#### Zeeshan Ahmad.

Ecommerce Web Application

Technical Document

Development plan

Version 1.0

Approved 17 july 2018

.

Table of Contents

1. Introduction
   1. Purpose
   2. Tools
   3. Architure
   4. Web app key Requirements
2. Internal Structure of the Web app

* Operations and controller

1. Model for Application
2. Migration for the application
3. Seeder for the Database
4. Database description

* Database view

1. API Endpoints
2. Laravel passport

* Clients

1. Controllers

* User controller
* Product controller
* Order controller

1. Routing
2. A landing view and homepage
3. App Container
4. Child pages
5. Login and registration flow design
6. Interference description

* User dashboard view
* Admin dashboard view

1.Introduction

Purpose

The purpose of this document is to provide a low-level description of the

Shopify Web App providing insight into the structure and design iteration required of each component.

Tools

* PHP7.x
* Laravel 5.3+
* [Vue Router](https://router.vuejs.org/en/)(handle the routing between the components of our Vue application)
* JavaScript and the [VueJS](https://vuejs.org/) framework

Architecture

This Shopify Web application makes selling goods and services online with the the of shopify BOT which provides a listing of the products the seller wishes to sell showing their prices. A Web page where we can see all the details of the single product selected, and finally, where to pay for the product and set how you wish to receive it.

With this in mind, we know that to have a useful Shopify application, we’d need to develop the following:

* A listing page to view all our products.
* A single page to view product information.
* A checkout page where the buyer can enter a delivery address and pay.
* A simple dashboard for a buyer to see all purchased products, and know if they have shipped.
* A way for the seller to see which orders users placed.
* A way for the seller to show a product has been shipped to the user.
* A way for the seller to see and update all product information.

**Web App key requirements:**

* Home page
* Child pages
* DataBase
* Admin Dashboard(where Shopify can automatically update products as well as admin can also modify products manually)
* User Dashboard
* Payment Process integration
* Mail Confirmation
* User login
* Checkout

2.Internal Structure Of the Web Application

Application will have two user types

**Administrator – The owner of the store.**

**Buyer – The person who wants to buy**

These products will be ordered and you need a way to store and track those orders. These would form the basis of any Shopify platform**.**

**The basic details we need to store in the different parts:**

|  |  |
| --- | --- |
| **Table** | **Fields** |
| User | Name, email, home address |
| Order | product ,quantity ,address delivery confirmation |
| Product | name, description,units, image(Shopify bot work) |

The basic operations we need the app to carry out, and where these are handled, are:

**Operation Controller**

Login UserController

Register UserController

User profile UserController

View all orders by a single user UserController

View product listing ProductController

View a single product ProductController

Edit a product ProductController

Order product ProductController

View all orders ProductController

Deliver an order ProductController

Delete an order ProductController

3.Model For application

We will make the models that will interact with our database and hold business logic.

We will be using Models in Laravel to interact with the database.

We will be making three models

1. Product model (This model establishing a relationship with orders placed on the application)
2. Order model (shows which user made an order or which product was ordered.)
3. User model (Laravel installations mostly come with the User model)

4.Migration for the application

We will be using migration to create and maintain our application’s database.It essentially defines how tables should be created or modified.

Reason:

Migrations are useful because they help you manage the database tables, columns and keys. You can share migration files instead of raw SQL, and because migration files are run chronologically, they make it easy to work with git, so it’s great for teams.

There are many methods in the Blueprint class for migrations. we can [read more here](https://laravel.com/docs/5.6/migrations)

5.Seeder for the Database

We will be using Seeders to pre-populate our database with the dummy data.

**A new admin user and a new product table in the database.**

1. Make a seeder class to create the user account for our administration.
2. seeder class for our products table in directory

I.e UserTableSeeder.php at the database/seeds directory

I.e database/seeds/ProductTableSeeder.php

6.Database Description

We will be using a non-file based database like [MySQL](https://www.mysql.com/).For actually defining the the database itself. We are going to use [SQLite](http://www.sqlitetutorial.net/download-install-sqlite/).

Database View

Bot

Our DB

Front end of our

application

Other Shopify

Web app

Scraping product

info

Injecting product info

Into DB through admin forms

7.Defining and Securing out end points

Our application uses Laravel and Vue to create the best application experience. This means we would need to define APIs to provide our Vue components with data.Laravel, by default, has support for **web** and **API** routes. Web routes handle routing for dynamically generated pages accessed from a web browser, while API routes handle requests from clients that need a response in mostly JSON or XML format.

Our application will have APIs for most requests. We need to secure our APIs to ensure only authorised users will access it. For this, we will use [Laravel Passport](https://laravel.com/docs/5.6/passport)

8.Laravel Passport

Laravel Passport comes with it’s own migrations.we will be creating the necessary keys for securing our application

Creating encryption keys needs to generate

clients

|  |
| --- |
| **Secure access tokens** |
| **Personal access** |
| **Password grant** |

, which will be used to generate access tokens.

We will be using the Laravel Passport HasApiToken trait in the Usermodel. This trait will provide a few helper methods to our model, which allow you to inspect the authenticated user’s token and scopes.

9.Controller

We will be using Laravel’s [Route-Model Binding](https://laravel.com/docs/5.6/routing#route-model-binding) to automatically inject our model instance into the controller.

Caveat “The variable name used for the binding has to be the same as the one defined in the route as well”

We will be defining several traditional classes methods for the User,Product and Order but with time we will be defining more according to our requirements of the shopify bot.

Architectural & Component-Level Design

User controller

For now Following class method will be define in our User controller

1. index() – returns all users with their orders.
2. login() – authenticates a user and generates an access token for that user. The createToken method is one of the methods Laravel Passport adds to our user model.
3. register() – creates a user account, authenticates it and generates an access token for it.
4. show() – gets the details of a user and returns them.
5. showOrders() – gets all the orders of a user and returns them

Product controller

For now Following class method will be define in our Product controller

1. index() – fetches and returns all the product records.
2. store() – creates a product record.
3. show() – fetches and returns a single product.
4. uploadFile() – uploads the image for a product we created and returns the url for the product.
5. update() – updates the product record.
6. updateUnits() – adds new units to a product.
7. delete() – deletes a product.

Order controller

For now Following class method will be define in our order controller

1. index() – fetches and returns all the orders.
2. deliverOrder() – marks an order as delivered.
3. store() – creates an order.
4. show() – fetches and returns a single order.
5. update() – updates the order.
6. destroy() – deletes an order.

10.Routing

we will be define routes for our application

1. We have to Define the routes in a way that laravel can differentiate from APIs route and web Routes.
2. We would like to check when we access the route.we are checking if the route requires authentication and if it is restricted to administrators or regular users only and auth:api will be use for ensuring any calls to the routes in the group must be authenticated
3. Route will be identify invoke by the object name.

Frontend of the application interacted with the back end using web request these web request then enter the backend of application at a single point Following a Web route.

Application Front End

Entry point

Application backend

Web requests

Web Route

Entry point for our backend application

Front End

11.A landing view and homepage

Landing view and Homepage

Component

Contents of the Products

Loop

image , name , id , price,etc

Updating from DB through ApIs

12.App container

Application container

Child component

With required methods

Check login

state

Logout

Default Values for

login

login

13.Child pages

1. Login
2. Checkout
3. Confirmation
4. Admin
5. Register
6. Product Page
7. UserBoard

14.Login and Registration

User

Login

Successful

Authentication using APIs

NO

Yes

Saving User data

Saving

Acces token

15.Interface description

User Dashboard View

User will view all his order in the Dashboard

User Dashboard

Order 1

Order 2

Order 3

Requirements

Style

Functionality

script

Looping the order information

1. The designing of the user dashboard shall be in HTML.
2. For Functionality we fetch all the user data from model and use iteration for the displaying the data

Admin Dashboard View

Parent Admin Dashboard

New Products

Products modification

Orders Delivered

Components

Main

APIs for other 3 components

Order

Fetching order details

Deliver button

Required deliver method

To firing the APIs for delivering order

User

Product

Components

Product model

Functionalities

Fetch user data

Functionalities

Admin can create new products ,modify existing products and can set orders as delivered.(on our case this will be handle mostly by bot)

Components

Admin dashboard will be having 3 component and one addition component main for ApIs of other three component and returning data.