BigCommerce Candidate Assignment

This document provides design and implementation details for Candidate Assignment.

# Authored By:

Vipul Jajal

[jajalvipul@gmail.com](mailto:jajalvipul@gmail.com)

0402822585

Table of Contents

[Authored By: 1](#_Toc528570839)

[Assignment Tasks 3](#_Toc528570840)

[User Control Flow 3](#_Toc528570841)

[Wireframe mock ups 3](#_Toc528570842)

[Customer order count screen 4](#_Toc528570843)

[Customer order history details 4](#_Toc528570844)

[Architecture Concepts 5](#_Toc528570845)

[Models 5](#_Toc528570846)

[Views 5](#_Toc528570847)

[Service 5](#_Toc528570848)

[Service Provider 6](#_Toc528570849)

[Controllers 6](#_Toc528570850)

[Cache 7](#_Toc528570851)

[Test Cases 7](#_Toc528570852)

[Classes summary 7](#_Toc528570853)

[BigCommerce Client APIs 8](#_Toc528570854)

[Env file 8](#_Toc528570855)

[Improvements 8](#_Toc528570856)

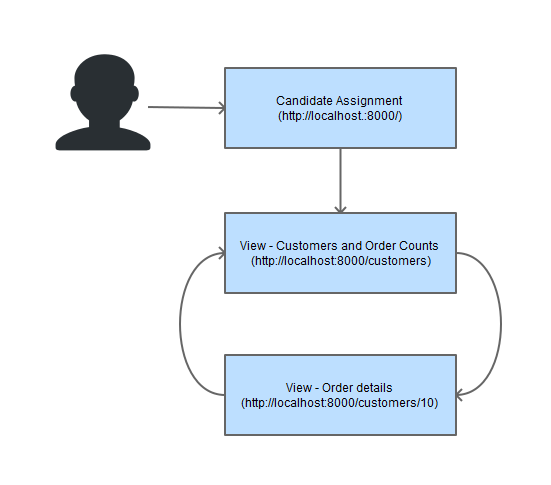
# Assignment Tasks

This application is to connect to a live BigCommerce store via the V2 API. The application will consist of the following screens:

* A list of Customers, including the total number of orders they have placed
* A Customer Details screen that displays the Order History for that Customer and their Lifetime Value (defined as the total value of all of their orders)

# User Control Flow

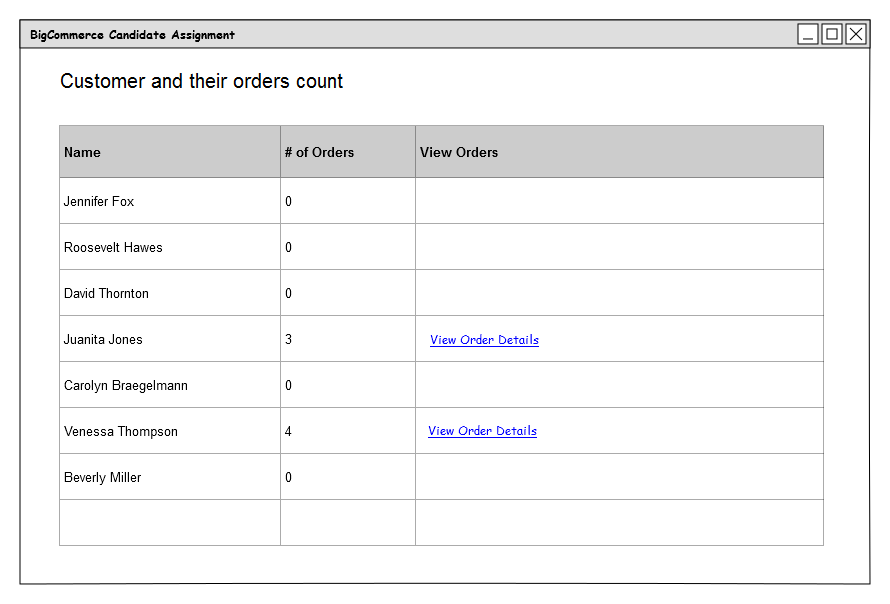
This section illustrates the control flow from the end user point of view.



# Wireframe mock ups

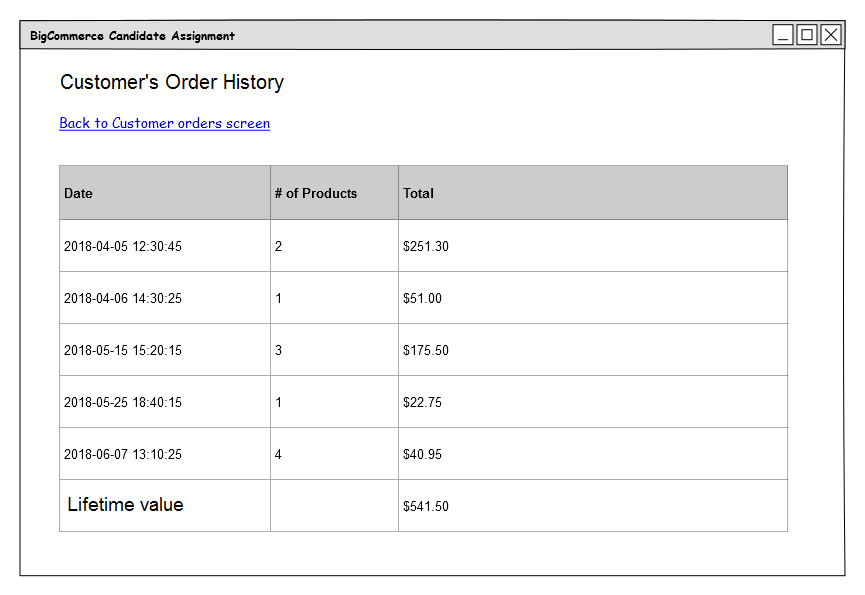
This section provides the screen wireframes for proposed screens. Note that the mock up screens are for illustrative purpose only.

## Customer order count screen



* The column “Name” displays customer’s first name and last name.
* The column “# of Orders” displays total number of orders placed by the customer
* The column “View Orders” displays the hyper link “View Order Details”. This hyperlink is visible only in those customer rows where the total order count is greater than zero.

## Customer order history details



* The title of the screen displays Customer’s Order History. The word “Customer” will be replaced by first name of customer for e.g. “Juanita's Order History”.
* The “Date” column displays the date when the order was created. The date format will be 'Y-m-d H:i:s' for e.g. “2018-04-05 12:30:45”.
* The “# of Products” column displays the total number of products present in the order.
* The “Total” column displays the total amount of an order, including the tax.
* The “Lifetime value” will display the total value of all of customer’s orders.

# Architecture Concepts

In the given candidate assignment, following functionalities are implemented to complete the given tasks.

## Models

The new models CustomerOrderCount and CustomerOrderDetail are added to provide customer and order related data to the views to render information on screens.

These models on their own do not contain any business logic. They are primary used to transfer the data obtained after invoking the BigCommerce V2 APIs using a service CustomerOrderService.

## Views

The views customers.blade.php and details.blade.php are based on the Blade templates and they use newly added models to display the customer and order details on the screen.

The view customers.blade.php mainly displays the customers (first\_name and last\_name) present in the given store.

For each customer, the total number of orders (total\_orders) they have placed is displayed next to the customer name column.

The end user can view the order history by clicking on the hyperlink “View Order details”. This hyperlink is visible only when the count of total orders for a customer is greater than zero.

As per the given suggestion in the assessment, the efforts in implementing user interface are kept to minimal level in order to focus on the server side functionalities. The pagination and search customer by name or total order count would be useful functionality to implement as an added improvements.

## Service

The newly added service CustomerOrderService is responsible to use the BigCommerce PHP API Client to get the customer and order related data.

Following methods are implemented to consume the BigCommerce API V2.

* getCustomer($customerId)
* Invokes PHP API *Client::getCustomer($customerId)*.
* Returns customer object populated for the given $customerId.
* getCustomersAndTheirOrderCounts($filter)
* Invokes following PHP APIs
  + *Client::getCustomers($filter)*
    - Where $filter is an associative array containing the keys and values for “page” and “limit”.
    - Obtains the list of customers.
  + *Client::getOrdersCount($orderfilter)*
    - Where $orderfilter is an associative array containing ("customer\_id" => $customer->id) for each customer present in the Customers list.
    - Obtains the total number of orders placed by the customer.
* Return an array of CustomerOrderCount objects
* getCustomersAndTheirOrders($customerId, &$lifeTimeValue)
* Invokes PHP API *Client::getOrders($orderfilter)*
  + Where $orderfilter is an associative array containing ("customer\_id"=>$customerId)
  + Obtains the list of orders placed by the customer ($customerId)
* Returns an array of CustomerOrderDetail objects.

(Note: This method could be renamed to getCustomerOrdersDetails to convey its targeted functionality properly)

## Service Provider

The newly added service provider CustomerOrderCacheServiceProvider demonstrates the functionality of customised caching by implementing following functions.

* hasCustomerOrderCount()
  + Verify whether the customer order count data is present in the cache.
* getCustomerOrderCount()
  + Retrieves the customer order count data from the cache.
* putCustomerOrderCount()
  + Stores the customer order count data in the cache.

## Controllers

The existing controllers were enhanced to provide the customer and order details.

* CustomersController

The function index() uses an instance of CustomerOrderService injected by the Laravel framework and returns the view “customers” having customer and their order count data.

The pagination filter is applied for better user experience and performance and to utilise the customised caching mechanism.

$filter = array("page" => 1, "limit" => 250);

(Above parameters are defined in this function but ideally it should be passed from the View UI.)

It also demonstrates the use of **Flash messages** to display the use of customised cache in View UI.

* CustomerDetailsController

The function show() uses an instance of CustomerOrderService injected by the Laravel framework and returns the view “details” having customer, customer order details and Lifetime value (defined as the total value of all of their orders placed by a customer).

## Cache

The customised cache has been implemented using **file** based Facade Illuminate\Support\Facades\Cache.

This cache expires at an interval (in minutes) specified by the Environment (*.env*) key CUSTOMISED\_CACHE\_TIMEOUT.

To demonstrate the cache functionality, only the customer and related order count data, subject to the filter criteria ("page" => 1, "limit" => 250) is used in the CustomersController.

The key CUSTOMISED\_CACHE\_TIMEOUT with a value of 4 minutes has been added in the files “*.env*” and “*.env.example*”.

Note: To improve the performance, this cache could be improved to store the customer’s historical order details.

## Test Cases

The following new test cases are added.

* CustomerOrderServiceTest

This is a unit test to verify the functionality implemented in the CustomerOrderService service in relation to customers and their related orders.

* CustomerOrderTest

This is a feature test containing set of HTTP tests to verify following:

* + - Uri – “/”
    - Uri – “/customers”
    - Uri – “/customers/{id}”

# Classes summary

New Classes

1. Model Class – App\Models\CustomerOrderCount
2. Model Class – App\ Models\CustomerOrderDetail
3. Service Class - App\Services\CustomerOrderService
4. Service Provider (Custom Cache) Class –

App\ Providers\CustomerOrderCacheServiceProvider

Updated Classes

1. Controller Class – App\Http\Controllers\CustomersController
2. Controller Class – App\Http\Controllers\CustomerDetailsController
3. Service Provider Class - App\Providers\AppServiceProvider

Test Classes

1. Unit tests - Tests\Unit\CustomerOrderServiceTest
2. Feature tests - Tests\Feature\CustomerOrderTest

# BigCommerce Client APIs

This is a list of BigCommerce PHP APIs that are used in the given assignment.

1. Client::getTime()
2. Client::getCustomers($filter)
3. Client::getOrdersCount($orderfilter);
4. Client::getCustomer($id);
5. Client::getOrders($orderfilter);

# Env file

New key - CUSTOMISED\_CACHE\_TIMEOUT

This key defines the time (in minutes) for which the cache will be retained. On elapse of this timeout, the cache will expire and subsequence request to Customer Order Count Screen will result in re-populating the cache.

# Improvements

1. CustomerOrderCacheServiceProvider can be improved to provide the functionality to store/retrieve the order history of a customer.
2. View UI can be improved to add the search functionality to enable the end user to search customers by name and/or order counts.
3. This document could be improved to add a section for Class Diagram and a section for Sequence Diagram

-- End of document --