the analysis phase of a project to identify and partition system functionality. They separate the system into actors and use cases. Actors represent roles that can are played by users of the system. Those users can be humans, other computers, pieces of hardware, or even other software systems. The only criterion is that they must be external to the part of the system being partitioned into use cases. They must supply stimuli to that part of the system, and the must receive outputs from it.

Use cases describe the behavior of the system when one of these actors sends one particular stimulus. This behavior is described textually. It describes the nature of the stimulus that triggers the use case; the inputs from and outputs to other actors, and the behaviors that convert the inputs to the outputs. The text of the use case also usually describes everything that can go wrong during the course of the specified behavior, and what remedial action the system will take. The Figure 4.3 depicts the use case diagram of the proposed system.

**Figure 4.3 Use case diagrams for proposed system**

Represents the E-Commerce service provided by Mageionic.

**4.4 SEQUENCE DIAGRAM**

****

**Figure 4.4 Sequence Diagram of proposed system**

As shown in Figure 4.4, the proposed system enhances the E-Commerce solution to the customer, from the login or register to product delivery in Mageionic.

The above sequence diagram shows the flow of process for a checkout of the any product.

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 MODULE DESCRIPTION**

**5.1.1 Sample E-Commerce store Creation using Magento Framework**

Magento is the most superior CMS for building stores in terms of security, smoothness and much more. The framework follows MVC pattern (i.e.) Model-View-Controller pattern. It has three components; Model, View and Controller. Model summarizes the core application data and functionality. View displays information to client on the screen. Controller handles the interaction between user interface and user inputs, and initiates the creation of the application’s view. Here in Magento MySQL will be used for Model; PHP is used as Controller and Web technologies for View. This ordering need to be remain same for the entire application, PHP, MySQL and Web technologies cannot be interchangeably used for Controller, Model, or View.

**Setting up Multiple Stores**

[Setting up Multiple Stores Magento 2](https://www.mageplaza.com/kb/how-to-setup-multiple-stores-magento-2.html) is one of the variable functionalities Magento 2 brings to store owners. This feature allows us to **create stores** as many as we need only in the single configuration. All stores share the same domain, be set up as subdomains of the same domain, or have entirely different domain, yet possibly share the same backend for the easier administration.

The very first step is to determine how you want to place the store. Will the stores share the same domain? or will they have their own subdomain, or will they have distinctly different domains? For each store, do one of the following:

* To place the store one level below the primary domain, you don’t have to do anything.
* Set up a subdomain of your primary domain.
* Set up a different primary domain.
* On the Admin Panel, **Stores > Settings > All Stores.**
* Click on **Create Store**, then follow the below
  + Select the **Website** that is the parent of the new store. If there is only one website, use the default “Main Website”.
  + Fill the **Name** for the new store, just for the internal reference.
  + In the **Root Category** field, you will set the [root category](https://www.mageplaza.com/kb/how-create-a-new-root-category-in-magento-2.html) for the main menu of the new store. The created root categories are shown in the dropdown list, select one for the store.

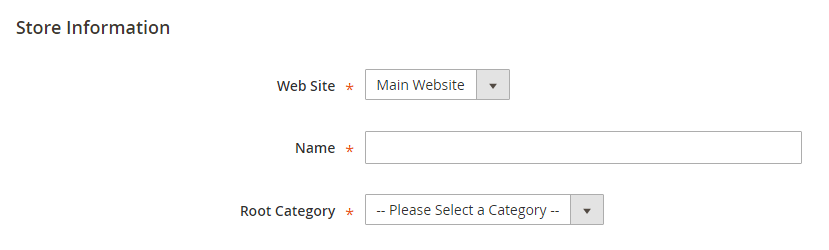


Figure 5.1.1.1 Store Information

* **Save Store** to complete.

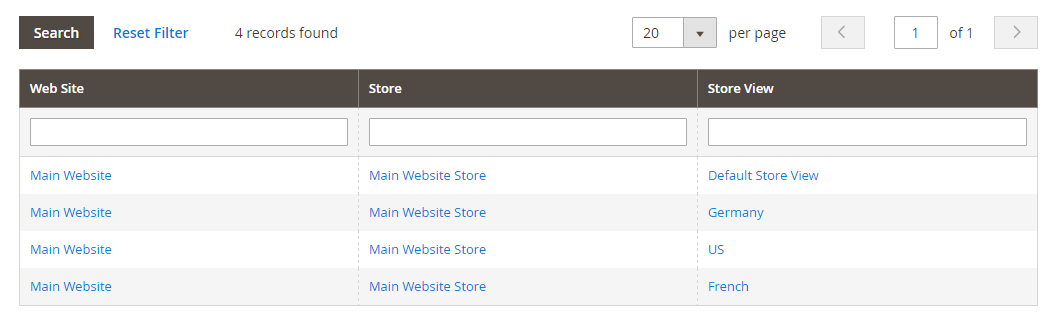


Figure 5.1.1.2.List of Store View

* Click on **Create Store View**, you can:
  + Assign the **Store** to the new store you’ve just created.
  + Set the **Name** for the store view.
  + Type the **Code** for the store view with lower characters.
  + Choose “Enabled” for the **Status**.
  + Specify the **Sort Order** that is position of the new store in the list.
* **Save Store View** to save all settings.

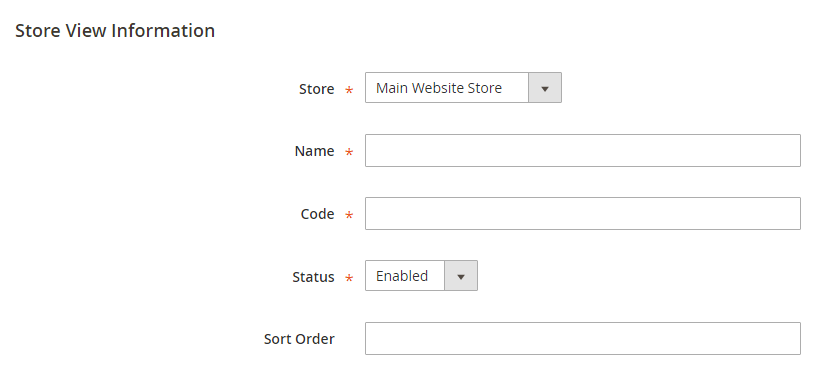


Figure 5.1.1.3. Store view details

* On the Admin Panel, go **Stores > Settings > Configuration.**
* On the left panel, under **General**, select Web.
* In the upper-left corner, choose the **Store View** to check the new store you added.
* Hit **OK** to verify the scope switching.

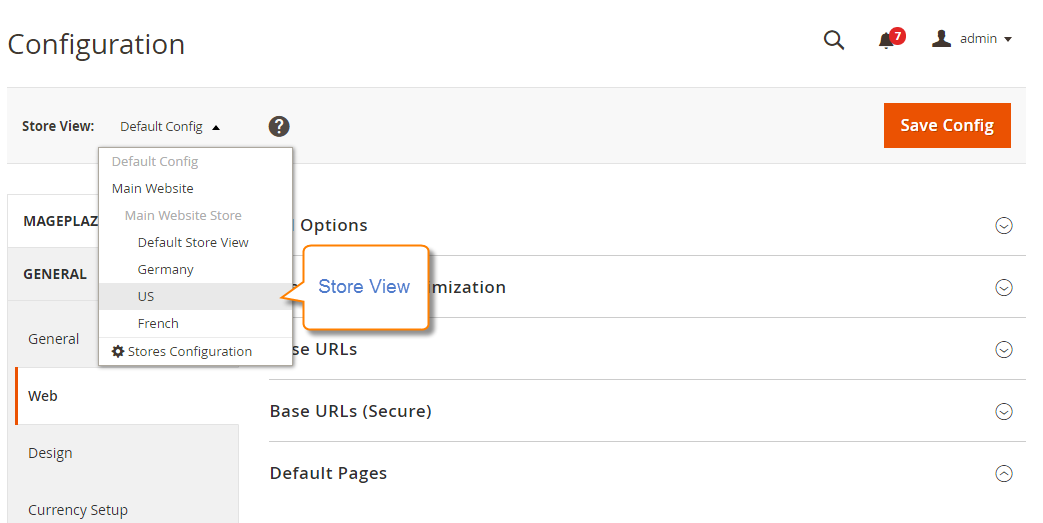


Figure 5.1.1.4 Store view Configuration

Open the **Base URLs** section,

* Clear the Use Website checkbox that is next to the Base URL field.
* Change the Base URL for the store.

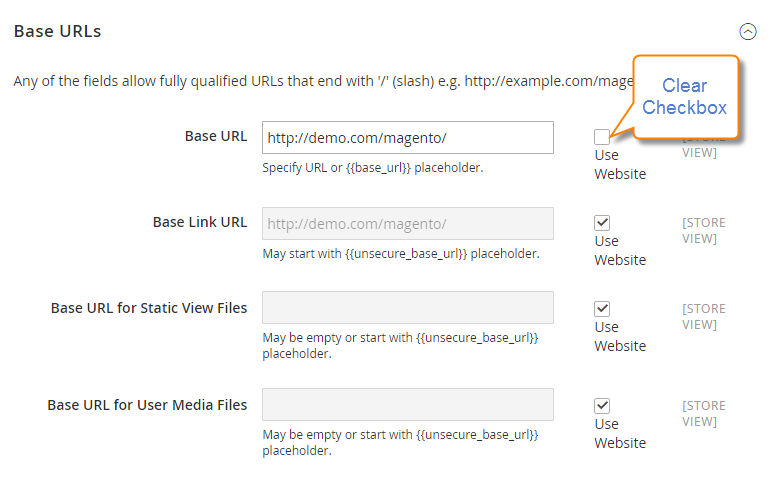


Figure 5.1.1.5 Base URL section

* Open the **Secure Base URLs** section below to set the store’s secure URL if needed.
* **Save Config** to complete.

The sample website was created to add products, product categories and descriptions for every product. The product's data table was created to handle the entire product ID, Name, quantity and category, every products are categorized with their type, design and cost. The main objective of dividing the products with category is to provide the filter feature for our site. Two additional operations allowed here are Sorting and Searching. Though it sounds simple, but it’s unlike the standard sorting and searching operations, with the help perfectly grained patterns it will provide both AND search and OR search. AND provides global search and OR provides local search. Through the help of local search the users can do ID search (i.e.) SKU [10], Name search, category search, design search and price search individually (for product's page), subsequent local searches are possible for every search tree that includes every category and products under those categories.

**5.1.2** **Adding Products**

Adding products to a site is a 6 step process which includes Naming a product, Description of the product, Product Category, Price details and other important details about store. A framework is a set of cooperating classes that make up a reusable design for a specific class of software. As per the concept of framework, magento provides the interfaces for adding products to the site is made easy by the magento developers. In the sequence of steps of a magento framework for adding product to the site is shown below.

**Step 1**: **Selecting the product type**

* On the Admin sidebar, click **Products**. Then under **Inventory**,

Select **Catalog**.

* In the upper-right corner on the **Add Product** menu, select

**Simple Product.**

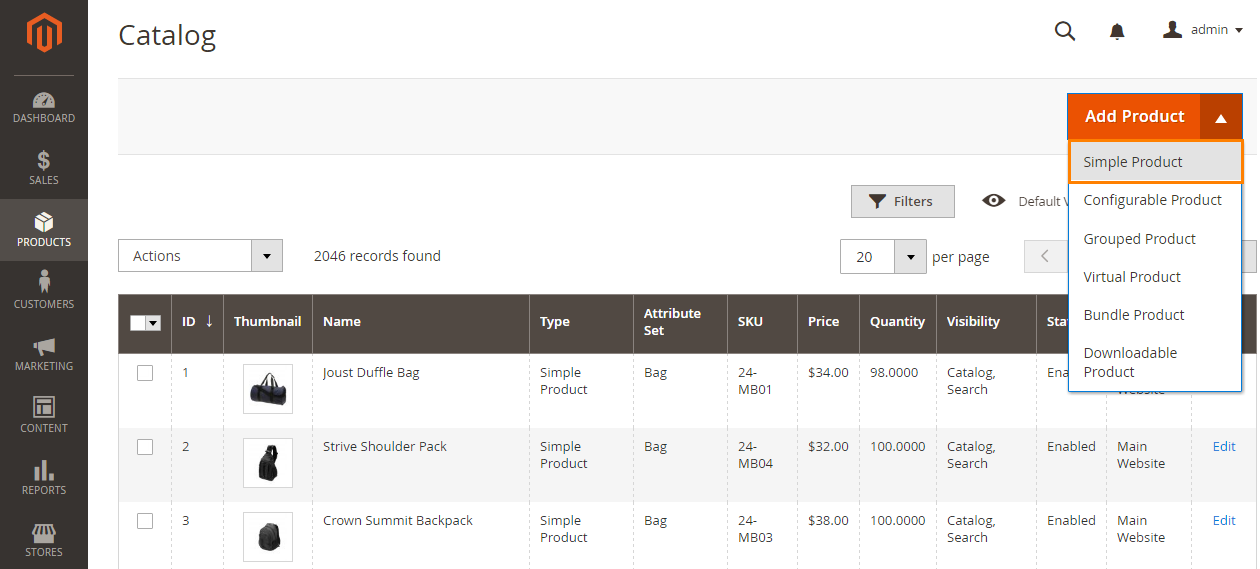


Figure 5.1.2.1 Product type

**Product Types**

In Magento products are classified into following 6 types

* Sample Product
* Configurable Product
* Grouped Product
* Virtual Product
* Bundle Product
* Downloadable Product

**Sample Product**

**Simple Product** is the most popular and unit of a store. In Magento, it is also defined as its name, it means that this kind of product is sold each single item (non-variation). Every single product has its own SKUcode.

*Example:*

The kind of Compete Track Tote Bags which is only sold in a fixed size, weight, material etc.

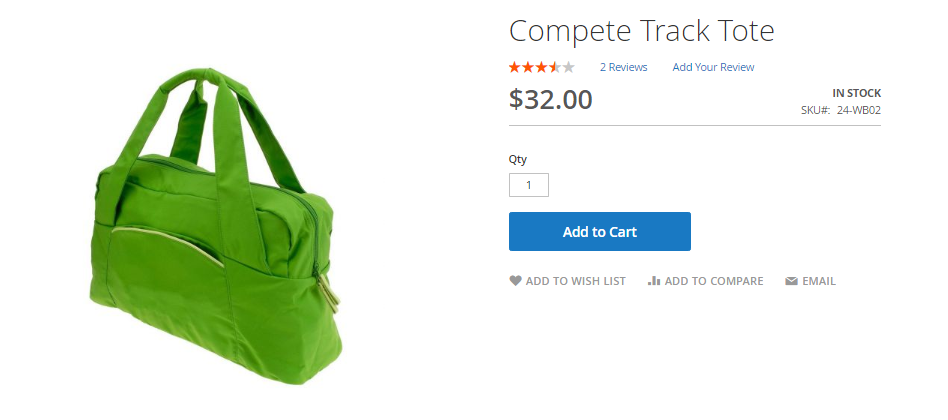


Figure 5.1.2.2 Sample Product

**Configurable Product**

This kind of product includes many **simple products**. The collection of various products with different options of colors and sizes but each single product has a separate **SKU**

***Example:***

For instant, one of **Configurable Product** is a kind of clothes coat such as following Stark Fundamental Hoodie product with 3 colors and different options size to choose, compare between them.

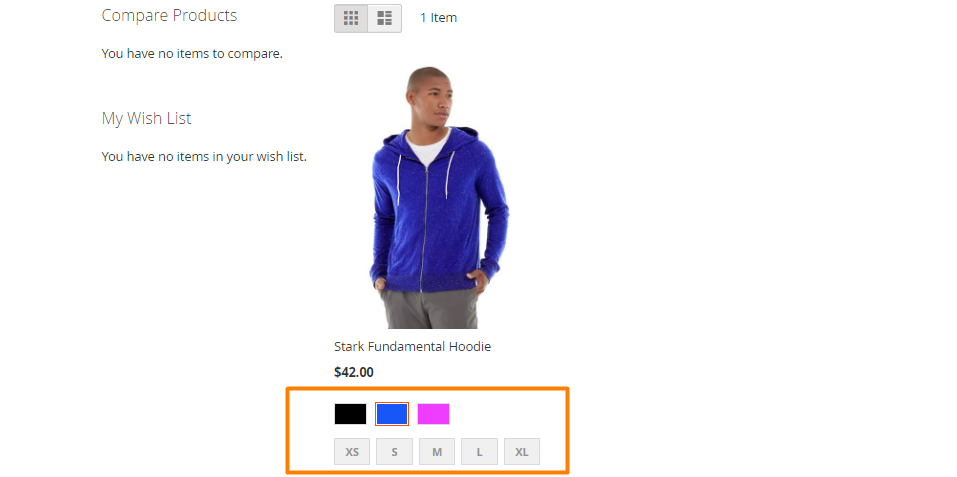


Figure 5.1.2.3 Configurable product

**Grouped Products**

It is a group of **Simple products** or **Virtual Product** which shares the same characteristics or related each other. It can save more than buying separately, increase the sale

*Example:*

Set of Sprite Yoga Straps under is example for **Grouped Product**, the buyers can choose to buy all 3 separated yoga straps or just one of them

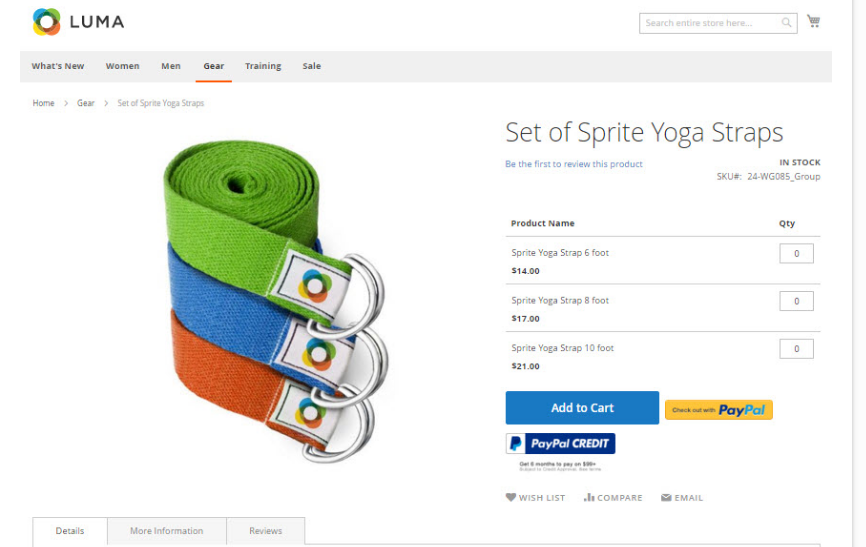


Figure 5.1.2.4 Grouped Product

**Bundle Products**

Which products customized are **Bundle Product** which includes various **Simple products** or **Virtual Product**

***Example:***

Sprite Yoga Companion Kit which has different items containing such as Ball, Yoga Brick, Yoga Strap and Foam Roller. Customers have no choice except for buying **ALL** items in that kit

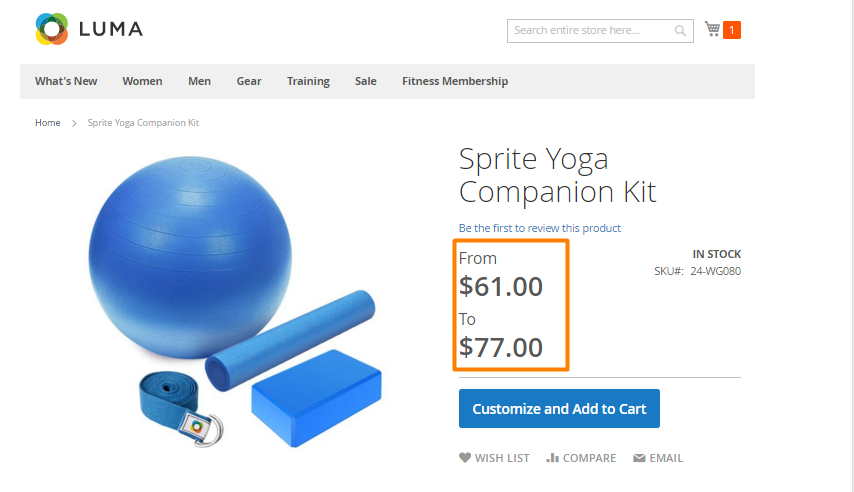


Figure 5.1.2.5 Bundle Product

**Downloadable Products**

Anything you are able to download such as software, files, ebook, video, etc. After customers watch the trailers, sample for testing, they will purchase, they just need download then can use the product

*Example:*

A kind of following Yoga video for learners is **Downloadable Product**. There are 3 trailers for buyers can overview the products before buying

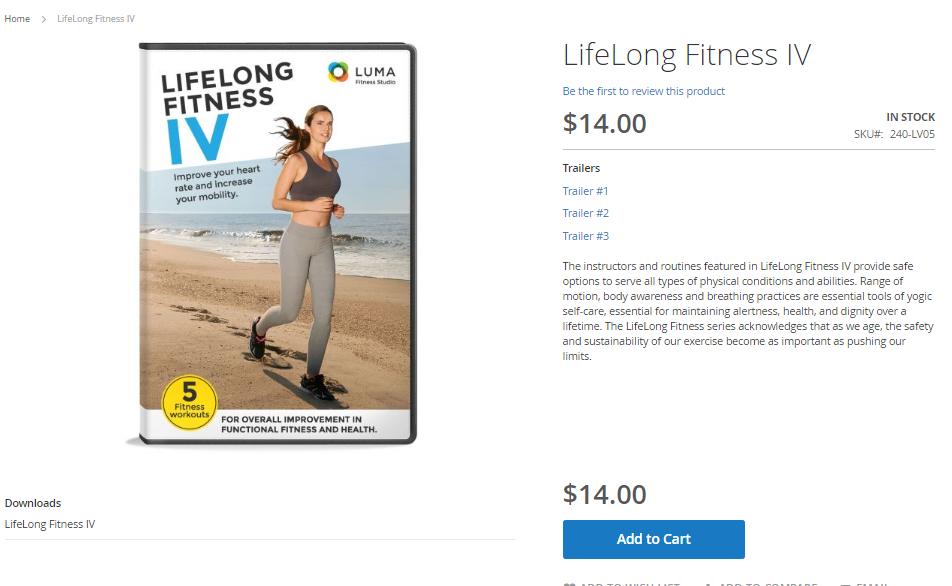


Figure 5.1.2.6 Downloadable Product

### Step 2: Choose the Product Template (Optional)

To choose the product template, do one of the following:

* In the Search box, enter the name of the template.
* In the list, choose the template that you want to use. The form is updated to reflect the template.

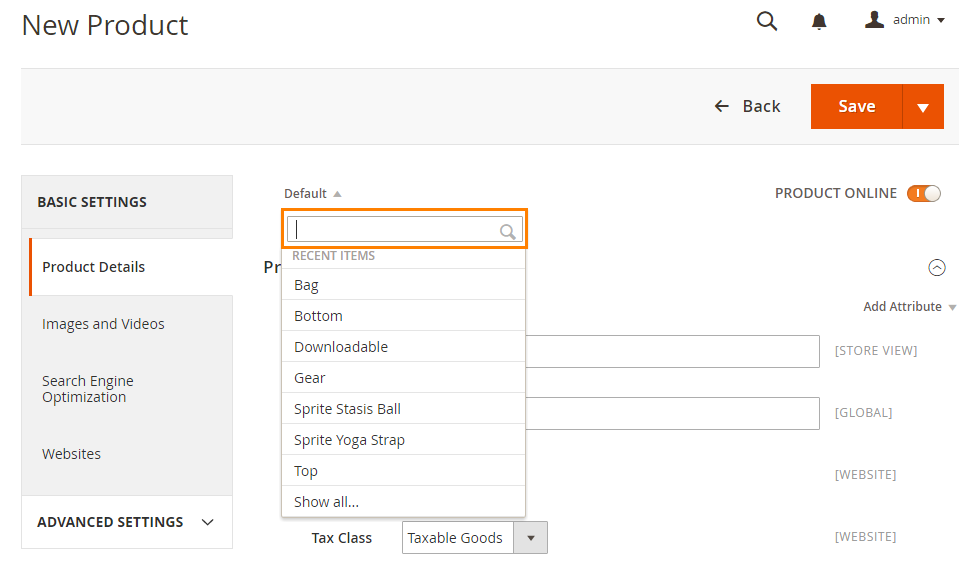


Figure 5.1.2.7 New product

### Step 3: Complete the Required Fields

* In the **Product Details** section. do the following:
  + Enter the product **Name**.
  + You can either use the default **SKU** that is based on the product name, or enter another.
  + Enter the product **Price**.
* Because the product is not yet ready to publish, set the **Product** **Online** switch to the “Off” position.
* Click **Save**, and continue with the next step. we can save the product settings after complete all the required fields complete.

### Step 4: Complete the Remaining Product Details

* Set **Tax Class** to one of the following:
  + None
  + Taxable Goods
* If you’re ready to add a product image, do one of the following:
  + Drag an image from your desktop, and drop it on the **camera** tile in the Images and Videos box.
  + In the Images box, click the camera tile, and navigate to the image file on your computer. Then, select the image, and click Open. A **placeholder** appears until a product image is uploaded.
* Enter the **Quantity** of the product that is currently in stock.
* Enter the product **Weight.**
* To assign the product to a **Category**, do one of the following:
  + Start typing to find a match. Then, choose the **Category**.
  + Click **Show Lis**t to see the category tree. Then, drill down and click on each category that you want to assign to the product.
  + Click **New Category**. Enter the **Category Name** and choose the **Parent Category** to determine its position in the menu structure. Then, click **Create Category**.

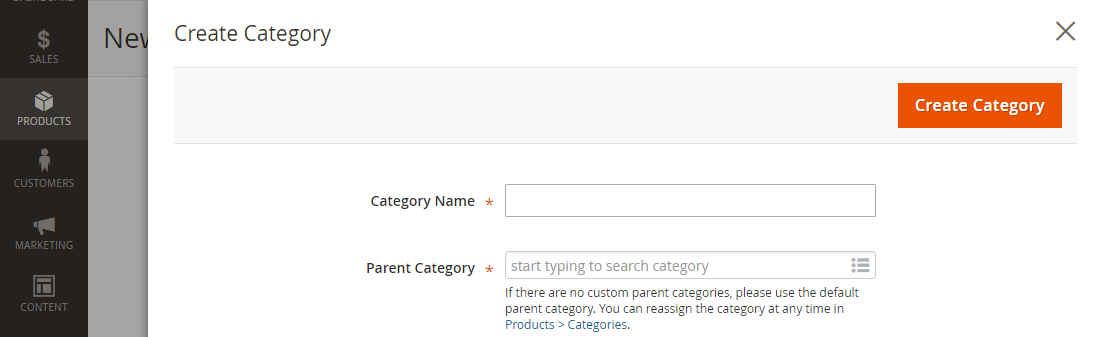


Figure 5.1.2.7 Create Category

* Enter the product **Description** directly into the text box, and format as needed. Then, click Submit. we can also use the **WYSIWYG** **Editor**, for additional control.

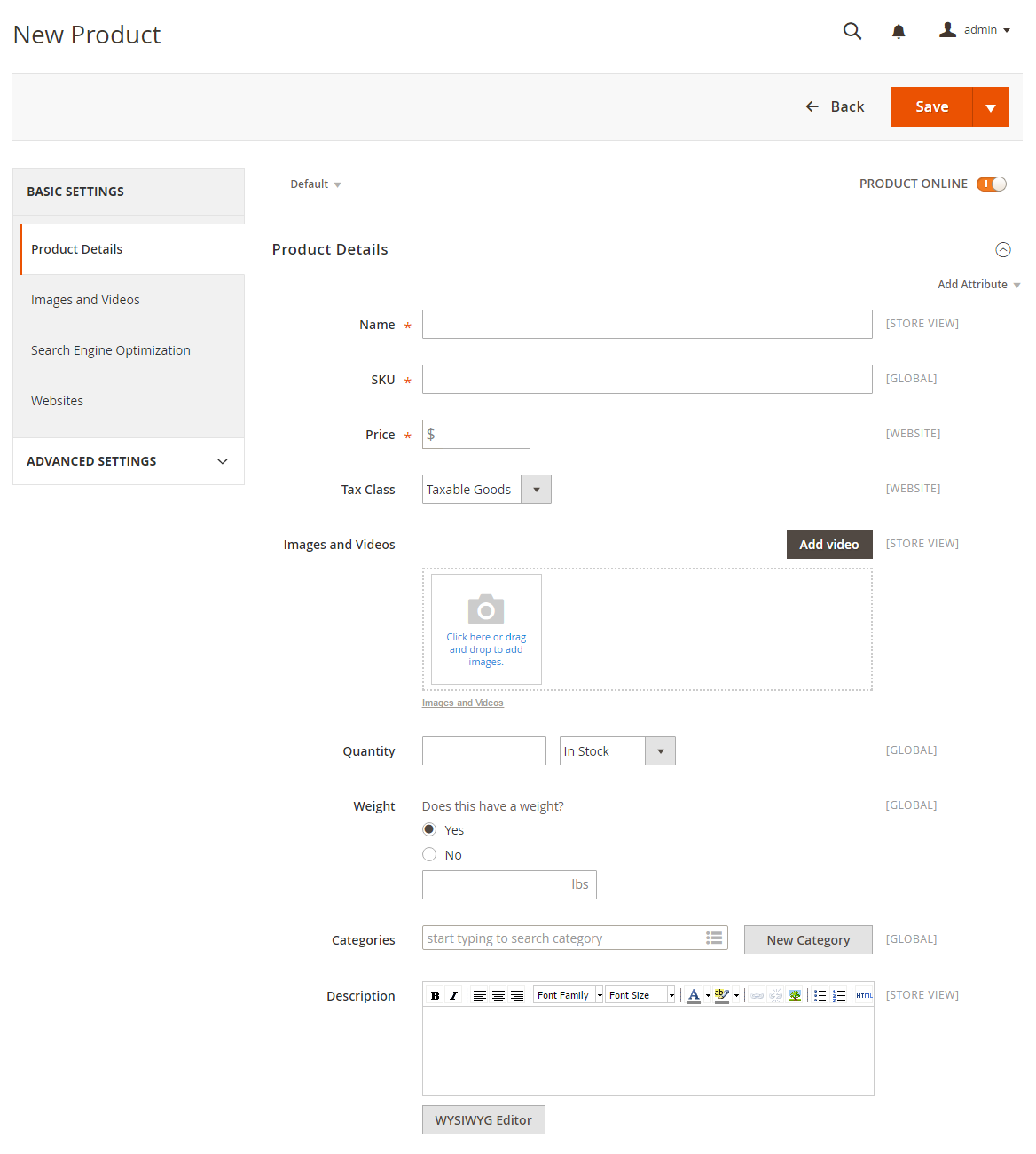


Figure 5.1.2.8 Add a product (Full)

### Step 5: Publish the Product

* If you are ready to publish the product in the catalog, set the **Product Online** switch to the “**On**” position.
* When complete, on the Save menu, select **Save & Close**.

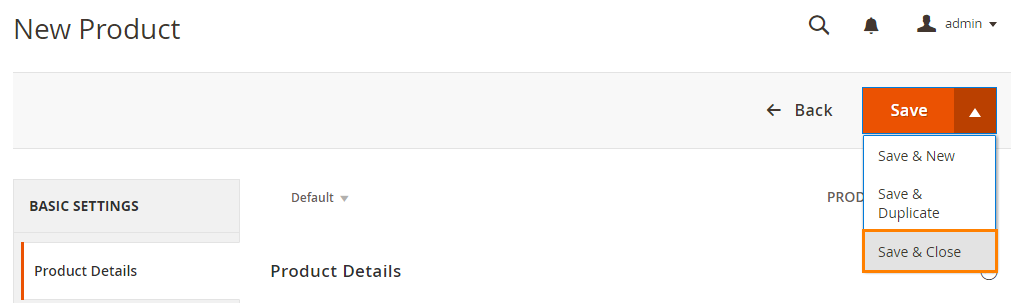


Figure 5.1.2.9 Save Product

### Step 6: View the Product in Your Store

* In the upper-right corner on the **Admin** menu, select **Customer View**.

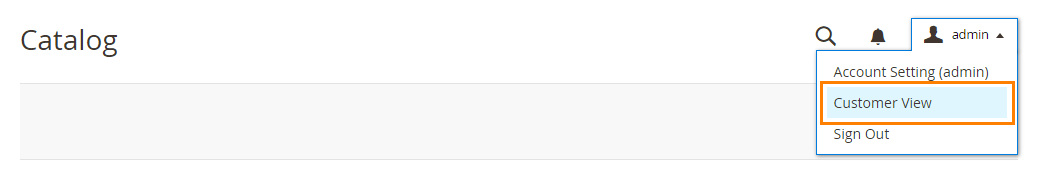


Figure 5.1.2.10 Set View for Product

When we add a product to the site it will stored in database in the following table format, we can view this product details at magento backend under the products tab.

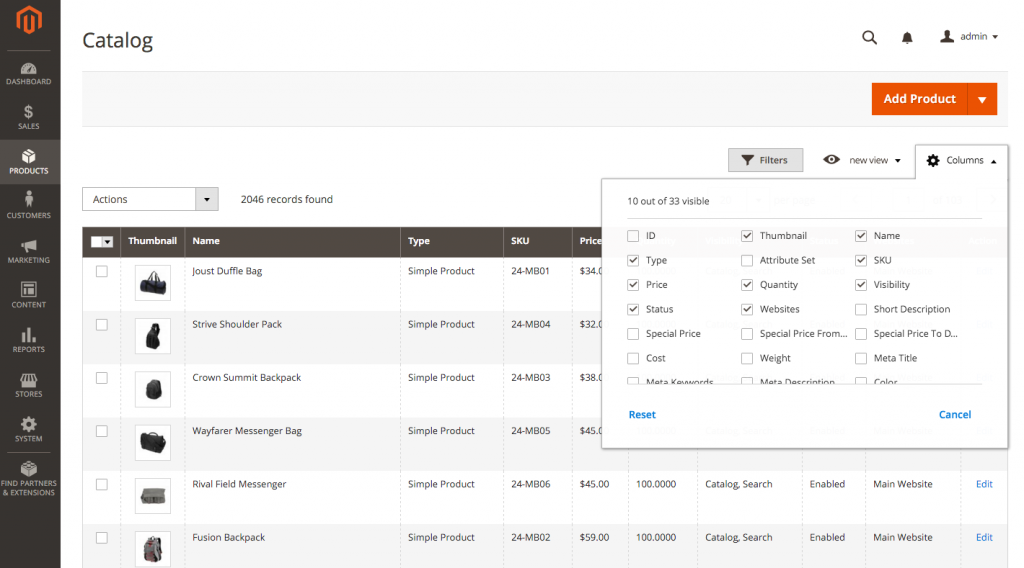


Figure 5.1.2.11. Catalog View

**SKU of the product**

SKU stands for Store Keeping Unit. It is the unique identifier for the product, which will be used to fetch the products from the backend whenever it required. Also it will be used by the seller to deliver the exact product which was purchased by the customer in the site.

**5.1.3** **Integrating the REST API with Magento**

Integrating the REST API in magento for fetching the products and data from backend to the Ionic, to display the contents on the mobile app requires the Ionic framework configured in the System.

In the Initial steps of our system development, we had discussed about the configuration part of the Ionic Framework in the development machine.

**Step 1: Copying the files into Magento**

First step in the integration of the REST API with magento starts with copying the required file (i.e.) the API files into our Magento sites’s Code directory. This api files contains the statements regarding fetching the products and details from magento backend as a json data

**Step 2: REST API as a Magento Module**

Run the following commands in the magento site directory to use the API files as a module in magento 2 framework.

* npm install
* bower install
* ionic setup saas
* ionic state restore –plugins

After executing this commands run the ionic serve command to view the frontend of the site via ionic in the browser with configured port number.

Then completion of copying the API files into the sites code folder as a module, the next step is to enable the module in the magento backend. To enable the module in the magento backend follow the steps below.

1. In magento 2 admin side panel goto **System > Web setup wizard**

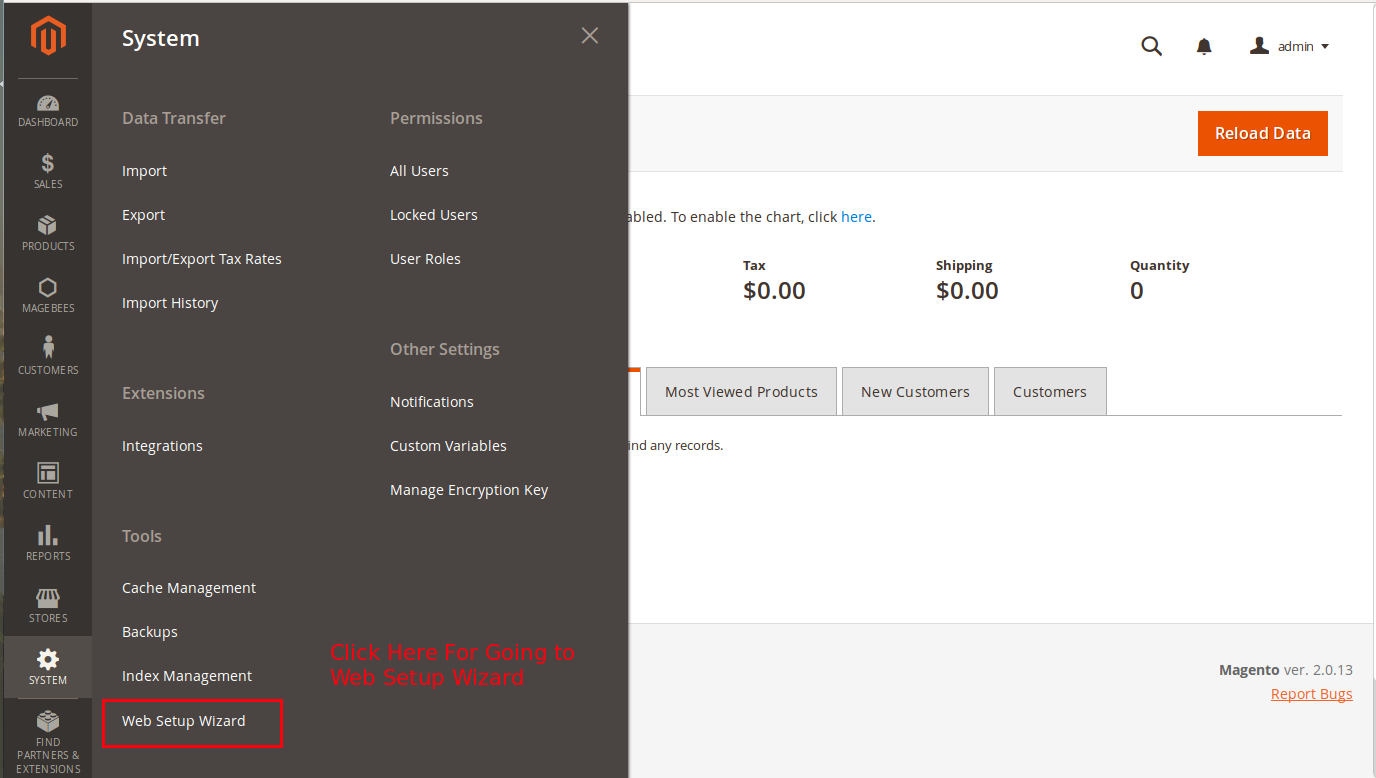


Figure 5.1.3.1 Integrating API

1. In the setup wizard page , click the component manager button,

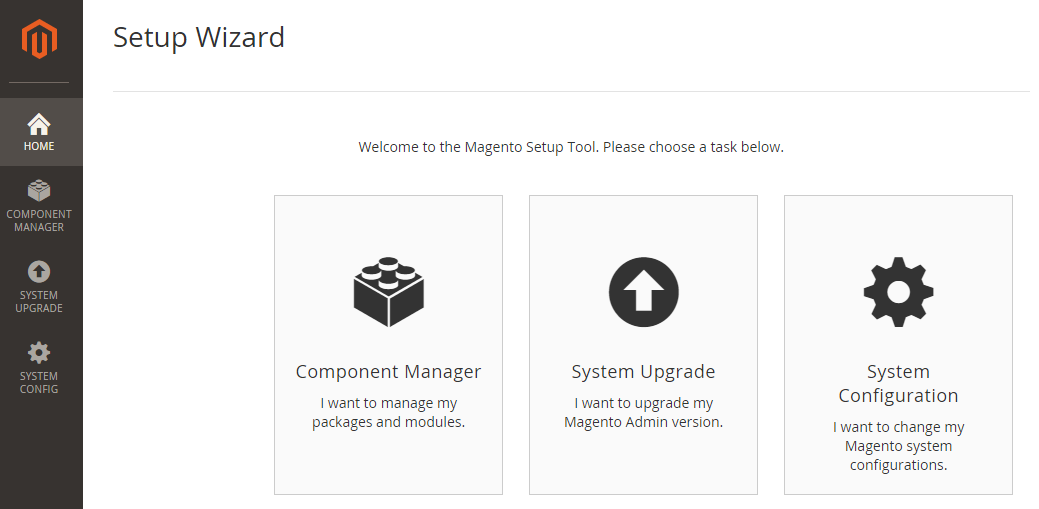


Figure 5.1.3.2 Component Wizard

1. In the components manager page, the recently added module (i.e.) our API module will be highlighted in red color to indicate that the module is disabled and also it requires the process of upgrading the magento setup to complete the integration of the module.
2. To upgrade the magento setup via command line, move to the magento site directory and execute the following command.

**./magento setup:upgrade**

1. The above command will clear the magento cache and upgrades the modules list of the magento. Now the module will be enabled and it will be in the green color that indicates that successful completion of upgrading modules.
2. Then go to **System** > **Integrations** page to activate the API plugin to the site, that enables the API to communicate between magento backend and ionic framework to interchange the request and product details.

Click the allow button to give the permission to the API for accessing products and database from plugin.

1. By clicking the allow button the magento will generate the consumer key, consumer secrete key, access key and access token secrete key.
2. Now we need to redirect the requests to the enabled plugin for rendering the the products and data in mobile view in Ionic App which we created.
3. Go to the www directory of the created Ionic App and open the **config.js** file, in the application url constrain give the url of the magento site and also the **Access Keys** and **Access Tokens**.
4. After this, our Ionic app will show the products from the magento’s backend.
5. This is How the Ionic fetches the product and product related data from the magento and displays it in the Ionic app.

**5.1.4 Evaluating the Quality of the Hybrid Web application**

While the jury is still out on which is the better way to build apps – native or hybrid, there is simply no resisting the charm of getting an elegantly functional, native quality app that can be developed once and used on both Android and iOS. And even though nothing beats the exquisite feel of a native app, modern hybrid apps have come a long way in providing you an immaculate performance, while saving you time and money. So unless you need a super extra high-performance app with specs like 60fps, 3D rendering, and holographic VR, you can bet on hybrid apps to fire up your business and keep users engaged.

Among other technologies, modern and powerful hybrid app development frameworks have a big role to play in the ascension of hybrid apps to near-native standards.

**CHAPTER 6**

**SYSTEM TESTING**

**6.1 SOFTWARE TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTS

There exist various types of testing to check a software and hardware component and its working. Among the abundant types, we make use of a few, given as follows

1. Unit testing
2. Functional testing
3. Integration testing
4. System testing

6.2.1 Unit Testing

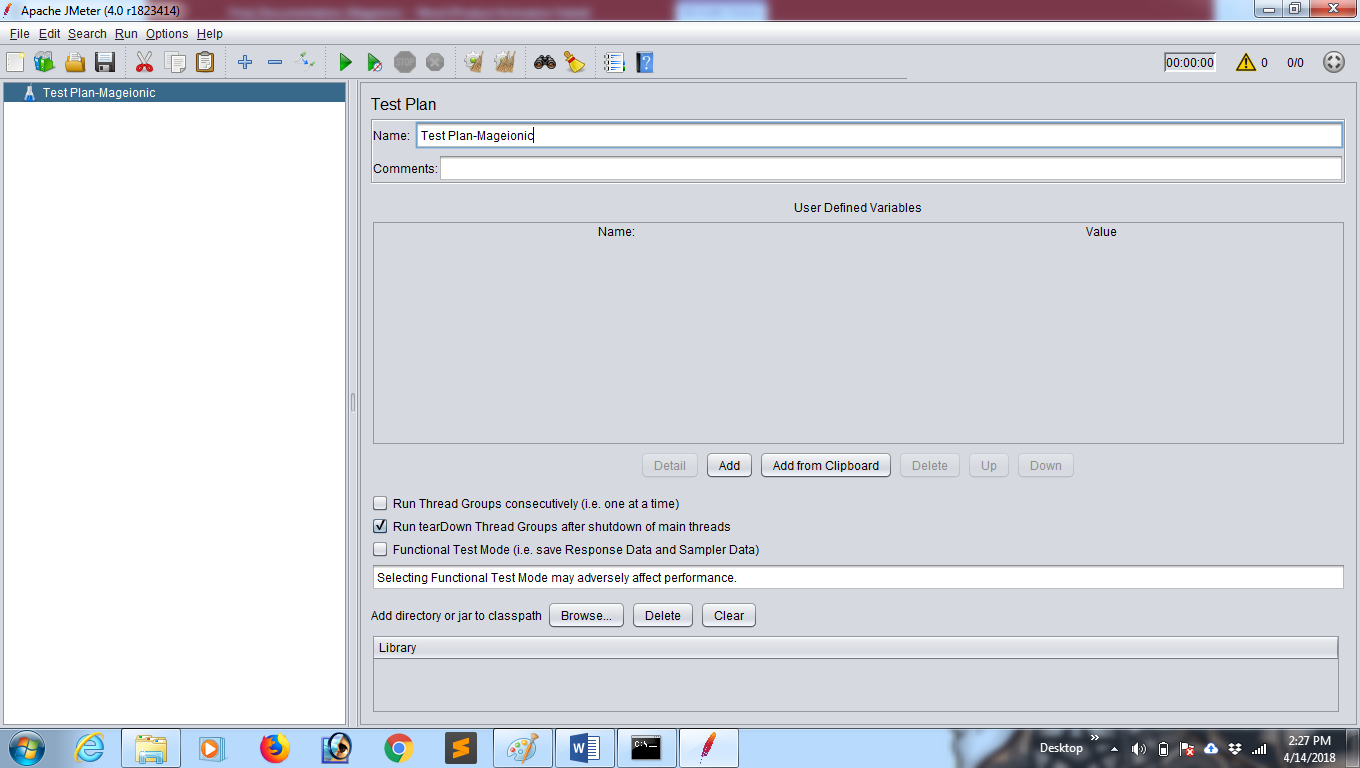
Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. The same logic applies to the physical or mechanical unit testing of machineries.

Units in this project solely includes software units, according to the particular module Injection of new patterns into Framework contains several units Action, DAO and Service. Unit testing can be performed for any of the mentioned units.

**6.2.2 Functional Testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Both the software companies and the automated sectors perform the functional testing with the intent of finding the functional defects in the product. Functional testing on a software is centered on the valid input, invalid input, function done or triggered, output obtained and the system or procedure followed to obtain the output. The functional testing by the automated sector follows the same strategy and gives the required raw material or product as input and checks the manufactured product or the stepped up raw material. In the proposed system, the Functional testing has been done by **Apache JMeter** tool.

Jmeter is a Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. Apache JMeter may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more). It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. You can use it to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load. The Test Plan with the name Mageionic-Testplan got created to simulate 5 users (or Threads) with 300 iterations each. Fig. 6.3 depicts the Thread Group.

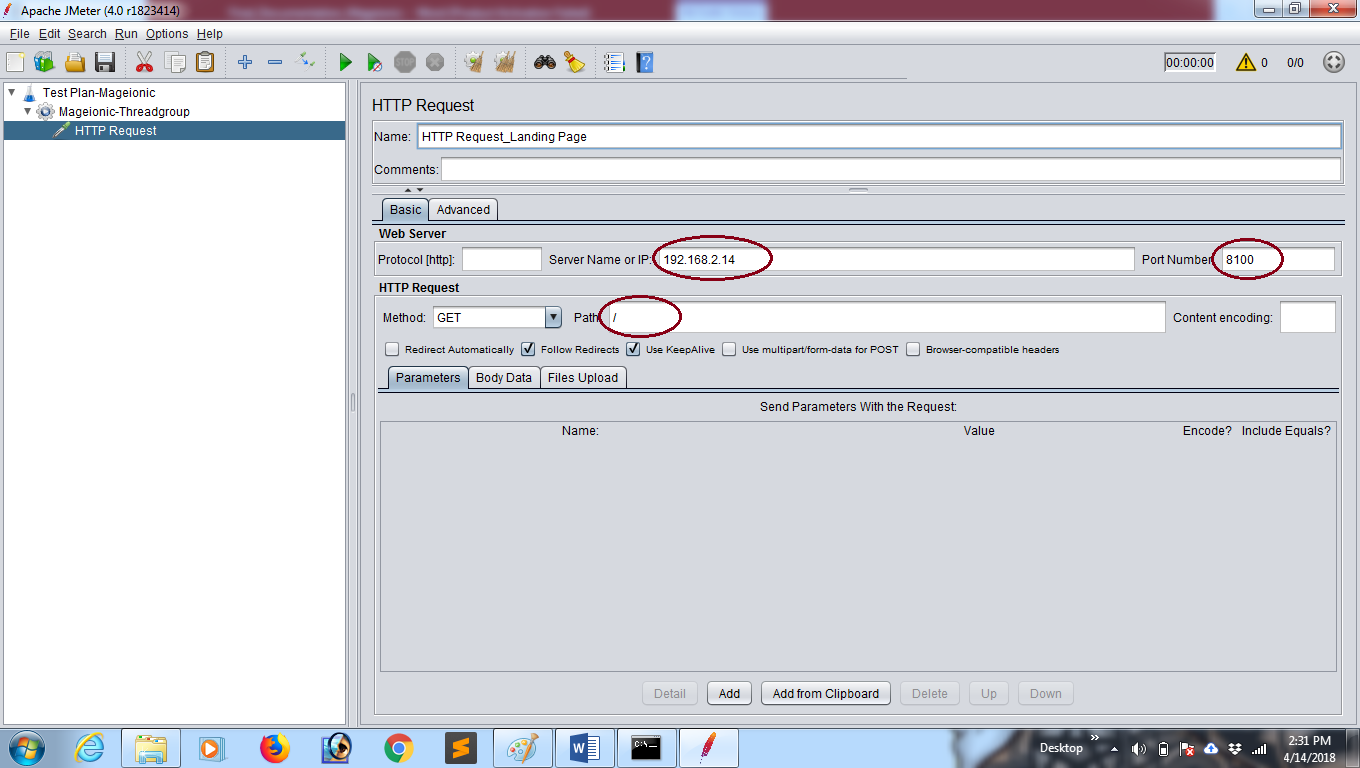


**Figure 6.1 Test Plan creation**

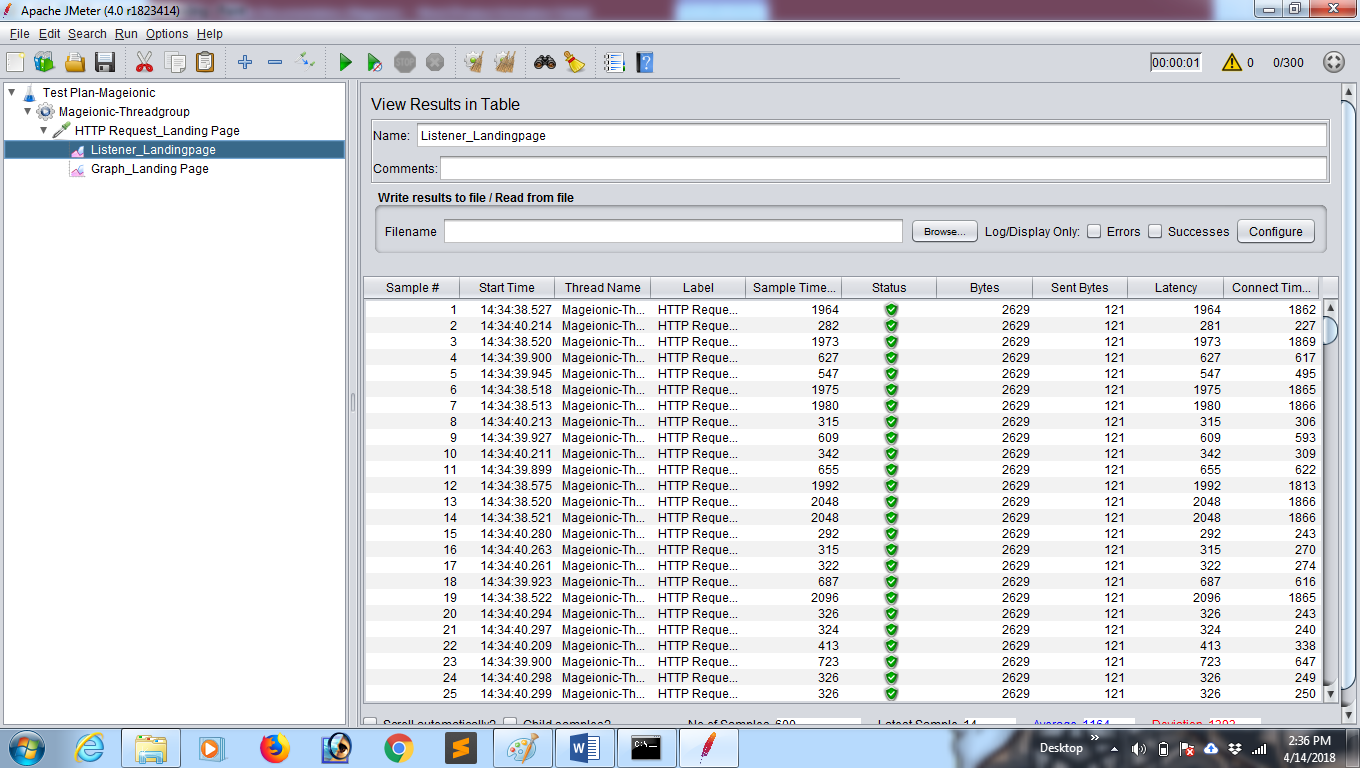
Since the created hybrid web application strictly follows MVC pattern, all the action Controllers has been taken for functional testing. All the product page’s HttpRequest has been considered for functional testing, Figure 6.3 shows the HttpRequest of landing page, Figure 6.4 shows the testing result in tabular form of landing page and Figure 6.5 shows the testing result in Graph form.



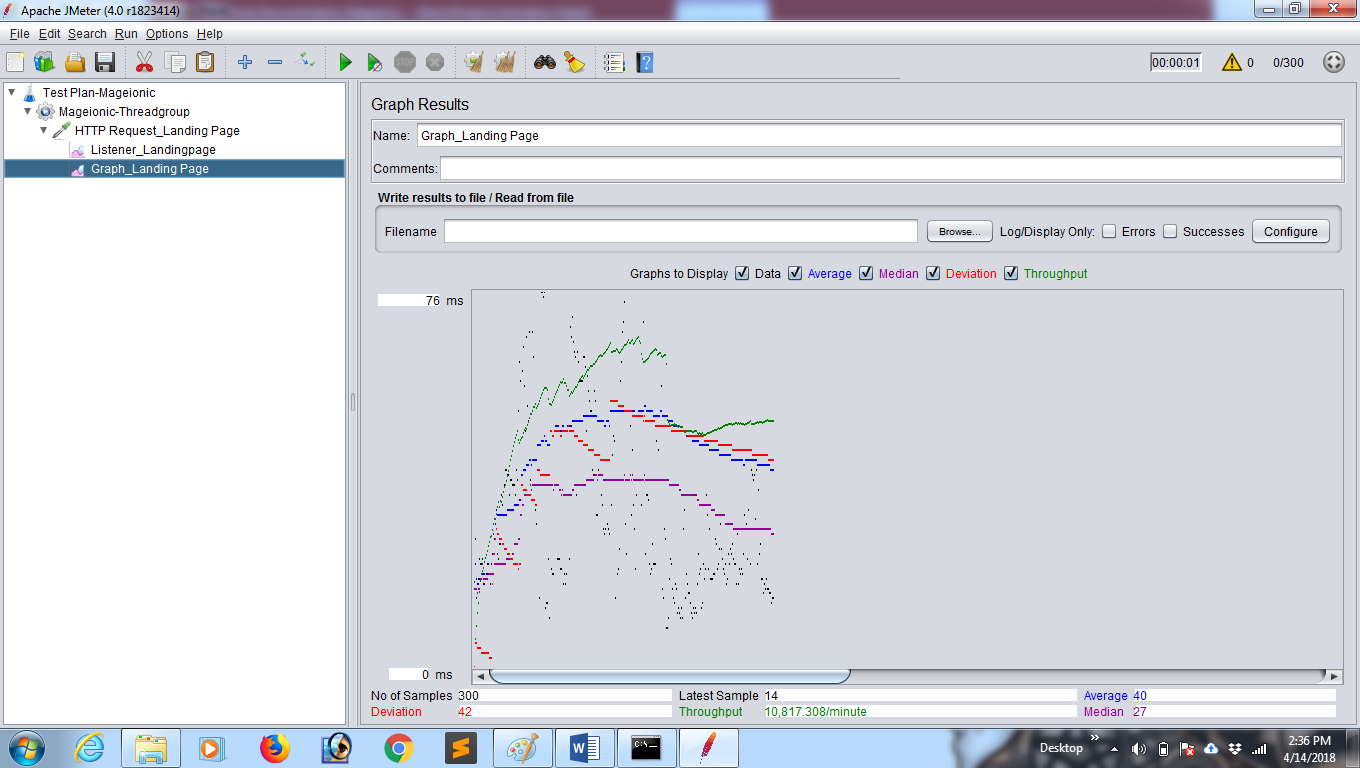
**Figure 6.2 Creation of Thread Group for Testing**

****

**Figure 6.3 Creation of HTTP Request for Landing page**

****

**Figure 6.4 Testing Results in Tabular form**

****

**Figure 6.5 Graph Result for Landing Page**

# 6.2.3. Integration Testing

# Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error. In the proposed system, Integration testing has been implicitly done by the framework itself.

**6.2.4** **System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. Thus the testing for the proper functioning of patterns under the framework to enhance the quality of the Hybrid Web application has been done by System testing.

**CHAPTER 7**

**RESULTS AND DISCUSSION**

**7.1 RESULTS**

The proposed approach encourages the use of the Hybrid Web application rather than the native apps. The performance of the proposed system by certain level and some metric based results is illustrated with the help of screenshots. The performance comparison between hybrid and native app for the light weight and non-graphical applications has been shown in the Also the usage scenario of hybrid web apps in mobility of current days.

**7.2 SCREENSHOTS**

Figure 7.2.1 Android Phone view

Figure 7.2.2 IPhone view

Figure 7.2.3 Windows Phone view

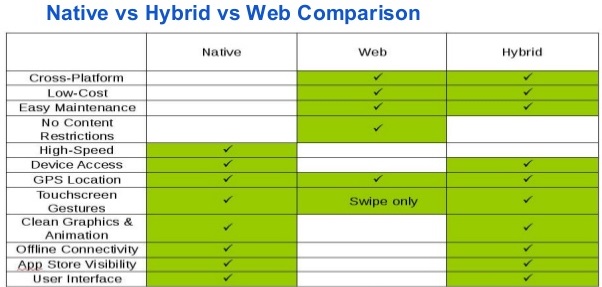
**7.3 PERFORMANCE ANALYSIS**

After the successful deployment of the app, 3 sample modules had been taken for quality analysis. Those modules are Payment method, Add to Cart, and Checkout. As we stated earlier three domains related metrics Native, Web, Hybrid web applications been considered for quality analysis. The subsequent graphs and tables depict the performance analysis before and after mageionic. Table 7.3.1 illustrates the results of LOC metric.

**Table 7.3.1 Results for LOC metric**

|  |  |  |
| --- | --- | --- |
| **Domains** | **Native** | **Hybrid** |
| Android | 250 | 70 |
| iOS | 180 | 70 |
| Windows | 350 | 70 |

As shown in Table 7.3.1, the LOC metric results of Hybrid apps had comes down from 250 to 70 and this clearly shows that the quality has been inevitably increased.



**Figure 7.3.1 Native vs Web vs Hybrid App**

The above table clearly shows the difference in quality between the three platforms.

**CHAPTER 8**

**CONCLUSION AND FUTURE WORK**

**8.1 CONCLUSION**

This work proposes an approach to enhance the quality of hybrid web application by implementing right platform of e-commerce sites as suitable for hybrid web applications. The proposed system is to not only reduce the source program’s lines of code, it also promises the intelligent and rapid app development for every complex sites.

**8.2 FUTURE WORK**

Our future work spans three main directions: first by considering the use of progressive web apps and to migrate to the progressive web apps from hybrid web apps; second by evaluating the quality and performance to for providing the graphical support with the proposed system. And third is to implement the Magento admin panel in the mobile apps.

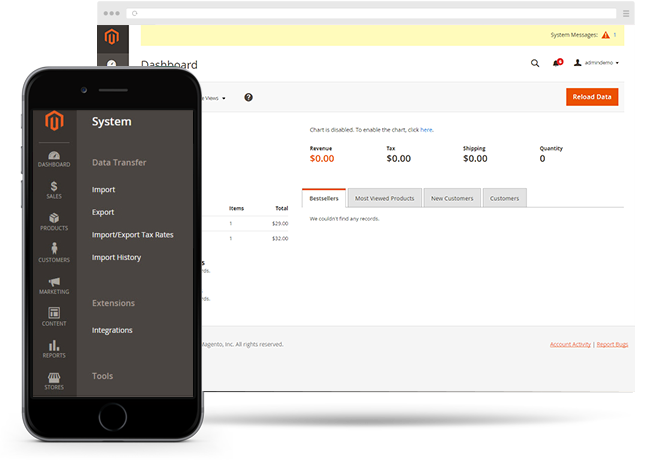


Figure 8.2.1 Future Enhancement-Mageionic

**REFERENCES**

E-commerce Development, Magento 2.2 Developer Documentation

[**https://devdocs.magento.com**](https://devdocs.magento.com)

Hybrid Web App Development, Ionic Framework Documentation

[**https://ionicframework.com/docs/**](https://ionicframework.com/docs/)

Configuring Magento 2.2 REST API Documentation

**http://devdocs.magento.com/guides/v2.0/rest/bk-rest.html**