Welcome to Chawkbazar
- Navigation
- Supported platforms & ecosystem for local development
- Requirements
- Special Notice

How it works
- Tech We Have Used
- Admin Dashboard (GraphQL & REST)
- Shop Frontend (GraphQL & REST)
- API

Getting Started
- Installation Video
- Prerequisites
    - Frontend
- Getting Started
    - For MAC and Linux(with sail and docker)
        - Prerequisites
        - Installation
        - Video
    - Getting Started with Frontend
        - For Admin :
        - For Shop :
        - Shop config
        - Admin (At the root of the chawkbazar directory, you can run the below commands)
        - Shop (At the root of the chawkbazar directory, you can run the below commands)

Upgrade
- ChawkBazar Update - Virutal Private Server
    - Step 1: Setup Git - Server
    - Install git
    - Config for first time
    - Prepare Git Repository
    - Git & Github
    - Step 2: Shut Down Current Process
    - Step 3: Local Repository & Updated Code
    - Step 4: Update API
    - Step 5: FrontEnd Project Build
    - Config Next Admin App For /admin Sub Directory
    - Install & Build
    - Build the project
    - Upload to GitHub
    - Step 6: Upload Frontend & Run
    - Run frontend app
- Server migration guide
    - How to upgrade the existing deployed laravel 8 server to laravel 9?
- Laravel version upgrade
    - How to upgrade the existing deployed laravel 9 server to laravel 10?
- Breaking changes
    - Version 2.2.2 to 3.0.0 update

Admin dashboard
    - A portion of order management. Order status change.
- Coupon Management
- Customer Management
- Tax Management
- Shipping Management

- Shipping Management
- Settings Management

Available Scripts:

Customization

- For customizing the template's default site settings:

Styles and Assets

- CSS styles:
- Icons:
- For Adding a custom Icon:

Menu

- Data Structure

Banner

- Data Structure
- Data Mapping
  - homeOneBanner
  - homeOneHeroBanner
  - homeTwoHeroBanner
  - homeThreeBanner
  - homeThreeMasonryBanner
  - bannerGrid
  - promotionBanner
  - promotionBannerTwo
  - bannerDataOne
  - bannerDataTwo
  - homeThreeProductsBanner
  - homeFourBanner
  - homeFourGridBanners
  - homeFiveBanner

Brand

Utilities

Backend Integration

- API Integration:
- Data Fetching
  - Uses Example:

Deployment

- AWS (Amazon Web Service)
  - How to create ec2 server?
  - Domain Setup
  - Login to Server
  - Change permission .pem
- Virtual Private Server (Automated Script)
  - Prerequisite
  - At first login your server from terminal
  - Upload api and deployment project to Virtual Server form your PC - RUN on Local PC
    - Server Environment setup script - RUN on Virtual Server
    - Nginx Setup And Settings - RUN on Virtual Server
    - Backend build - RUN on Virtual Server
    - Frontend build script - RUN on Local PC
    - Frontend run script - RUN on Virtual Server
- Vercel
  - vercel.com
  - API
  - Frontend
- Virtual Private Server (Manual)
  - Access Server
  - Install NodeJS & Required Application

- Install NodeJS
- Install Yarn
- Install Zip & Unzip
- Install PM2
- Setup Server
- Introduction
- Step 1 - Installing Nginx
- Step 2: Adjusting the Firewall
- Step 3 – Checking your Web Server
- Step 4 - Install MySQL
- Step 5 - Install PHP
- Step 6 - Create MySQL Database & User
- Step 7 - Change permission for the `www` folder
- Step 8 - Upload API to Server
- Step 9: Setting Up Server & Project
- Step 10 - Create New Nginx for the domain
- Secure Server
  - Step 1: Secure Nginx with Let's Encrypt
- Install API
  - Step 1: Build and Run `api`
- FrontEnd Project Build
- Step 1 - Config Next Admin App For /admin Sub Directory
- Step 2 - Install & Build
- Step 3 - Build the project
  - Shop
  - Admin
  - Image Config
  - Build Project
- Install FrontEnd And Run
- Run frontend app

- cPanel
  - Access Server
  - Create Subdomains
  - Install API
  - Install FrontEnd
  - FrontEnd Project Build
  - step 1 - Build Custom Server
    - shop
    - admin rest
  - Step 2 - Install & Build
  - Step 3 - Build the project
  - Install NodeJs Project

SEO and Analytics

- SEO
- Analytics

Laravel API

- Introduction
- Getting Started
  - Installation Video
  - Prerequisites
  - Resources you might need
  - Packages we have used
- Installation
- For MAC and Linux(with sail and docker)
    - Prerequisites

- Installation
- Configuration
- Console Commands
- Development
- REST API
  - Endpoints
  - Folder structure
  - config
  - database
  - Enums
  - Events
  - Listeners
  - Mail
  - Notifications
  - Providers
  - stubs
- GraphQL API
  - Laravel GraphQL API Endpoint
  - Alternatives
  - Folder Structure
  - Payment Gateway
- Extending The Functionality
- Deployment

Social Login

- Google
- Facebook

OTP (Mobile Number Verification)

- Configuration
  - Twilio Configuration:
  - MessageBird:
- FrontEnd Demo
  - Login with Mobile Number:
  - OTP Verification During Checkout Process:
  - OTP Verification For Updating Mobile Number:

Email Verification

- How to turn on this feature
- Admin view of this feature
- Shop view of this feature
- Configuration

Payment

- Stripe
  - Stripe integrate inside ChawkBazar.
  - How to create & setup Stripe information properly?
  - Special Notes for Stripe users.
  - How can I add card in my user profile for future payments in Stripe?
- PayPal
  - PayPal integrate inside ChawkBazar.
  - How to create & setup PayPal information properly?
- RazorPay
  - RazorPay integration inside PickBazar.
  - How to create & setup RazorPay information properly?
  - Special Notes for RazorPay.
- Mollie
  - Mollie integrate inside PickBazar.
  - Mollie Webhook settings
  - How to create & setup Mollie information properly?

- Special Notes for Mollie.
- Paystack
  - Paystack integrate inside PickBazar.
  - Paystack Webhook settings
  - How to create & setup Paystack information properly?
  - Special Notes for Paystack.
- Integratioin of new payment gateway
  - Getting Started with API
  - Step 1: Install and configure the payment gateway package
  - Step 2: Add payment gateway name in the Enum.
  - Step 3: Configure the Payment Facade for the new payment gateway.
  - Step 4: Using that payment gateway for submitting the order.
  - Step 5: How to Setup webhook in ChawkBazar-laravel follow the steps
  - Getting Started with admin dashboard.
  - Getting Started with shop front.
  - Redirect-base Payment Gateway
  - Non-redirect based payment gateway
- Multiple payment gateway
  - How to activate multiple payment gateways [in Admin] ?
  - Description [sequence by number in the screenshot]
  - How to use multiple payment gateways from customer end [in Shop] ?
  - 1. Select a payment gateway during place order
  - 2. Select different payment gateway (if needed)
  - Description [sequence by number in the screenshot]

Translation

- Existing Language
  - For Rest
- New Language
- Default Language

Multivendor

- Create New Shop
  - Dashboard:
  - Attributes:
  - Products:
  - Order:
  - Staff:
  - WithDraw:
- User Roles:
  - Super Admin:
  - Store Owner:
  - Staff:
- Withdraw Payment:
- FrontEnd Shop

Export Import

- Simple Product
- variable products

Settings Management

- Change Logo
  - API
- Change Site information
  - Shipping class, Tax class & Free Shipping
- Payment Gateway Management
  - Customer
- SEO
- Add or Remove Delivery Schedule
- Shop Settings & Google Map

New Static Page
Shop,
Admin,
FAQ:
How to change the default layout?
Why showing client_id error while Google social login
How to configure Stripe payment gateway?
Why am I facing "You may need an appropriate loader to handle this file type" during running shop rest?
I am changing schema files but changes is not working
Changing .env files but not getting the changes
Changing route but not getting the changes.
I have set `STRIPE_API_KEY` in .env but still getting error.
Can I use it with my existing laravel?
Why am I getting `Access denied for user`?
Why am I getting permission issue during deployment?
Why am I getting "The GET method is not supported for this route. Supported methods: HEAD"?
I am getting "Cannot use 'Mixed' as class name as it is reserved" after access graphql endpoint.
How to resolve the `Load More` `Infinity` loading issue?
I'm trying to upload images, but the images are not displayed on the frontend?
API
Admin
Shop
Build Frontend,
How to resolve the `502 Bad Gateway` error for frontend/admin?
Reason 1:
PORT:
Reason 2:
How to resolve `javascript heap out of memory` issue?
Image upload throw Internal Server Error; how to resolve that?
Sometimes my server shutdown or halts automatically. After restart, it works fine again; how to resolve that?
S3 uploading doesn't work after adding credentials to .env; how to resolve that?
How to rebuild the project?
How to increase upload size?
How to resolve `docker: invalid reference format: repository name must be lowercase.`?
How to remove the existing payment gateway?
Why checkout `Place Order` button is disabled?
How do I log in to the admin panel as the main administrator?
How to disable product popup view and redirect to a single product page?
Support
Thank You!

Welcome to Chawkbazar

Welcome to the ChawkBazar Laravel documentation! Fastest Ecommerce App built with React, NextJS, TypeScript, Laravel and Tailwind CSS.

Navigation

You can find different topics in the table of contents. On desktop, you should see it in the left sidebar. On mobile, you should see it after pressing an icon with Hamberger in the top right corner.

Supported platforms & ecosystem for local development

- Compatible Browsers (Firefox, Safari, Chrome, Edge)
- Node.js 16.15.0 or later
- PHP 8.1
- MySQL 8
- MacOS, Windows, and Linux are supported

Requirements

- Node.js 16.15.0 or later
- yarn
- PHP 8.1
- MySQL 8
- ext_curl
- ext_gmp
- editor: Visual Studio Code(recommended)

Special Notice

One new chapter named `Upgrade` was introduced along with some sub-chapters for easy migration process of app & server ecosystem. Please visit this chapter each time to check if any new changes are introduced. Breaking changes

> Please follow this video playlist with the documentation, and it'll make installation and customization process relatively easy.

How it works

Tech We Have Used

We have used monorepo folder structure with Yarn Workspace. In our app we have three different services:

- api
- shop
- admin

Tech specification for specific part is given below:

Admin Dashboard (GraphQL & REST)

- NextJs
- Typescript
- Tailwindcss

- React Hook Form
- React-Query

Shop Frontend (GraphQL & REST)

- NextJs
- Typescript
- Tailwindcss
- React Hook Form
- React-Query

API

- Laravel

Getting Started

For getting started with the template you have to follow the below procedure. For quick guide you can check below videos for installation.

Installation Video

> *Please follow this video with the documentation, and it'll make the installation process on windows relatively easy.*

Prerequisites

- PHP 8.1
- Composer
- Xamp/Wamp/Lamp/Mamp for any such application for apache, nginx, mysql
- PHP plugins you must need
    - simplexml
    - PHP's dom extension
    - mbstring
    - GD Library

Frontend

- node(16.15 or later)
- yarn
- editor: Visual Studio Code(recommended)

Getting Started

1. First download the file from codecanyon.
2. Unzip the downloaded file and folder structure you get

```
chawkbazar
|-- chawkbazar-api
|-- admin
|-- shop
```

3. From the above folder structure you should notice that our app has three parts `chawkbazar-api` , `shop` and `admin`. So you have to run all the parts separately and sequentially.

Getting started with api

- Run and start xamp/wamp/lamp/mamp and start all the services
- Create a database in your mysql and put those info in next step
- Rename .env.example file to .env and provide necessary credentials. Like database credentials stripe credentials, s3 credentials(only if you use s3 disk) admin email shop url etc.
  - Specially check for this `env` variables

```
DB_HOST=localhost
DB_DATABASE=chawkbazar_laravel
DB_USERNAME=root
DB_PASSWORD=
```

- Run `composer install`

```
▶ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: barryvdh/laravel-dompdf
Discovered Package: bensampo/laravel-enum
Discovered Package: cviebrock/eloquent-sluggable
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: ignited/laravel-omnipay
Discovered Package: intervention/image
Discovered Package: laravel/legacy-factories
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: mll-lab/laravel-graphql-playground
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: nuwave/lighthouse
Discovered Package: pickbazar/shop
Discovered Package: prettus/l5-repository
Discovered Package: spatie/laravel-medialibrary
Discovered Package: spatie/laravel-permission
Package manifest generated successfully.
95 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

  - run `php artisan key:generate`

```
▶ php artisan key:generate

Application key set successfully.
```

  - Run `php artisan marvel:install` and follow necessary steps.

```
Installing Pickbazar Dependencies...

 Do you want to migrate Tables? If you have already run this command or migrated tables then be aware, it will erase all of your data. (yes/no) [no]:
 > yes

Migrating Tables Now....
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (79.70ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (58.57ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (79.56ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (143.06ms)
Migrating: 2020_04_17_194830_create_permission_tables
Migrated:  2020_04_17_194830_create_permission_tables (839.79ms)
Migrating: 2020_06_02_051901_create_pickbazar_tables
Migrated:  2020_06_02_051901_create_pickbazar_tables (1,601.56ms)
Migrating: 2020_10_26_163529_create_media_table
Migrated:  2020_10_26_163529_create_media_table (71.72ms)
Tables Migration completed.

 Do you want to seed dummy data? (yes/no) [no]:
 > yes

Copying necessary files for seeding....
File copying successful
Seeding....
Seed completed successfully!

 Do you want to create an admin? (yes/no) [no]:
 > no

Copying resources files...
Installation Complete
```

- For image upload to work properly you need to run `php artisan storage:link`.

```
The [/var/www/html/public/storage] link has been connected to [/var/www/html/storage/app/public].
The links have been created.
```

- run `php artisan serve`

```
▶ php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sat Apr 10 16:35:26 2021] PHP 8.0.0 Development Server (http://127.0.0.1:8000) started
```

> NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost:8000/`

> For details api doc and requirements details you can go to Laravel API

For MAC and Linux(with sail and docker)

There is an alternate installation procedure for linux and mac. You can follow below procedure to getting started with `sail`

Prerequisites

- Docker

Installation

Video

- Run Docker application first
- Now go to your chawkbazar-laravel root directory and run `bash install.sh`. It will guide you through some process. Follow those steps carefully and your app will be up and running
- Navigate to `api` then `sail down` to stop the container. If you want to remove the volumes then `sail down -v`

> NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost/`

> For details api doc and requirements details you can go to Laravel API

Getting Started with Frontend

4. After configuring API & running it successfully you can choose the directory where you need to work

Below are the directories where you will choose to work for frontend stuffs

```
cd admin/rest
cd shop
```

After choosing your working directory
Go to specific folder and rename the `.env.template` => `.env` and put your api endpoint here. You will find `.env.template` file at the root of your `admin/{chosen-directory-name}` or `shop`

5. Run yarn at the root directory.

```
# on chawkbazar/root directory
yarn
```

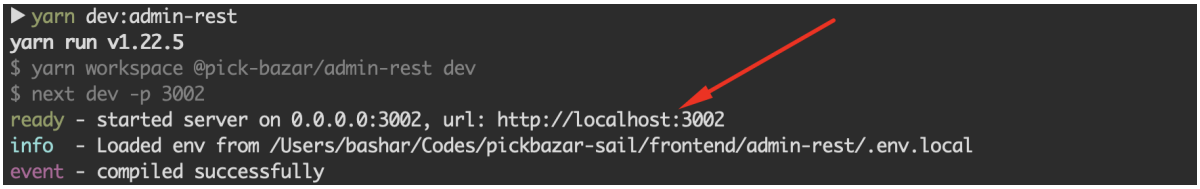
6. ## Scripts To Run the fronted App

For Admin :

For starting the admin dashboard part with corresponding api data run below commands.

- using workspace (At the root of the chawkbazar directory, you can run the below commands)

```
yarn dev:admin-rest
```



- without workspace(if you want to run the command within specific project root of `admin/{chosen-directory-name}`)

```
# for dev mode run below command
# REST
yarn dev
```

This command will run the app in development mode. Open the suggested url in your terminal. like => http://localhost:3000 .

** Note: **

- The page will automatically reload if you make changes to the code. You will see the build errors and lint warnings in the console.
- If you saw any error while running make Sure you setup your API endpoint properly at `.env` file.

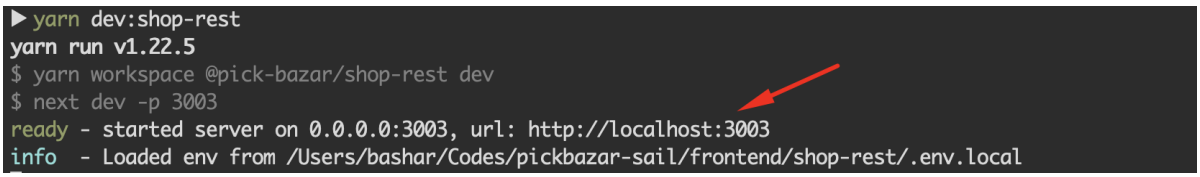
For Shop :

Shop config

For starting the shop part with corresponding api data run below commands.

- using workspace (At the root of the chawkbazar directory, you can run the below commands)

```
yarn dev:shop-rest
```



- without workspace(if you want to run the command within specific project root of `shop`)

```
yarn dev
```

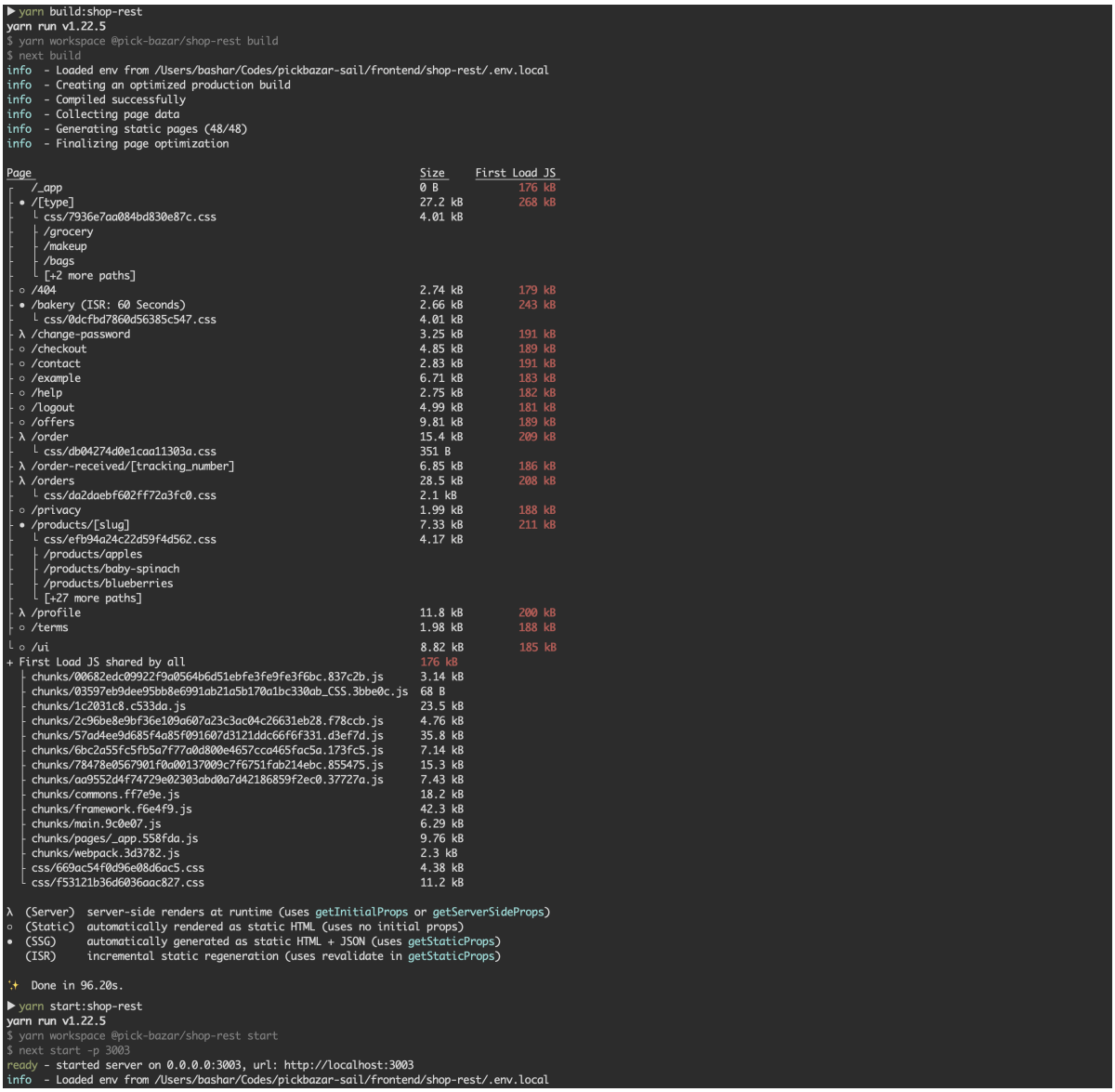
** If you want to test your production build admin or shop in local environment then run the below commands. **

Admin (At the root of the chawkbazar directory, you can run the below commands)

```
# build admin for production
yarn build:admin-rest

#start admin in production mode
yarn start:admin-rest
```

Shop (At the root of the chawkbazar directory, you can run the below commands)

```
# build shop for production
yarn build:shop-rest

# start shop in production mode
yarn start:shop-rest
```



** Note **:
Please see `package.json` file for other builtin helper commands.

7. For development purpose we use yarn workspace if you want to use it then see the `package.json` file at root, for various workspace specific command.

- if you prefer single template then just copy the required template folder and separate them. you'll find their `package.json` file within them and follow the command for dev, build, start.

8. For further development & customization check our Frontend Customization guide.

Upgrade

ChawkBazar Update - Virutal Private Server

> If you follow this Virtual Private Server docs to host your site, then follow this documentation to update ChawkBazar to a new version.

To build the frontend you've to update the `API` first. But before that this time, we'll use git and GitHub to make upload and download relatively easy.

Step 1: Setup Git - Server

> This step is only for first update. From second update start from Step 2

At first, we've to install git on our server and config it.

Install git

```
sudo apt install git
```

Config for first time

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Make sure you change `you@example.com` and `Your Name` with your `email` and `name`.

Prepare Git Repository

At first, go to your `chawkbazar` directory on your server,

```
cd /var/www/chawkbazar
```

2. Then initialize a git on that folder,

```
git init
```

3. Create a new `.gitignore` file

```
nano .gitignore
```

and paste this code to that `.gitignore`,

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring
files.

# dependencies
node_modules
/.pnp
.pnp.js

# testing
/coverage
# *~
*.swp
tmp/

# misc
.DS_Store

# If Packages have their independent repo
# /packages
```

```
# ignore log files
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.idea/
.vscode/
node_modules/
.DS_Store
*.tgz
my-app*
lerna-debug.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
/.changelog
.npm/
packages/server/node_modules
packages/server/error.log
packages/server/debug.log
packages/web/.env
packages/reusecore
packages/cloud

.docz
.now
.vercel
```

After that, `save` that file and use this command for the initial commit,

```
git init
```

```
git add .
```

```
git commit -m "Initial commit"
```

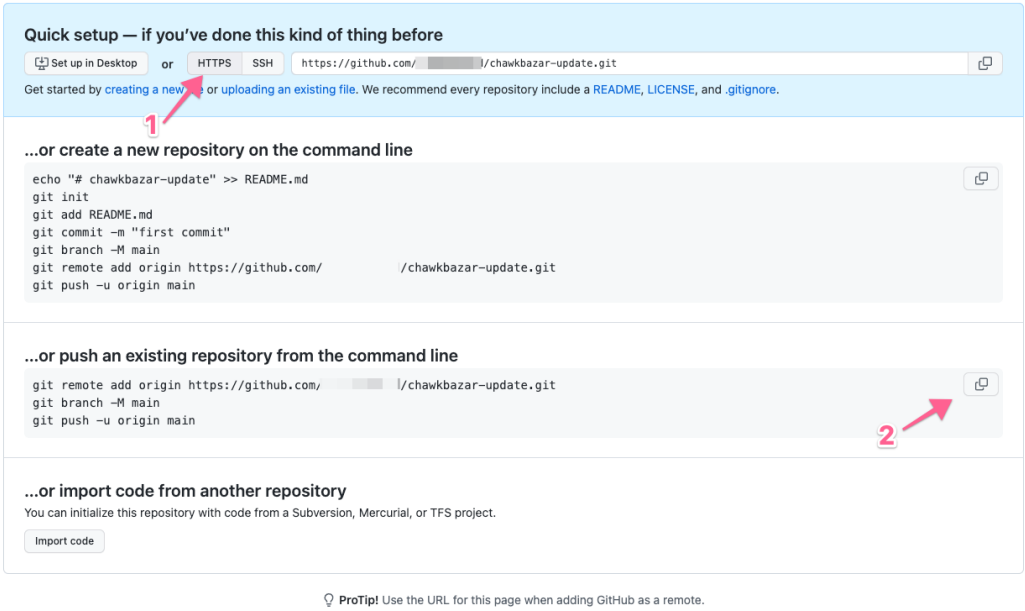
Create a separate branch to maintain the updated code.

```
git branch chawkbazar
```

Git & Github

1. At first, go to https://github.com/ and create an account first. If you already have an account then `Sign In` on that account.
2. Then create a new repository,

3. After creating the repository you'll get a page like this and from this page copy the `second command block` and go to your `server` using `SSH & terminal`,



And `paste` that copied command to `pickabazar` folder and press `enter`

> It'll ask your GitHub `username` and `password`, provide your GitHub `username` and `password`

Your existing repository is successfully connected with GitHub.

Step 2: Shut Down Current Process

At first use this command to shut down all the applications for update,

```
pm2 stop 'all'
```

Step 3: Local Repository & Updated Code

In this step,

1. Clone that `GitHub repository` to your `local computer`
2. Download `update package` from `CodeCanyon`
3. Open your terminal and clone that repository to your computer
4. Checkout to `chawkbazar` branch

```
git checkout -b chawkbazar
```

5. replace `repository` file with downloaded `chawkbazar-laravel` folder content.
6. Then use this command to add all files to git

```
git add .
```

```
git commit -m "Update ChawkBazar API"
```

7. Merge new code with `main` branch

```
git checkout -b main
```

```
git merge chawkbazar
```

> In this step, you'll face a merge conflict issue. Make sure you resolve all the conflicts to maintain your customization with the updated code. You can check this video about resolve merge conflict.

After `resolve` and `commit`, push the code to GitHub.

```
git push origin main
```

Step 4: Update API

In this step, go to your server terminal and go to `/var/www/chawkbazar` directory and pull all updated code,

```
git pull origin main
```

After pull go to `api` folder,

```
cd api
```

and install composer package and optimize compiled file,

```
composer install
```

```
php artisan optimize:clear
```

With that updated API will be installed. To check go to your `YOUR_API_DOMAIN/products`

Step 5: FrontEnd Project Build

Typescript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move the folder to the server then serve the project. We'll do the second method in this tutorial.

> We'll suggest you build the frontend part on your computer and then move the build file to the server using git and github.

Go to your `local git repository` using terminal.

Config Next Admin App For /admin Sub Directory

Edit `admin/rest/next.config.js`,

add `basePath` for '/admin'

```
const withPWA = require("next-pwa");
const runtimeCaching = require("next-pwa/cache");
const { i18n } = require("./next-i18next.config");

module.exports = withPWA({
  basePath: "/admin",
  i18n,
  pwa: {
    disable: process.env.NODE_ENV === "development",
    dest: "public",
    runtimeCaching,
  },

  images: {
    domains: [
      "via.placeholder.com",
      "res.cloudinary.com",
      "s3.amazonaws.com",
      "18.141.64.26",
      "127.0.0.1",
      "localhost",
      "picsum.photos",
      "pickbazar-sail.test",
      "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
      "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
      "lh3.googleusercontent.com",
    ],
  },
```

Install & Build

go to your `root` folder again

Clean previous build file,

```
yarn clean
```

To install all the npm packages run this command,

```
yarn
```

Build the project

```
yarn build:shop-rest
yarn build:admin-rest
```

Upload to GitHub

Use this command to add all files to git,

```
git add .
```

```
git commit -m "Build frontend"
```

```
git push origin main
```

Step 6: Upload Frontend & Run

At first go to your server `chawkbazar` or `git` folder and use this command to pull all the build file,

```
git pull origin main
```

Then install all the node packages,

```
yarn
```

Run frontend app

Use this command to run frontend app as `PM2` again.

```
pm2 start 'all'
```

Now go to Now, go to your `YOUR_DOMAIN` to access the shop page and `YOUR_DOMAIN/admin` for the access admin section.

Server migration guide

How to upgrade the existing deployed laravel 8 server to laravel 9?

At first, remove all the existing `php 7.4` and its extensions by using this command,

```
sudo apt purge php-fpm php-mysql
sudo apt purge php-mbstring php-xml php-bcmath php-simplexml php-intl php-
mbstring php7.4-gd php7.4-curl php7.4-zip composer
```

Then remove the composer,

```
sudo rm /usr/bin/composer
```

Then delete the `vendor` folder from the `chawkbazar-laravel -> chawkbazar-api` folder.

```
cd /var/www/chawkbazar-laravel/chawkbazar-api

sudo chown -R $USER:www-data storage
sudo chown -R $USER:www-data bootstrap/cache

rm vendor -rf
```

Then install PHP 8, and its extensions,

```
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

```
sudo apt install php8.0-fpm php8.0-mysql
```

```
sudo apt install php8.0-mbstring php8.0-xml php8.0-bcmath php8.0-simplexml
php8.0-intl php8.0-gd php8.0-curl php8.0-zip php8.0-gmp
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d
64d92e6563bbde02') { echo 'Installer verified'; } else { echo 'Installer
corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

```
sudo mv composer.phar /usr/bin/composer
```

Then update 7.4 fpm to 8.0 fpm from,

```
/etc/nginx/sites-enabled/chawkbazar
```

```
    charset utf-8;

    # For API
    location /backend {
        alias /var/www/pickbazar/api/public;
        try_files $uri $uri/ @backend;
            location ~ \.php$ {
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $request_filename;
            fastcgi_pass   unix:/run/php/php8.0-fpm.sock;
         }
    }

    location @backend {
       rewrite /backend/(.*)$ /backend/index.php?/$1 last;
    }

    # For FrontEnd -> Rest
    location /{
         proxy_pass http://localhost:3000;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection 'upgrade';
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
    }

    location /admin{
         proxy_pass http://localhost:3002/admin;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection 'upgrade';
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
    }

    error_page 404 /index.php;

    location ~ \.php$ {
         fastcgi_pass unix:/var/run/php/php8.0-fpm.sock;
         fastcgi_index index.php;
         fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
         include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
         deny all;
```

Then install composer packages,

```
cd /var/www/chawkbazar-laravel/chawkbazar-api
```

```
composer install
```

```
php artisan optimize:clear
```

```
php artisan marvel:install
```

> php artisan marvel:install will remove all of your existing data. Ensure you export or backup your data before using that command.

```
sudo chown -R www-data:www-data storage
sudo chown -R www-data:www-data bootstrap/cache
```

Laravel version upgrade

How to upgrade the existing deployed laravel 9 server to laravel 10?

At first, remove all the existing `php 7.4` and its extensions by using this command,

```
sudo apt purge php8.0-fpm php8.0-mysql
sudo apt purge php8.0-mbstring php8.0-xml php8.0-bcmath php8.0-simplexml php8.0-
intl php8.0-gd php8.0-curl php8.0-zip php8.0-gmp composer
```

Then remove the composer,

```
sudo rm /usr/bin/composer
```

Then delete the `vendor` folder from the `chawkbazar-laravel -> chawkbazar-api` folder.

```
cd /var/www/chawkbazar-laravel/chawkbazar-api

sudo chown -R $USER:www-data storage
sudo chown -R $USER:www-data bootstrap/cache

rm vendor -rf
```

Then install PHP 8.1, and its extensions,

```
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

```
sudo apt install php8.1-fpm php8.1-mysql
```

```
sudo apt install php8.1-mbstring php8.1-xml php8.1-bcmath php8.1-simplexml
php8.1-intl php8.1-gd php8.1-curl php8.1-zip php8.1-gmp
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d
64d92e6563bbde02') { echo 'Installer verified'; } else { echo 'Installer
corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

```
sudo mv composer.phar /usr/bin/composer
```

Then update 8.0 fpm to 8.1 fpm from,

```
/etc/nginx/sites-enabled/chawkbazar
```

```
    # For API
    location /backend {
        alias /var/www/chawkbazar-laravel/chawkbazar-api/public;
        try_files $uri $uri/ @backend;
            location ~ \\.php$ {
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $request_filename;
            fastcgi_pass   unix:/run/php/php8.1-fpm.sock;
         }
    }

    location @backend {
       rewrite /backend/(.*)$ /backend/index.php?/$1 last;
    }

    # For FrontEnd
    location /{
        proxy_pass http://localhost:3003;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location /admin{
        proxy_pass http://localhost:3002/admin;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    error_page 404 /index.php;

    location ~ \\.php$ {
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
```

Then install composer packages,

```
cd /var/www/chawkbazar-laravel/chawkbazar-api
```

```
composer install
```

```
php artisan optimize:clear
```

```
php artisan marvel:install
```

> php artisan marvel:install will remove all of your existing data. Ensure you export or backup your data before using that command.

```
sudo chown -R www-data:www-data storage
sudo chown -R www-data:www-data bootstrap/cache
```

Breaking changes

In this chapter the breaking changes will be listed here version wise sequentially. So that our customers can easily track down the changes & take necessary steps during their migration process.

Version 2.2.2 to 3.0.0 update

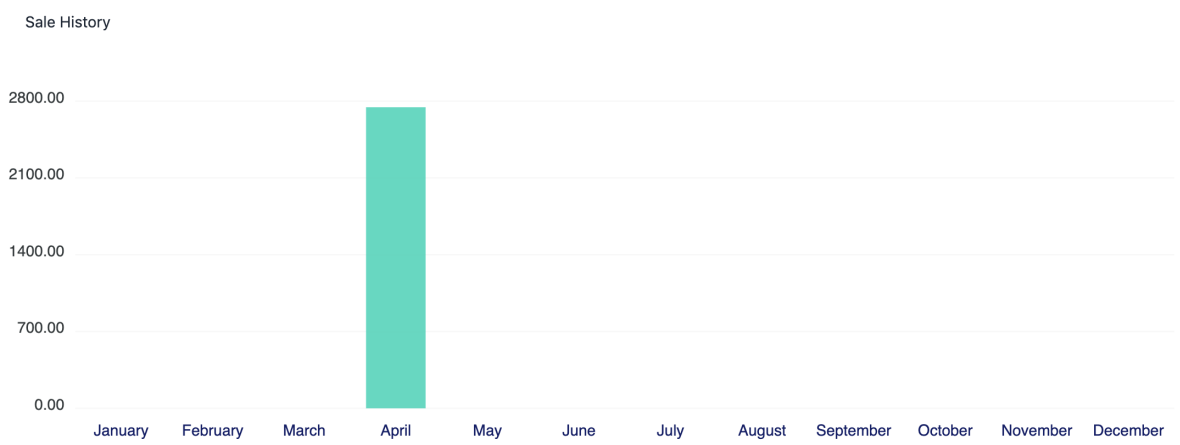
- Laravel version 8 to 9 upgradation process was introduced. Laravel 8 to 9 upgrade
- There are changes coming in database section. Before update & run any database migration command, please keep the database backup.
    - Order table was updated in the database.
    - Order-Status table was deleted. Now, order statuses are managed dynamically
- Before update please keep your existing code backup. Because there are changes inside order code architecture & payment gateway integration system.
- Product inventory system was managed based on new order & payment status.

Admin dashboard

- ## Analytics Dashboard

You will get a complete analytics dashboard to know the overview of your shop.

Total Revenue (Last 30 Days)	Total Order (Last 30 Days)	New Customer (Last 30 Days)	Todays Revenue
$7,844.05	25	16	$0.00



Recent Orders

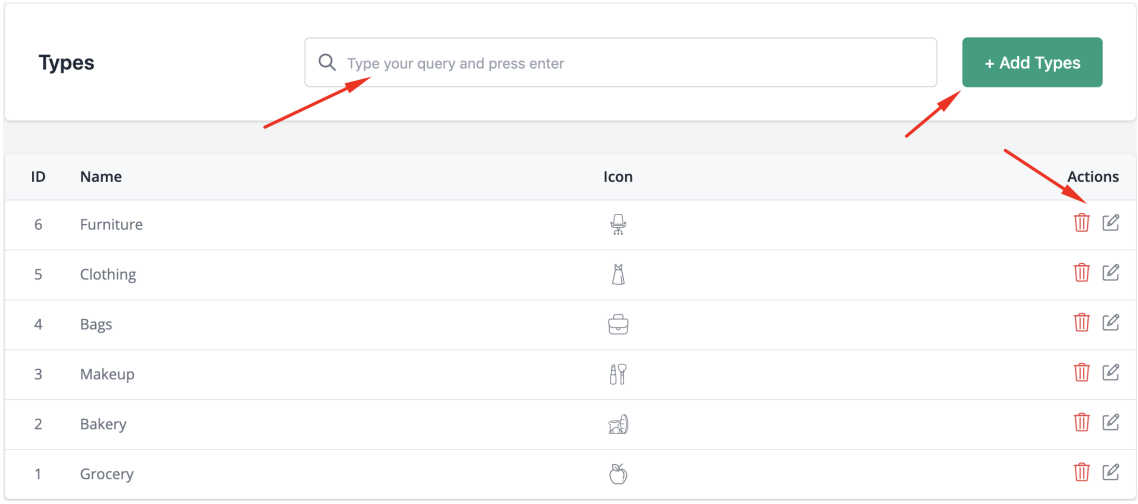
Tracking Number	Total	Order Date	Status
GkszijsVAcae	$2,744.33	5 days ago	Order Received
lrrdghqshkys	$1,748.30	14 days ago	Order Dispatched
iTUVhT87QdE8	$526.98	17 days ago	Order Dispatched
A9mvTnzhGxkc	$294.49	17 days ago	Order Received
BlOnTtmsHHHE	$12.80	17 days ago	Order Received
rulb1kz4UllN	$2.45	19 days ago	Shipment Refused by Consignee
NqlykeIW6awe	$97.80	18 days ago	Delivered
TfzYmld1AR32	$18.00	18 days ago	Ready To Dispatch
PXqs8RXQBGm5	$7.20	19 days ago	Delivered
K2isul12Cetf	$540.00	19 days ago	Ready To Dispatch

Popular Products

ID	Name	Type	Price/Unit	Quantity
4	Brussels Sprout		$5.00	17
3	Blueberries		$3.00	30
2	Baby Spinach		$0.60	10
6	Clementines		$3.00	50
5	Celery Stick		$6.00	18
7	Sweet Corn		$5.00	50
8	Cucumber		$2.50	25
103	Armani Purse		$80.00	50
9	Dates		$10.00	50
102	Armani Leather Purse		$50.00	50

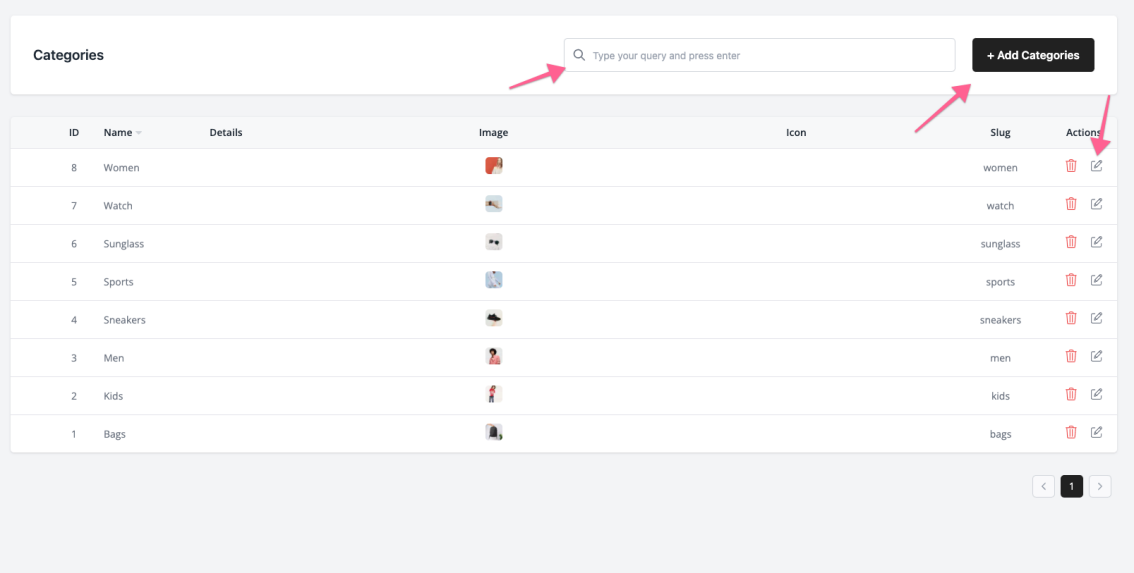
- # Manage Product Type

  In `Types` menu you will get the product types and you can add, remove or modify product type from there.



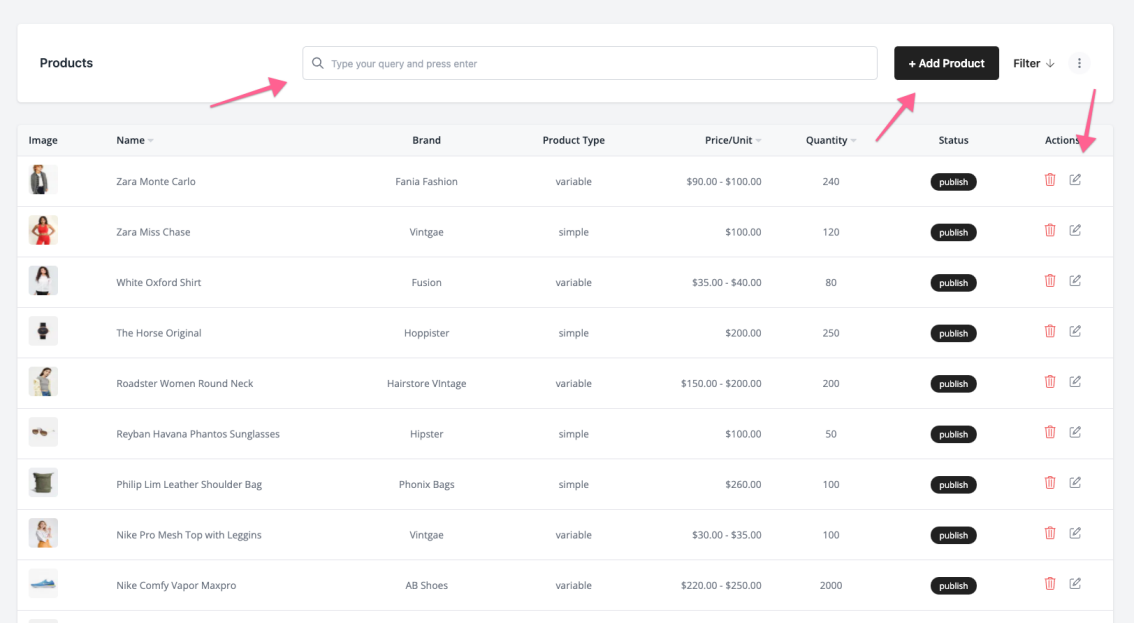
- # Manage Product Category

  In `Categories` menu you will get the product types and you can add, remove or modify product categories from there.

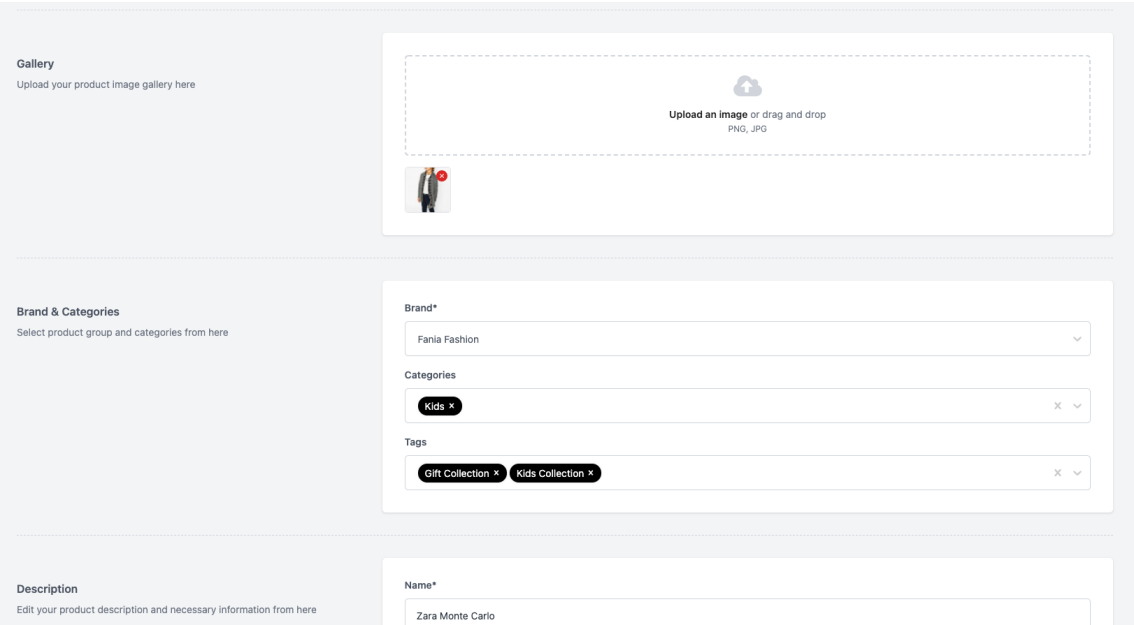


- # Product Management

In `Products` menu you will get the products and you can add, remove or modify products from there.

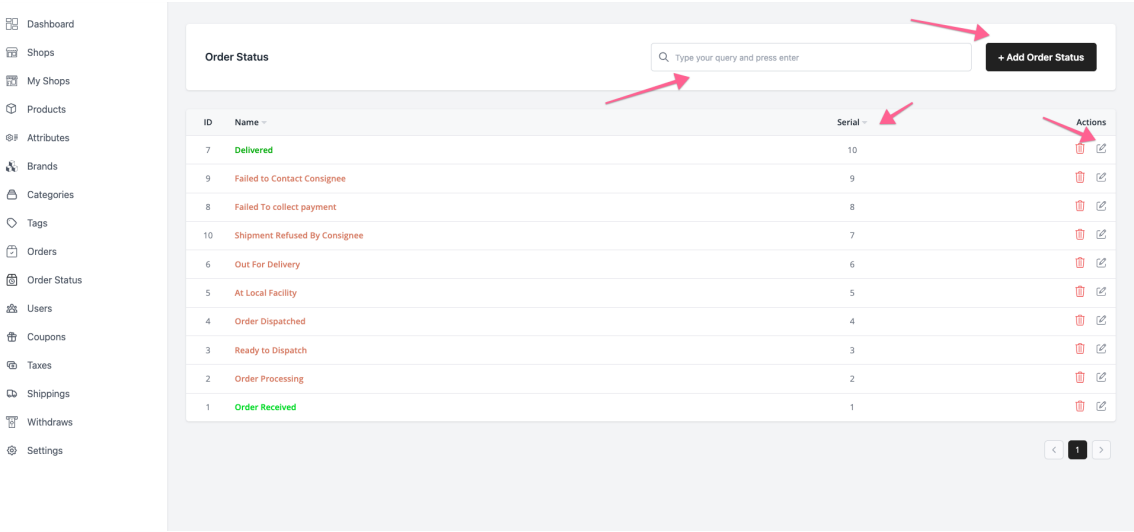


A portion of product form



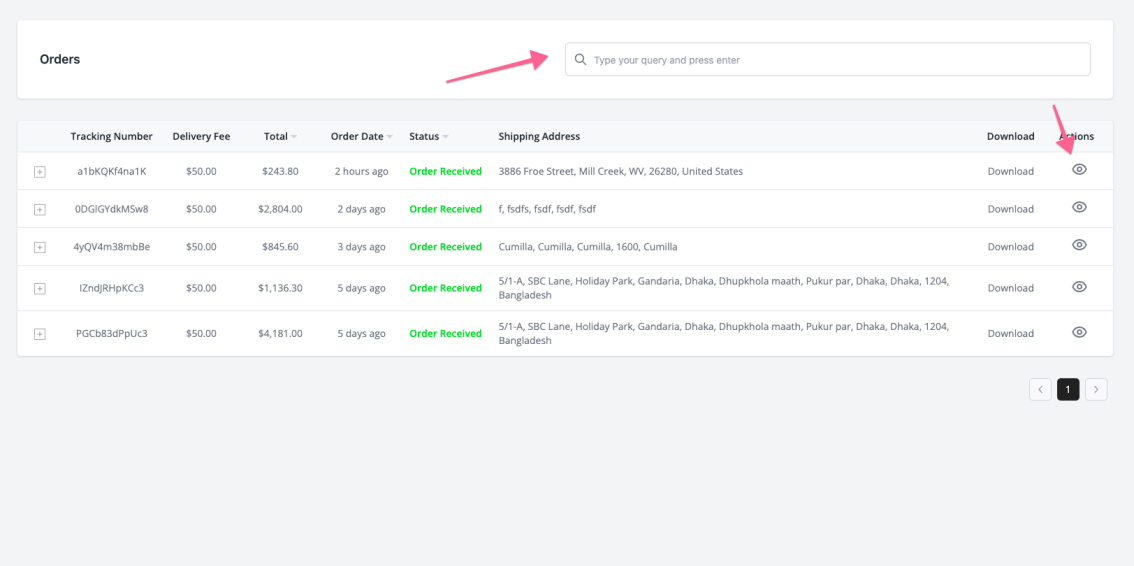
- ## Order Status

In `Order Status` menu you will get the order status list and you can add, remove or modify order status from there.

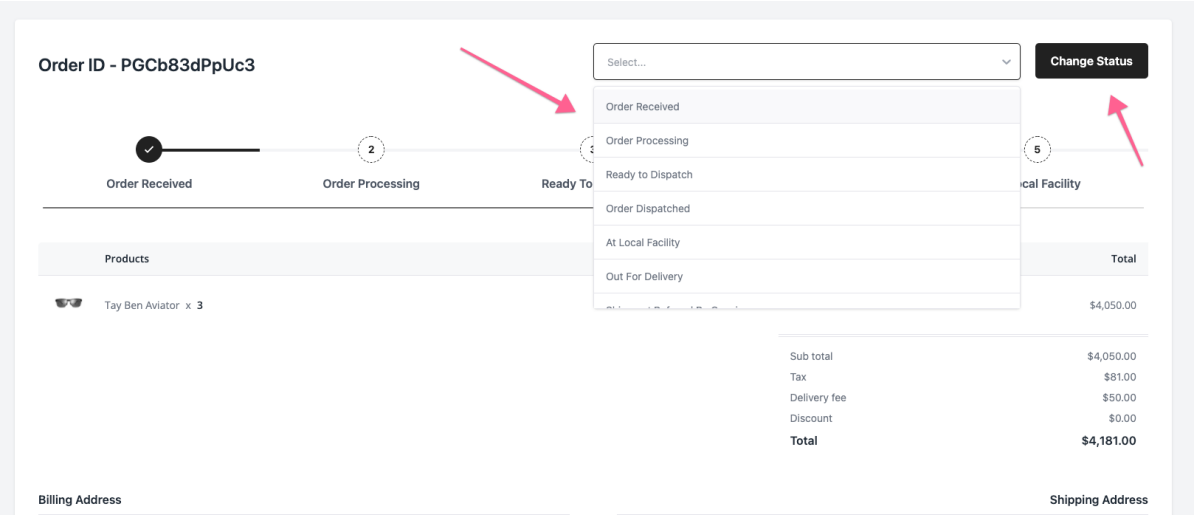


Order Management

In `Order` menu you will get the order list and you can add, remove or modify order from there.

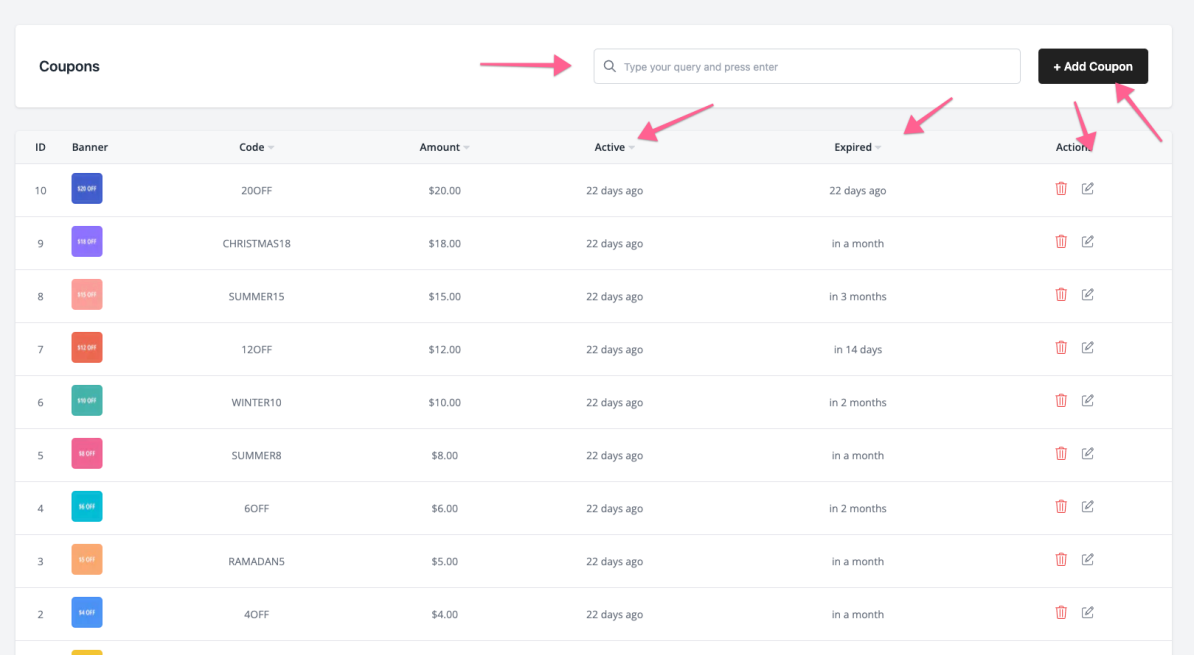


A portion of order management. Order status change.



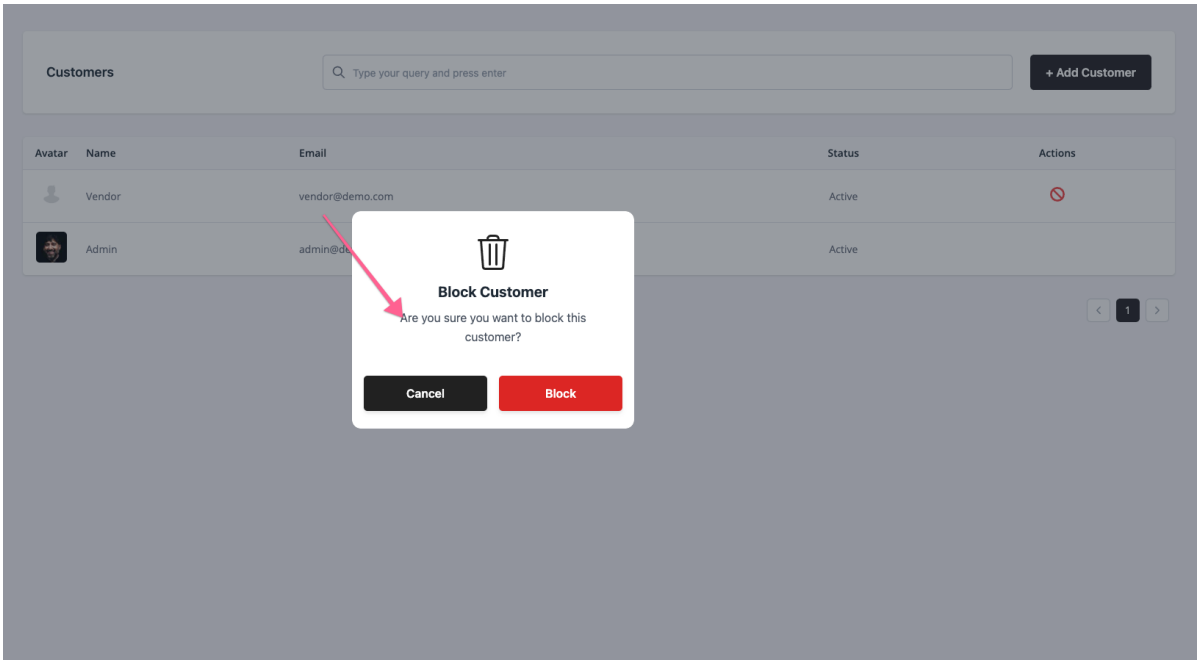
Coupon Management

In `Coupon` menu you will get the Coupon list and you can add, remove or modify Coupon from there.



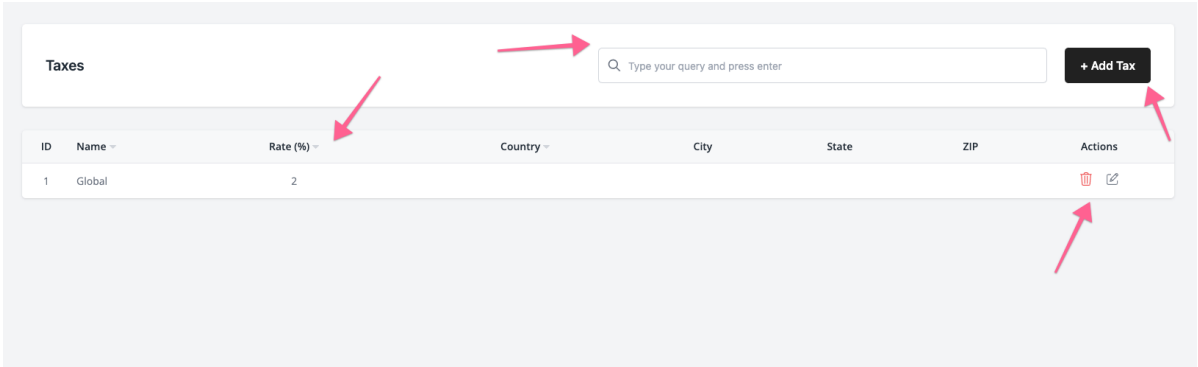
Customer Management

In `Customer` menu you will get the Customer list and you can add, remove or modify Customer from there.



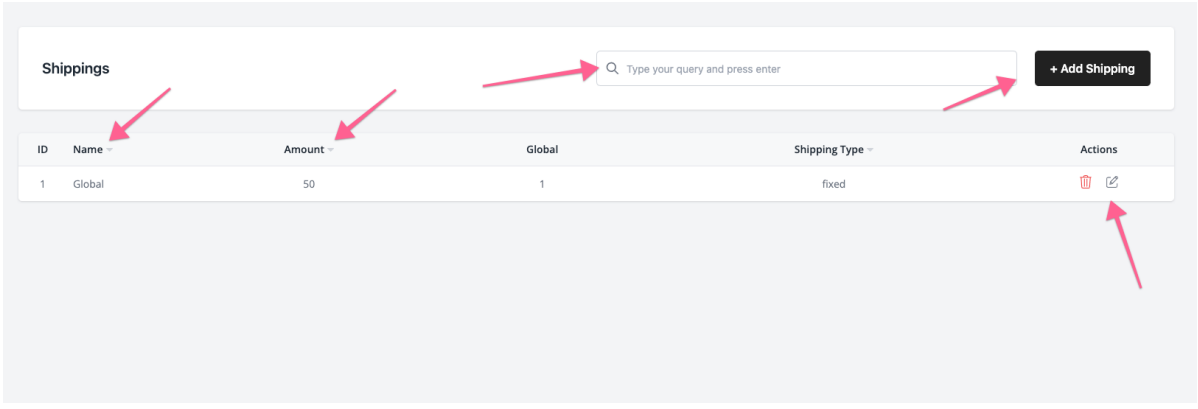
Tax Management

In `Tax` menu you will get the Tax list and you can add, remove or modify Tax from there.



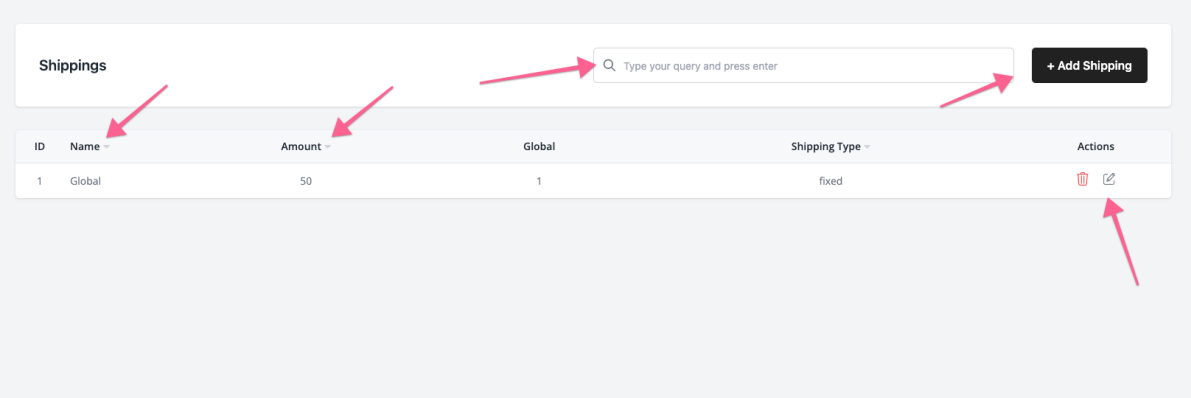
Shipping Management

In `shipping` menu you will get the shipping list and you can add, remove or modify shipping from there.



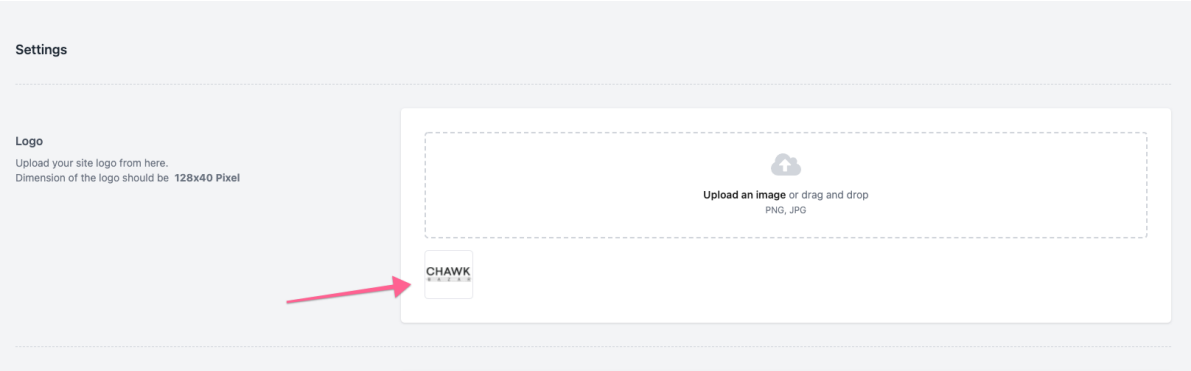
Shipping Management

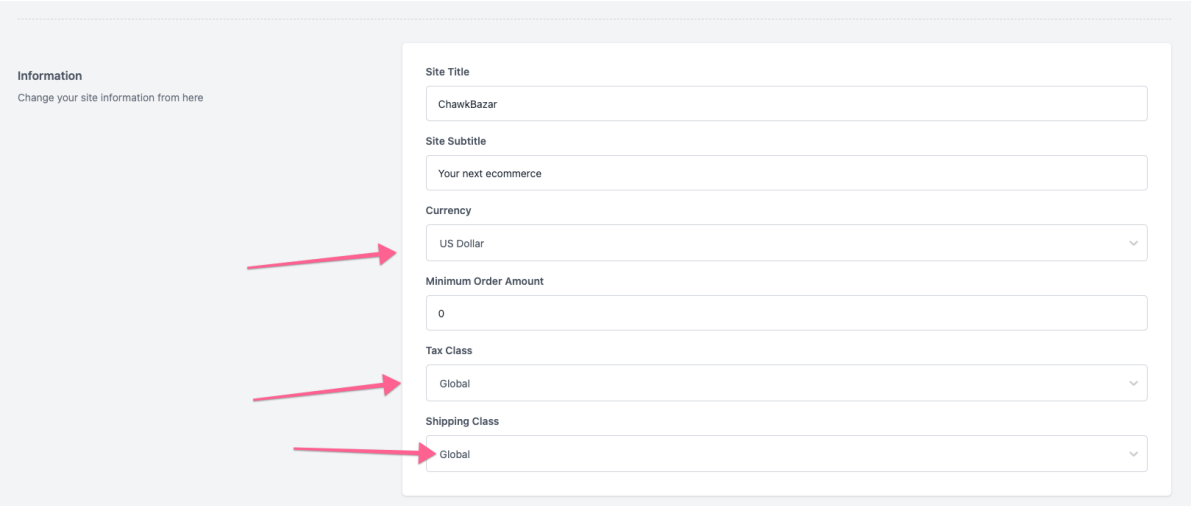
In `shipping` menu you will get the shipping list and you can add, remove or modify shipping from there.



Settings Management

In `settings` menu you will get the settings management form there.





Available Scripts:

You can run below commands in the root folder for your need.

```
"clean": "yarn workspaces run rimraf \"
{.next,node_modules,__generated__,.cache,src/graphql/*.d.ts,src/framework/graphql
/**/*.d.ts}\" && rimraf node_modules",
"dev:shop-rest": "yarn workspace @chawkbazar/shop dev",
"build:shop-rest": "yarn workspace @chawkbazar/shop build",
"start:shop-rest": "yarn workspace @chawkbazar/shop start",
"dev:admin-rest": "yarn workspace @marvel/admin-rest dev",
"build:admin-rest": "yarn workspace @marvel/admin-rest build",
"start:admin-rest": "yarn workspace @marvel/admin-rest start",
"prepare": "husky install",
```

** Note: ** Also, individual Scripts are available under every individual package.
You can check out them from their individual package.json file.

Customization

For customizing the template's default site settings:

> *[**your-frontend-project**] = admin [`rest`] or shop*

If you want to customize the site settings such as site info, default(logo, language , currency code),navigation etc, you can easily change those settings from

```
[your-frontend-project]/src/settings/site.settings.ts
```

In this file, find your required settings key and change it's value according to your need.

For example, If you want to change the currencyCode : please go to `[your-frontend-project]/src/settings/site.settings.ts` and find the following portions and change it.

```
currencyCode: "USD";
```

NOTE ** Some of these options are customizable through ADMIN Dashboard.

Styles and Assets

CSS styles:

> [your-frontend-project] = admin [`rest`] or shop

We use tailwindcss framework with some customization which you find at :

```
open [your-frontend-project]/tailwind.config.js
```

For tailwindcss documentation:

Go to Tailwindcss

Icons:

for our icons

```
open [your-frontend-project]/src/components/icons
```

For Adding a custom Icon:

To add a custom icon please follow this procedure.

1. Open your custom SVG icon file in the code editor and copy all the code.
2. Then Go to src -> components -> icons folder and create a new .tsx file.
3. Then here create a function component and paste your copied SVG code inside the return statement.
4. Then covert all the SVG's kebab-cases properties into camelCase format except the data-name property. For ex. change the stroke-width and fill-color into strokeWidth and fillColor. (for reffernce you can see one of our icon.)
5. If your custom SVG code has any single custom color then change them into fillColor.

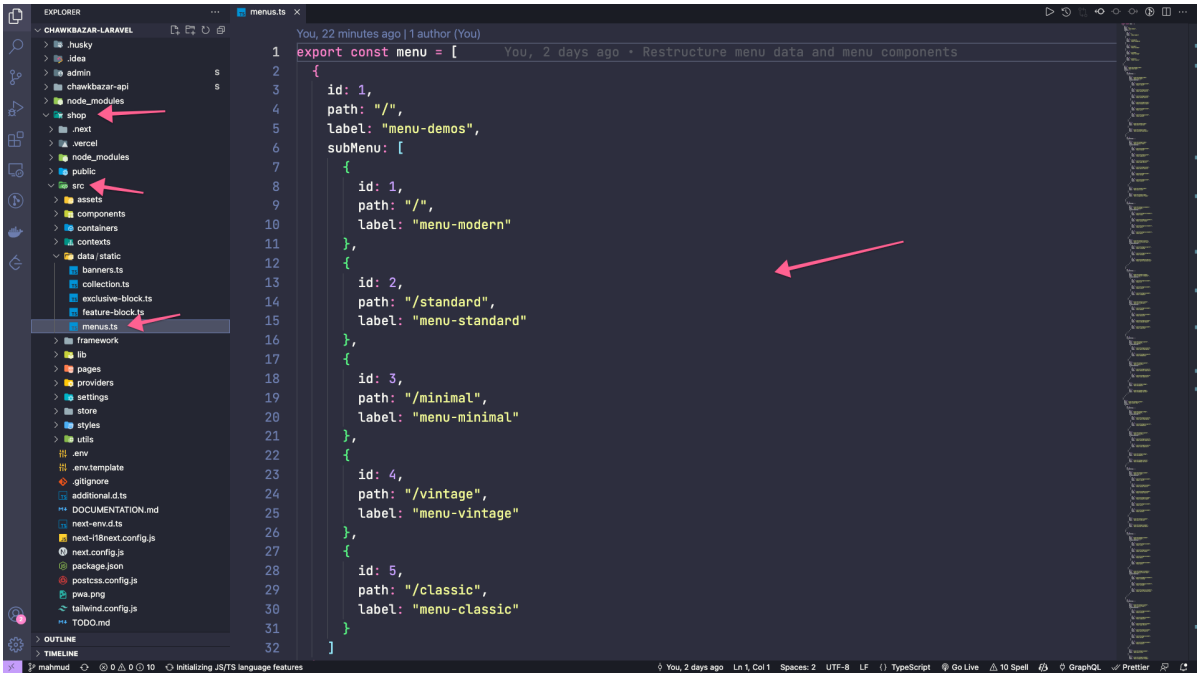
Menu

> Please follow this video with the documentation, and it'll make the menu build process relatively easy.

https://www.youtube.com/watch?v=ewNGG_i3r-Y&feature=emb_title

Mega menu for the `shop` are coming from a JSON file which is located at,

```
shop -> public -> data -> menus.json
```



You've to update this JSON to build the mega menu.

Data Structure

```
id -> ID for menu or submenu
path -> After clicking the menu in frontend which path will be redirect
label -> Label of that men
subMenu -> JSON for submenu
mobileMenu -> Different menu for mobile.
```

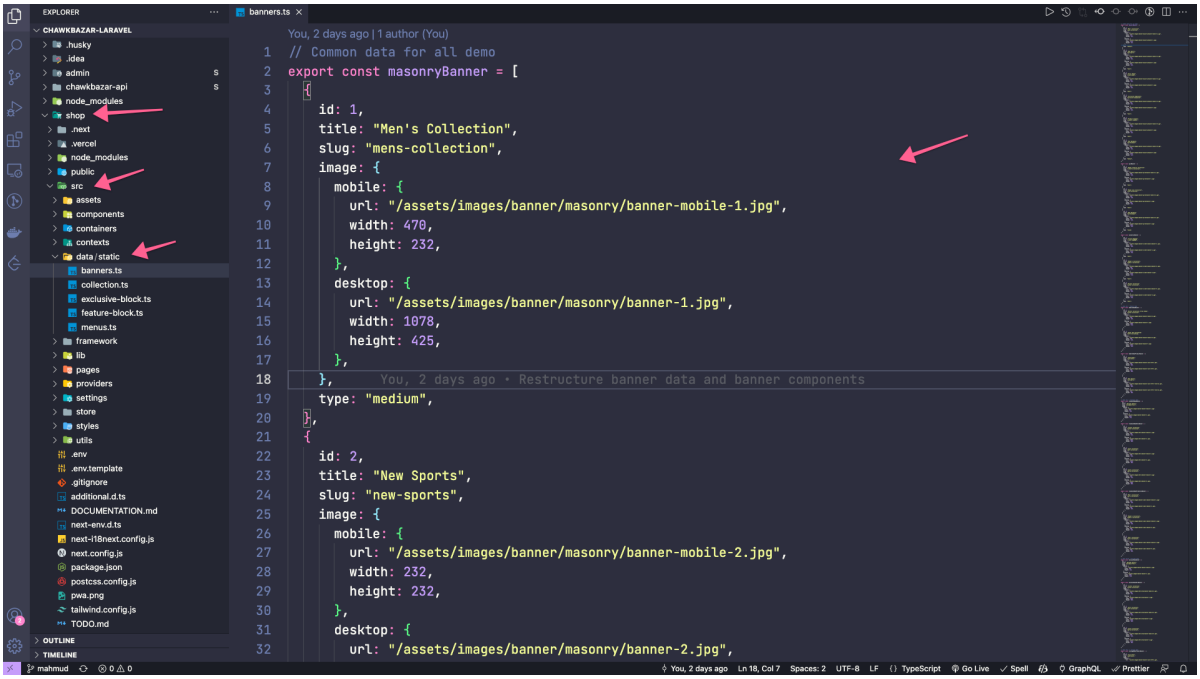
Banner

> Please follow this video with the documentation, and it'll make the banner update process relatively easy.

https://www.youtube.com/watch?v=AMOexuHTKa0

The banner data are coming from a JSON file which is located at,

```
shop -> public -> data -> banner.json
```



You've to update this JSON to build the mega menu.

Data Structure

```
id -> ID for a banner
title -> Title for a banner
slug -> This is a tag slug
```

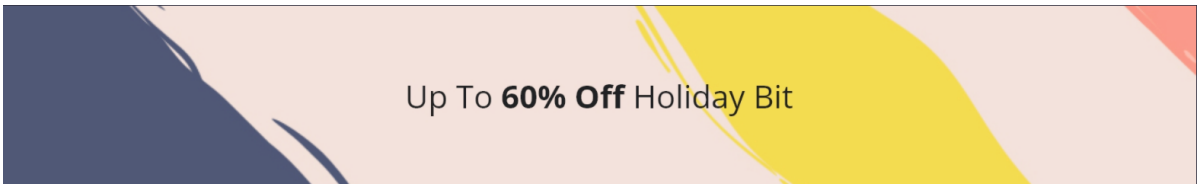
> At first, create the tag and add that tag slug on the `slug` path.

> For example, if you want to start a special campaign for some products, just create a tag for that campaign in admin, add that tag to their related products, and on `banners.json`, add that tag as a slug.

```
image -> Image path
mobile -> JSON object for mobile image
desktop -> JSON object for desktop image
url -> Path of the image
width -> Width of the image for next/image optimization
height -> Height of the image for next/image optimization
```

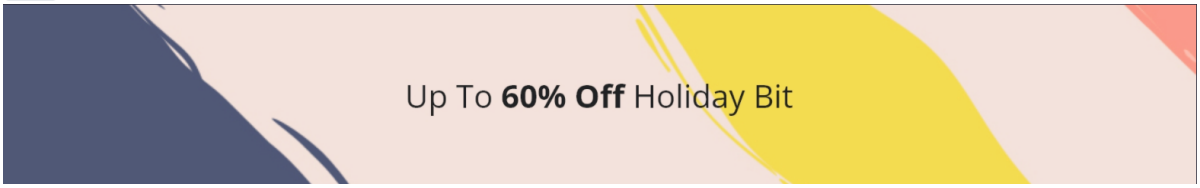
Data Mapping

homeOneBanner

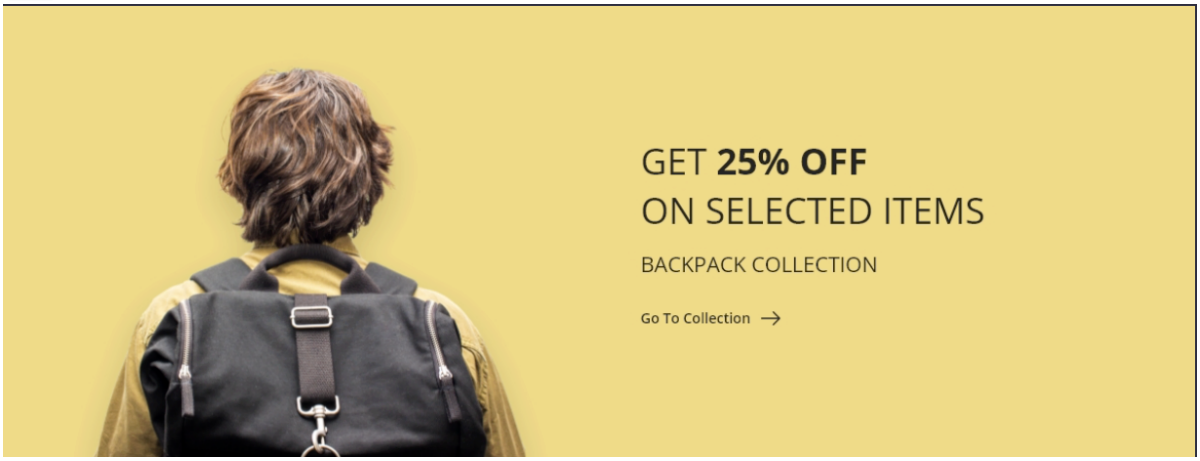


homeOneHeroBanner

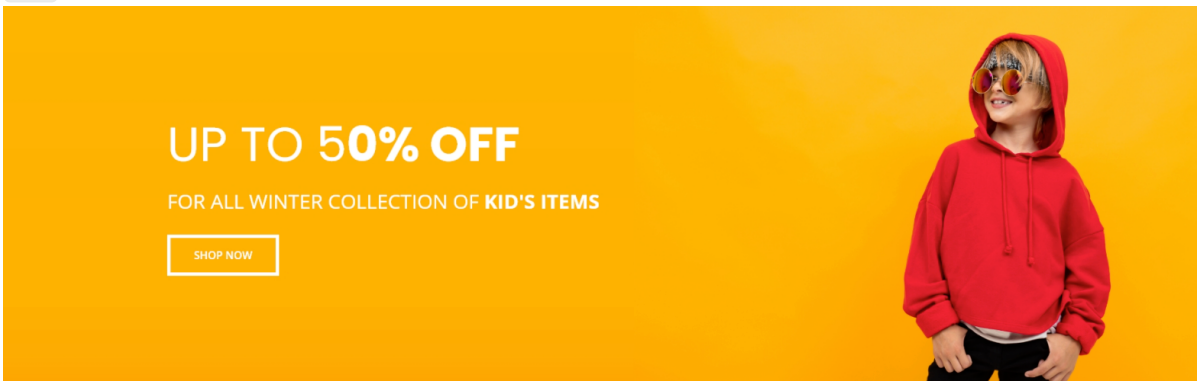
[1]



[2]

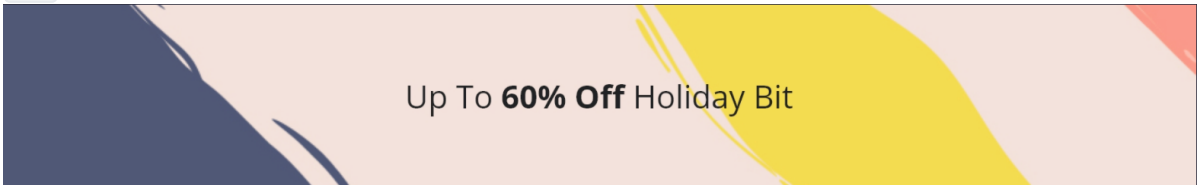


[3]

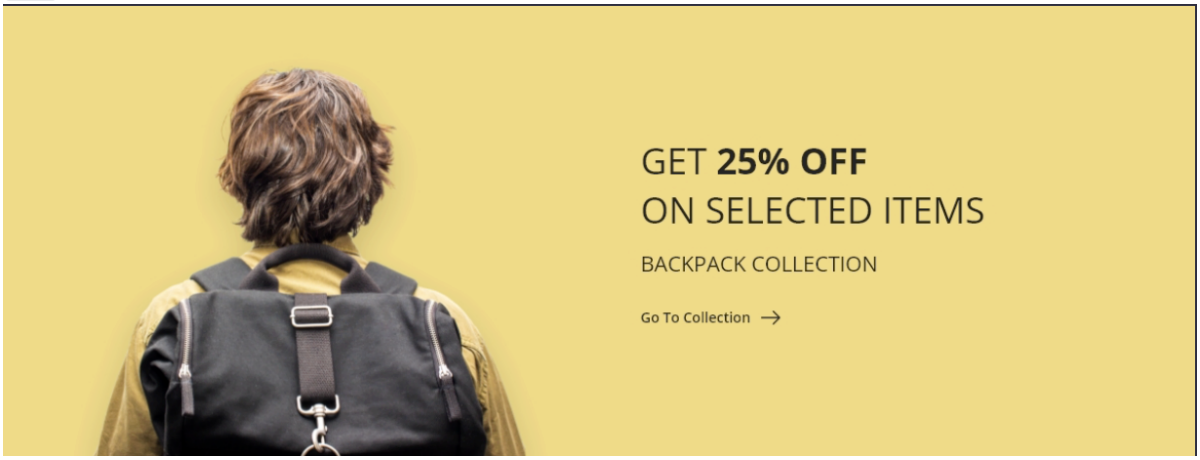


homeTwoHeroBanner

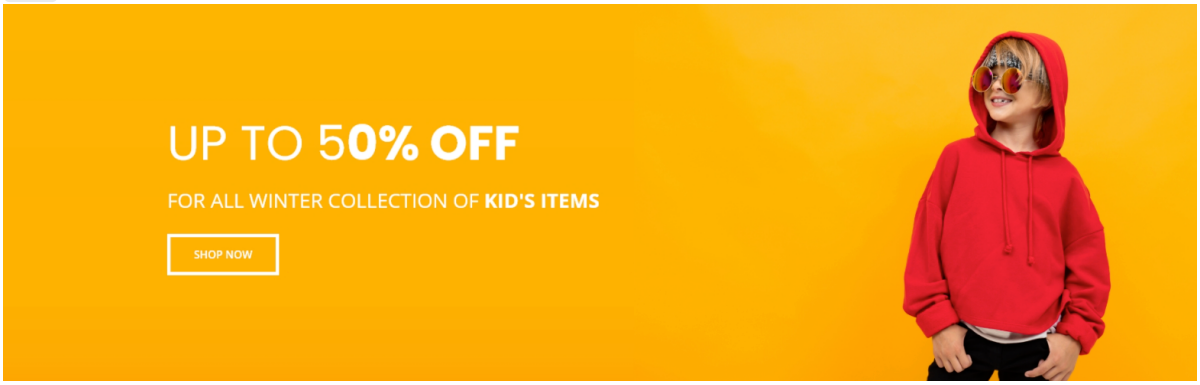
[1]



[2]

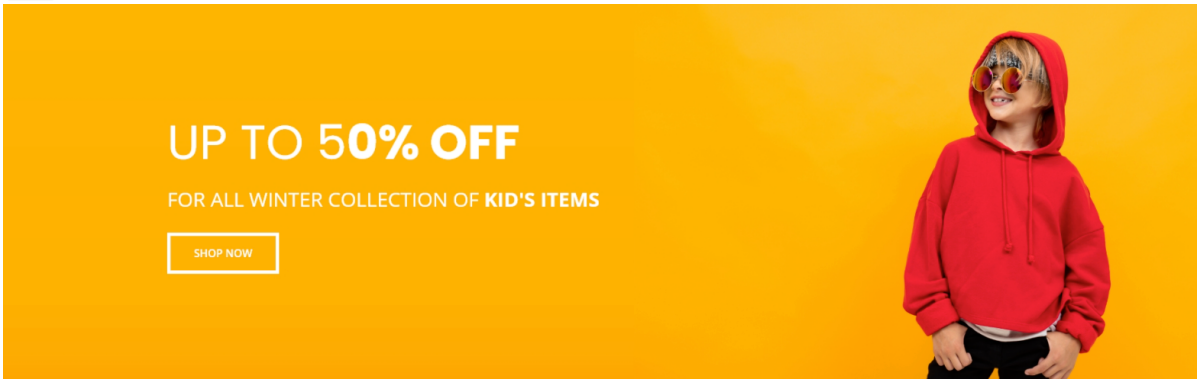


[3]

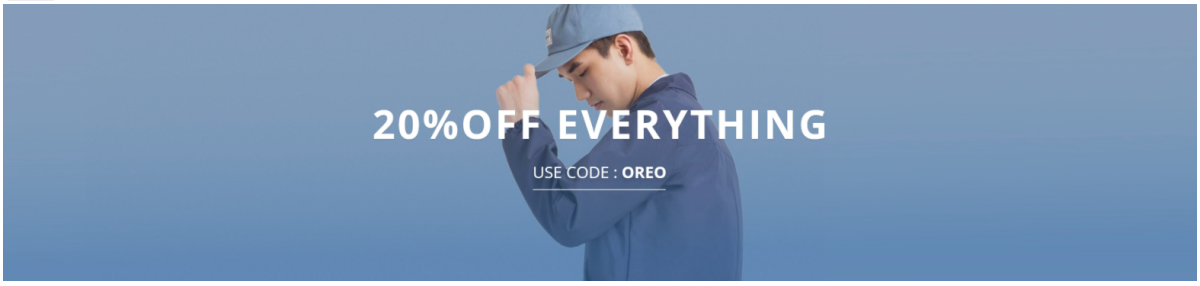


homeThreeBanner

[1]

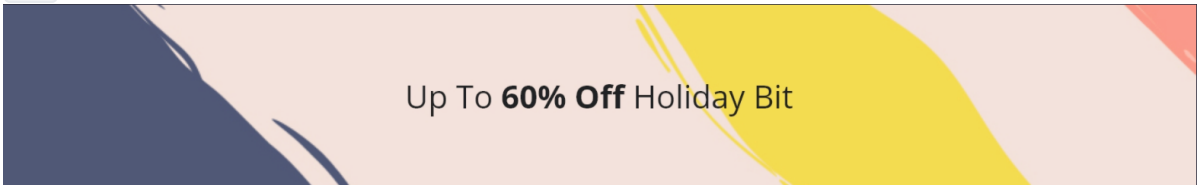


[2]

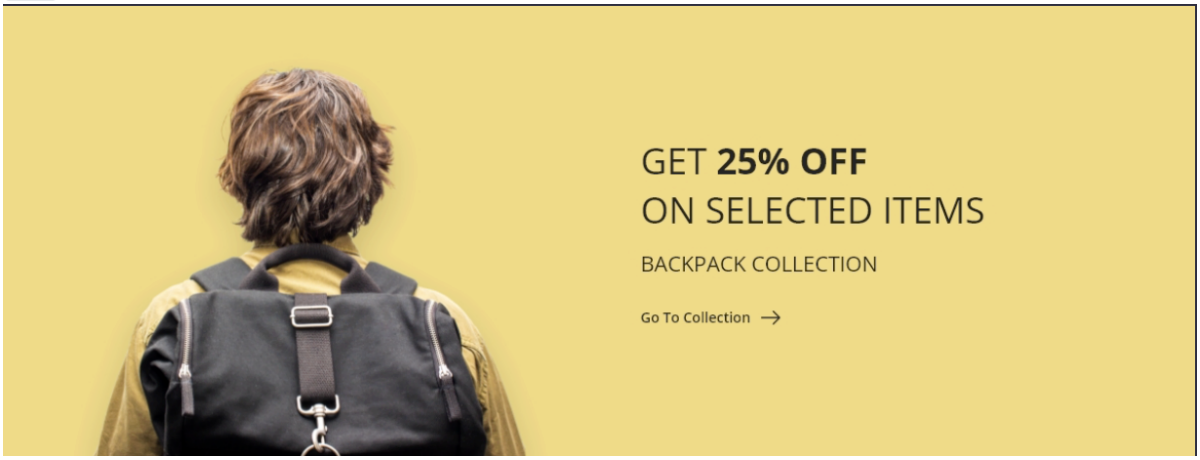


homeThreeMasonryBanner

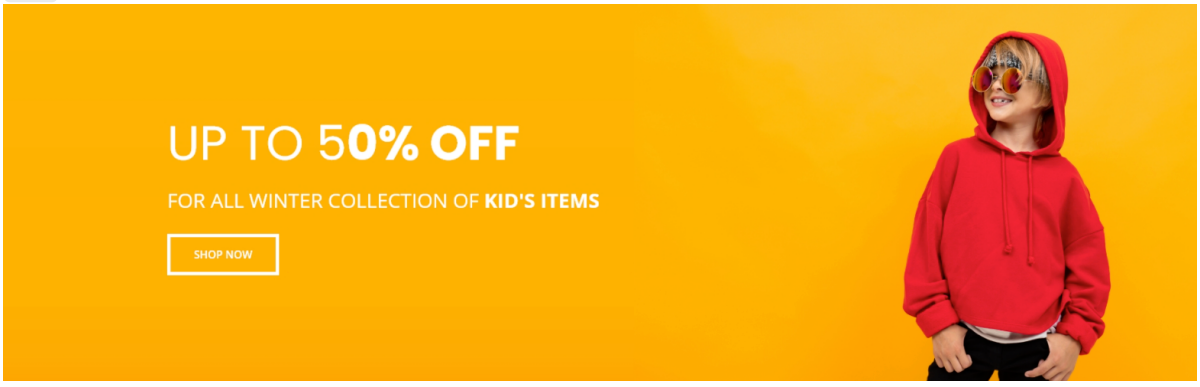
[1]



[2]



[3]

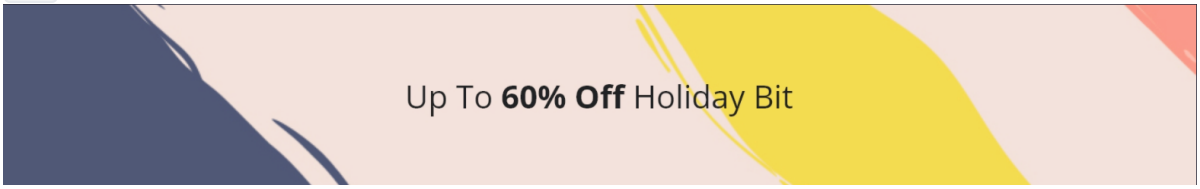


[4]

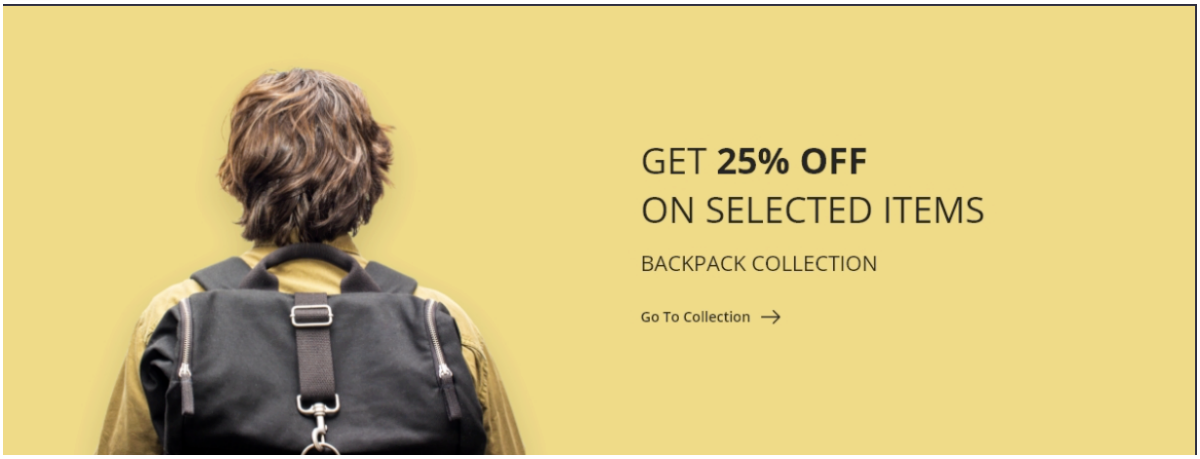
[5]

bannerGrid

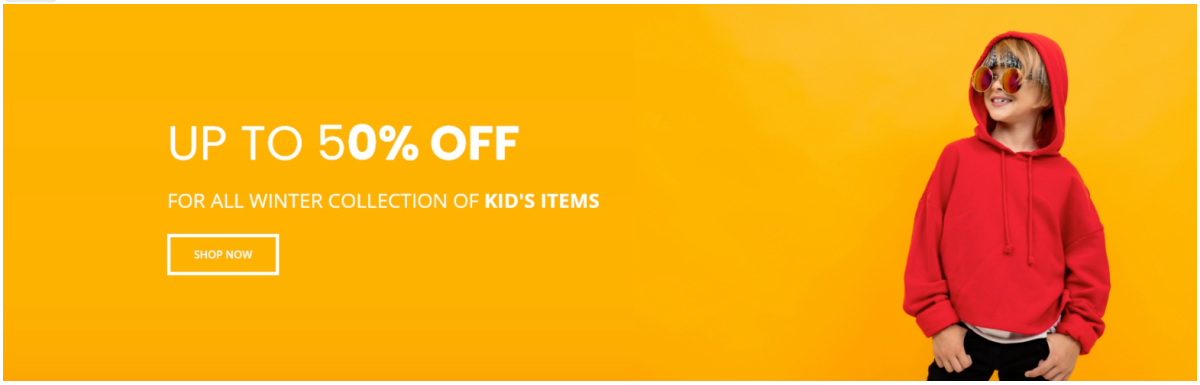
[1]



[2]

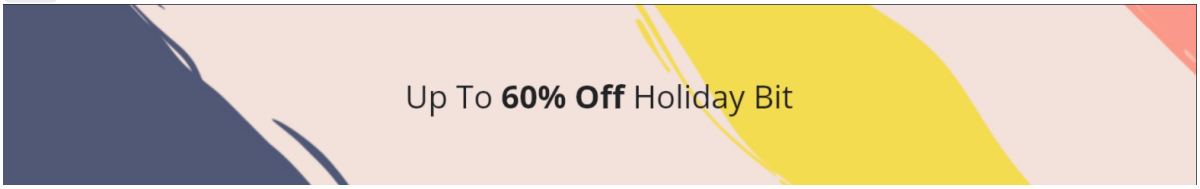


[3]

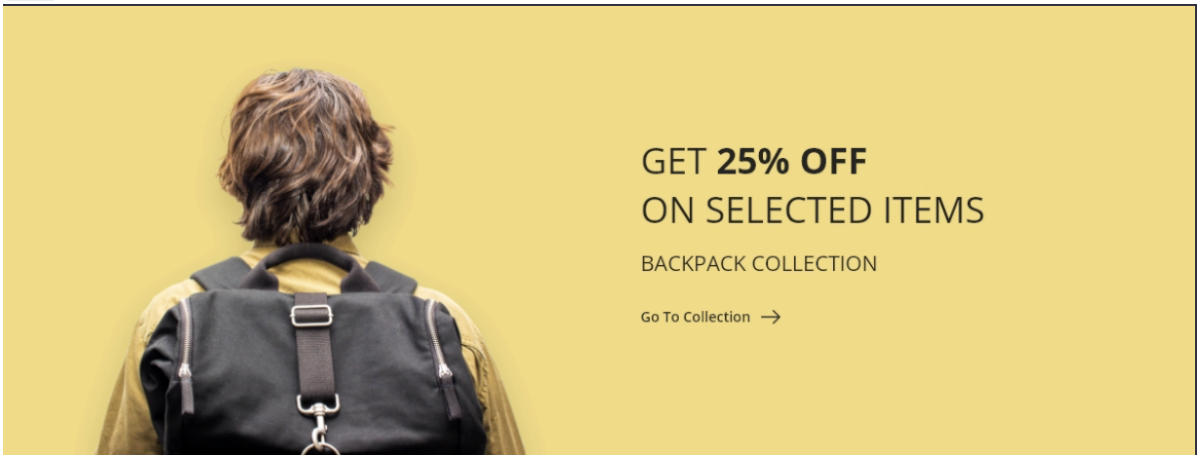


promotionBanner

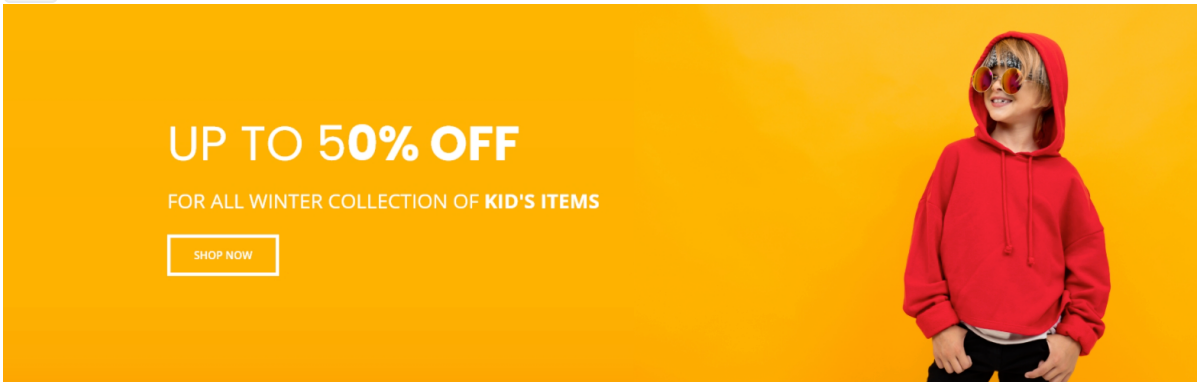
[1]



[2]

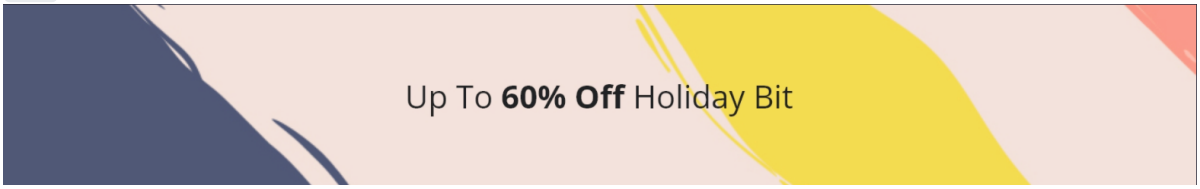


[3]

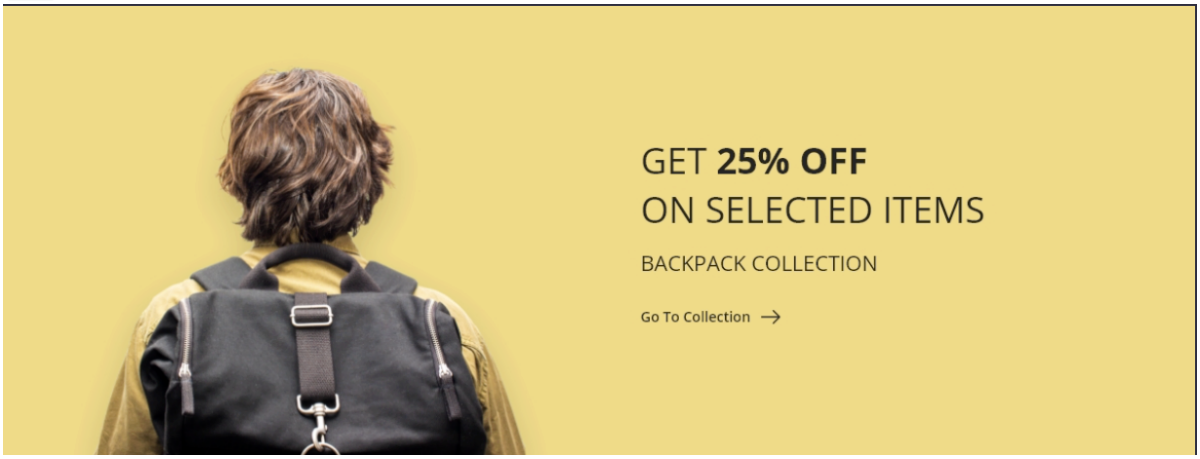


promotionBannerTwo

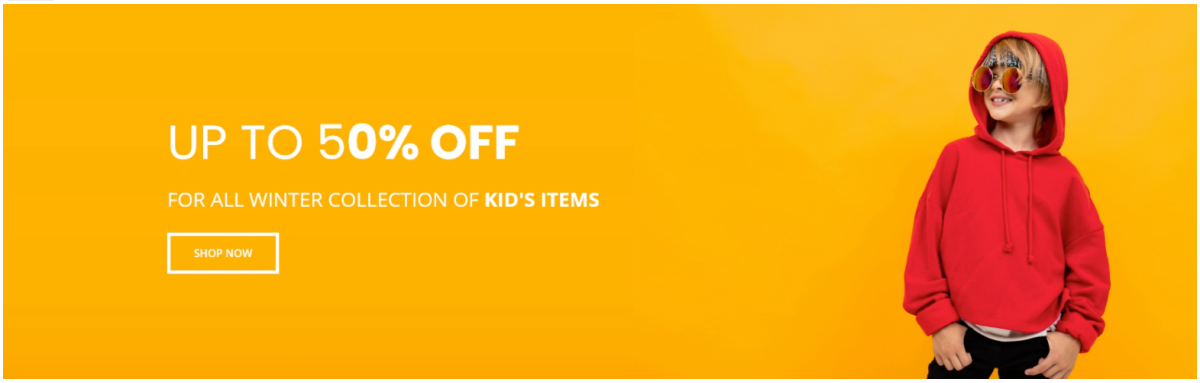
[1]



[2]

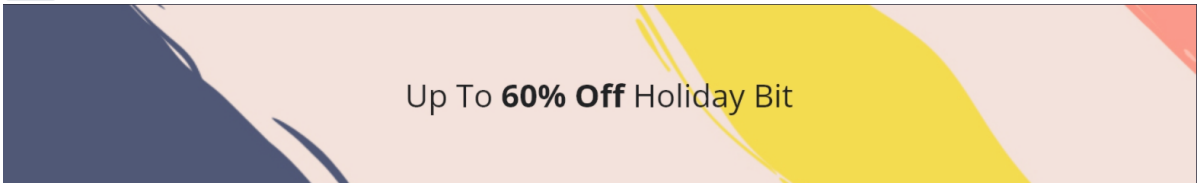


[3]

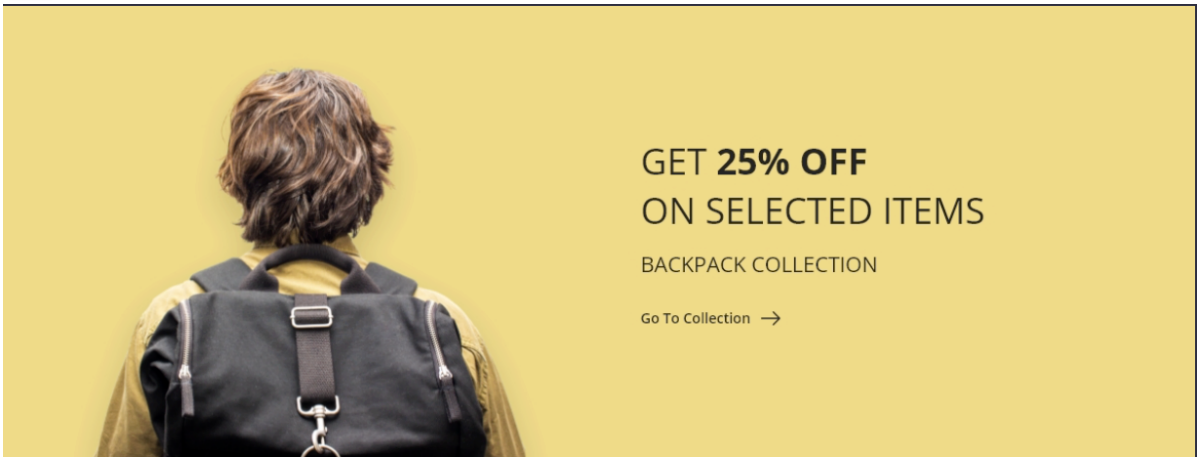


bannerDataOne

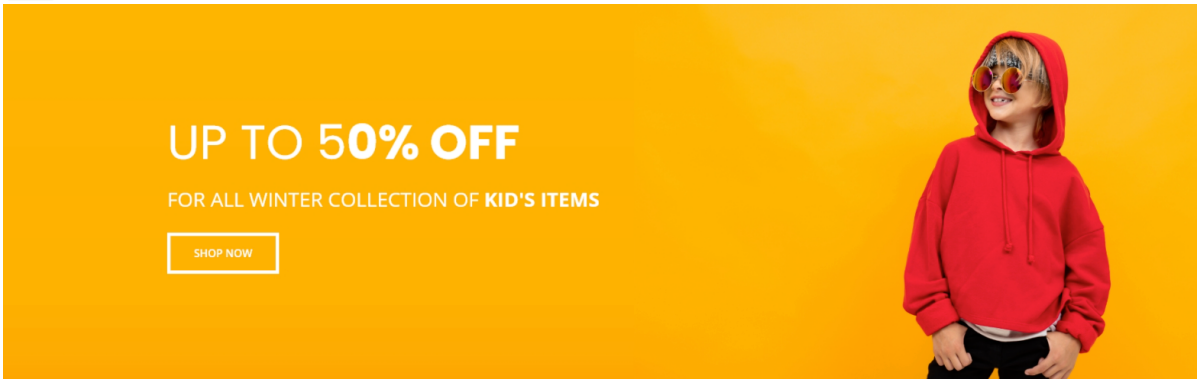
[1]



[2]

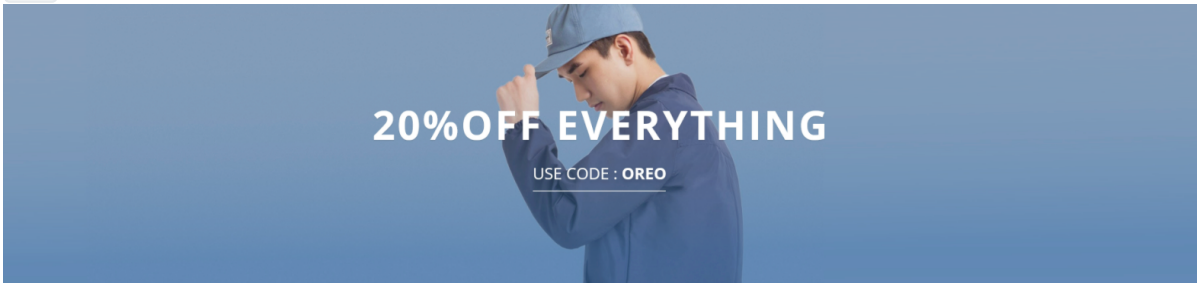


[3]



bannerDataTwo

[1]



[2]

[3]

homeThreeProductsBanner

[1]

[2]

homeFourBanner

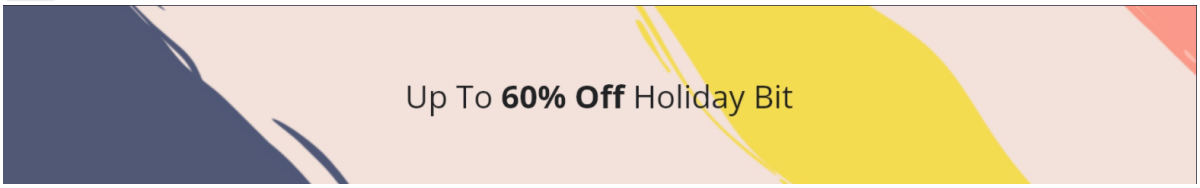
[1]

[2]

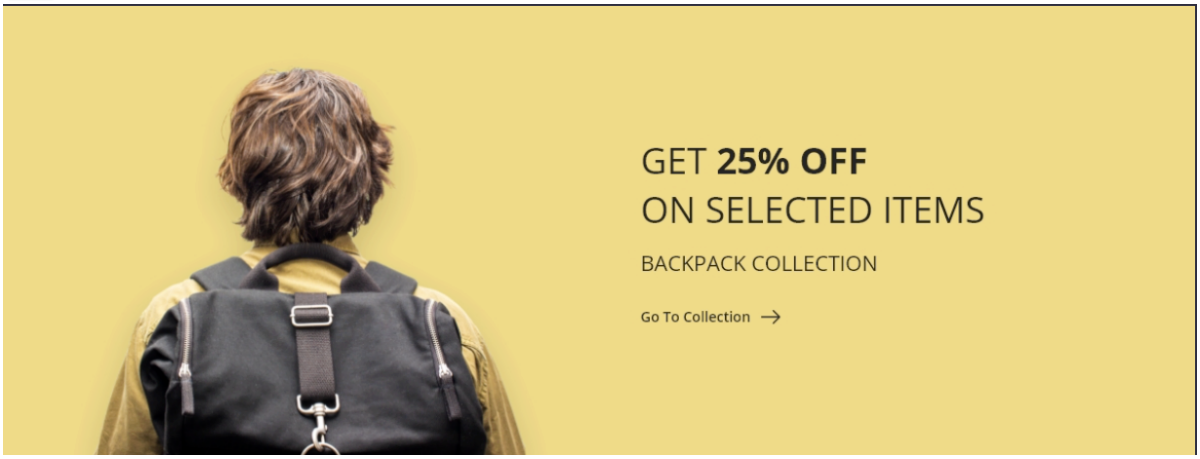
[3]

homeFourGridBanners

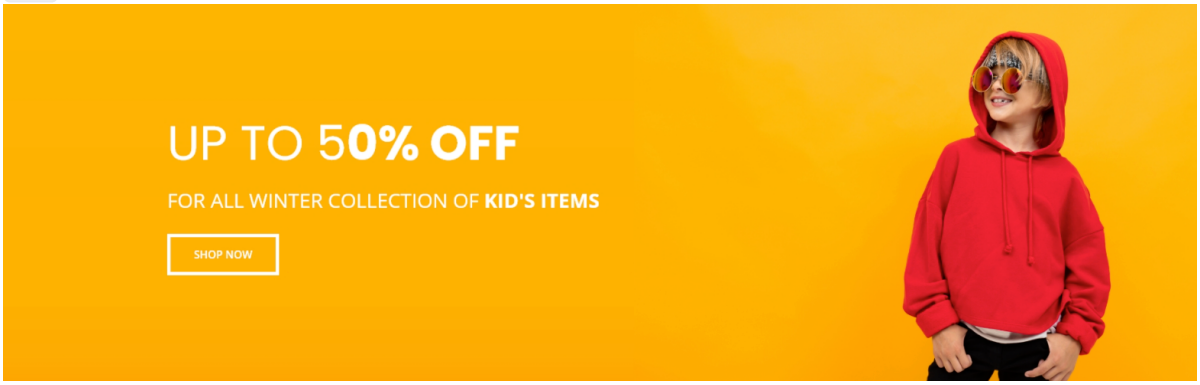
[1]



[2]

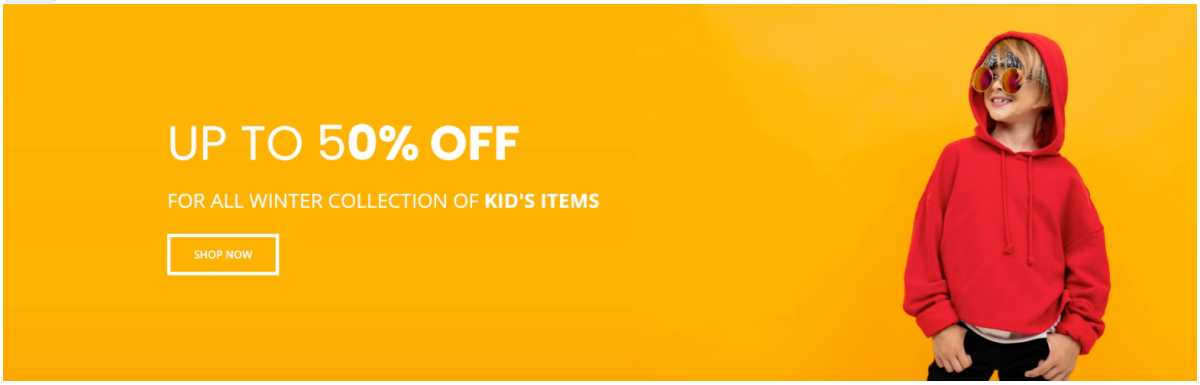


[3]



homeFiveBanner

`[1]`



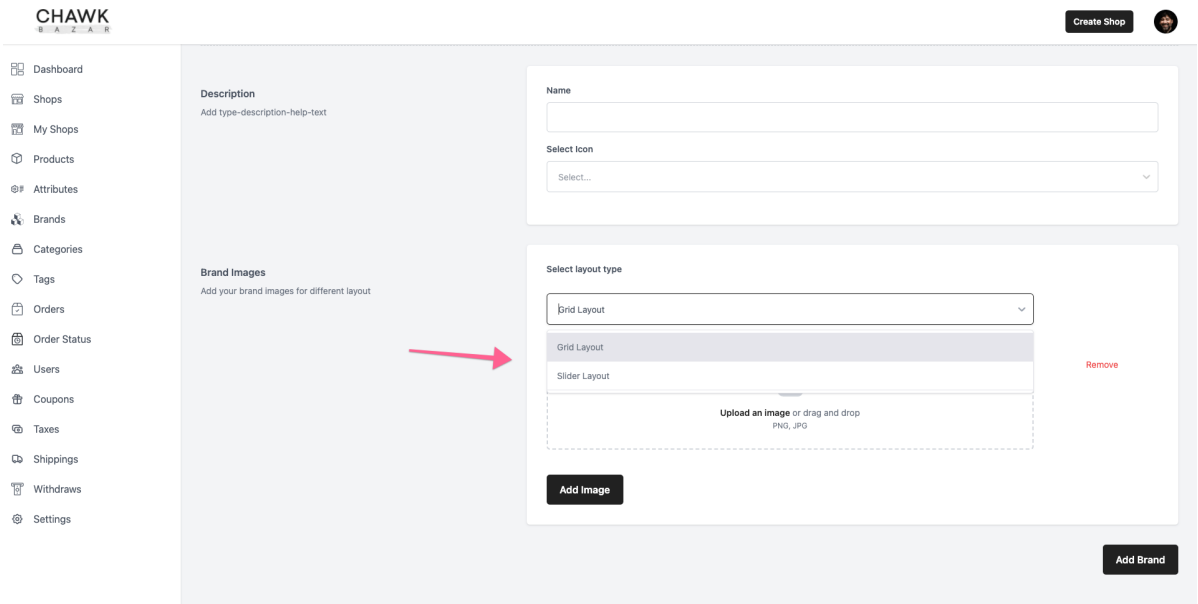
Brand

We have multiple types of brand layouts.

1. Slider Layout -> For Minimal Demo
2. Grid Layout -> For Others Demo

For slider layout, you've to upload a single image, and for grid layout, you've to upload two images; one is a cover, and another is a logo.

You'll get a layout option when you create a brand,



If you use a minimal home page, then add a slider layout; otherwise, use Grid Layout.

Utilities

In this template, We have used some custom helper functions which is located in

> [your-frontend-project] = admin [`rest`] or shop

```
cd [your-frontend-project]/src/utils/
```

You can use or customize these helper fuctions according to your needs.

Backend Integration

API Integration:

For API Integration:

> *`[your-frontend-project]`* = *`admin`* *`[rest]`* *or* *`shop`*

We have used env variables using `.env` file format. You have

to put your API url within this file.

For example:

Put that url in the `shop/.env` and `admin/rest/.env`

```
NEXT_PUBLIC_REST_API_ENDPOINT= '{put_your_api_url_here}'
```

And run your frontend project

Data Fetching

> `[your-frontend-project]` = `admin` `[rest]` or `shop`

We have used React Query and Axios which is defined in

```
[your-frontend-rest-project]/src/data/
```

For details documentation:

- React-Query
  You can check and customize it as your need.

Uses Example:

- Query

```
import { useProductsQuery } from "@data/product/use-products.query";

const {
  isFetching: loading,
  isFetchingNextPage: loadingMore,
  fetchNextPage,
  hasNextPage,
  isError,
  data,
  error,
} = useProductsQuery({
  type: "grocery",
  text: "",
  category: "grocery",
```

```
});
```

- Mutation

```
import { useCreateProductMutation } from "@data/product/product-create.mutation";

const {
  mutate: createProduct,
  isLoading: creating,
} = useCreateProductMutation();

function handleSubmit(inputVaues) {
  createProduct(
    {
      ...inputValues,
    },
    {
      onError: (error: any) => {
        Object.keys(error?.response?.data).forEach((field: any) => {
          setError(field, {
            type: "manual",
            message: error?.response?.data[field][0],
          });
        });
      },
    }
  );
}
```

Deployment

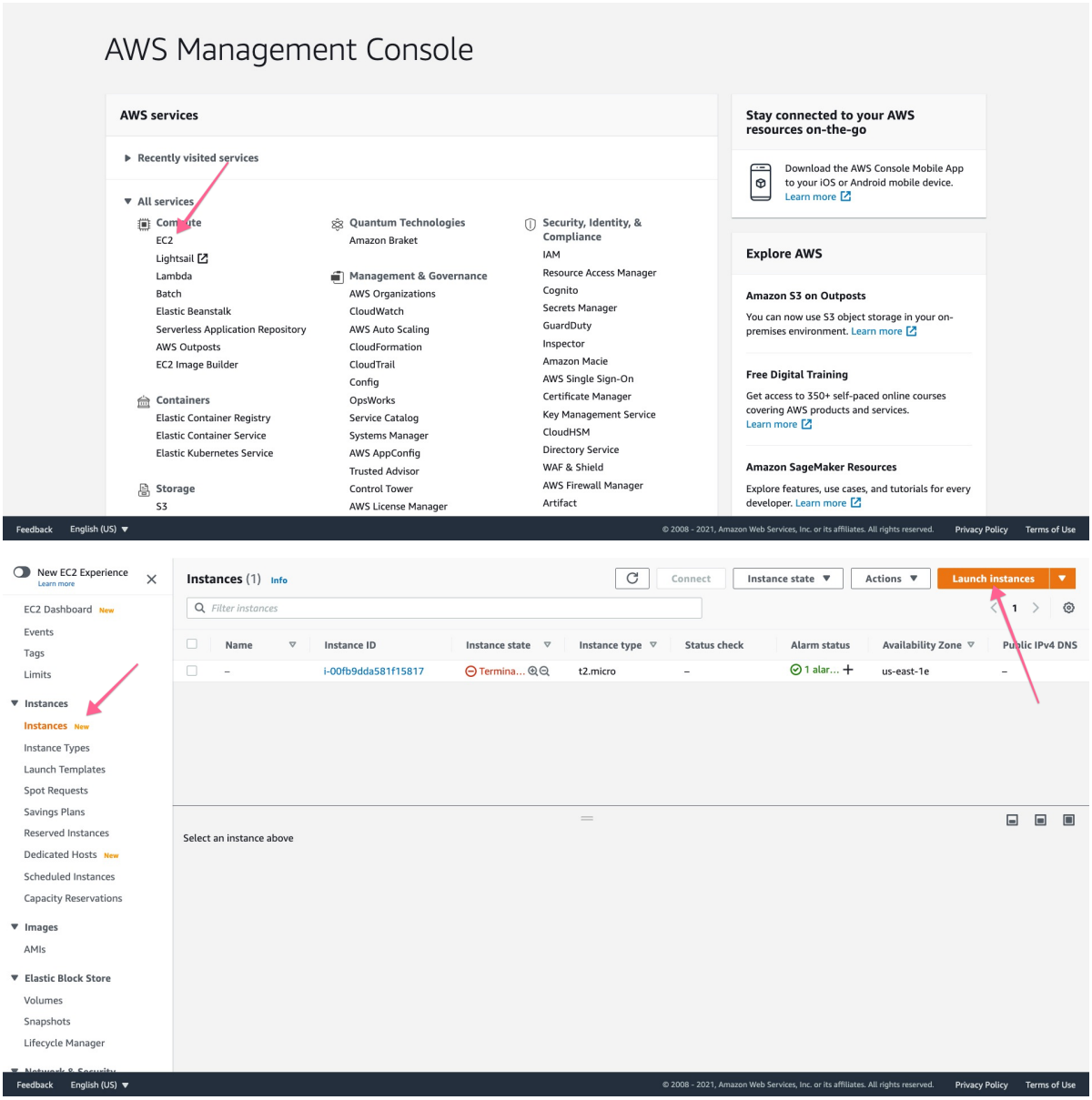
AWS (Amazon Web Service)

> If you want to use all the scripts (`shop`, `admin`, `api`) on the same server as this tutorial, then we recommend creating a blank ubuntu-based server with at least 2+ CPU cores and 2GB+ memory.

How to create ec2 server?

In this AWS tutorial, we're going to create an ec2 server. To do that at first, login to your AWS account and then click,

```
ec2 -> Instance -> Launch Instance
```



Then select a ubuntu 20.04 server

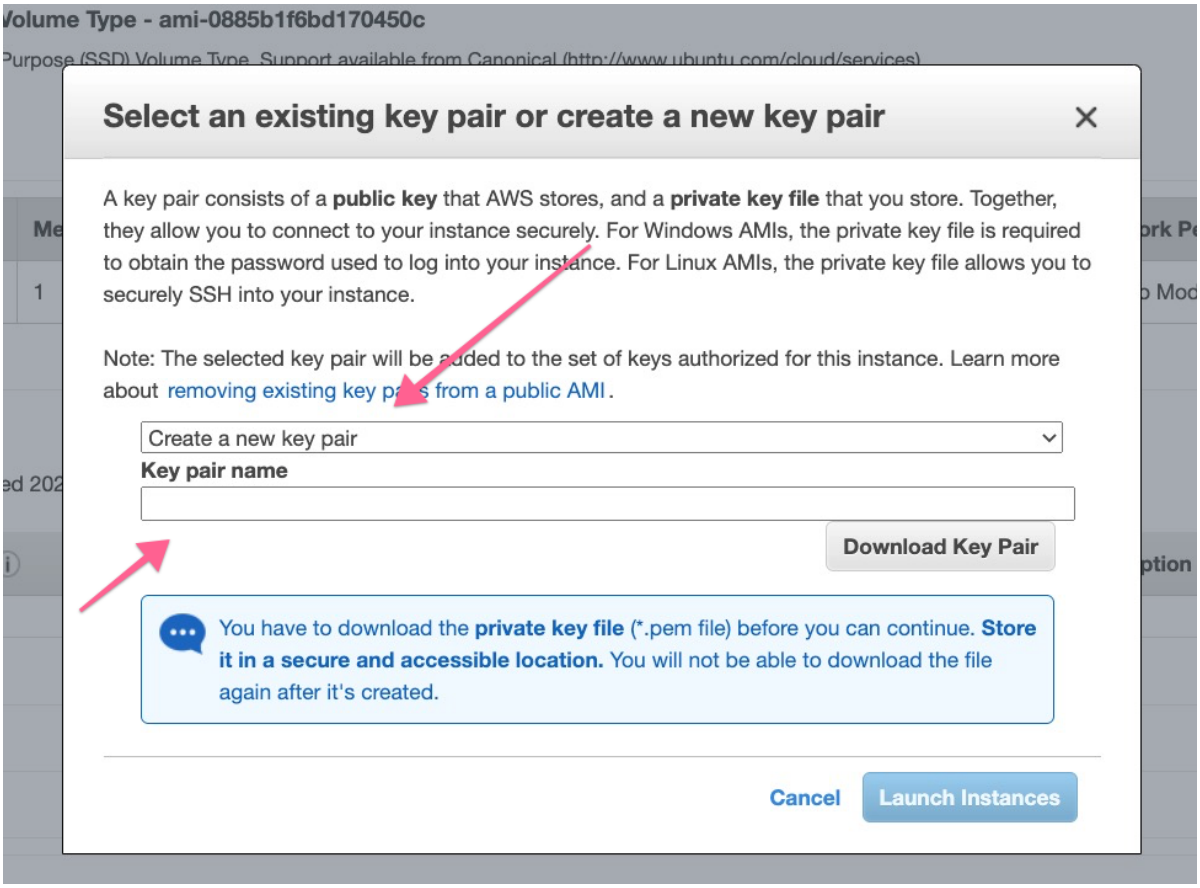
After that, click `Next` -> `Next` -> `Next` -> `Next`

And on security pages, add a rule for `HTTP`, `HTTPS` and `SSH`,

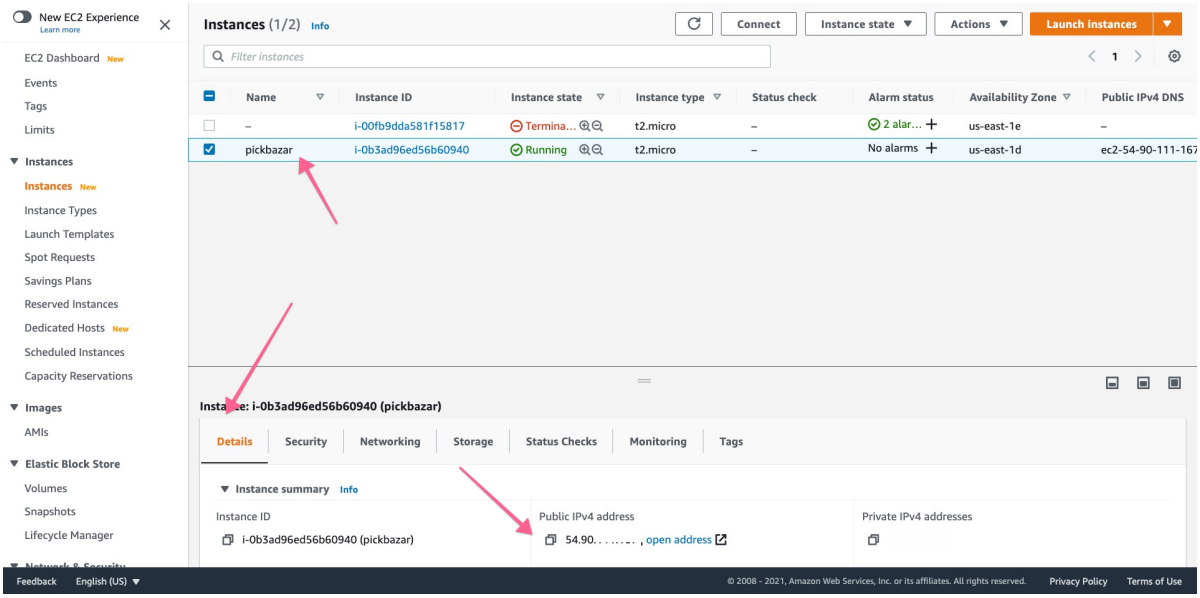
> Our automation scripts setup `HTTPS` in your domain so you should open `HTTPS`

After review, click `Launch`, and you'll get and popup for KeyPair, which will be required to login to the server using ssh.

If you already have a previous KeyPair, you can use that; otherwise, you can create a new one. After completing that, make sure you download that KeyPair.

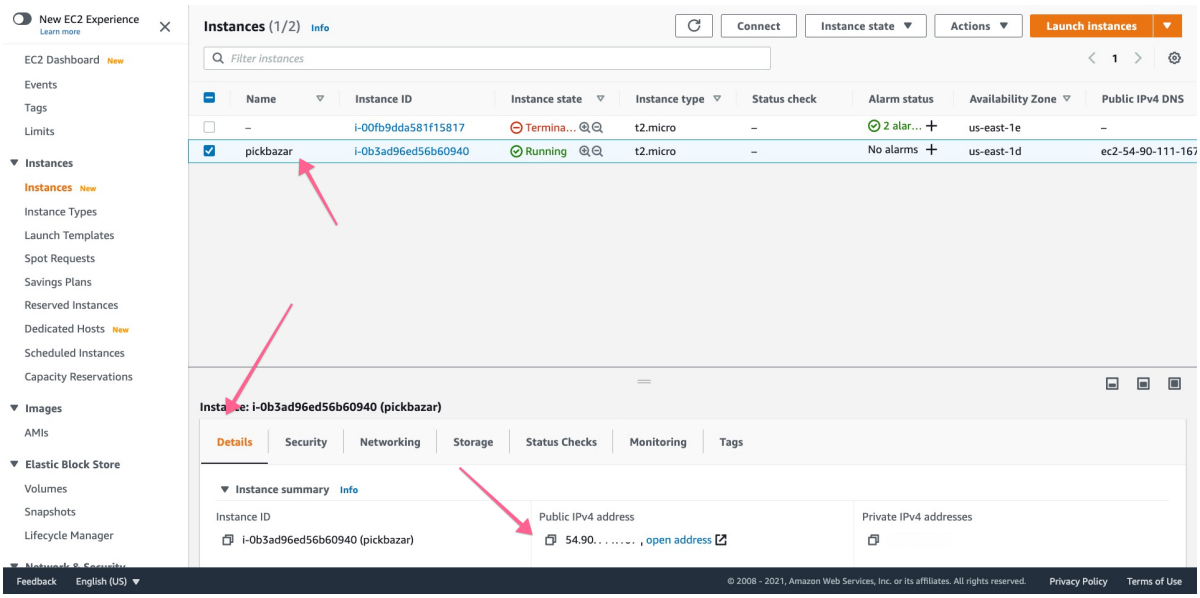


After launching the instance, you'll get the server IP, which will be required to login into ssh.



Domain Setup

Now copy the server IP and connect it with your domain name.



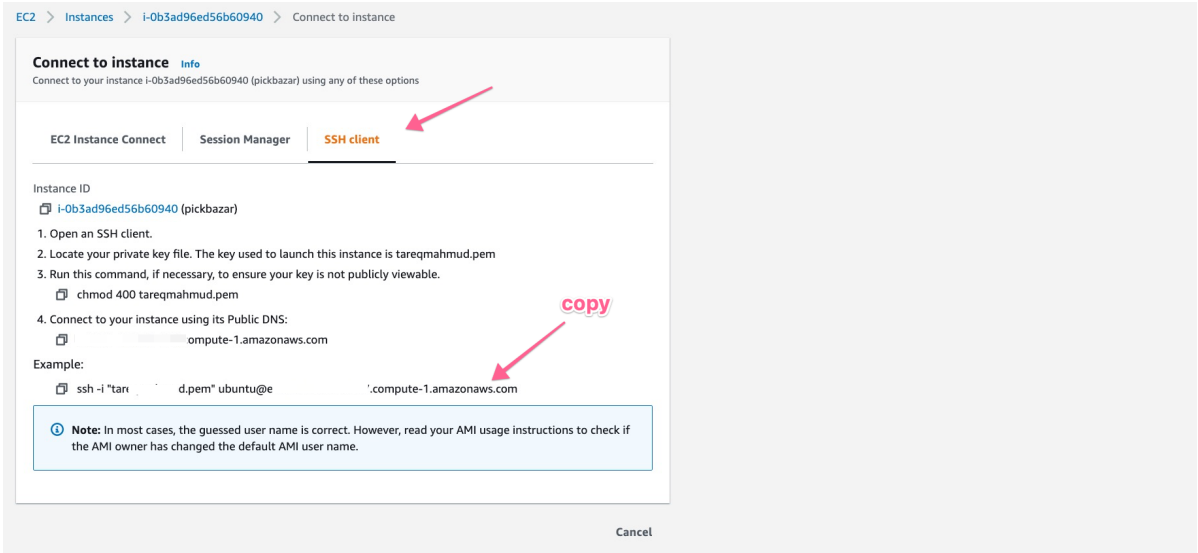
> *Please contact your domain provider for detailed explanation of how to do that.*

Login to Server

At first, login to your `AWS` server using ssh. to do that, go to the folder from the terminal where `KeyPair` is downloaded.

then click `Connect`

From the `Connect` dashboard, go to `SSH Client` and copy the example line and paste it to your terminal.



With this command, you will successfully connect to your server throw ssh.

Change permission .pem

You've to change the permission downloaded `.pem` file to `400` to access the server. To do that, at first go to the location where .pem store then run,

```
chmod 400 chawkbazar.pem
```

Change the `chawkbazar.pem` filename if you use a different name during generate the key.

> Now go to the `VPS Server` section for deploy the `Chawkbazar Laravel`

Virtual Private Server (Automated Script)

> *If you want to deploy the app using automated script then* follow this

With this tutorial, you can install ChawkBazar to any type of blank or empty ubuntu server. For example, `Digital Ocean Droplets, Amazon Lightsail, AWS, Google Cloud Virtual Private Server, Azure Ubuntu Virtual Private Server`, etc.

> If you want to use all the scripts (`shop`, `admin`, `api`) on the same server as this tutorial, then we recommend creating a blank ubuntu-based (`v20.0.4 lts`) server with at least 2+ CPU cores and 2GB+ memory.

> Please connect your `domain` with `server`. We don't recommend/support deployment the project via `IP`.

> Please follow this video with the documentation, and it'll make the installation process relatively easy.

Prerequisite

> This automated script is for the *nix system. So if you are using mac or Linux, then you're good to go. But if you are using windows, then install WSL on your computer and use this script using WSL or follow this manual installation

Before starting, the procedure ensures that NodeJS 16 (the latest) is installed on your computer.

```
npm i -g yarn zx
```

Now you can follow the script installation procedure,

At first login your server from terminal

```
ssh SERVER_USERNAME@SERVERIP
```

> Make sure that you are logged in your server then follow the next step and run suggested command.

Upload api and deployment project to Virtual Server form your PC - RUN on Local PC

To upload the zipped `chawkbazar-api` and `deployment` files to server you need to run the below command form your chawkbazar project root

> while running below command you will asked for enter your server `username` and `ip address` by entering and a successful connection you will also asked for enter your `chawkbazar-api.zip` and `deployment.zip`
> files path and the path will be look like `/home/your_project_folder_path/chawkbazar-laravel/chawkbazar-api.zip` for chawkbazar-api.zip file so forth for `deployment.zip`

```
bash deployment/deployment.sh
```

Then login your server using `ssh` and,

Server Environment setup script - RUN on Virtual Server

```
bash /var/www/chawkbazar-laravel/deployment/nodesetup.sh
```

Nginx Setup And Settings - RUN on Virtual Server

```
zx /var/www/chawkbazar-laravel/deployment/setenv.mjs
```

Backend build - RUN on Virtual Server

```
sudo zx /var/www/chawkbazar-laravel/deployment/backendbuildscript.mjs
```

Frontend build script - RUN on Local PC

Run the below command from your chawkbazar project root

```
zx deployment/frontendbuildscript.mjs
```

Frontend run script - RUN on Virtual Server

```
zx /var/www/chawkbazar-laravel/deployment/frontendrunscript.mjs
```

Vercel

vercel.com

If you want to host the template in vercel.com then follow the below command

API

It's not possible to host the API to vercel. Vercel doesn't support laravel API deployment. So you've to host the API on a separate server. We suggest you create a VPS server and host the API there. For more details, follow this VPS Deployment docs.

After the host, you'll get your API URL.

Frontend

Now for frontend add API URL and other necessary con
fig details to,

```
shop -> vercel.json
```

```
admin -> rest -> vercel.json
```

after that, install vercel-cli on your computer using this command,

```
npm i -g vercel-cli
```

After that, log in to vercel using this command,

```
vercel login
```

Then go to the `shop` directory and use this command to deploy,

```
vercel
```

Similarly, go to the `admin -> rest` directory and use this command to deploy,

```
vercel
```

For more details,

Please follow [nextjs deployment docs:](#)

Virtual Private Server (Manual)

With this tutorial, you can install ChawkBazar to any type of blank or empty ubuntu server. For example, `Digital Ocean Droplets, Amazon Lightsail, AWS, Google Cloud Virtual Private Server, Azure Ubuntu Virtual Private Server`, etc.

> If you want to use all the scripts (`shop`, `admin`, `api`) on the same server as this tutorial, then we recommend creating a blank ubuntu-based server with at least 2+ CPU cores and 2GB+ memory.

> Please follow this video with the documentation, and it'll make the installation process relatively easy.

https://www.youtube.com/watch?v=cESkbAmmDPc

Access Server

At first login your server using `SSH` and `Terminal`

Install NodeJS & Required Application

Install NodeJS

At first, we've to install NodeJS and npm to run the chawkbazar app. To install NodeJS and npm, run this command on your terminal,

```
sudo apt-get update
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Install Yarn

Chawkbazar is highly dependent on `yarn`, it would be best to handle all the script parts using `yarn`. So to install yarn, use this command,

```
sudo npm i -g yarn
```

If you face any permission issue, then please check this official doc to resolve that,

Npm Permission Issue

Install Zip & Unzip

```
sudo apt install zip unzip
```

Install PM2

Now we will install `PM2`, which is a process manager for Node.js applications. `PM2` provides an easy way to manage and daemonize applications (run them in the background as a service). To install `PM2` use this command,

```
sudo npm install -g pm2
```

After restarting the server or if the server crash, then pm2 will halt the process. To prevent that, we'll add pm2 as a `startup` process to run automatically after restart the server.

```
pm2 startup systemd
```

Setup Server

Introduction

Nginx is one of the most popular web servers in the world. In this deployment tutorial, we're going to use Nginx to host our website. In this tutorial, we're going to use ubuntu 20.04 to host chawkbazar

Step 1 - Installing Nginx

After creating the server, make sure the apt library is up to date. To update the apt library, use this command,

```
sudo apt update
```

After the update apt, we're going to install Nginx. To do that, use this command

```
sudo apt install nginx
```

Step 2: Adjusting the Firewall

Before testing Nginx, the firewall software needs to be adjusted to allow access to the service. Nginx registers itself as a service with `ufw` upon installation, making it straightforward to allow Nginx access.

To check the `ufw` list, use this command,

```
sudo ufw app list
```

You will get a listing of an application list like this,

At first, add ssh to the firewall,

```
sudo ufw allow ssh
sudo ufw allow OpenSSH
```

After that, to enable Nginx on the firewall, use this command,

```
sudo ufw allow 'Nginx HTTP'
```

Now enable the firewall,

```
sudo ufw enable
sudo ufw default deny
```

You can verify the change by typing:

```
sudo ufw status
```

The output will be indicated which HTTP traffic is allowed:

```
Status: active

To                         Action      From
--                         ------      ----
Nginx HTTP                 ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Step 3 – Checking your Web Server

Now check the status of the Nginx web server by using this command,

```
systemctl status nginx
```

You'll get an output like this,

```
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2021-01-12 07:35:20 UTC; 7min ago
       Docs: man:nginx(8)
   Main PID: 2940 (nginx)
      Tasks: 2 (limit: 1164)
     Memory: 5.1M
     CGroup: /system.slice/nginx.service
             ├─2940 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─2941 nginx: worker process
```

Step 4 - Install MySQL

```
sudo apt install mysql-server
```

Step 5 - Install PHP

```
sudo apt install php-fpm php-mysql
sudo apt install php-fpm php-mysql php-mbstring php-xml php-bcmath php-simplexml
php-mbstring php7.4-gd php7.4-curl composer
```

Step 6 - Create MySQL Database & User

```
sudo mysql
CREATE DATABASE chawkbazar;

CREATE USER 'chawkbazar_user'@'%' IDENTIFIED WITH mysql_native_password BY
'chawkbazar1';

GRANT ALL ON chawkbazar.* TO 'chawkbazar_user'@'%';

FLUSH PRIVILEGES;
```

We use MySQL user name `chawkbazar_user` and MYSQL password `chawkbazar1`. Make sure you change at least `MySQL` password for security.

Step 7 - Change permission for the `www` folder

```
sudo chown -R $USER:$USER /var/www/
```

Step 8 - Upload API to Server

At first, use this command to create a directory on `/var/www/`

```
mkdir /var/www/chawkbazar
```

Then, go to your `local computer`

1. Extract the `chawkbazar` package that you download from `CodeCanyon.`
2. On that folder, you'll get another `folder` called `chawkbazar-laravel`.
3. On that folder, you'll get a folder called `chawkbazar-api`

Now upload this `chawkbazar-api` folder to the server `/var/www/chawkbazar/`

Step 9: Setting Up Server & Project

In this chapter, we'll set up our server and also will set up Reverse Proxy to host all of our sites from the same server.

At first, we'll disable the default configuration.

```
sudo rm /etc/nginx/sites-enabled/default
```

Step 10 - Create New Nginx for the domain

```
sudo touch /etc/nginx/sites-available/chawkbazar
sudo nano /etc/nginx/sites-available/chawkbazar
```

Add this Nginx config file to that edited file,

```
server {
    listen 80;
```

```
    server_name YOUR_DOMAIN.com;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.html index.htm index.php;

    charset utf-8;

    # For API
    location /api {
        alias /var/www/chawkbazar/chawkbazar-api/public;
        try_files $uri $uri/ @api;
            location ~ \.php$ {
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $request_filename;
            fastcgi_pass   unix:/run/php/php7.4-fpm.sock;
         }
    }

    location @api {
       rewrite /api/(.*)$ /api/index.php?/$1 last;
    }

    # For FrontEnd -> Rest
    location /{
         proxy_pass http://localhost:3003;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection 'upgrade';
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
    }

    location /admin{
         proxy_pass http://localhost:3002/admin;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection 'upgrade';
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
    }

    error_page 404 /index.php;

    location ~ \.php$ {
         fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
         fastcgi_index index.php;
         fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
         include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
         deny all;
```

```
    }
}
```

> Make sure you change `YOUR_DOMAIN.com` to your specific `domain name`

> You can change `api` path, if you want to change the the domain path for the laravel application

> You can change `admin` path, if you want to change the the domain path for the frontend admin

Save and close the file by typing `CTRL` and `X,` then `Y` and `ENTER` when you are finished.

Then enable the config

```
sudo ln -s /etc/nginx/sites-available/chawkbazar /etc/nginx/sites-enabled/
```

Make sure you didn't introduce any syntax errors by typing:

```
sudo nginx -t
```

Next, restart Nginx:

```
sudo systemctl restart nginx
```

Secure Server

Step 1: Secure Nginx with Let's Encrypt

```
sudo apt install certbot python3-certbot-nginx
sudo ufw status

sudo ufw allow 'Nginx Full'
sudo ufw delete allow 'Nginx HTTP'

sudo ufw status
sudo certbot --nginx -d YOUR_DOMAIN
```

> After this command, you'll get several command prompt. Make sure you take the necessary steps and provide information on that command prompt.

Install API

Step 1: Build and Run `api`

At first, go to the `chawkbazar-api` folder, then copy `.env.example` to `.env,`

```
cd /var/www/chawkbazar/chawkbazar-api
cp .env.example .env
```

Edit `.env`

```
nano .env
```

And add `MySQL,` `stripe,` `mail` or others configuration.

Also, add `https://YOUR_DOMAIN.COM/api` to `APP_URL`. Without this, the `upload` function will be broken.

```
APP_NAME=ChawkBazar
APP_ENV=production
APP_KEY=base64:dladdfdsfdfdsfssdf/sdfdsfer
APP_DEBUG=true
APP_URL=https://YOUR_DOMAIN.COM/backend
APP_SERVICE=chawkbazar.test
APP_NOTICE_DOMAIN=CHAWKBAZAR_

DUMMY_DATA_PATH=chawkbazar

LOG_CHANNEL=stack
LOG_LEVEL=debug

```

Then install all the packages and install `api`

```
composer install

php artisan key:generate

php artisan marvel:install
```

> You'll get several confirmations for migration, dummy data, and admin account. Make sure you check the confirmation step and take the necessary actions based on your requirement.

Enable `laravel storage,`

```
php artisan storage:link
```

Then give proper `permission` for laravel folder,

```
sudo chown -R www-data:www-data storage
sudo chown -R www-data:www-data bootstrap/cache
```

Now, when you go to the `YOUR_DOMAIN/api` you'll get a `welcome` page like this

FrontEnd Project Build

> Typescript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move the folder to the server then serve the project. We'll do the second method in this tutorial.

> We'll suggest you build the frontend part on your computer and then upload the build file to the server.

Go to your `chawkbazar-laravel` folder from your `local computer`.

Step 1 - Config Next Admin App For /admin Sub Directory

Edit `admin/rest/next.config.js`,

add `basePath` for '/admin'

```
const withPWA = require("next-pwa");
const runtimeCaching = require("next-pwa/cache");
const { i18n } = require("./next-i18next.config");

module.exports = withPWA({
  basePath: "/admin",
  i18n,
  pwa: {
    disable: process.env.NODE_ENV === "development",
    dest: "public",
    runtimeCaching,
  },

  images: {
    domains: [
      "via.placeholder.com",
      "res.cloudinary.com",
      "s3.amazonaws.com",
      "18.141.64.26",
      "127.0.0.1",
      "localhost",
      "picsum.photos",
      "pickbazar-sail.test",
      "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
      "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
      "lh3.googleusercontent.com",
    ],
  },
```

Step 2 - Install & Build

go to your `chawkbazar-laravel` folder again

To install all the npm packages run this command,

```
yarn
```

Step 3 - Build the project

At first, we've to copy the sample `.env.template` to production `.env` for the shop and admin first.

Shop

Go to `shop` and copy `.env.template` to `.env`

Then edit `.env` and add your `API` url to that file

```
NEXT_PUBLIC_REST_API_ENDPOINT="https://YOUR_DOMAIN/api"
```

Also, update all the necessary configurations like `OTP,` `Mailchimp Token,` `Instagram Token,` `Google MAP API,` etc.

Admin

After that, Go to `admin -> rest` and copy `.env.template` to `.env`

Then edit `.env` and add your `API` url to that file

```
NEXT_PUBLIC_REST_API_ENDPOINT="https://YOUR_DOMAIN/api"
```

Also, update all the necessary configurations like `Google MAP API,` etc.

Image Config

Then open `shop -> next.config.js` and `admin -> rest -> next.config.js`

and add your domain to `images` object

```
const withPWA = require("next-pwa");
const runtimeCaching = require("next-pwa/cache");
const { i18n } = require("./next-i18next.config");

module.exports = withPWA({
  i18n,
  pwa: {
    disable: process.env.NODE_ENV === "development",
    dest: "public",
    runtimeCaching,
  },

  images: {
    domains: [
      "YOUR_API_DOMAIN",
      "via.placeholder.com",
      "res.cloudinary.com",
      "s3.amazonaws.com",
      "18.141.64.26",
      "127.0.0.1",
      "localhost",
      "picsum.photos",
      "pickbazar-sail.test",
      "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
      "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
      "lh3.googleusercontent.com",
    ],
  },
```

> If your API is hosted on a subdomain, then add that subdomain with root domain on `next.config.js`

Build Project

Now go to the `chawkbazar-laravel` folder,

run this for build `admin`

```
yarn build:admin-rest
```

And for the `shop`,

at first, run the shop in one terminal

```
yarn dev:shop-rest
```

and then build the shop,

```
yarn build:shop-rest
```

Now zip `admin`, `shop`, `package.json`, `babel.config.js` and `yarn.lock` files and upload them to the server `/var/www/chawkbazar`

Now go to the server `/var/www/chawkbazar` using terminal

Then `unzip` that `zip` file.

Install FrontEnd And Run

Then install all the node packages,

```
yarn
```

Run frontend app

For `shop` app, use this command,

```
pm2 --name shop-rest start yarn -- run start:shop-rest
```

Then to run the `admin` app, use this command,

```
pm2 --name admin-rest start yarn -- run start:admin-rest
```

Now go to Now, go to your `YOUR_DOMAIN` to access the shop page and `YOUR_DOMAIN/admin` for the access admin section.

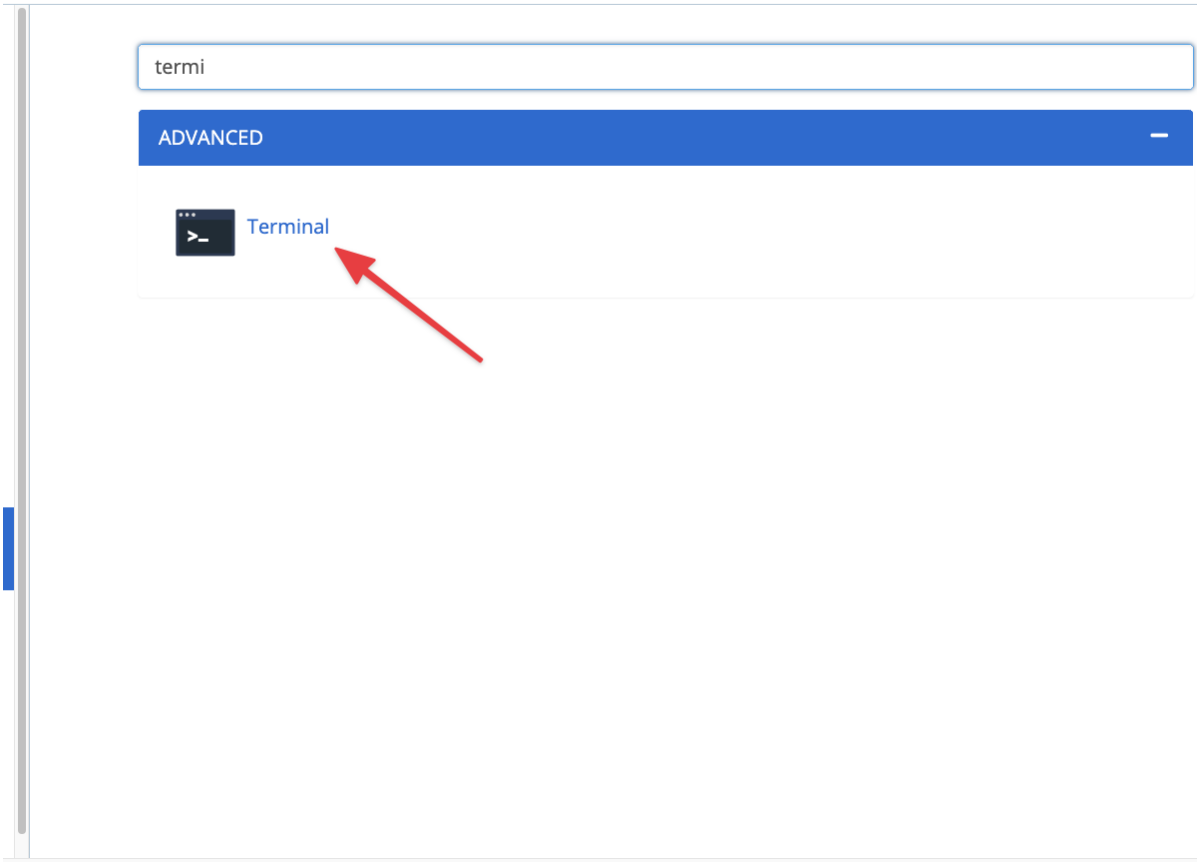
cPanel

> It's quite hard to debug any deployment issue on the cPanel or any managed server as the provider manages this type of server, and they've complete control of the server. And for that, We don't recommend Cpanel or any managed server for deployment. We suggest you use any VPS server where you have complete control of it. you can purchase any $5 – $10/mo server from amazon lightsail, ec2 or digitalocean or any ubuntu server

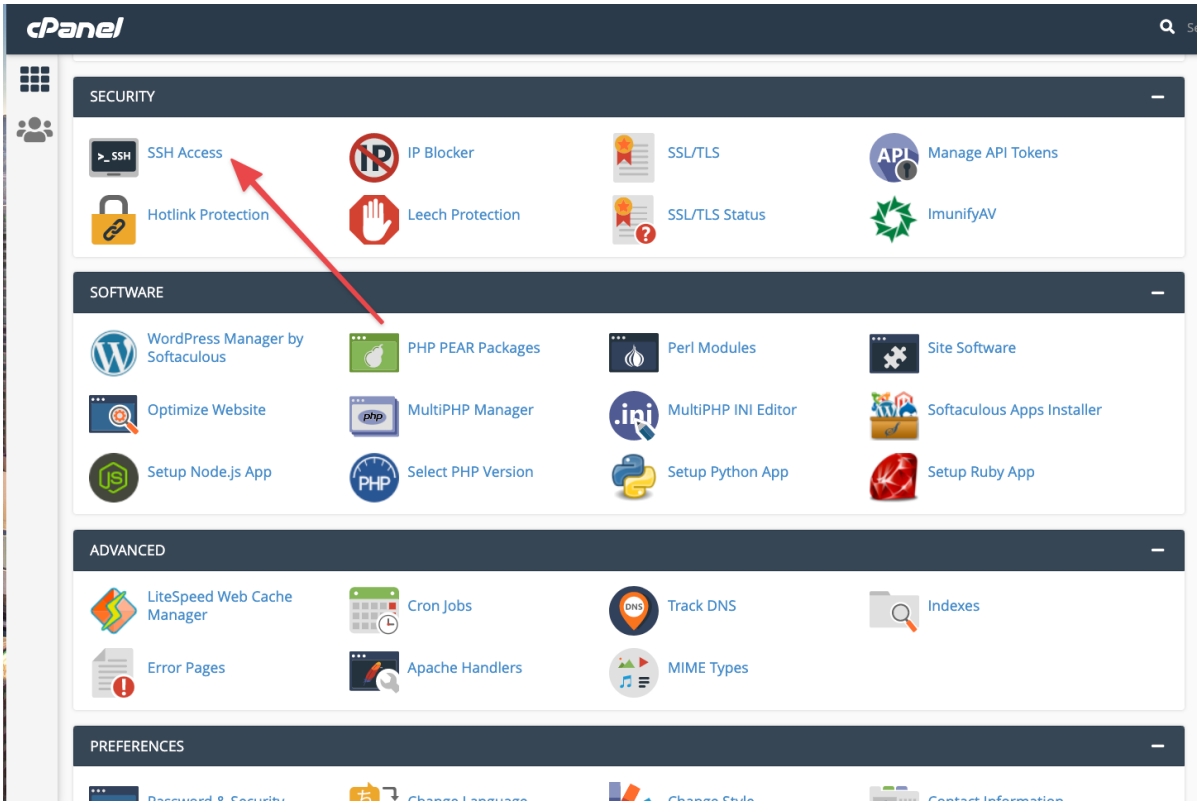
> If you still decide to proceed with panel based server, our support team won't be able to help you. We have put some resources for Cpanel in this documentation section to help our users to get started but other than that, we don't have much to offer with Cpanel or any Managed Server.

Access Server

To install the API, access the server using the cPanel terminal first,



If you don't find the terminal, then login to your local computer terminal or putty for Windows using SSH.



After enabling the ssh login to your server using ssh,

> If you dont't see any option, then contact your hosting provider as cPanel control by hosting provider.

After logging in, Check if the composer is already installed or not using this command,

```
composer -v
```

If composer is not installed then, install `composer` to your server.

Check this YouTube Video for install `composer` on your server,

After that, check the PHP version using,

```
php -v
```

make sure it's `8.1`

Create Subdomains

Now create two subdomains, for example,

```
-> your_domain.com -> host frontend store.
-> api.your_domain.com -> host laravel API.
-> admin.your_domain.com -> host admin dashboard.
```

Or if you want to host all the script on subdomains, then create subdomains like this,

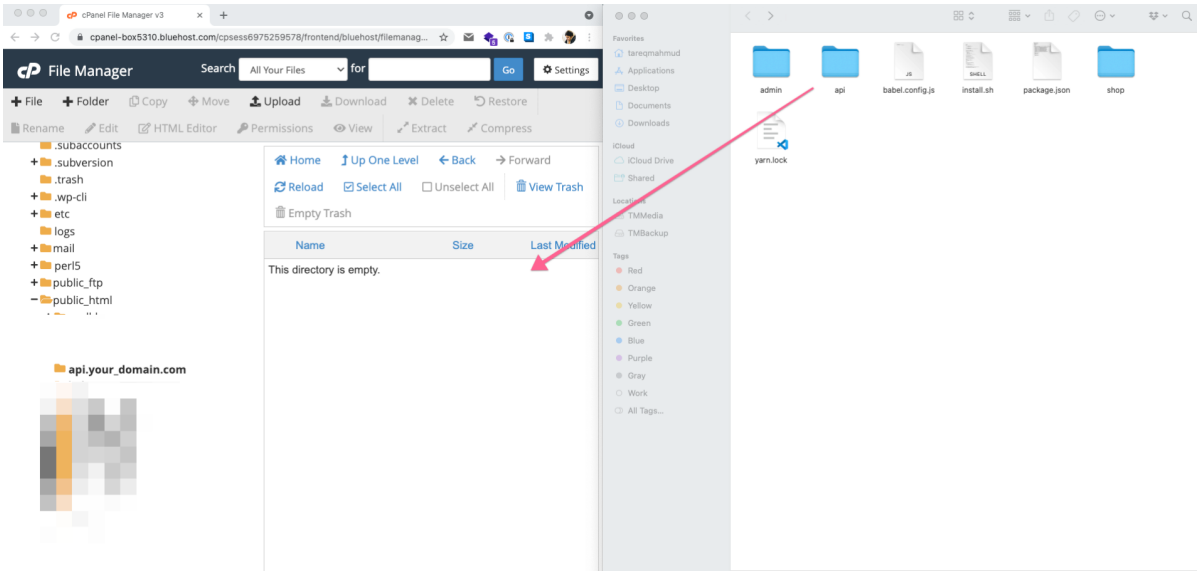
```
-> store.your_domain.com -> host frontend store.
-> api.your_domain.com -> host laravel API.
-> admin.your_domain.com -> host admin dashboard.
```

> After creating domain/subdomains, make sure all the domain/subdomains are HTTPS enabled. Please contact your hosting provider to enable this, as most hosting providers provide some sort of free SSL.

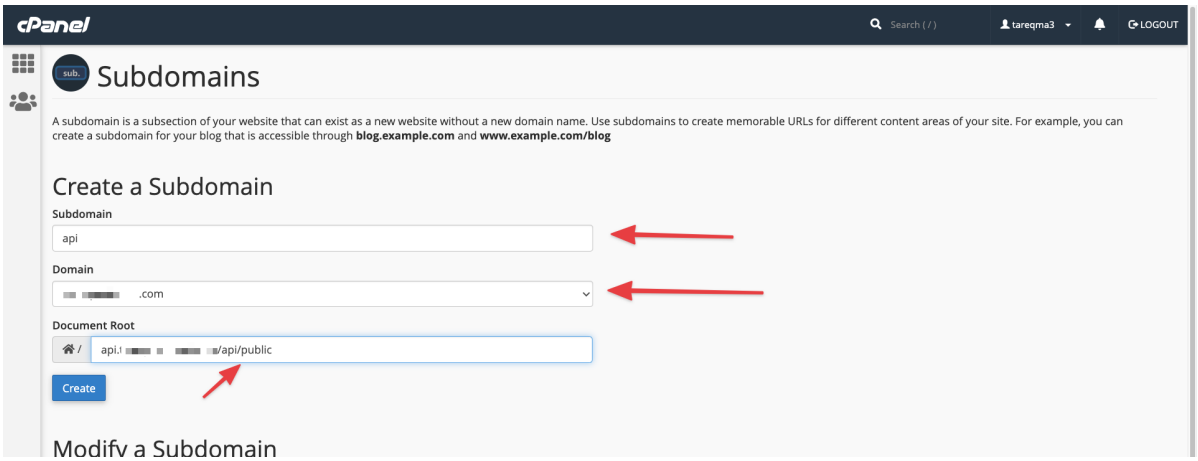
Install API

1. Extract the `chawkbazar` package that you download from `CodeCanyon.`
2. On that folder, you'll get another `zip` called `chawkbazar-laravel.zip`.
3. Now extract this `chawkbazar-laravel.zip` file.
4. On that file, you'll get a folder called `api`

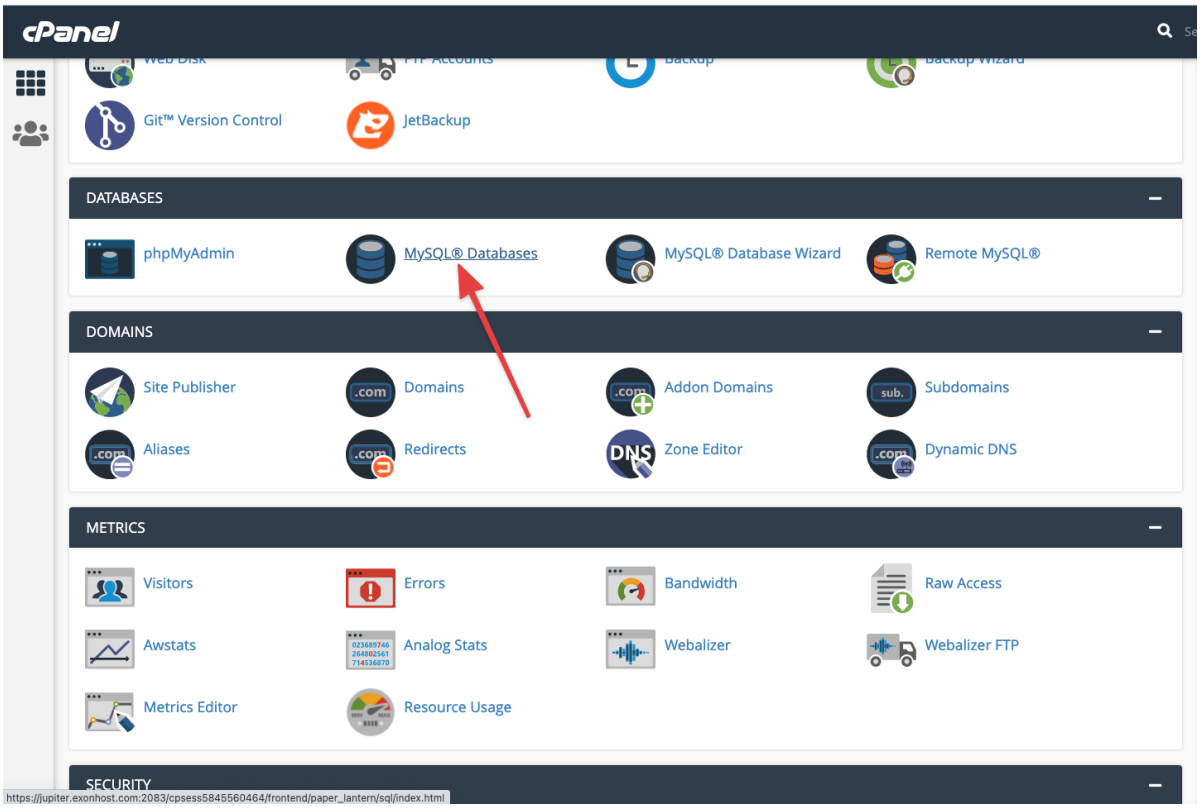
Now upload this `api` folder to the `api.your_domain.com` folder in your server



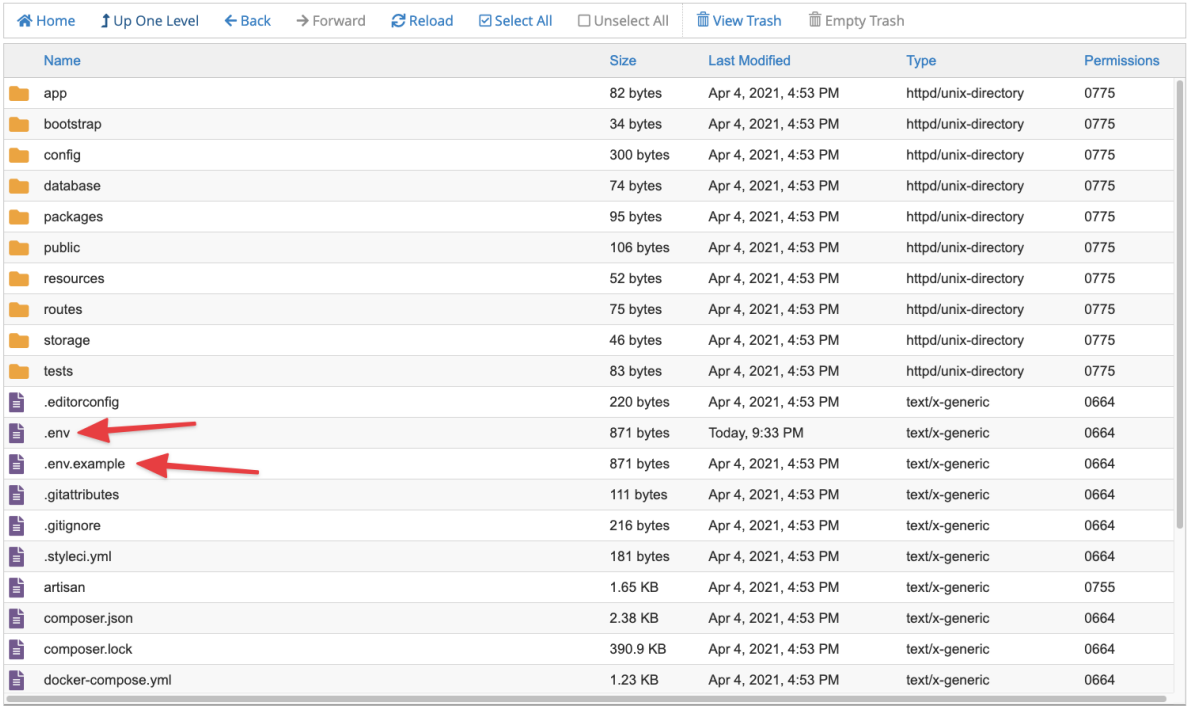
> Make sure your `api.your_domain.com` subdomain `Document Root` points to that `api/public` folder.



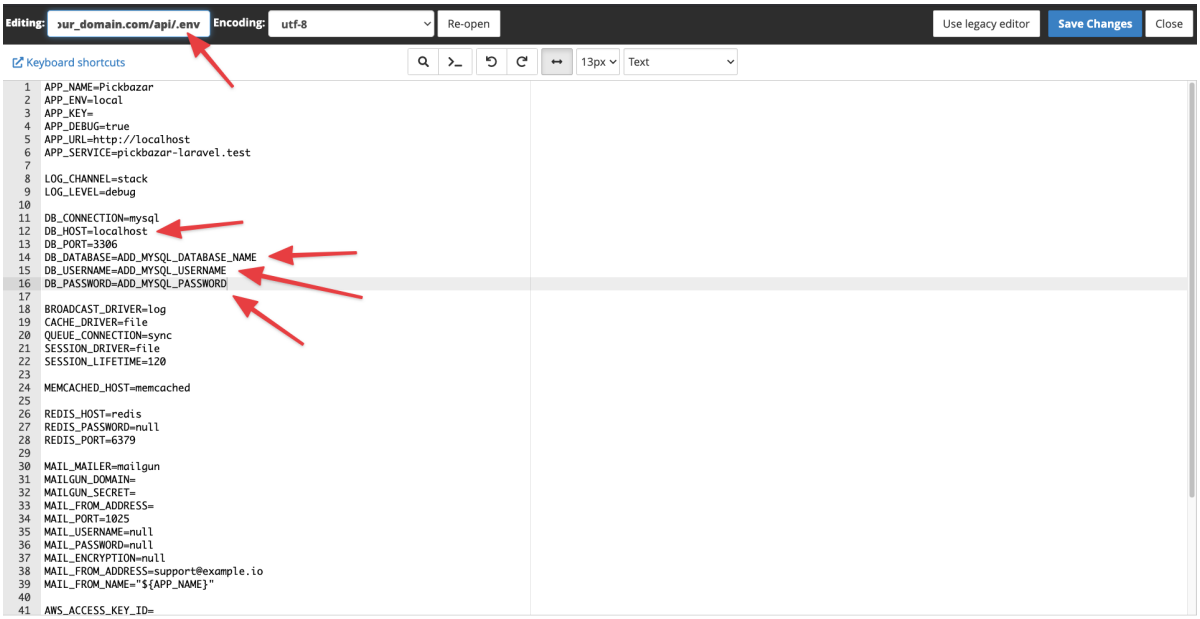
Now create a MySQL database and user from MySQL wizard



After creating the MySQL database, go to your `api` folder from your cPanel file manager and copy `.env.example` to `.env`.



After the copy, edit `.env` and add MySQL credentials,



Also, add `https://YOUR_DOMAIN.COM/api` to `APP_URL`. Without this, the `upload` function will be broken.

Then go to your `ssh terminal` again and,

go to `api` folder and run,

```
composer install
```

If `composer` installs all the packages successfully, then run this command on the `api` folder,

```
php artisan key:generate

php artisan marvel:install
```

> You'll get several confirmations for migration, dummy data, and admin account. Make sure you check the confirmation step and take the necessary actions based on your requirement.

After that, run this command to link storage,

```
php artisan storage:link
```

After install, go to your `api.your_domain_name.com,` and you'll get a webpage like this,

Install FrontEnd

Before proceeding next step, make sure you already create two subdomains like this,

```
-> your_domain.com -> host frontend store.
-> admin.your_domain.com -> host admin dashboard.
```

OR

```
-> store.your_domain.com -> host frontend store.
-> admin.your_domain.com -> host admin dashboard.
```

FrontEnd Project Build

Typescript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move the folder to the server then serve the project. We'll do the second method in this tutorial.

> We'll suggest you build the frontend part on your computer and then upload the build file to the server.

step 1 - Build Custom Server

go to your `chawkbazar-laravel` folder

shop

Create custom server for `shop`,

create a `server.js` file on,

```
shop/server.js
```

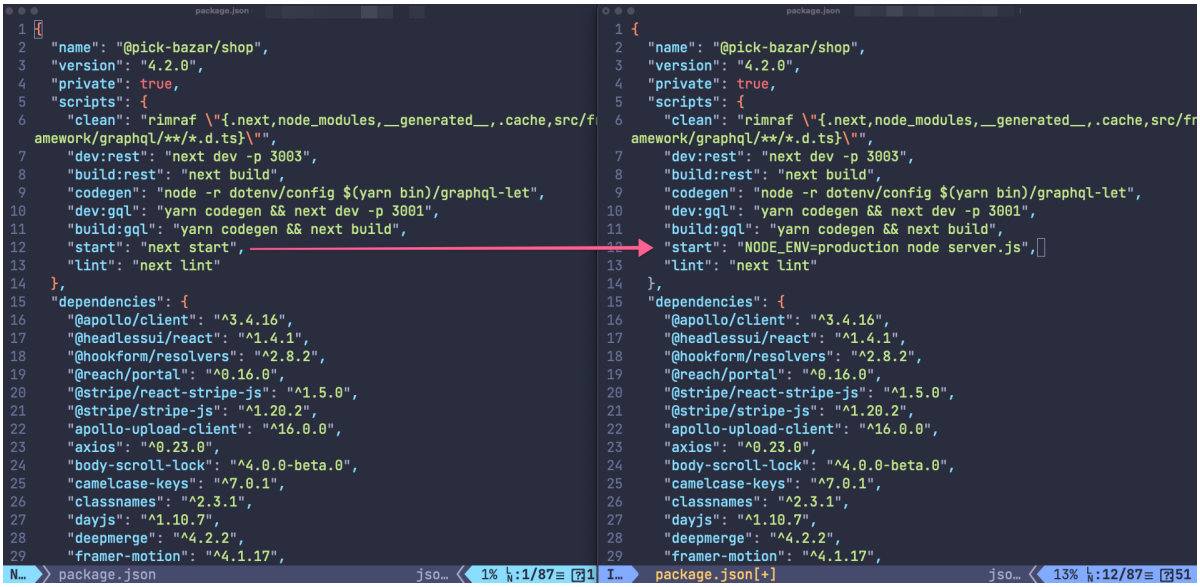
and paste this code,

```
// server.js
const { createServer } = require('http')
const { parse } = require('url')
const next = require('next')

const dev = process.env.NODE_ENV !== 'production'
const app = next({ dev })
const handle = app.getRequestHandler()

app.prepare().then(() => {
  createServer((req, res) => {
    // Be sure to pass `true` as the second argument to `url.parse`.
    // This tells it to parse the query portion of the URL.
    const parsedUrl = parse(req.url, true)
    const { pathname, query } = parsedUrl

    if (pathname === '/a') {
      app.render(req, res, '/a', query)
    } else if (pathname === '/b') {
      app.render(req, res, '/b', query)
    } else {
      handle(req, res, parsedUrl)
    }
  }).listen(3003, (err) => {
    if (err) throw err
    console.log('> Ready on http://localhost:3003')
  })
})
```

Now update package.json for `shop/package.json`,

and replace `start` script with this,

```
"start": "NODE_ENV=production node server.js"
```



admin rest

Similarly, create custom server for `admin rest`,

create another `server.js` file on,

```
admin/rest/server.js
```

and paste this code,

```
// server.js
const { createServer } = require('http')
const { parse } = require('url')
const next = require('next')

const dev = process.env.NODE_ENV !== 'production'
const app = next({ dev })
const handle = app.getRequestHandler()

app.prepare().then(() => {
  createServer((req, res) => {
    // Be sure to pass `true` as the second argument to `url.parse`.
    // This tells it to parse the query portion of the URL.
    const parsedUrl = parse(req.url, true)
    const { pathname, query } = parsedUrl

    if (pathname === '/a') {
      app.render(req, res, '/a', query)
    } else if (pathname === '/b') {
      app.render(req, res, '/b', query)
    } else {
      handle(req, res, parsedUrl)
    }
  }).listen(3002, (err) => {
```

```
    if (err) throw err
    console.log('> Ready on http://localhost:3002')
  })
})
```

Now update package.json for `admin/rest/package.json`,

and replace `start` script with this,

```
"start": "NODE_ENV=production node server.js"
```

Step 2 - Install & Build

go to your `chawkbazar-laravel` folder

To install all the npm packages run this command,

```
yarn
```

Step 3 - Build the project

At first, we've to copy the sample `.env.template` to production `.env` for the `shop` and `admin` first.

Go to,

```
cd shop
```

then use this command to copy,

```
cp .env.template .env
```

Now edit .env and add you `API` url to `.env`

```
nano .env
```

and use

```
NEXT_PUBLIC_REST_API_ENDPOINT=https://api.YOUR_DOMAIN.com/
```

After that, go to the `admin -> rest` folder,

```
cd ../admin/rest
```

then use this command to copy,

```
cp .env.template .env
```

```
nano .env
```

and use

```
NEXT_PUBLIC_REST_API_ENDPOINT=https://api.YOUR_DOMAIN.com/
```

Now go to the `chawkbazar-laravel` folder,

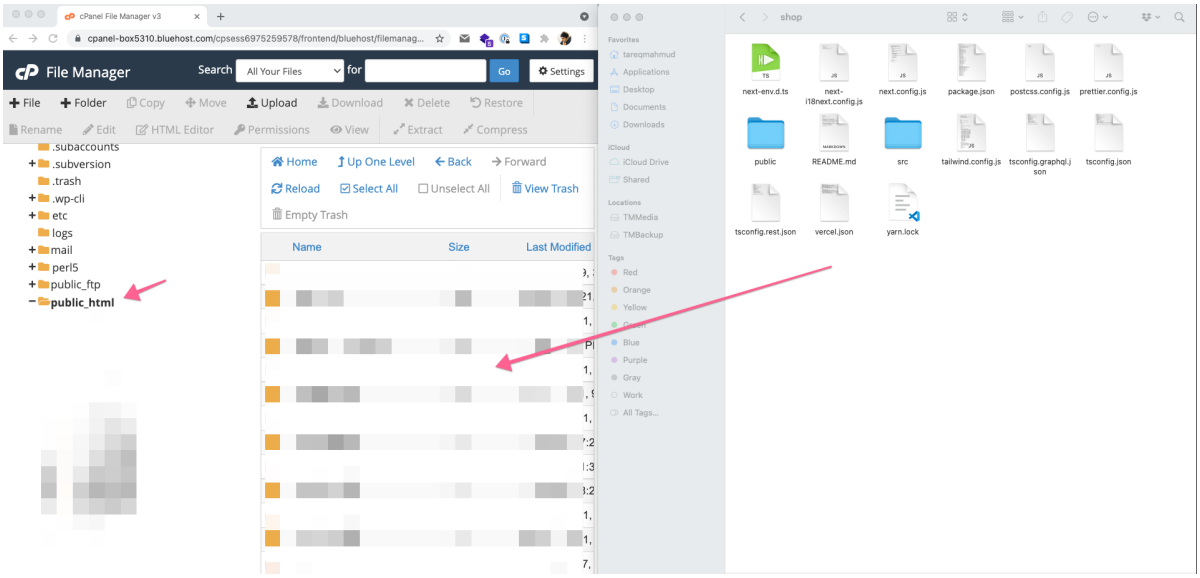
and run,

```
yarn build:shop-rest
yarn build:admin-rest
```

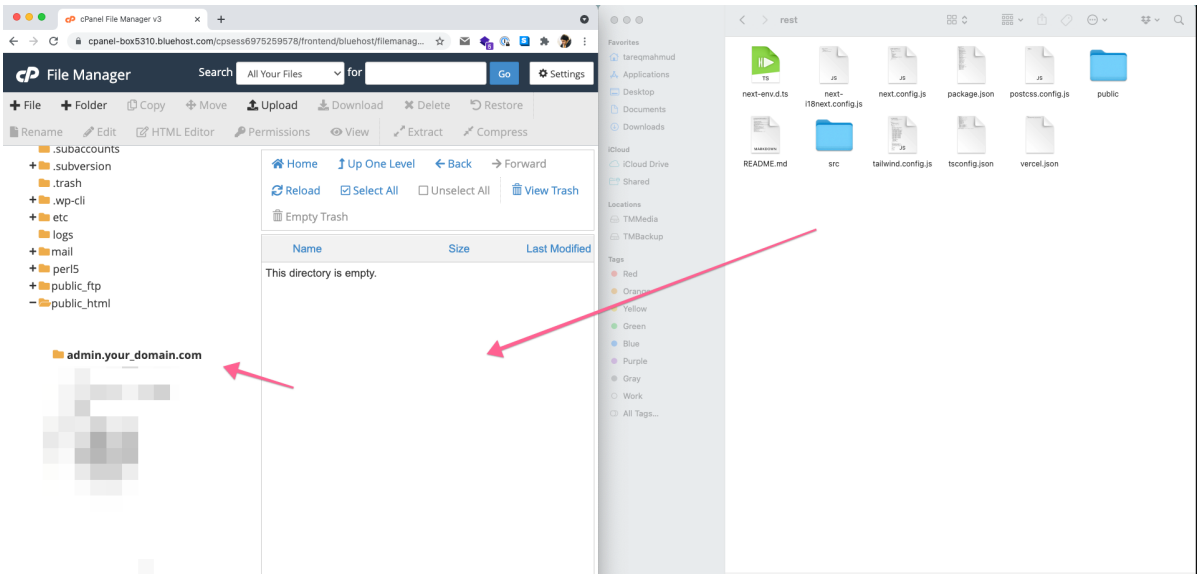
After build the project upload the

- `shop` to `root_domain -> public_html` folder
- `admin-rest` to `admin.your_domain.com` folder

`shop`,



`admin-rest`,

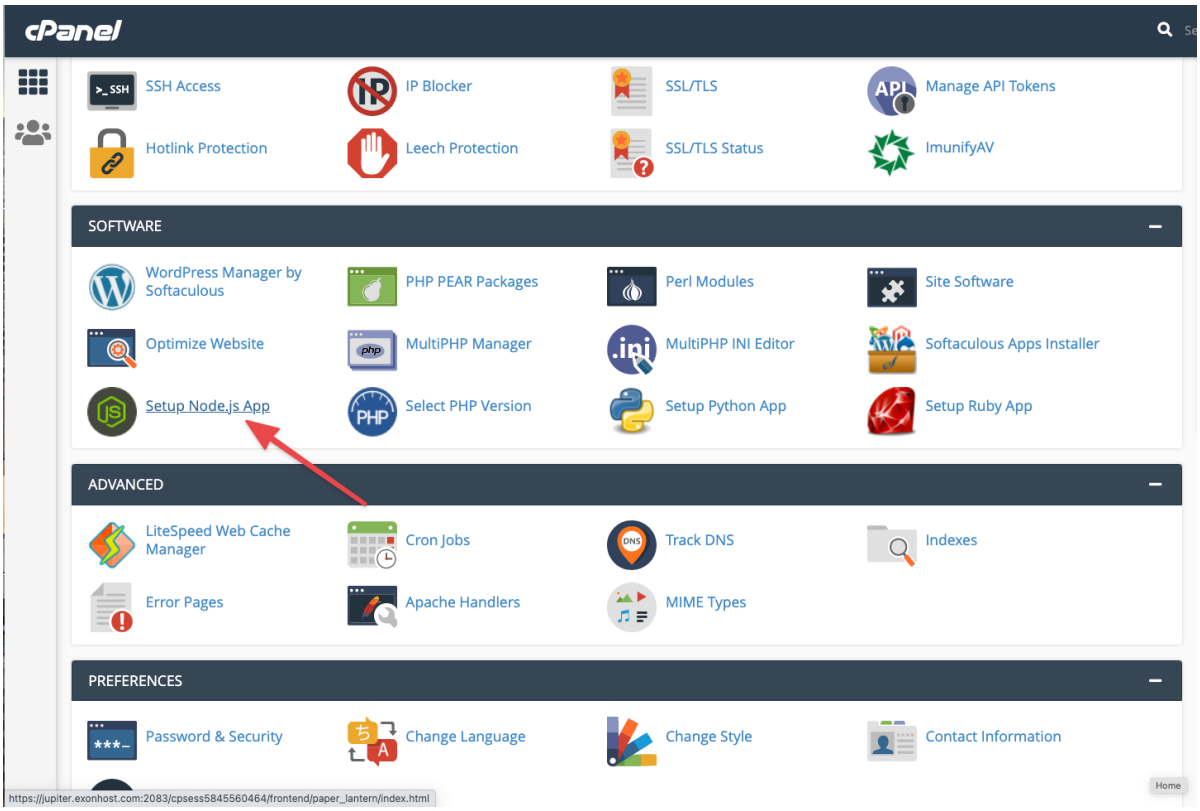


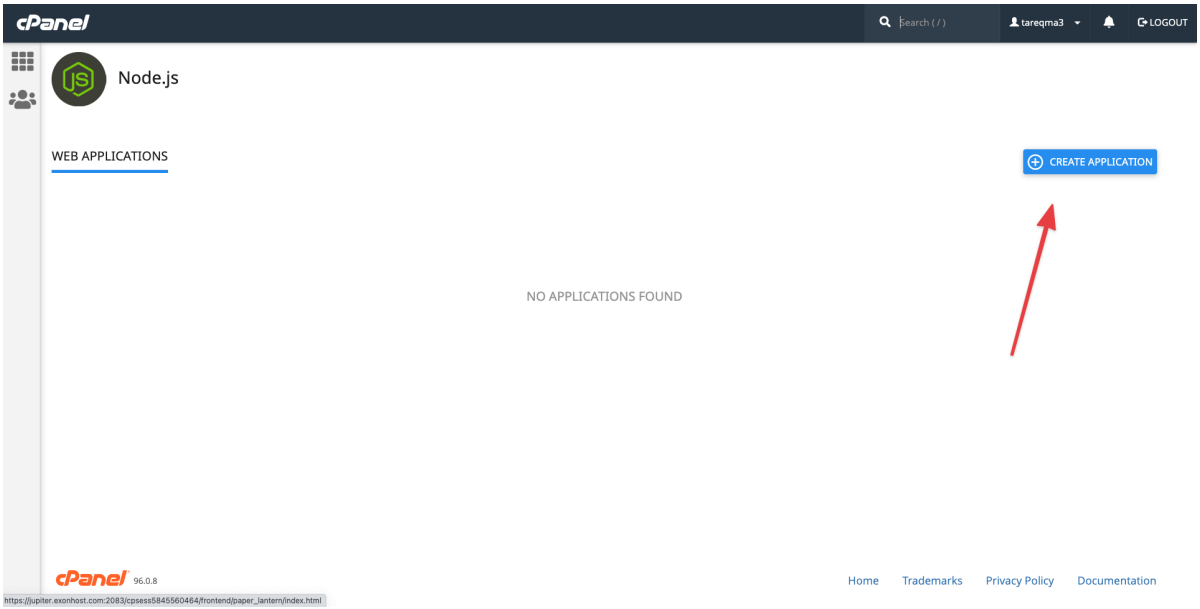
Install NodeJs Project

We'll run both `shop` and `admin` using the cPanel NodeJs application in this step.

To do that at first go to the NodeJS section from your cPanel,

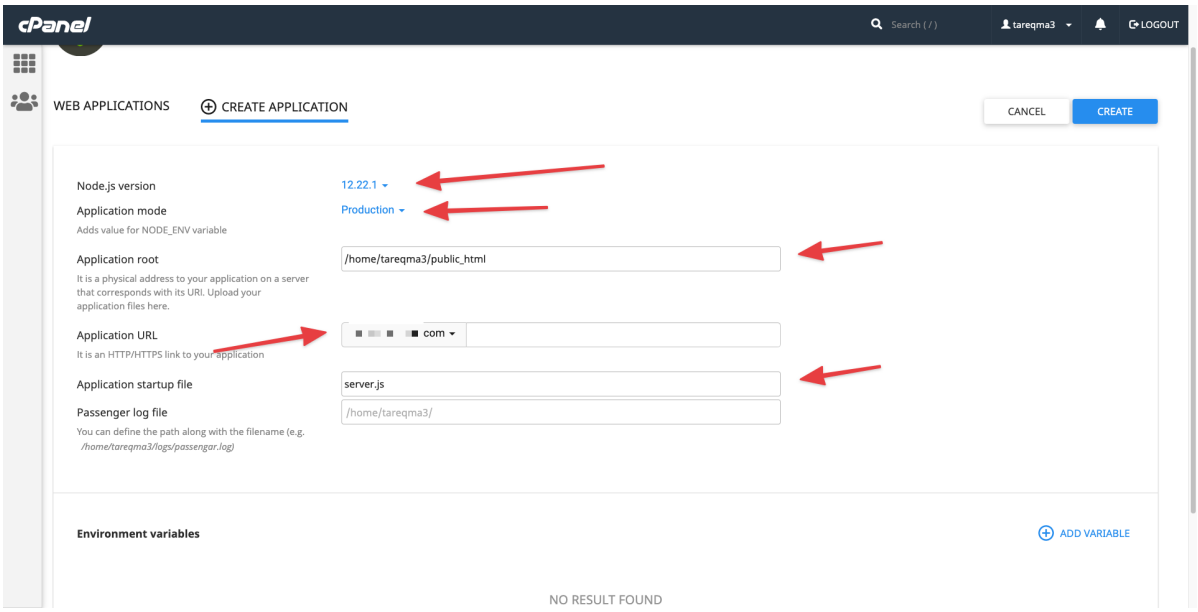
For `shop`,



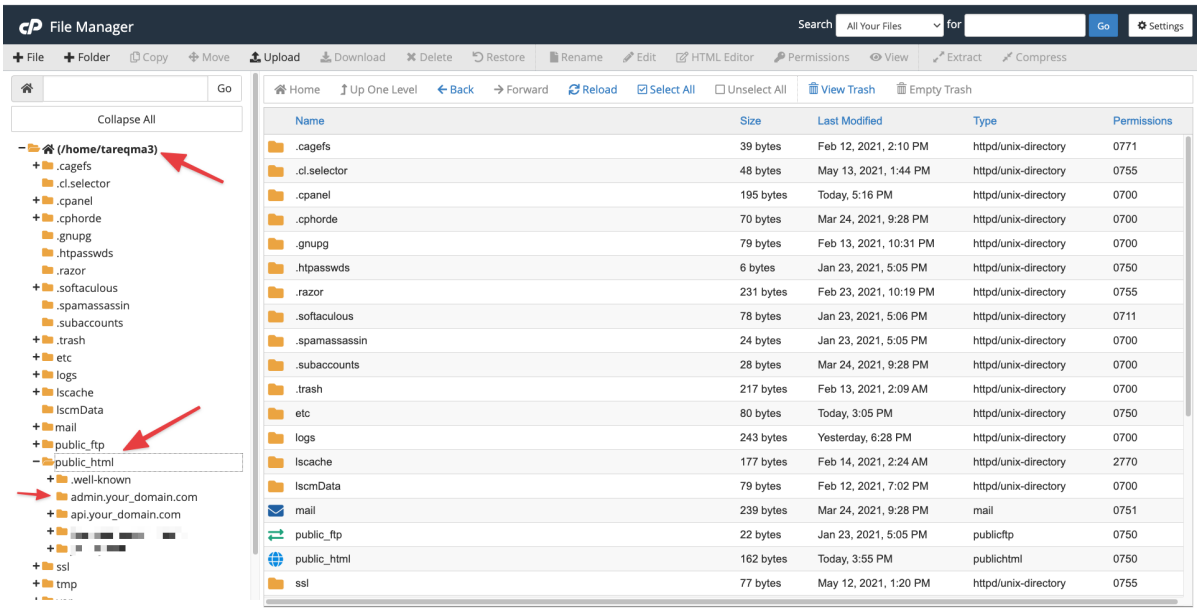


Now,

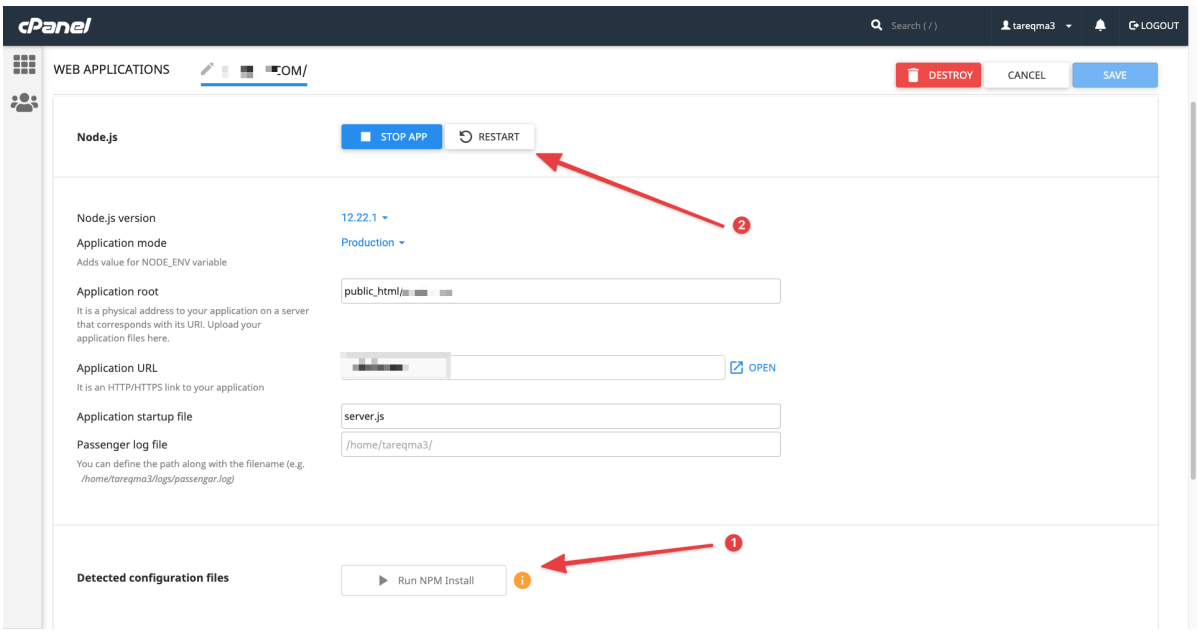
- Select NodeJS version
- Make environment `production.`
- Set Application Root
- And application startup file as `server.js`



You can get the Application Path from your cPanel file manager



After create NodeJS app, `install all the packages` and `restart the app`,



For `admin`,

Similarly, create a another NodeJS application for admin with `admin subdomain` and `admin subdirectory`

> After installing and run both NodeJS application, you can access your domain to check ChawkBazar,

Thank You!

SEO and Analytics

SEO

For SEO, we use `Next SEO` packages, and we provide two boilerplates at

```
shop -> src -> components -> seo
```

With that boilerplate, check `Next SEO` docs to use SEO on your chawkbazar site,

https://github.com/garmeeh/next-seo

Analytics

We have not used any analytics integration in this template yet. But, you can easily integrate any analytics using Next JS examples.

Laravel API

Introduction

Chawkbazar is a laravel multi api package for ecommerce. This package is for building dynamic ecommerce site using chawkbazar package.

Getting Started

For getting started with the template you have to follow the below procedure. For quick guide you can check below videos for installation.

Installation Video

Prerequisites

- PHP 8.1
- Composer
- Xamp/Wamp/Lamp for any such application for apache, nginx, mysql
- PHP plugins you must need
    - simplexml
    - PHP's dom extension

- mbstring
- GD Library

Resources you might need

1. https://laravel.com/docs/10.x
2. https://lighthouse-php.com/5/getting-started/installation.html
3. https://github.com/spatie/laravel-medialibrary
4. https://github.com/andersao/l5-repository
5. https://spatie.be/docs/laravel-permission/v3/introduction

Packages we have used

```
"require": {
    "php": "^8.0|^8.1",
    "barryvdh/laravel-dompdf": "2.0.1",
    "doctrine/dbal": "3.6.0",
    "guzzlehttp/guzzle": "7.5.0",
    "laravel/framework": "10.1.5",
    "laravel/socialite": "5.6.1",
    "laravel/tinker": "2.8.1",
    "marvel/shop": "dev-master",
    "messagebird/php-rest-api": "3.1.4",
    "openai-php/client": "^0.4.1",
    "psr/log": "2.0.0",
    "srmklive/paypal": "3.0.19",
    "stevebauman/purify": "5.1.1",
    "symfony/http-client": "6.2.6",
    "symfony/mailgun-mailer": "6.2.5",
    "twilio/sdk": "6.44.4"
},
"require-dev": {
    "spatie/laravel-ignition": "1.3.1",
    "fakerphp/faker": "1.20.0",
    "laravel/sail": "1.15.1",
    "mockery/mockery": "1.5.0",
    "nunomaduro/collision": "6.2.1",
    "phpunit/phpunit": "9.5.21",
    "squizlabs/php_codesniffer": "3.7.1"
},
```

Installation

- Make sure you have run xamp/mamp/wamp/lamp for mysql and php
    - Create a database in your mysql and put those info in next step
    - Rename .env.example file to .env and provide necessary credentials. Like database credentials stripe credentials, s3 credentials(only if you use s3 disk) admin email shop url etc.
      Specially check for this `env` variables

```
DB_HOST=localhost
DB_DATABASE=chawkbazar_laravel
DB_USERNAME=root
DB_PASSWORD=
```

- Run `composer install`

```
▶ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
```

```
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: barryvdh/laravel-dompdf
Discovered Package: bensampo/laravel-enum
Discovered Package: cviebrock/eloquent-sluggable
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: ignited/laravel-omnipay
Discovered Package: intervention/image
Discovered Package: laravel/legacy-factories
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: mll-lab/laravel-graphql-playground
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: nuwave/lighthouse
Discovered Package: pickbazar/shop
Discovered Package: prettus/l5-repository
Discovered Package: spatie/laravel-medialibrary
Discovered Package: spatie/laravel-permission
Package manifest generated successfully.
95 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

- run `php artisan key:generate`

```
▶ php artisan key:generate

Application key set successfully.
```

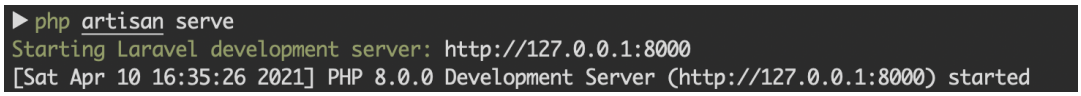
- Run `php artisan marvel:install` and follow necessary steps.

```
Installing Pickbazar Dependencies...

 Do you want to migrate Tables? If you have already run this command or migrated tables then be aware, it will erase all of your data. (yes/no) [no]:
 > yes

Migrating Tables Now....
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (79.70ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (58.57ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (79.56ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (143.06ms)
Migrating: 2020_04_17_194830_create_permission_tables
Migrated:  2020_04_17_194830_create_permission_tables (839.79ms)
Migrating: 2020_06_02_051901_create_pickbazar_tables
Migrated:  2020_06_02_051901_create_pickbazar_tables (1,601.56ms)
Migrating: 2020_10_26_163529_create_media_table
Migrated:  2020_10_26_163529_create_media_table (71.72ms)
Tables Migration completed.

 Do you want to seed dummy data? (yes/no) [no]:
 > yes

Copying necessary files for seeding....
File copying successful
Seeding....
Seed completed successfully!

 Do you want to create an admin? (yes/no) [no]:
 > no

Copying resources files...
Installation Complete
```

- For image upload to work properly you need to run `php artisan storage:link`.

```
The [/var/www/html/public/storage] link has been connected to [/var/www/html/storage/app/public].
The links have been created.
```

- run `php artisan serve`



> NB: You must need to run `php artisan marvel:install` to finish the installation. Otherwise your api will not work properly. Run the command and follow the necessary steps.

> NB: your frontend `NEXT_PUBLIC_GRAPHQL_API_ENDPOINT` env value will be `localhost:8000/graphql`

> NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost:8000/`

For MAC and Linux(with sail and docker)

There is an alternate installation procedure for linux and mac. You can follow below procedure to getting started with `sail`

Prerequisites

- Docker

Installation

- Run Docker application first
- Now go to your chawkbazar-laravel root directory and run `bash install.sh`. It will guide you through some process. Follow those steps carefully and your app will be up and running
- Navigate to `api` then `sail down` to stop the container. If you want to remove the volumes then `sail down -v`

> NB: your frontend `NEXT_PUBLIC_GRAPHQL_API_ENDPOINT` env value will be `localhost/graphql`

> NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost/`

Configuration

All the configurations files are in `packages/marvel/config` folder. You can change any necessary configuration from these files. You can also publishes the shop configuration using `artisan vendor:publish --provider="Marvel\ShopServiceProvider" --tag="config"` command in your root folder.

- Create .env file from our example.env file and put necessary configuration
- By default s3 is using for media storage but you can also use local folder. Change `MEDIA_DISK` IN `.env` file as your need. Supported options are `public` and 's3`
- Set Payment related configuration to `STRIPE_API_KEY` `.env` variable
- Set `ADMIN_EMAIL`, `SHOP_URL` and necessary Database credentials.

Console Commands

- `php artisan marvel:install` complete installation with necessary steps
- `php artisan marvel:seed` seeding demo data
- `php artisan marvel:copy-files` copy necessary files

- `php artisan vendor:publish --provider="Marvel\ShopServiceProvider" --tag="config"` published the configuration file

All of the above custom command you will find in `packages/marvel/src/Console` folder.

Development

We have provided below two api.

- REST API
- GraphQL API

REST API

All the rest routes is resides in `packages/marvel/src/Rest/Routes.php` file and you can easily navigate to corresponding controller and necessary files.

Endpoints

https://documenter.getpostman.com/view/11693148/UVC5Fo9R

Folder structure

config

The `packages/marvel/config` folder contains all the `config` for our app.

database

The `packages/marvel/database` folder contains all the `factories` and `migrations`.

- **Http:**

  Contains two folders. `Controllers` and `Requests`. All the necessary controllers and requests are in this two folder.

- **Database:**

  Contains `Models` and `Repositories`. For repositories we have used `l5-repository` (https://github.com/andersao/l5-repository).

Enums

All the `enums` that are used throughout the app is in `packages/marvel/src/Enums` folder.

Events

All the events are in `packages/marvel/src/Events` folder.

Listeners

All the listeners corresponding to the above events are in `packages/marvel/src/Listeners` folder

Mail

All the mailables are in `packages/marvel/src/Mails` folder.

Notifications

Notifications related to order placed is reside `packages/marvel/src/Notifications`. Currently we have provided mail notification but you can easily add others notification system using laravel conventions.

Providers

All the secondary service providers that we have used in our app resides in `packages/marvel/src/Providers` folder. The main `ShopServiceProviders` reside in `packages/marvel/src/` folder.

stubs

The `packages/marvel/stubs` folder contains all the necessary email templates and demo data related resources for the app.

GraphQL API

Laravel GraphQL API Endpoint

```
your_domain/graphql
```

To access the graphql api you will need a graphql client. GraphiQL is one of them. You can install it from below link(https://www.electronjs.org/apps/graphiql)

Alternatives

- GraphQL Playground
- https://altair.sirmuel.design/

Folder Structure

All the code specifically related to GraphQL reside in `packages/marvel/src/GraphQL/` folder.

- **Mutations:**

  Folder contain necessary mutations files which is connected to rest Controller file for code reusability.

- **Queries:**

  Folder contain necessary queries files which is connected to rest Controller file for code reusability.

- **Schema**

  This is the most important part of the graphql api. Check the lighthouse-php(https://lighthouse-php.com/4.18/getting-started/installation.html) doc for understanding how schema works. We have provided schema for all the models in our app. If you check the above `lighthouse-php` doc you will understand how schema works and how you can modify it to your need.

> ### Before Finishing up
>
> Before you finishes the installation process make sure you have completed the below steps.
>
> - Copied necessary files and content to your existing laravel projects(if using existing projects)
> - Installed all the necessary dependencies.
> - Ran `marvel:install` commands and followed the necessary steps.
> - Created a .env file with all the necessary env variables in the provided projects.
> - Put `DISK_NAME` configuration for `public` or 's3`
> - Set Payment related configuration to `STRIPE_API_KEY`

Payment Gateway

We have used `omnipay` for payment and given `stripe` and `cash_on_delivery` default. We have used `ignited/laravel-omnipay` by forking it in our packages due to some compatibility issue with Laravel 8.

Extending The Functionality

If you want to extend the functionality of the app you can easily do it on your app. You would not need to modify code from our packages folder. Like you can add any `routes` and corresponding `controller` in your laravel app as your need. We highly suggest you to do all the modification in your app so you can update the package easily.

Deployment

Its a basic laravel application so you can deploy it as any other laravel application. Make sure you have installed all the php required plugins we mentioned above.

Social Login

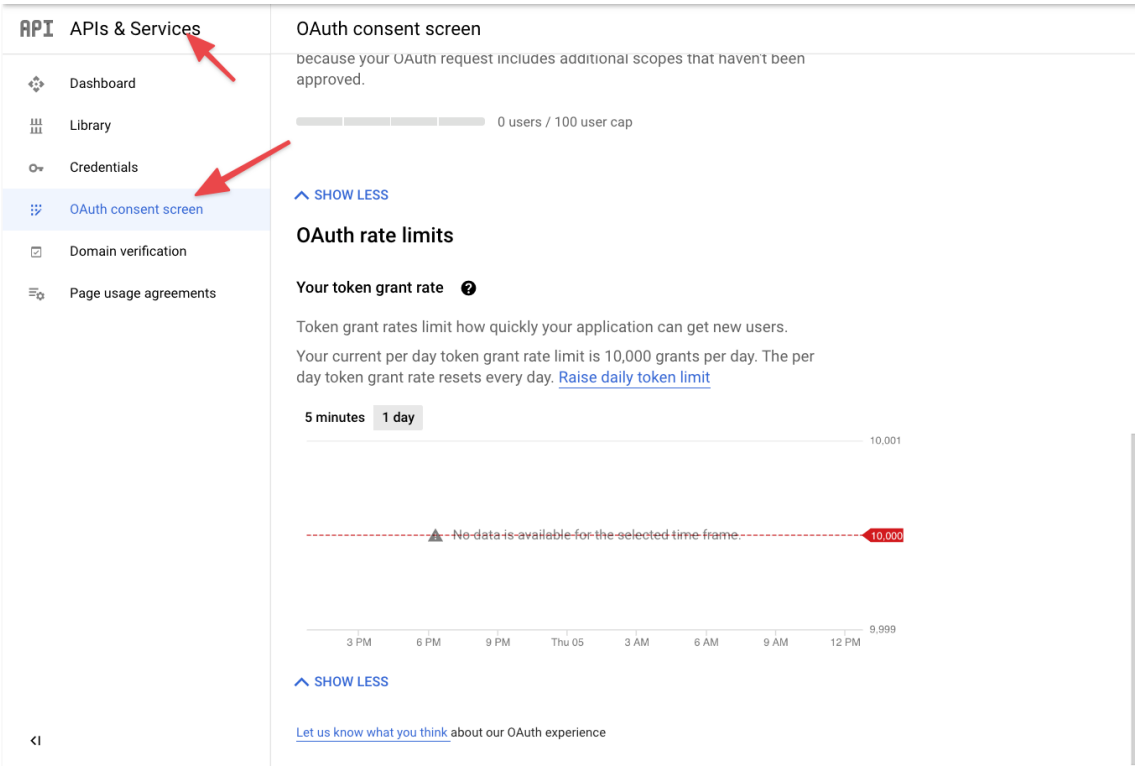
You can use a social network like `Google` or `Facebook` as a login provider like email/username and password. To set up social login, follow the below procedure,

Google

For Google, follow this procedure,

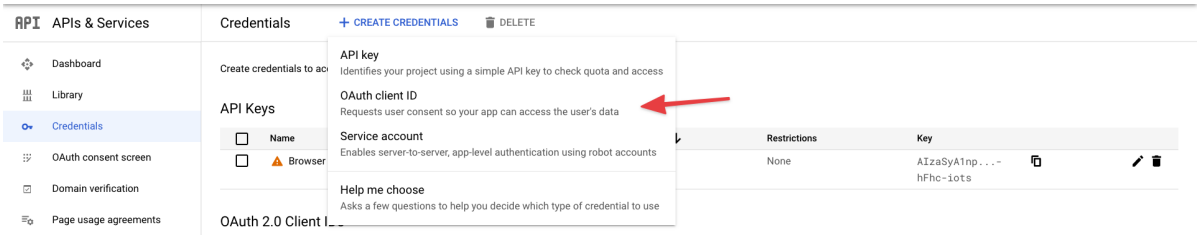
1. At first go to Google Console dashboard (https://console.cloud.google.com/)
2. From the console, create a new project

3. After creating the project, go to `APIs & Services -> OAuth consent screen.`



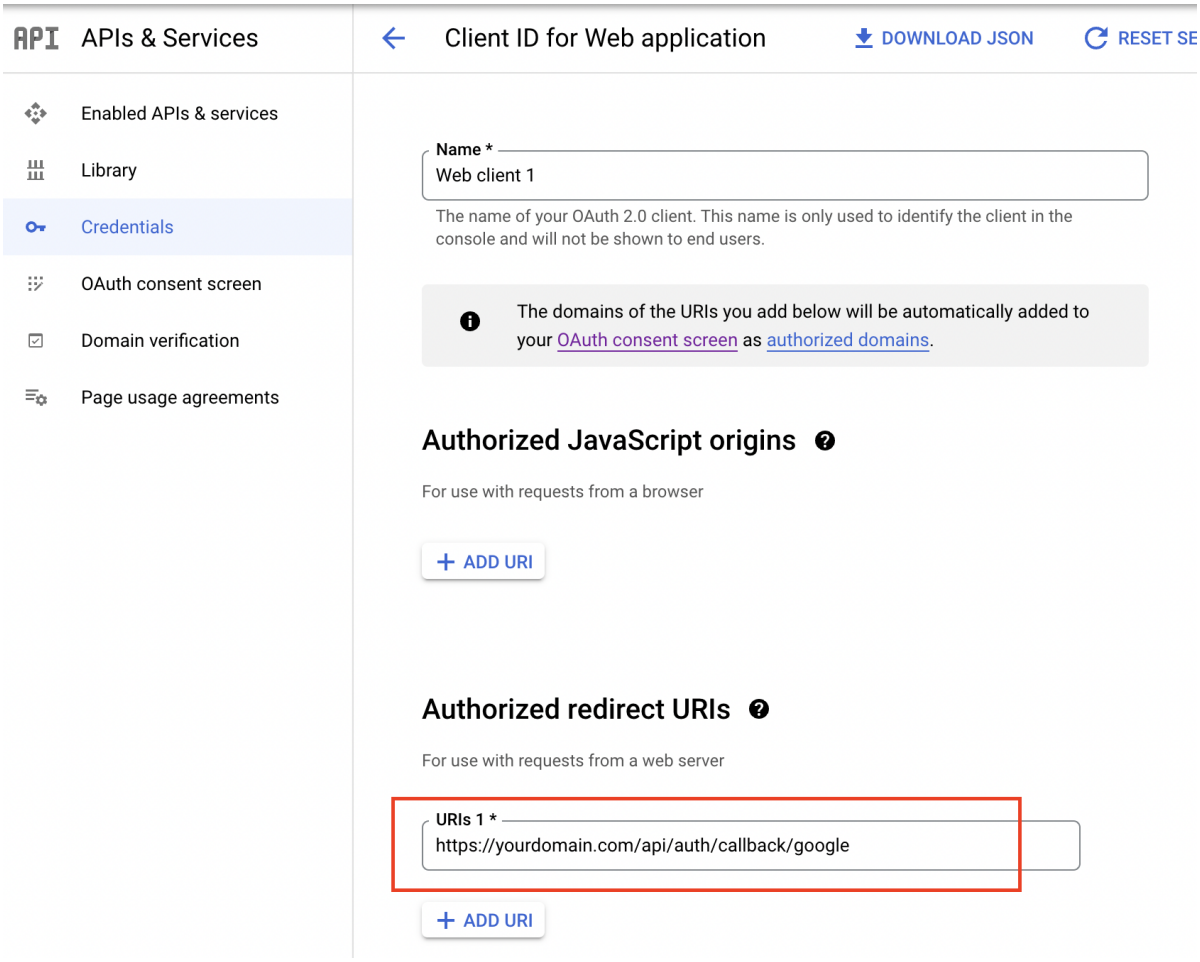
4. Create an `External OAuth Consent.`

5. Then, go to the `credentials` section and create `OAuth Client ID.`

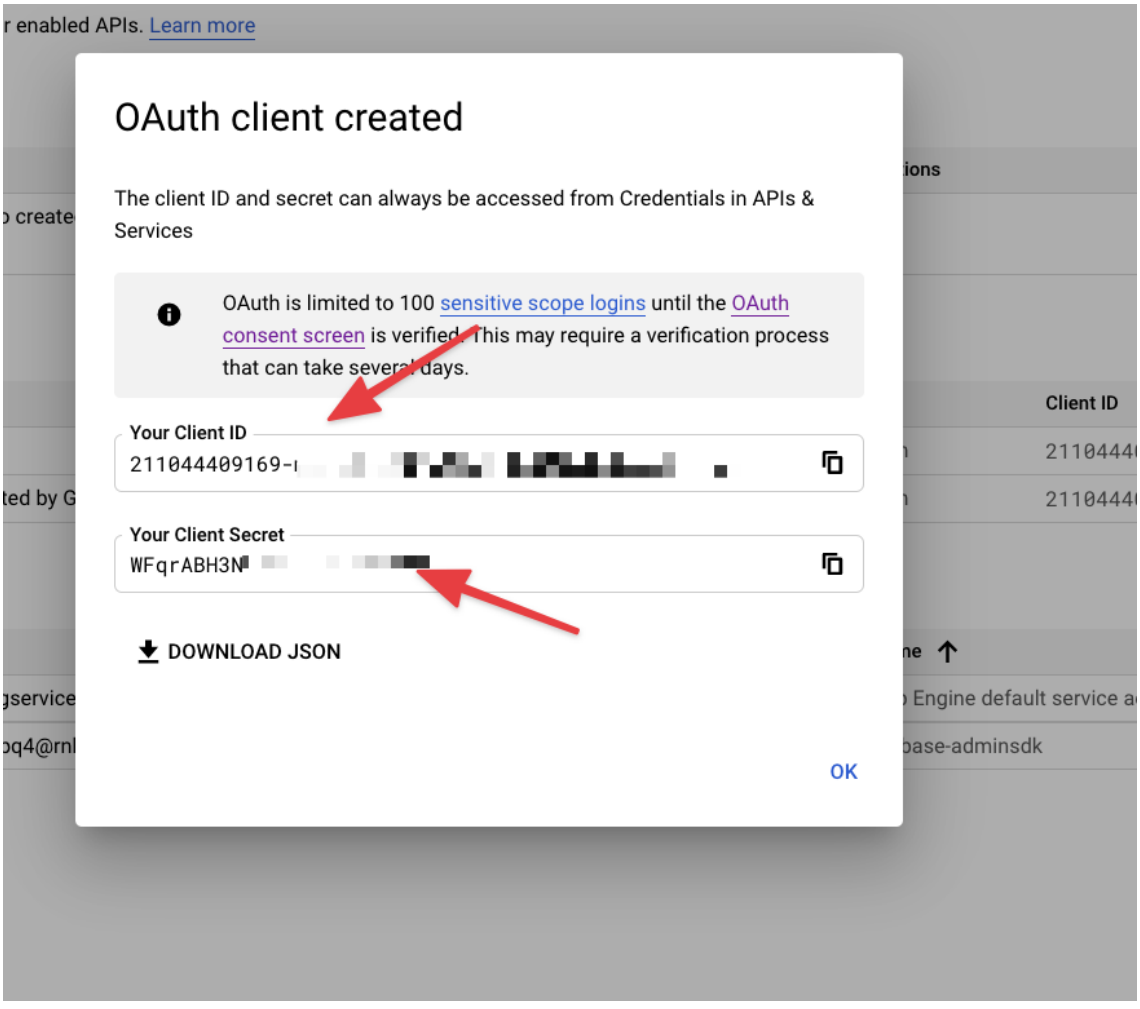


Add an `Authorized redirect URIs`. The format of the URIs should be:

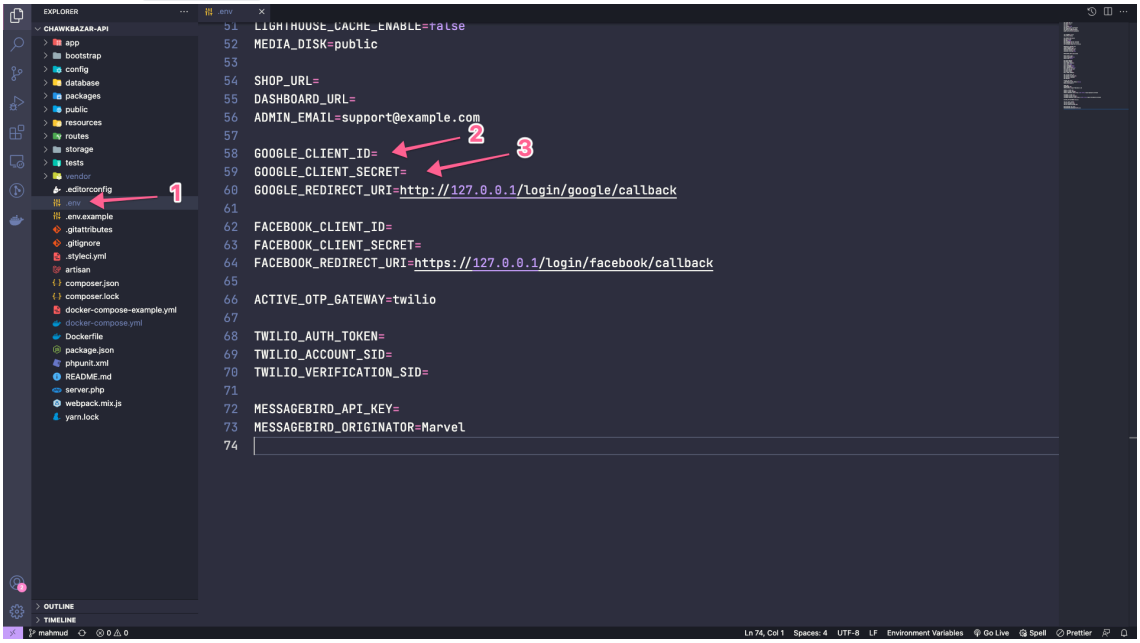
```
http://yourdomain.com/api/auth/callback/google
```



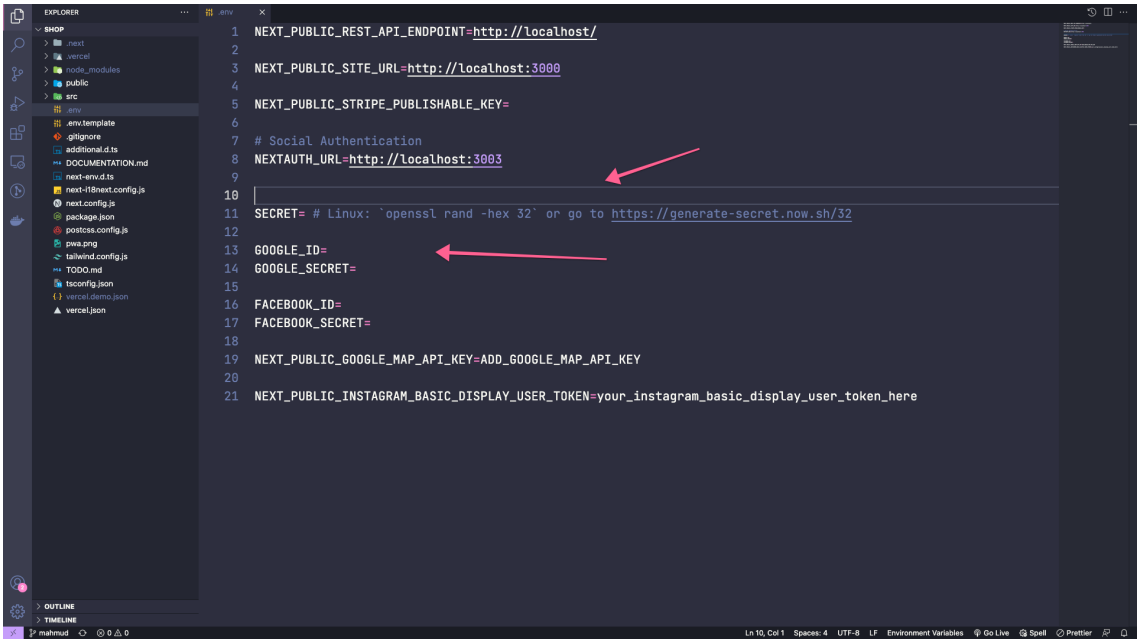
6. Then you'll get your credentials,



7. Then open `api/.env` and add API ID and credentials,



8. Then go to this `https://generate-secret.now.sh/32` site and generate a secret key.
9. Then open `shop/.env` and add credentials to `GOOGLE_CLIENT_ID` and `GOOGLE_CLIENT_SECRET` also add a `SECRET` code.



After configuration, make sure you rebuild your project using this command,

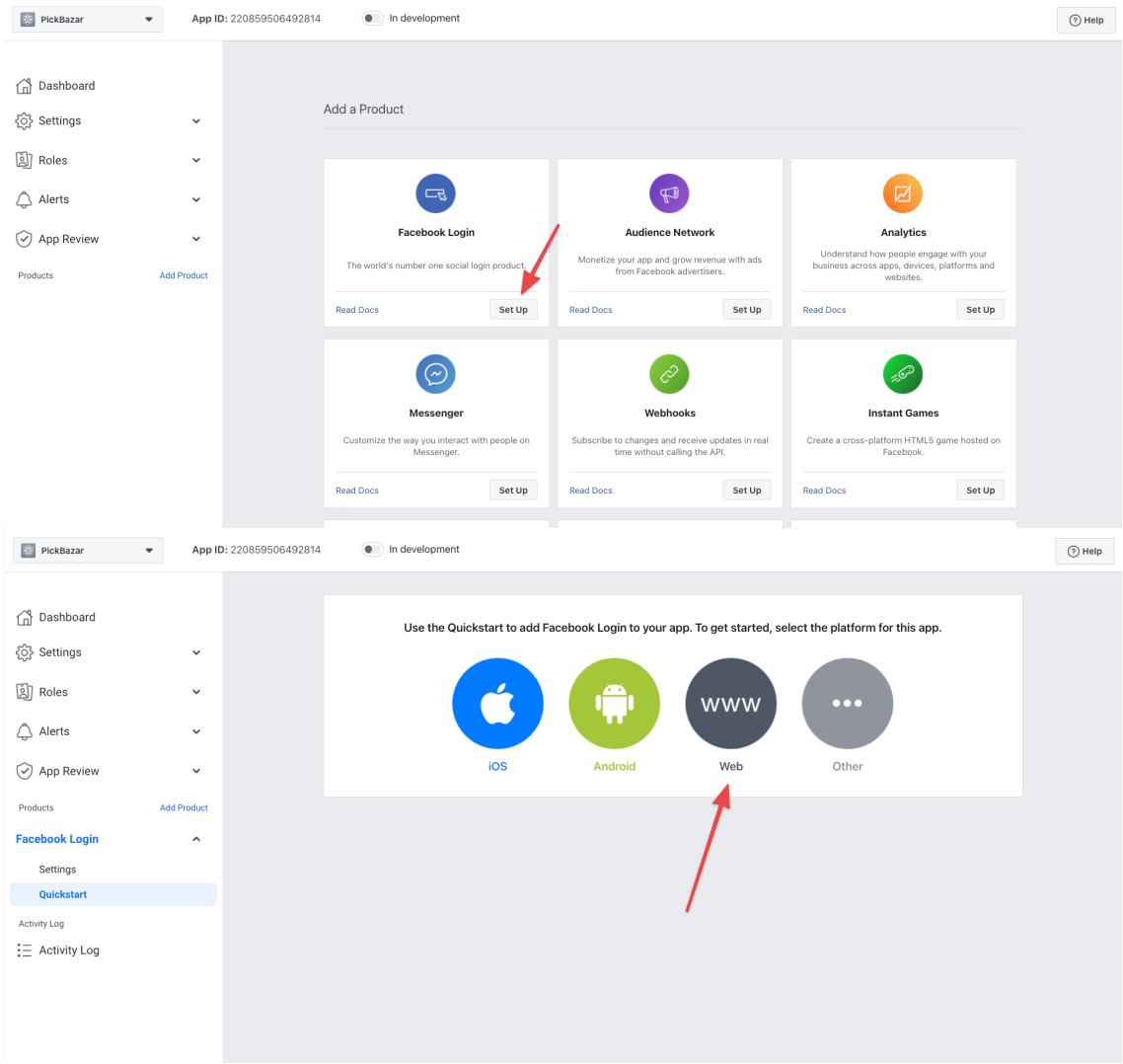
For REST API

```
yarn build:shop-rest
yarn build:admin-rest
```

Facebook

1. Go to Facebook developer dashboard (https://developers.facebook.com/apps/)
2. Then create a new app for chawkbazar

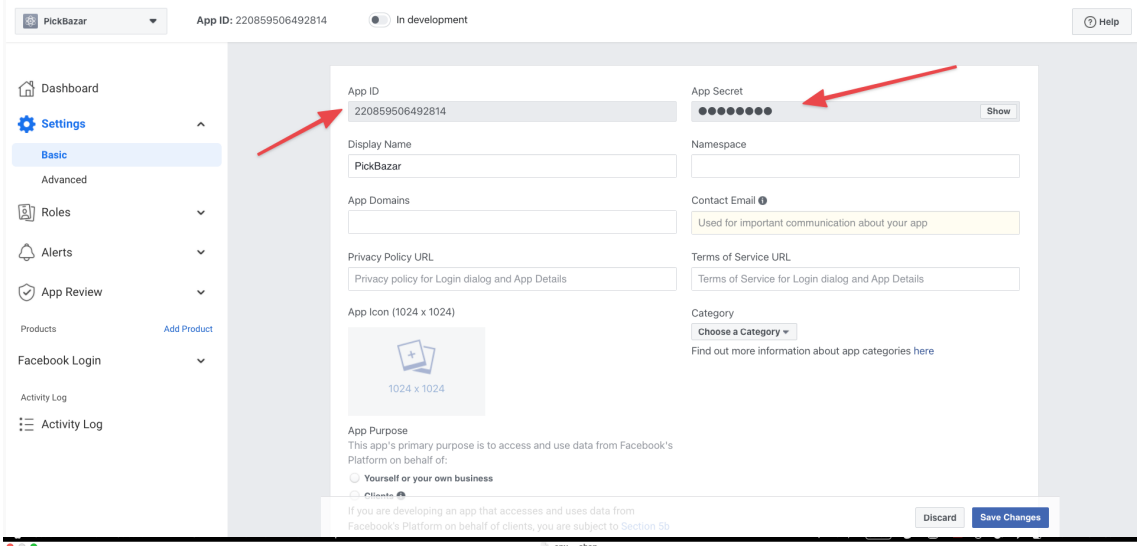
3. After that, set up `Facebook Login.`

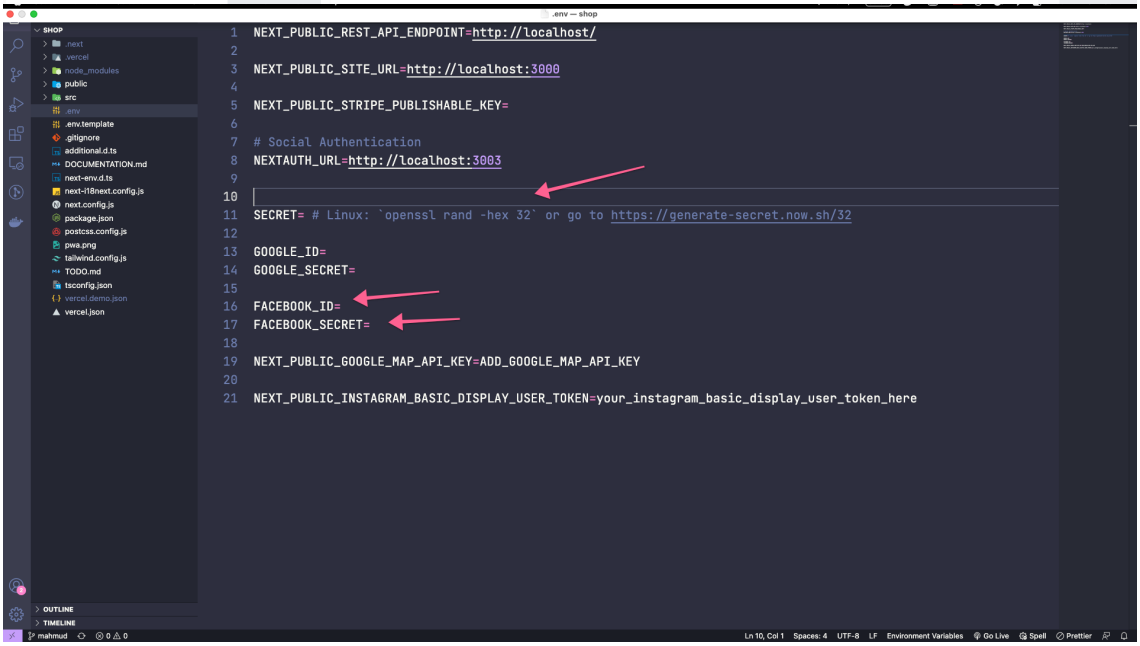


4. Then provide all the necessary information.

5. After that, go to the settings page, and on that page, you'll get all the credentials.

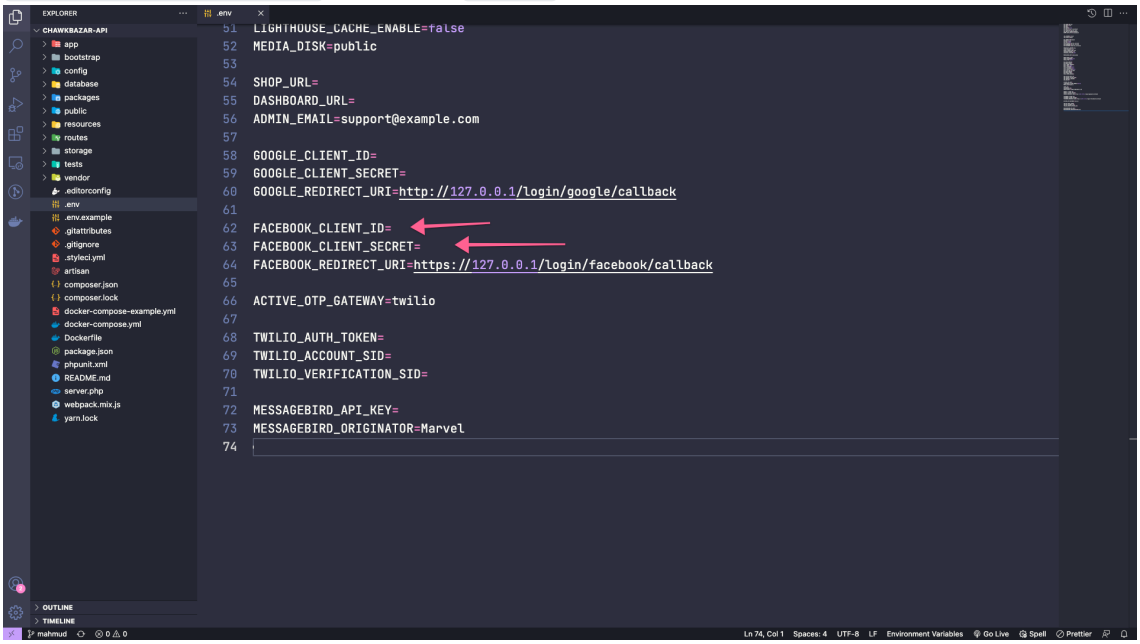
6. Then open `api/.env` and add API ID and credentials,





7. Then go to this `https://generate-secret.now.sh/32` site and generate a secret key.

8. Then open `shop/.env` and add credentials to `FACEBOOK_CLIENT_ID` and `FACEBOOK_CLIENT_SECRET` also add a `SECRET` code.



> For Facebook, make sure you apply verification for the APP. Otherwise, the Facebook login won't work. (https://developers.facebook.com/docs/app-review/)

After configuration, make sure you rebuild your project using this command,

For REST API

```
yarn build:shop-rest
yarn build:admin-rest
```

OTP (Mobile Number Verification)

We implement the `OTP` feature on ChawkBazar `v2.2.0`. So if you want to use `OTP` with `ChawkBazar,` then make sure your ChawkBazar is `v2.2.0` or later.

With the OTP feature, you can do,

```
- Login with Mobile Number
- OTP verification for updating Mobile Number in profile update section
- OTP verification during checkout process in customer contact section
```

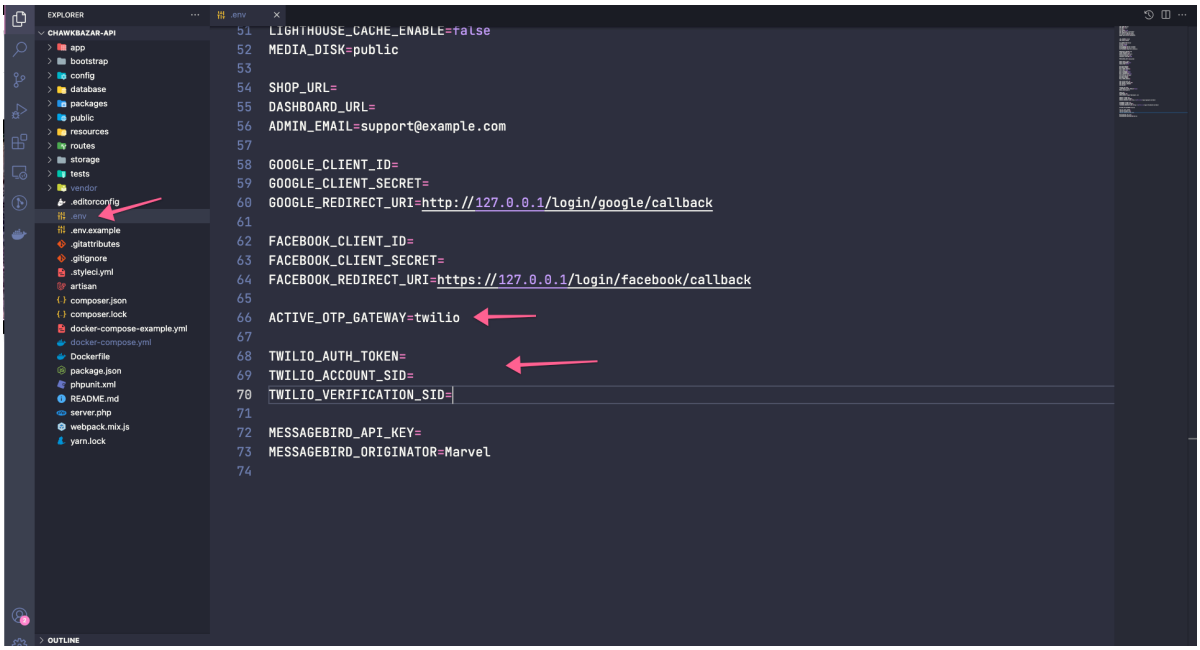
Configuration

As of today, we implement two service providers for OTP. One is `Twilio,` and another is `MessageBird.`

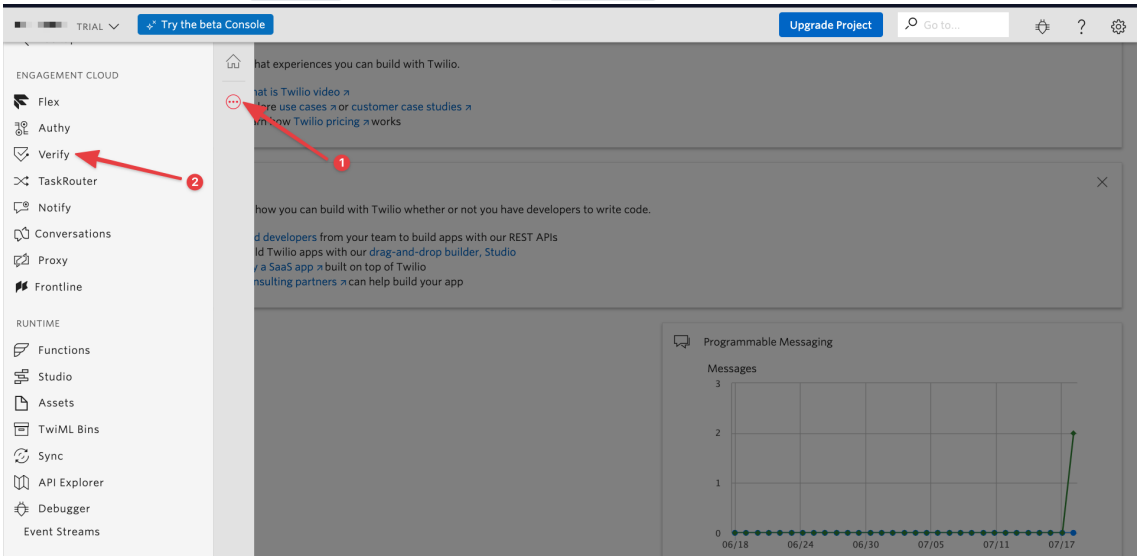
Twilio Configuration:

If you want to use `Twilio` as your OTP service provider, then follow this procedure,

1. Create an account at Twilio
2. After that, go to the `Twilio Console Panel` and copy the `ACCOUNT SID,` and `AUTH TOKEN` key and add this key to your `api/.env` file

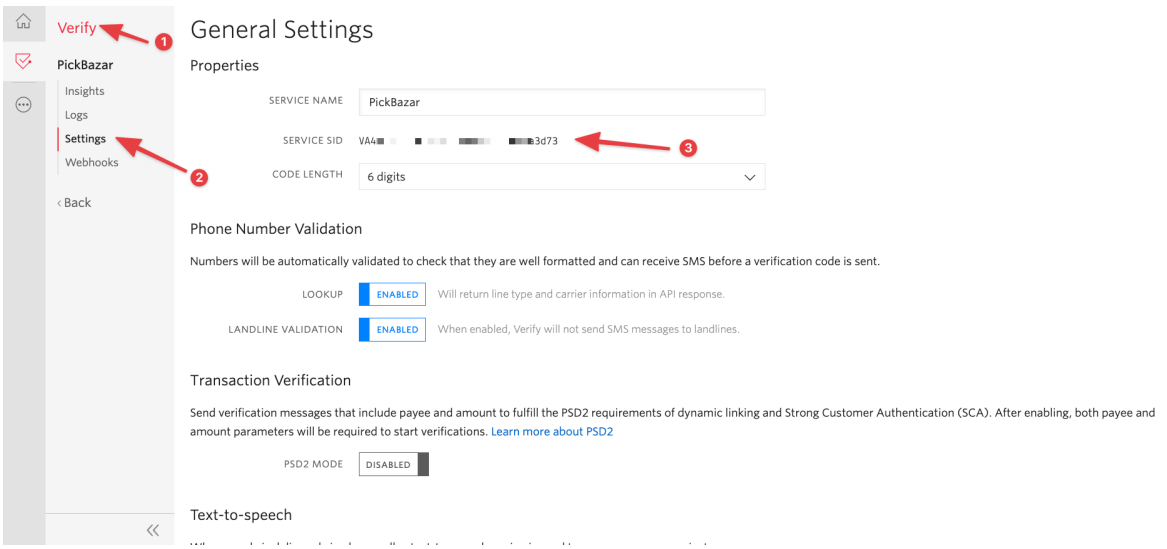


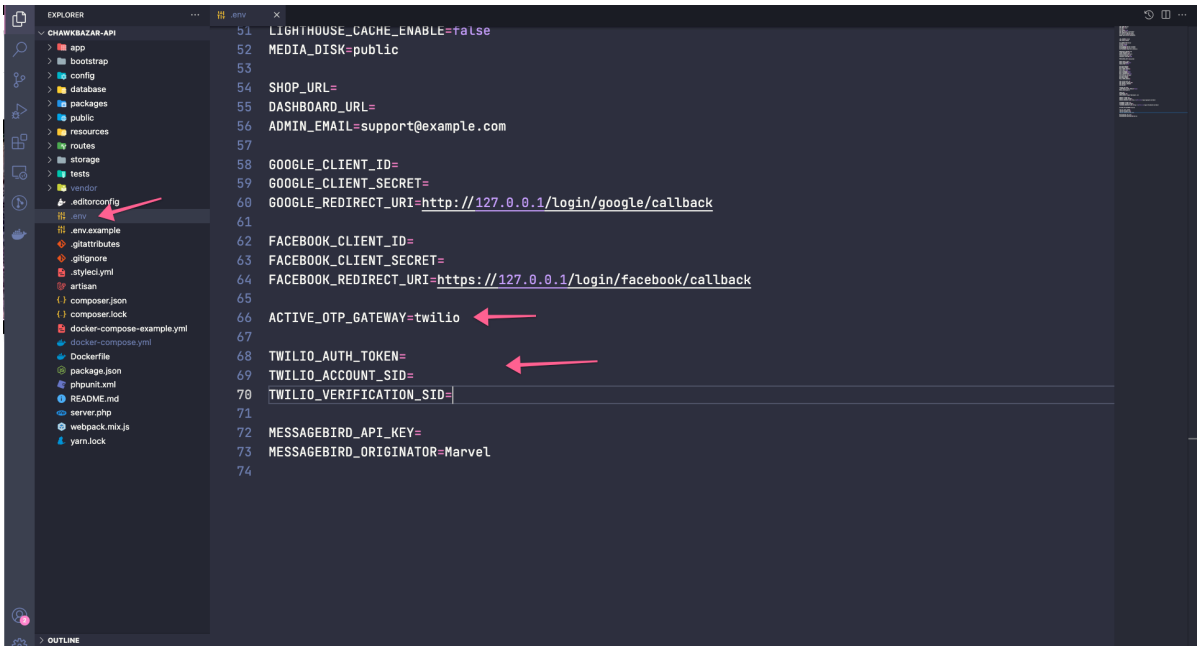
3. After that, go to the `Verify` option from the `Sidebar` panel,



4. And from `verify dashboard,` create a service.

5. Add add that service key to `api/.env -> TWILIO_VERIFICATION_SID`





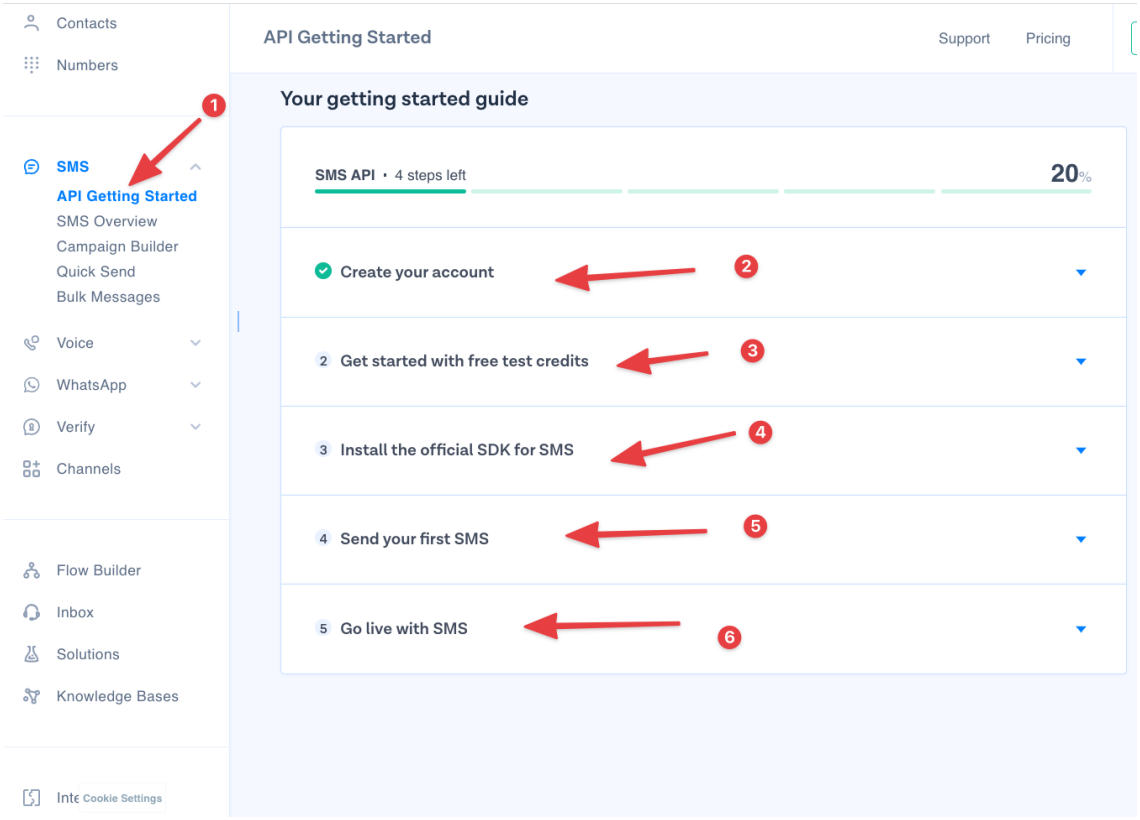
With that, your configuration is complete.

MessageBird:

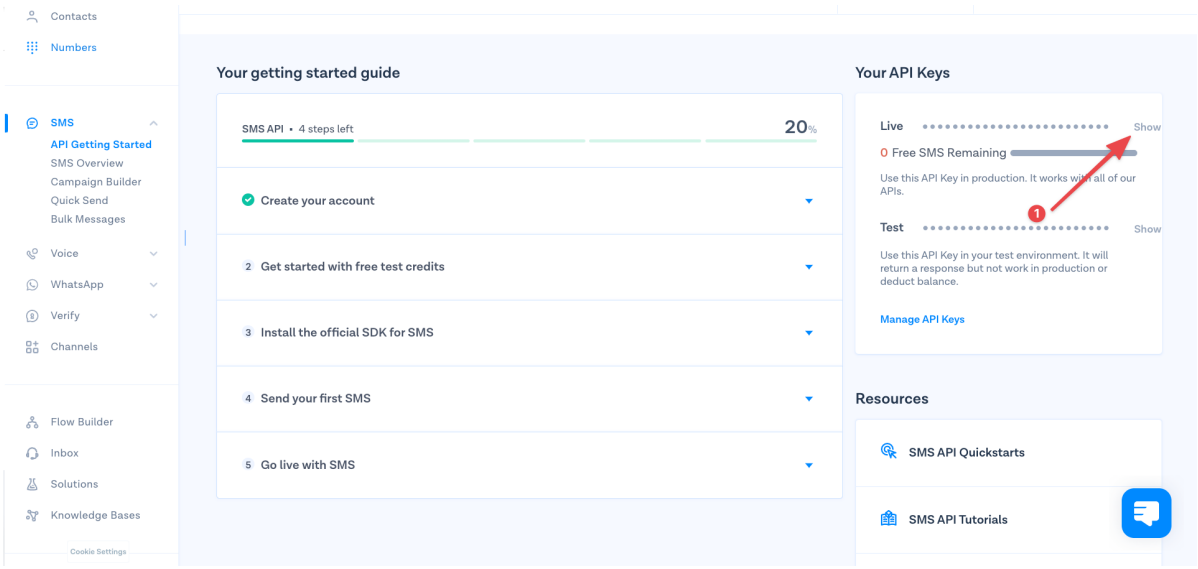
> Use only one service provider for OTP service, so use this `MessageBird` configuration only if you want to use `MessageBird` instead of `Twilio.`

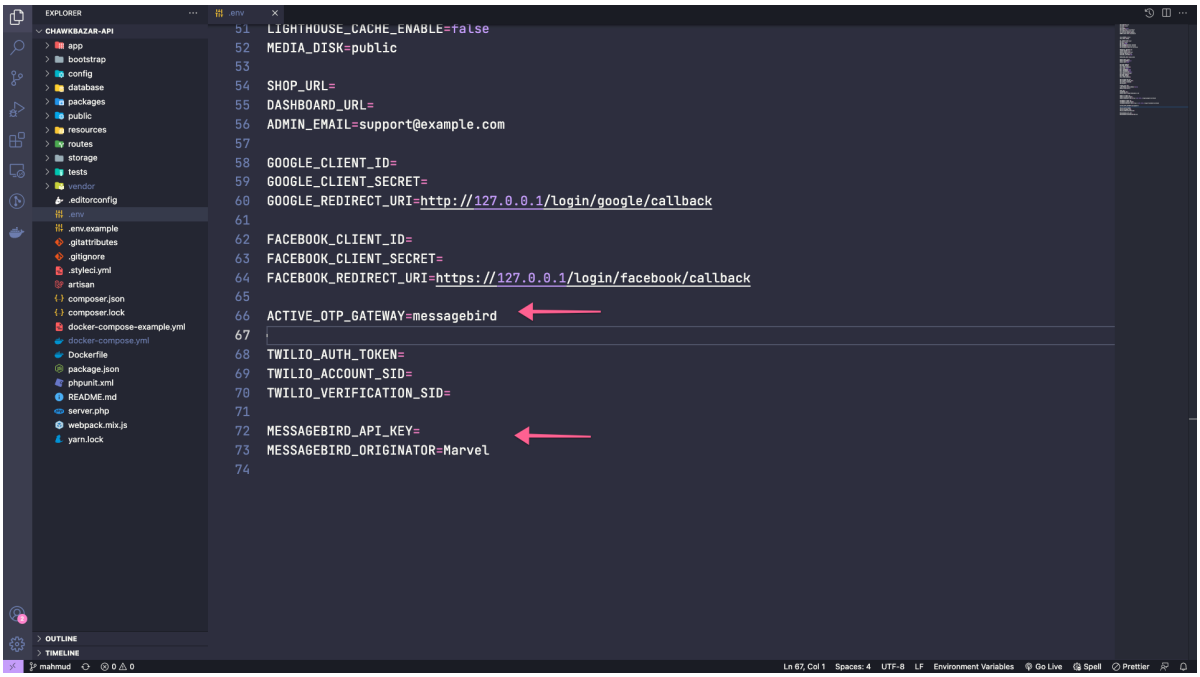
To configure `MessageBird,` follow this procedure,

1. Create an account from MessageBird
2. After creating the account, go to the `API Getting Started` and follow all the procedures step by step.



3. After completing the profile, copy the `API` key from `MessageBird` and paste it to `API/.env`



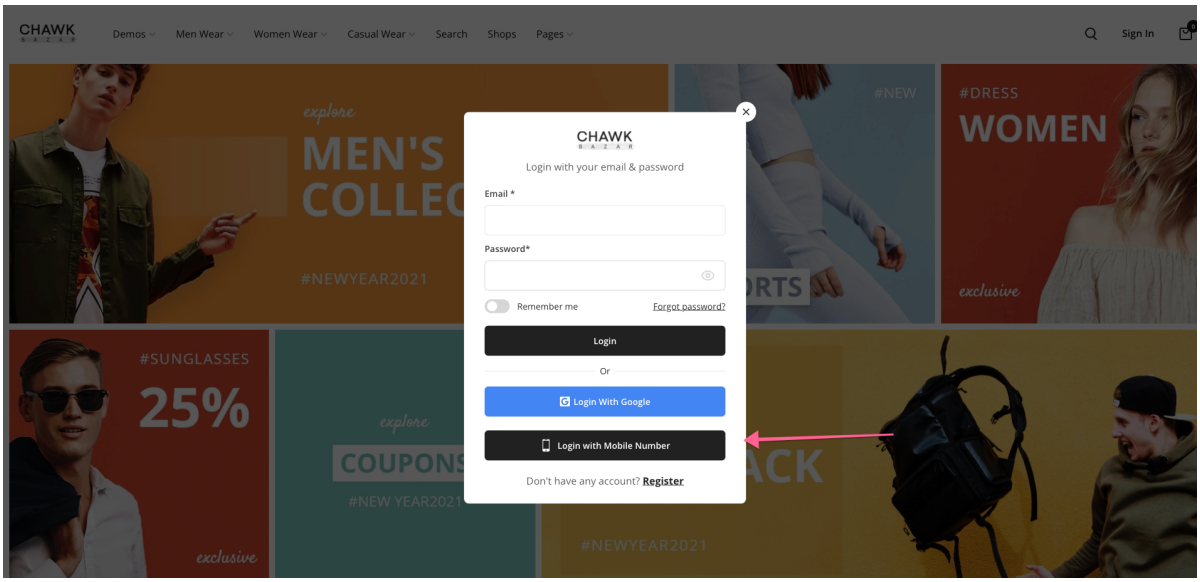


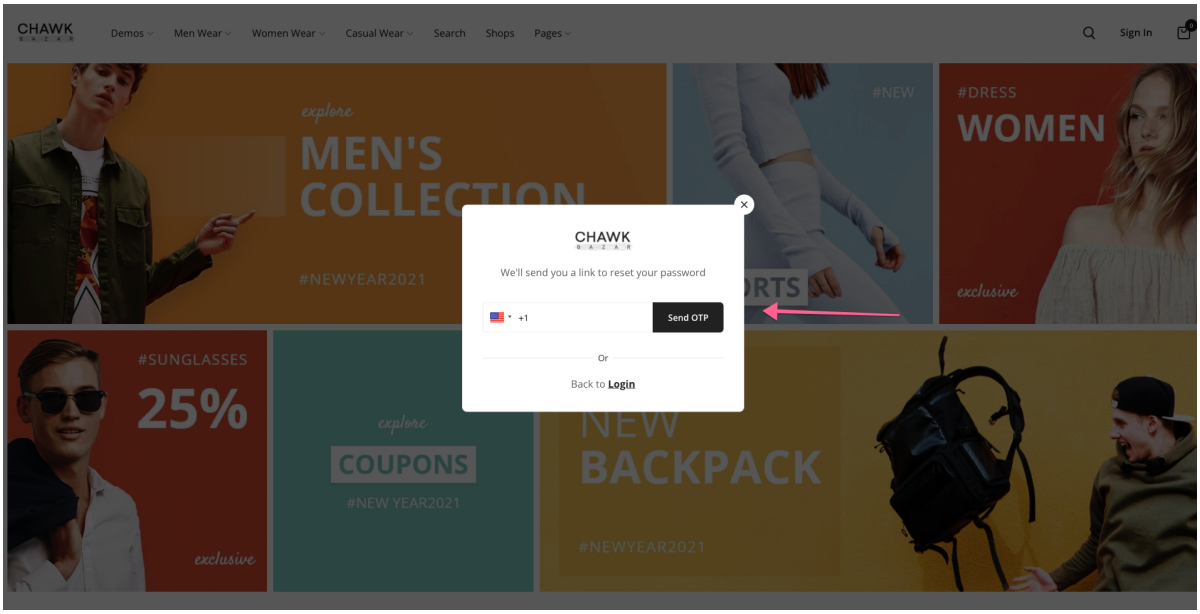
With that, `MessageBird` is appropriately configured for the OTP Service.

FrontEnd Demo

Login with Mobile Number:

After configuration, you will be able to use `mobile number` as a login provider. When users select the `Join` button, they'll get an option `Login with mobile` to use a mobile number as a login provider.

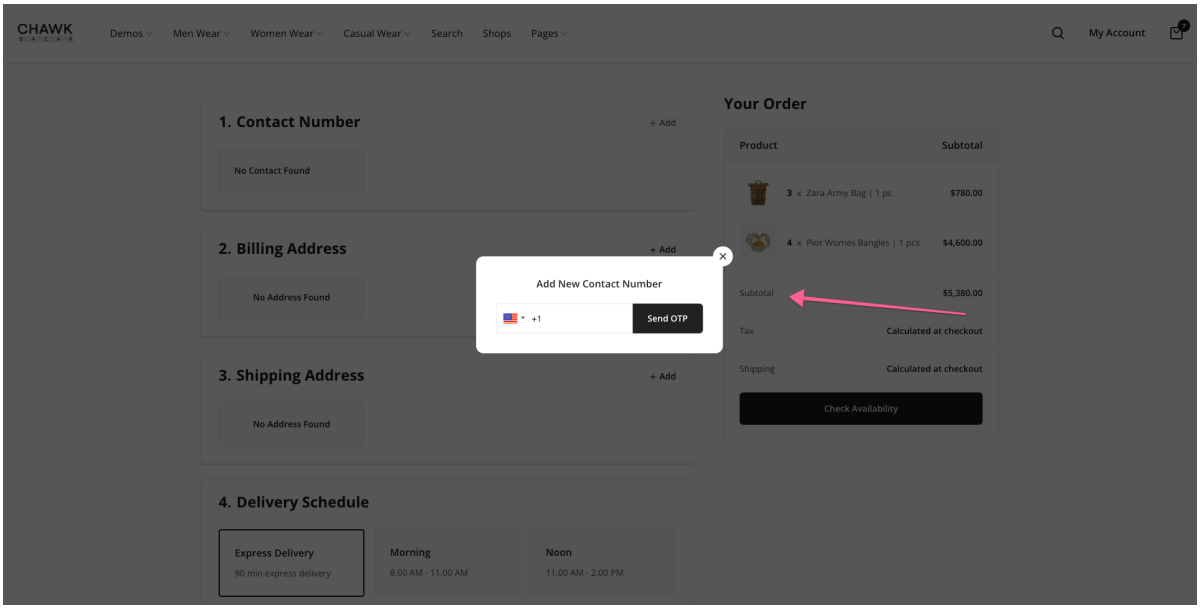




OTP Verification During Checkout Process:

Similarly, when users go to the checkout page, they'll get a form to verify their mobile number.

For the front end, you don't have to do anything. When a user goes to the `checkout` page, they get a phone number box to add their number and verify it.



OTP Verification For Updating Mobile Number:

Users also can change the number from their profile.

Email Verification

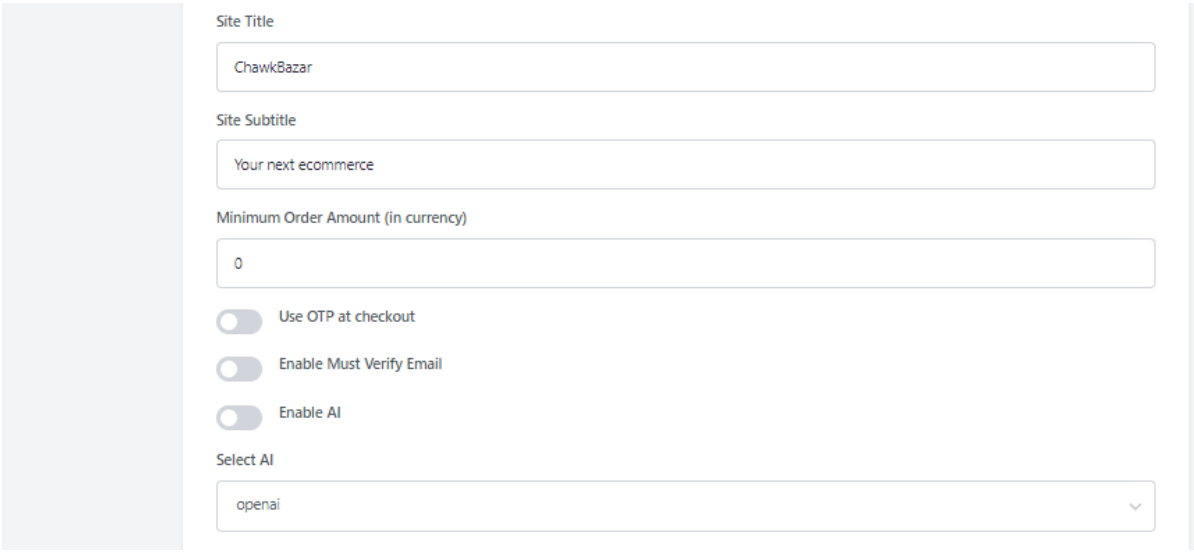
We implement the `Email Verification` feature on Chawkbazar `v5.1.0`. So if you want to use `Email Verification` with `Chawkbazar,` then make sure your Chawkbazar is `v5.1.0` or later.

With the Email Verification feature, you can do,

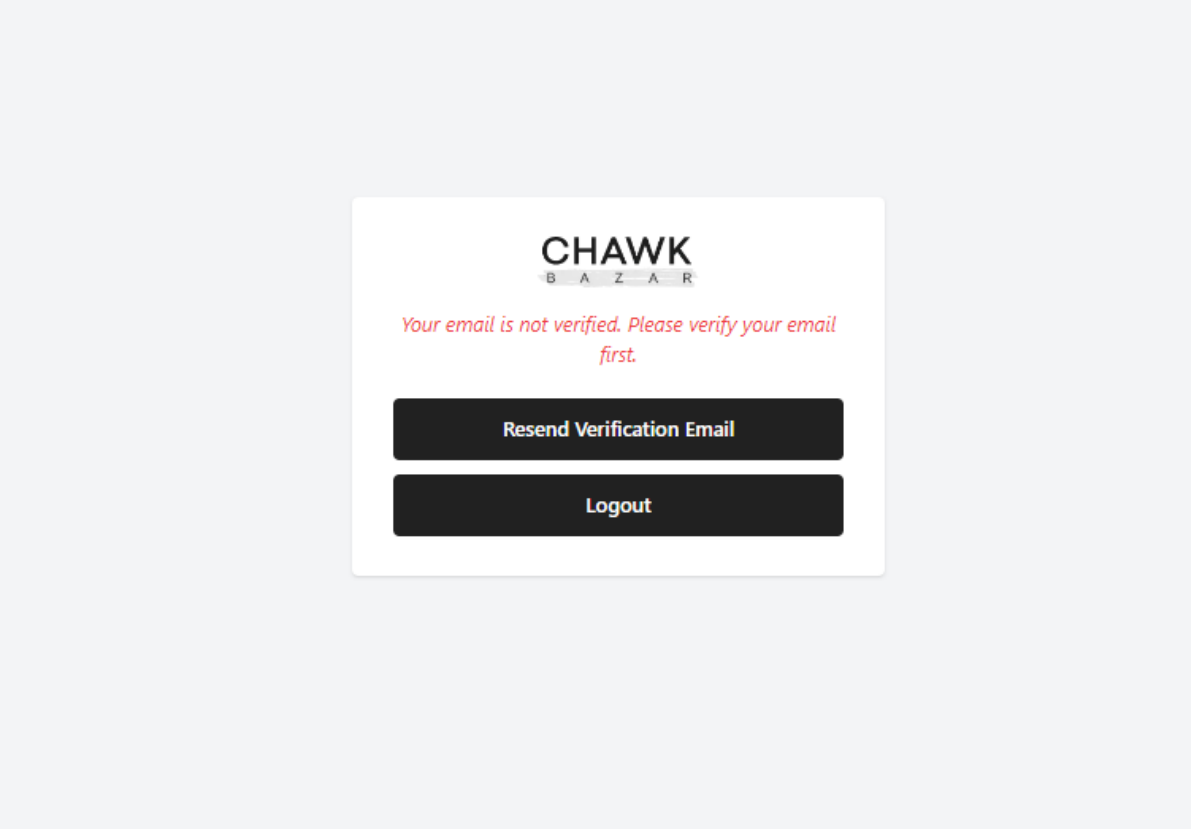
```
- Get Email After Registration
- Email verification for updating Email in profile update section
- When Email verification is on in settings then users will be forced to verify
their email.
```

How to turn on this feature

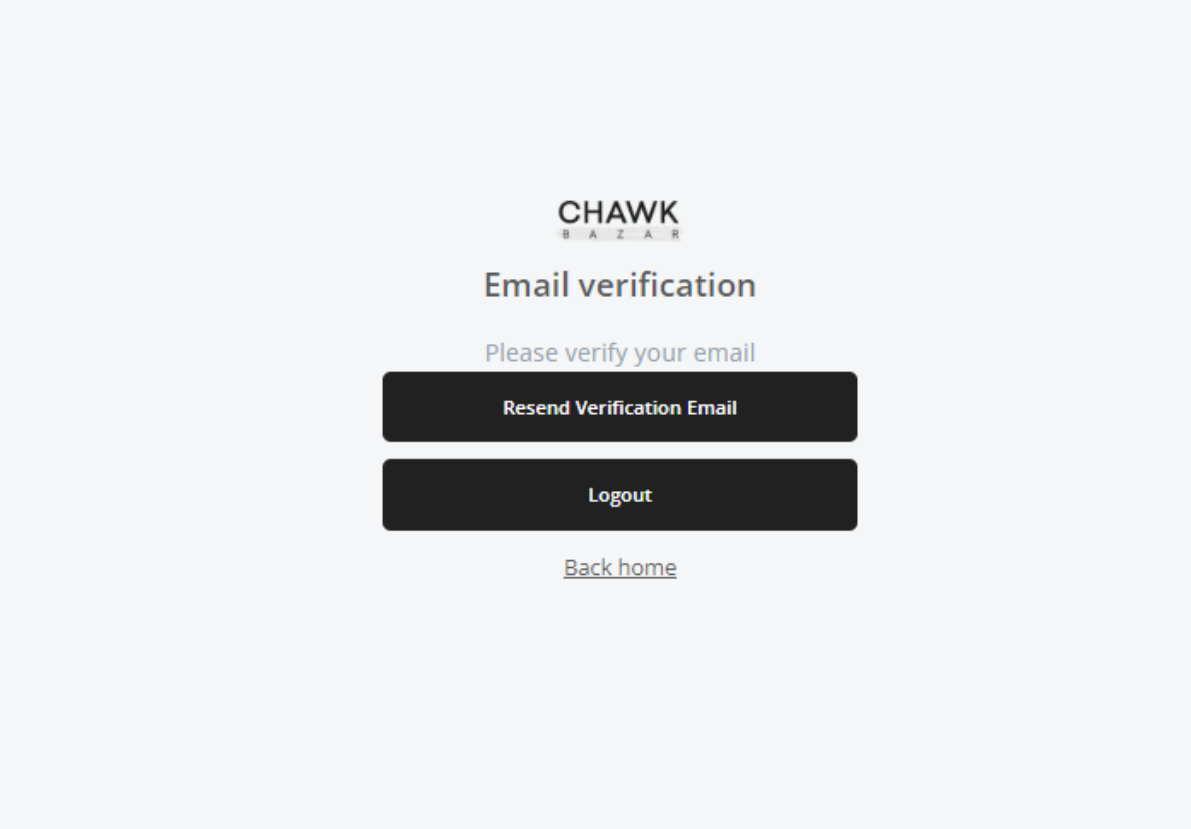
To turn on this feature go to settings of your site then find `Enable Must Verify Email` switch.



Admin view of this feature



Shop view of this feature



Configuration

To use this feature all you need to configure mail in your application.

```
MAIL_MAILER=mailgun
MAILGUN_DOMAIN=
MAILGUN_SECRET=
MAIL_FROM_ADDRESS=
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
ADMIN_EMAIL=support@example.com
```

Payment

We have introduced the new `Payment` architecture feature on ChawkBazar `v3.0.0`. So if you want to use `Payment` with `ChawkBazar`, then make sure your ChawkBazar is `v3.0.0` or later.

We have plan to enrich this feature area by integrating more payment gateways in future. List of available payment gateways now.

- **Stripe**
- **PayPal**
- **Rezorpay**
- **Mollie**
- **Paystack**

Let's discuss those sequentially.

Stripe

Stripe is a financial service & Software as a service (SaaS) company. It offers payment processing software & API for e-commerce applications. In ChawkBazar we have integrated Stripe API for payment system. Though currently `Card` based features are available only, `Stripe Element` for other payment options will be integrate in future updates.

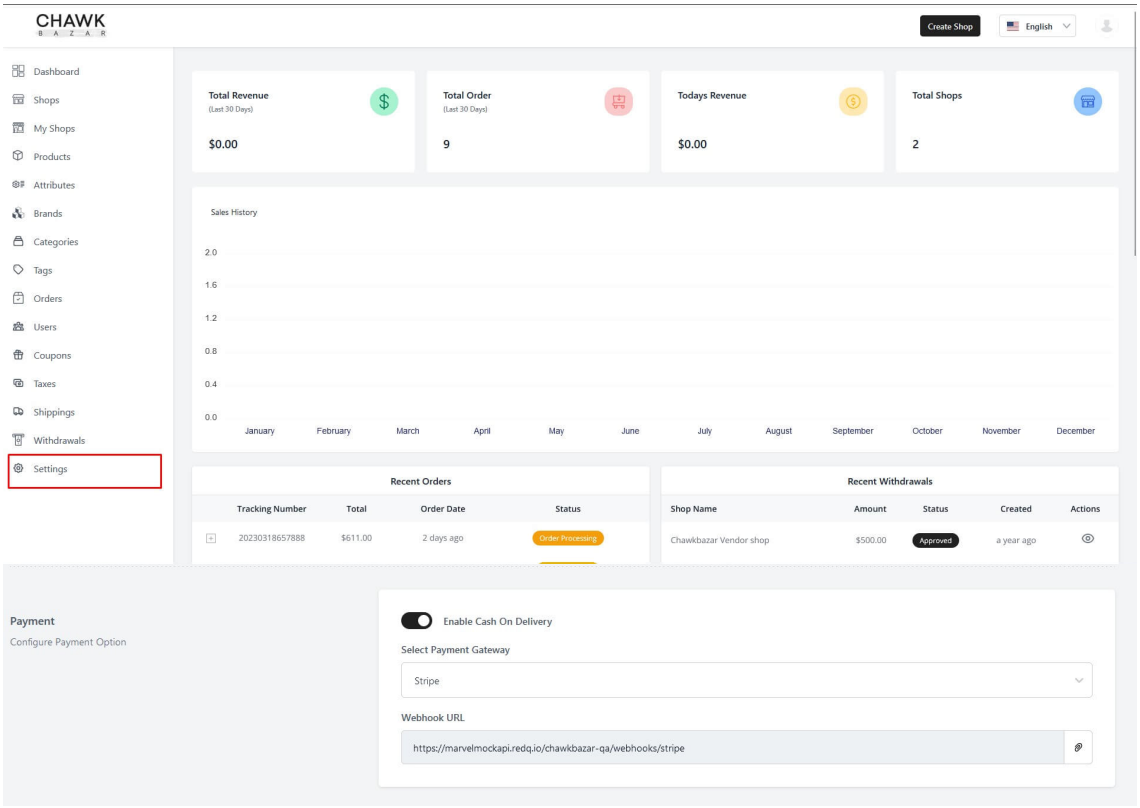
Stripe integrate inside ChawkBazar.

Please follow & complete this steps for stripe integration for your e-commerce system.

- Inside api there is `.env` file. You have to copy & paste this line of codes inside the `.env` file. We will discuss later about how to create those API keys in stripe.

```
STRIPE_API_KEY=[YOUR_STRIPE_API_SECRET_KEY]
STRIPE_WEBHOOK_SECRET_KEY=[YOUR_STRIPE_API_WEBHOOK_SECRET_KEY]
```

- To activate Stripe go to settings from ChawkBazar admin dashboard. Inside settings you will find configure payment option.(e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

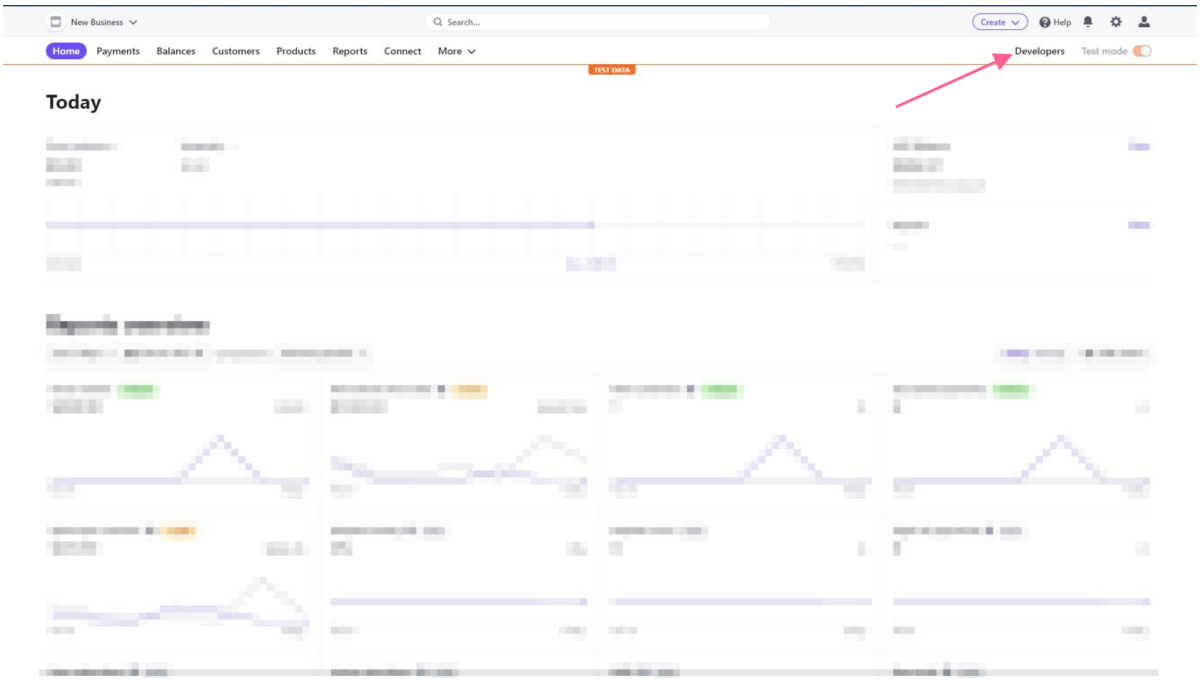


- Add Stripe publishable key inside `.env` file of ChawkBazar shop.

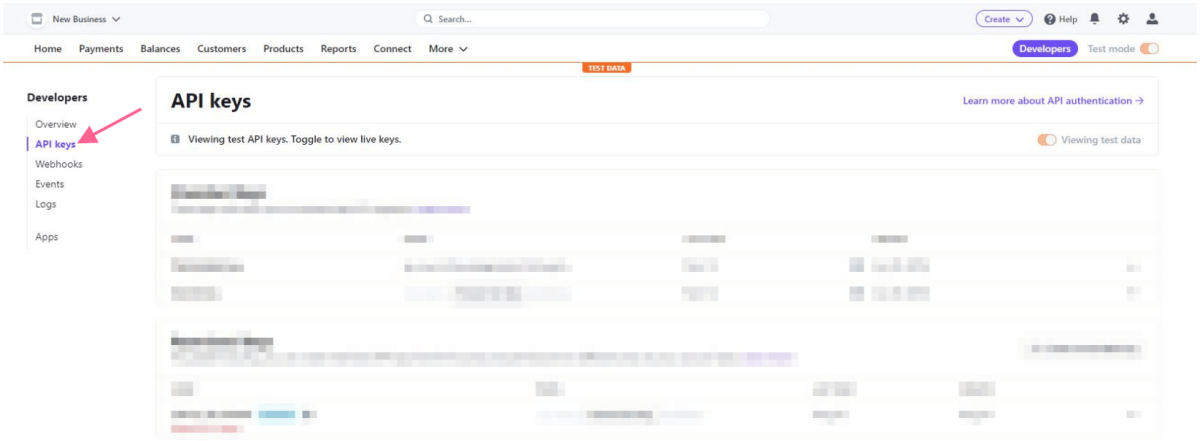
```
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=[YOUR_STRIPE_PUBLISHABLE_KEY]
```

How to create & setup Stripe information properly?

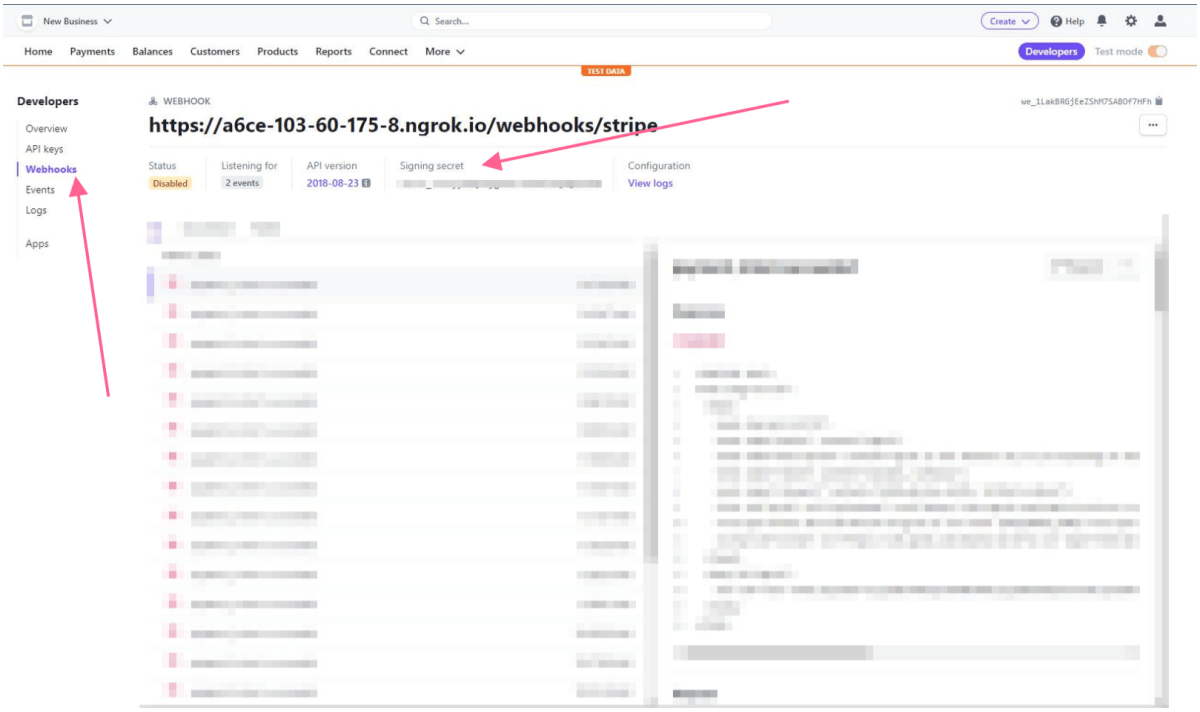
- Go to the stripe website and login https://stripe.com If you aren't a registered user, the complete the stripe registration first.
- After logged in into stripe dashboard, follow the developer link to create the API keys.



- Then create API keys from there. for more details follow up this official documentation https://stripe.com/docs/keys



- Create Webhook secret key if you decide to up & running webhooks in your App.



- Create this two webhook events for monitoring the payment flow.
    - **payment_intent.succeeded**
    - **payment_intent.payment_failed**

Special Notes for Stripe users.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

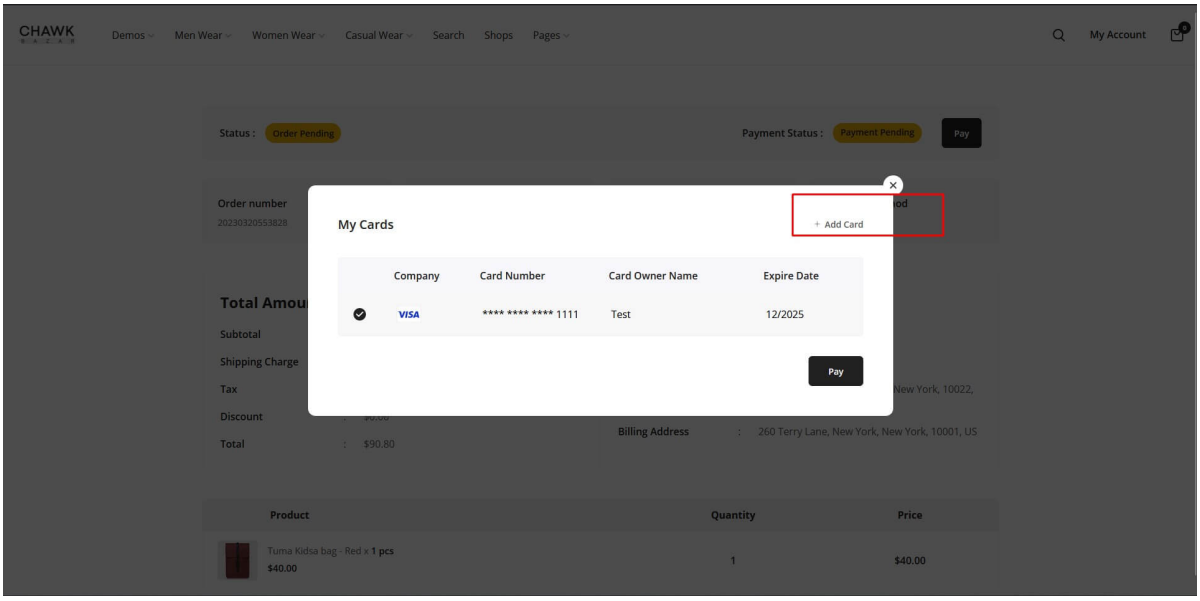
- Stripe DOCS [home page] -- https://stripe.com/docs
- Stripe API Docs -- https://stripe.com/docs/api
- Stripe JavaScript Docs -- https://stripe.com/docs/js

How can I add card in my user profile for future payments in Stripe?

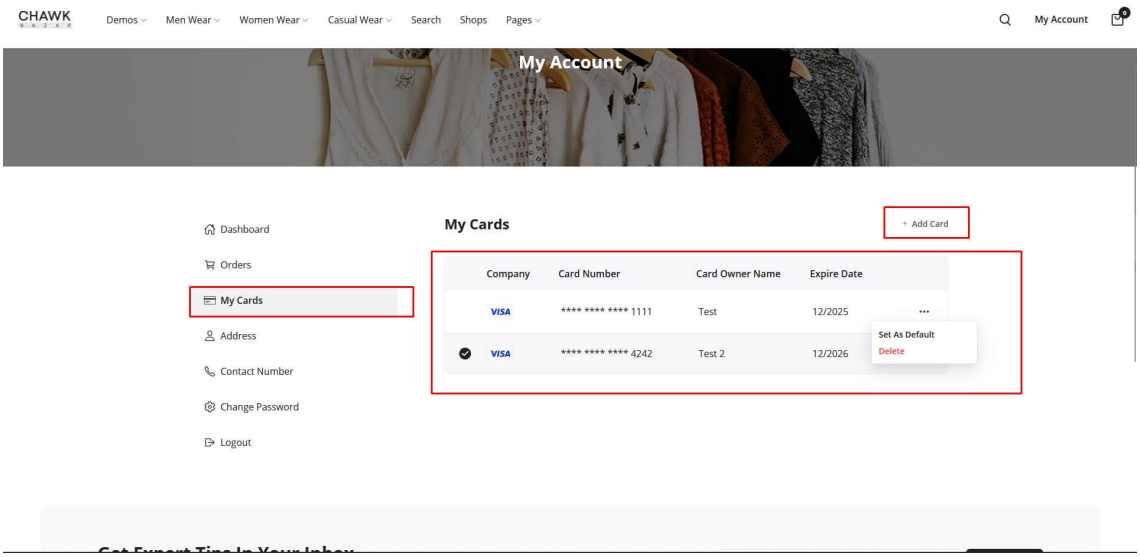
In ChawkBazar we have provided an feature for saving card in case of future usages. This is an `on-session` process. So, if a customer wants to pay via Cards, s/he must have present in the application lively. No `off-session` payment was applied here.

A customer can save a card via two process.

- Save card during checkout process.



- Save card in his user profile `My Card` section.



There are something needs to keep in mind.

Please note

- No confidential information was saved in this features. By maintaing the guideline only available information which are permitted to save via Stripe is implemented here.
- Guest user can't save card for future payment.

PayPal

PayPal is an American multinational financial technology company operating an online payments system. In ChawkBazar we have integrated PayPal APIs which may cover a vast area of PayPal supported region. It will help your business to grow and reach to a wide region.

PayPal integrate inside ChawkBazar.

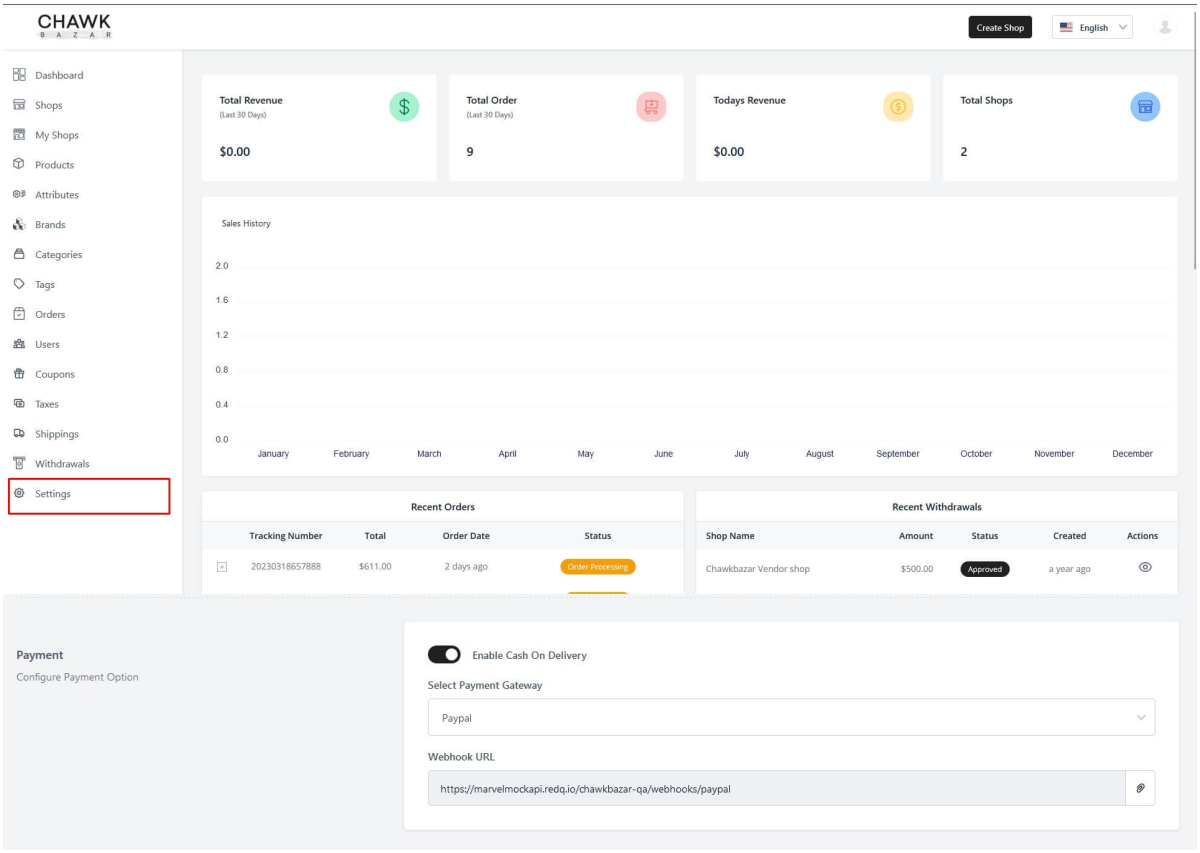
Please follow & complete this steps for PayPal integration for your e-commerce system.

- Inside api there is `.env` file. You have to copy & paste this line of codes inside the `.env` file. We will discuss later about how to create those API keys in PayPal.

```
# Values: sandbox or live (Default: live)
PAYPAL_MODE=sandbox

# Add currency like USD
PAYPAL_CURRENCY=USD

# Change this accordingly for your application.
PAYPAL_NOTIFY_URL=

# force gateway language  i.e. it_IT, es_ES, en_US ... (for express checkout
only)
PAYPAL_LOCALE=en

#Validate SSL when creating api client.
PAYPAL_VALIDATE_SSL=

# PayPal Setting & API Credentials -> sandbox

PAYPAL_SANDBOX_CLIENT_ID=[YOUR_PAYPAL_SANDBOX_CLIENT_ID]
PAYPAL_SANDBOX_CLIENT_SECRET=[YOUR_PAYPAL_SANDBOX_CLIENT_SECRET_KEY]

# PayPal Setting & API Credentials -> live

PAYPAL_LIVE_CLIENT_ID=[YOUR_PAYPAL_LIVE_CLIENT_ID]
PAYPAL_LIVE_CLIENT_SECRET=[YOUR_PAYPAL_LIVE_CLIENT_SECRET_KEY]

# PayPal Webhook settings

PAYPAL_WEBHOOK_ID=[YOUR_PAYPAL_WEBHOOK_URL]
SHOP_URL=[YOUR_SHOP_URL]
```

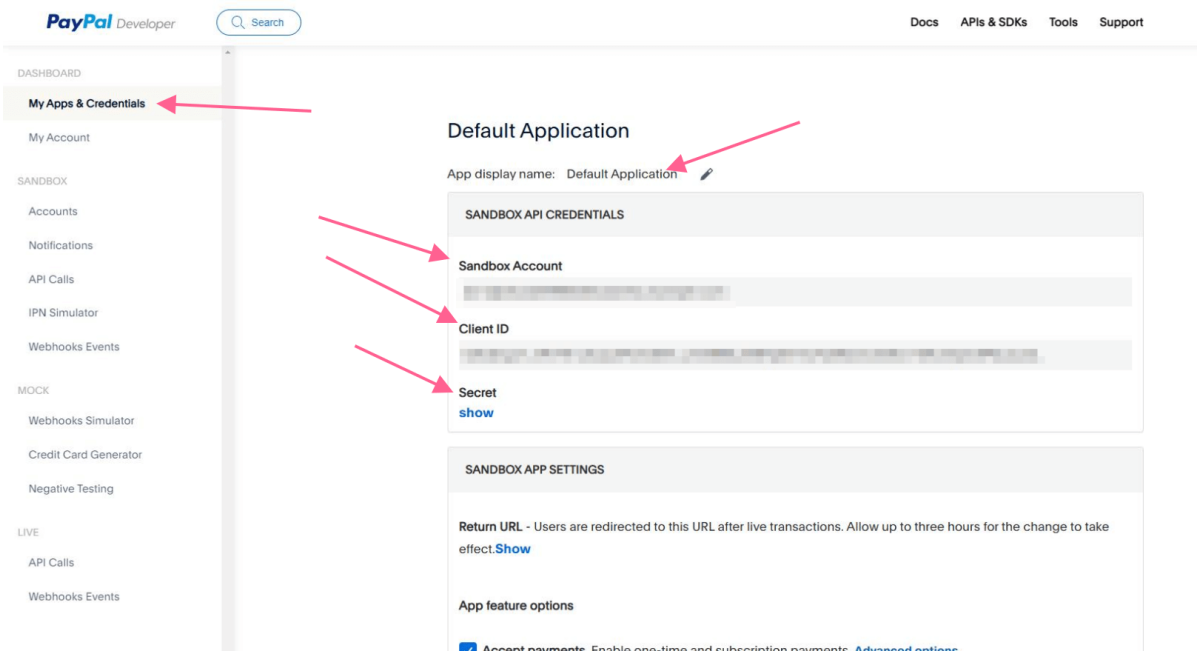
SHOP_URL=[YOUR_SHOP_URL] This parameter is must have in .env file when PayPal is using. Otherwise the payment redirection will be broken.

- To activate Paypal go to settings from ChawkBazar admin dashboard. Inside settings you will find configure payment option. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

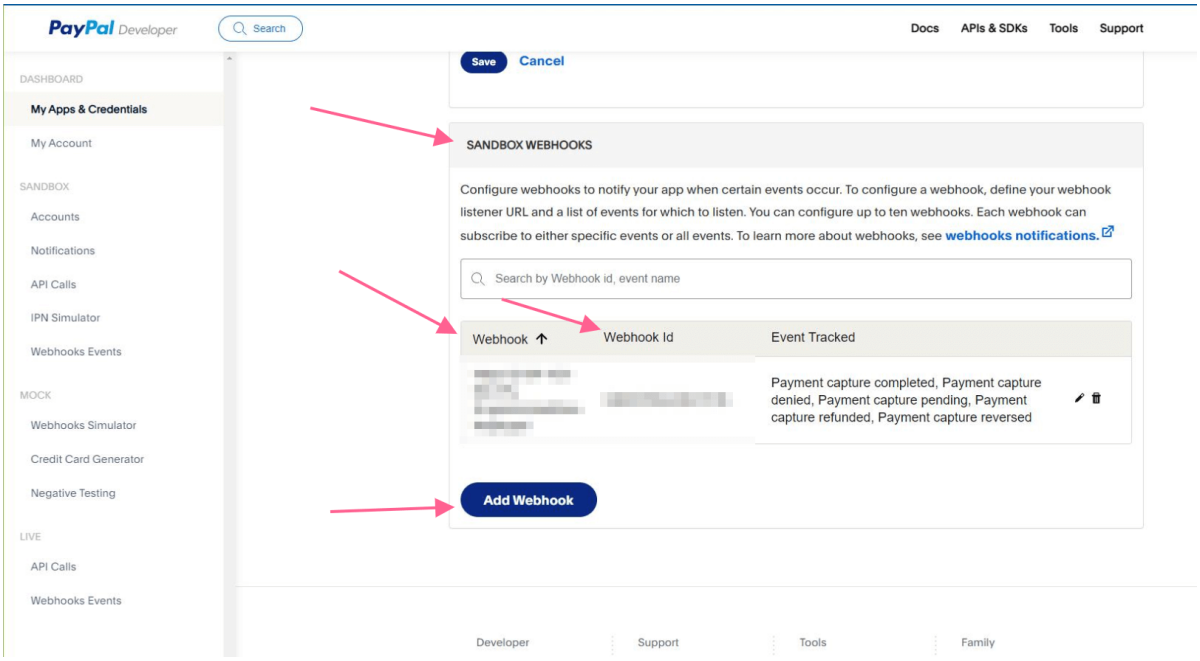


How to create & setup PayPal information properly?

- Create an account in https://developer.paypal.com
- Choose sandbox if your are testing your development environment.
- Click on `Default Applicaiton`. or create a new App.
- You will get your Sandbox API credentials. Copy & paste those inside ChawkBazar app .env file like mentioned above.



- You will find the Webhook ID and link too.



- Then Add the webhook events if you want to up & running the services.
    - **PAYMENT.CAPTURE.COMPLETED**
    - **PAYMENT.CAPTURE.PENDING**
    - **PAYMENT.CAPTURE.CANCELLED**
    - **PAYMENT.CAPTURE.REVERSED**
- At last, For going live with your application please follow this official documentation. https://developer.paypal.com/docs/archive/paypal-here/sdk-dev/going-live/

RazorPay

Razorpay is the only payments solution in India that allows businesses to accept, process and disburse payments with its product suite. It gives you access to all payment modes including credit card, debit card, netbanking, UPI and popular wallets including JioMoney, Mobikwik, Airtel Money, FreeCharge, Ola Money and PayZapp.

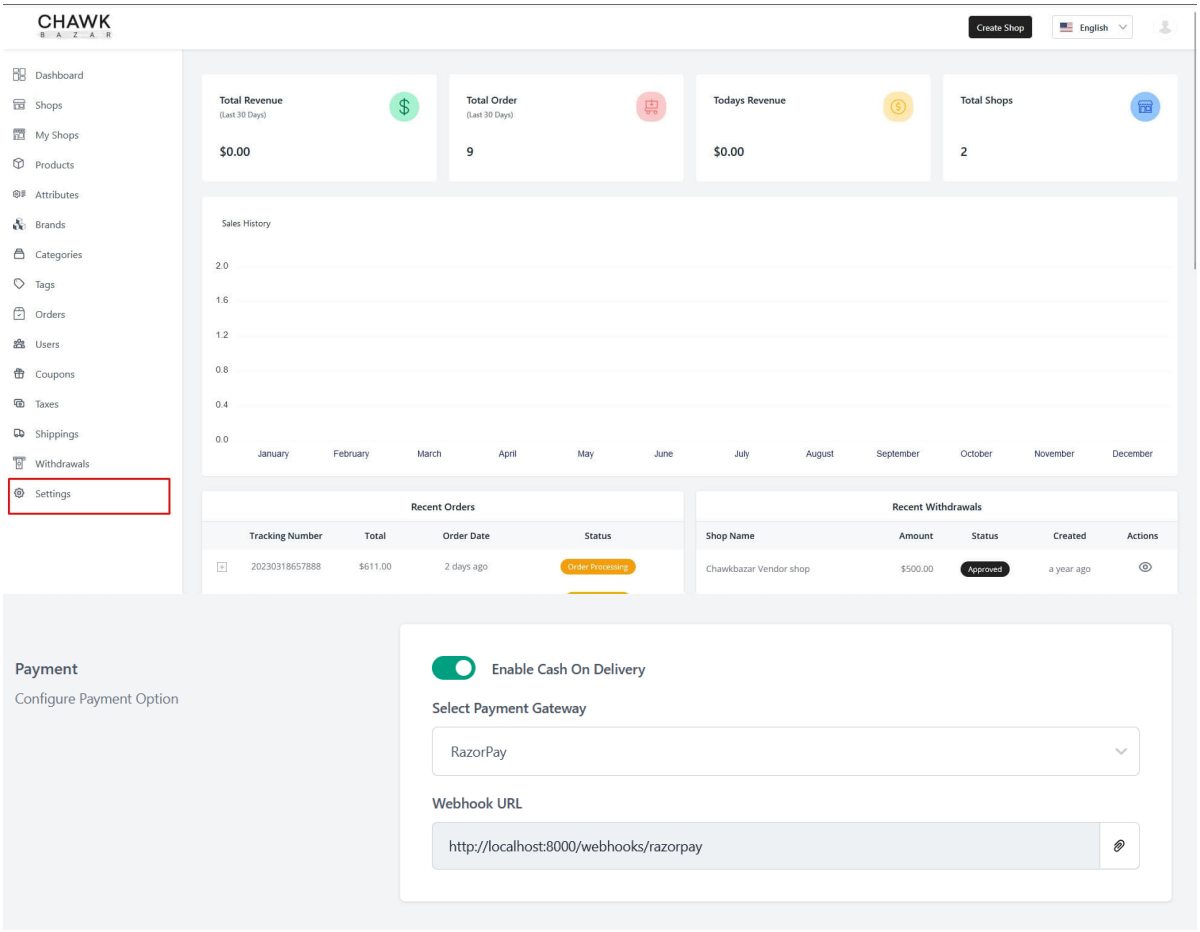
RazorPay integration inside PickBazar.

Please follow & complete this steps for RazorPay integration with your e-commerce system.

- Inside api there is `.env` file. You have to copy & paste this line of codes inside the `.env` file. We will discuss later about how to create those API keys in RazorPay.

```
RAZORPAY_KEY_ID=[YOUR_RAZORPAY_KEY_ID]
RAZORPAY_KEY_SECRET=[YOUR_RAZORPAY_KEY_SECRET]
RAZORPAY_WEBHOOK_SECRET_KEY=[YOUR_RAZORPAY__WEBHOOK_URL]
```

- Activate RazorPay from PickBazar admin dashboard. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

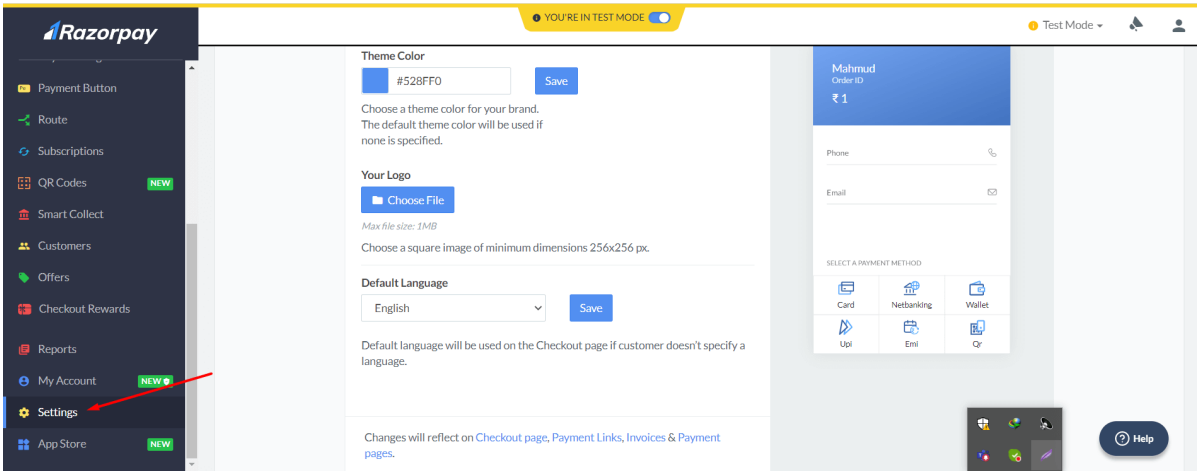


- Add RazorPay publishable key inside `.env` file of pickbazar shop.

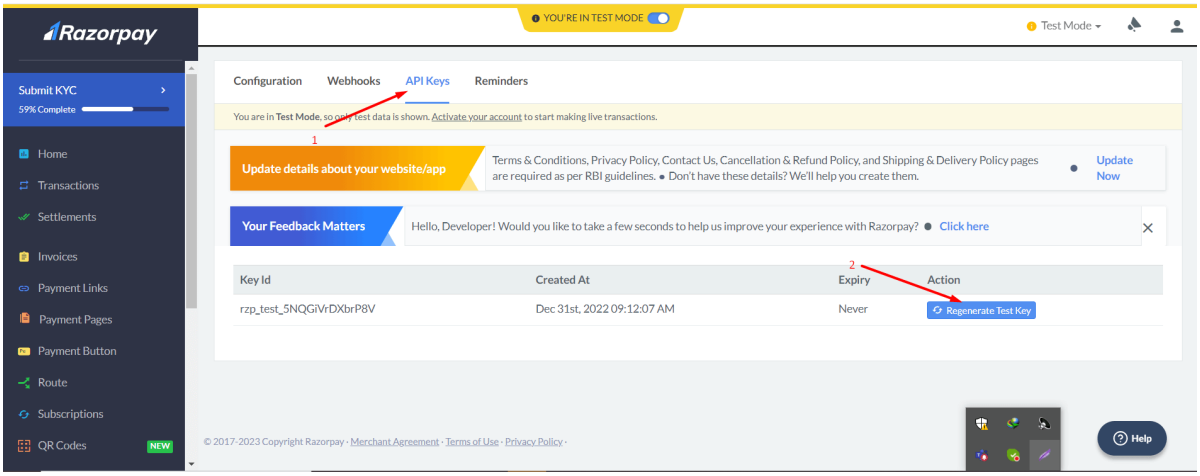
```
NEXT_PUBLIC_RAZORPAY_KEY=[YOUR_RAZORPAY_KEY_ID]
```

How to create & setup RazorPay information properly?

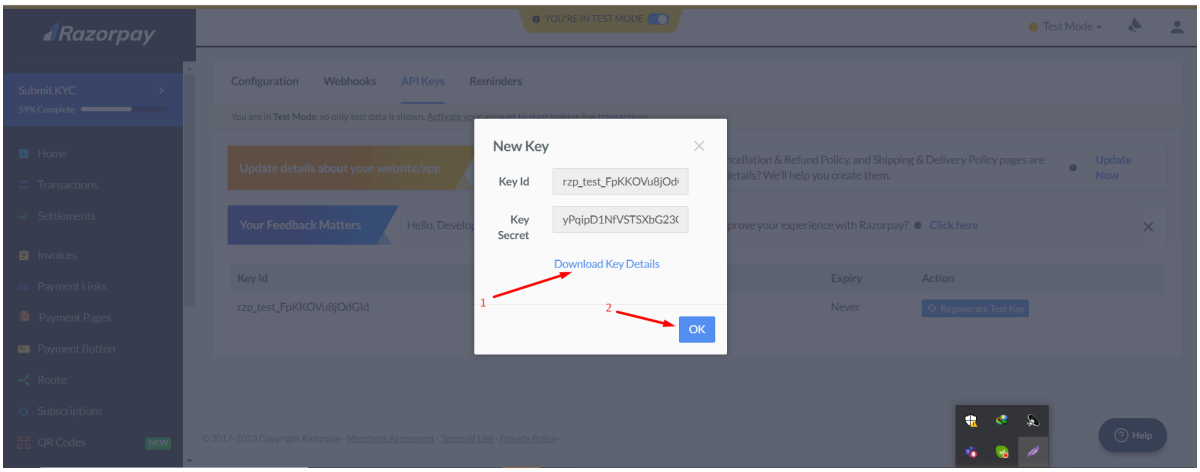
- Go to the RazorPay website & login https://razorpay.com If you aren't a registered user, Complete the RazorPay registration first.
- After logged in into RazorPay dashboard, Click on Settings



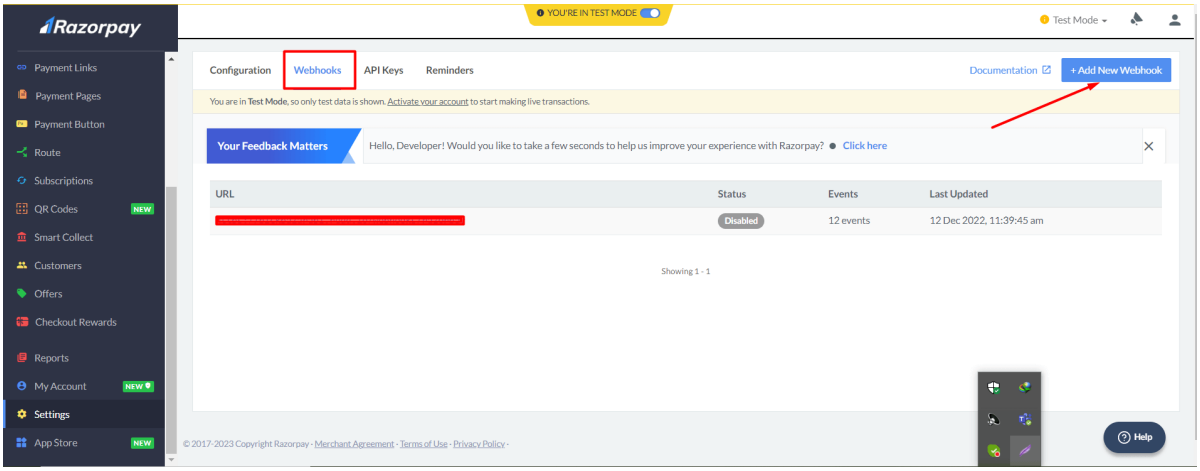
- Select API Keys option & click on Generate Test Key.



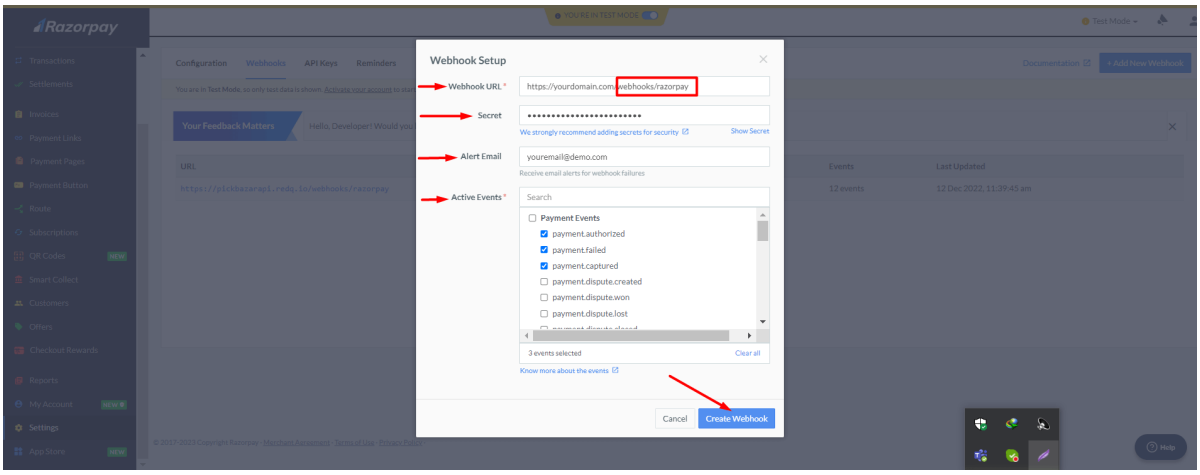
- The Client Id and Secret Key will appear. Download these keys and click on OK



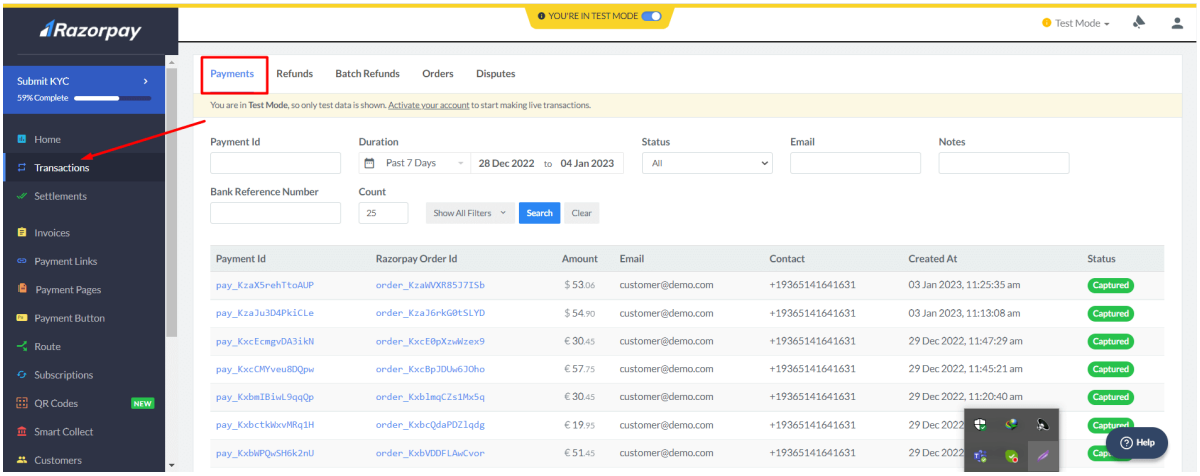
- Select Webhooks option & click on Add New Webhooks.



- Enter Webhook URL , Secret key , Alert Email and tick mark the checkbox in Active Event then click on create Webhook



- Copy & paste those inside PickBazar app .env file like mentioned above. after that test your RazorPay Payment Status



Special Notes for RazorPay.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

- RazorPay DOCS -- https://razorpay.com/docs/#home-payments
- RazorPay Webhooks -- https://razorpay.com/docs/webhooks/setup-edit-payments

Mollie

Mollie is a Payment Service Provider (PSP) that processes online payments for companies. If you buy something online from one of our merchants, we make sure that your money is transferred safely from your bank to the merchant's bank. Since we arrange the payment process, you may see Mollie or Stg Mollie Payments on your bank statement.

Mollie integrate inside PickBazar.

Please follow & complete this steps for Mollie integration for your e-commerce system.

- Inside api there is `.env` file. You have to copy & paste this line of codes inside the `.env` file. We will discuss later about how to create those API keys in Mollie.

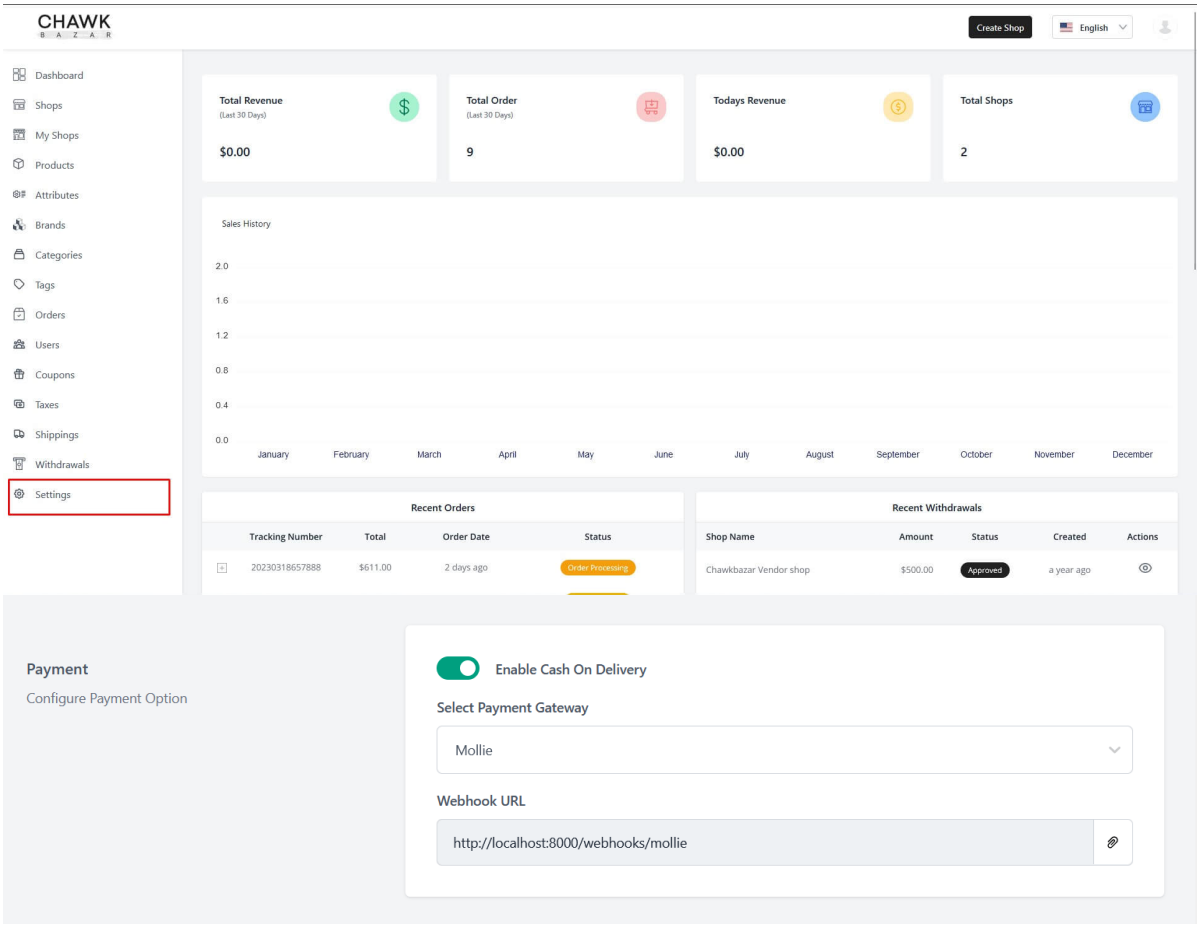
```
MOLLIE_KEY=[YOUR_MOLLIE_API_KEY]
```

Mollie Webhook settings

```
SHOP_URL=[YOUR_SHOP_URL]
MOLLIE_WEBHOOK_URL=[YOUR_MOLLIE_WEBHOOK_URL]
```

SHOP_URL=[YOUR_SHOP_URL] This parameter is must have in .env file when Mollie is using. Otherwise the payment redirection will be broken.

- To activate Mollie go to settings from PickBazar admin dashboard. Inside settings you will find configure payment option. (e.g. Mollie Webhook URL is coming from local development. This static link will dynamically generated in live environment)



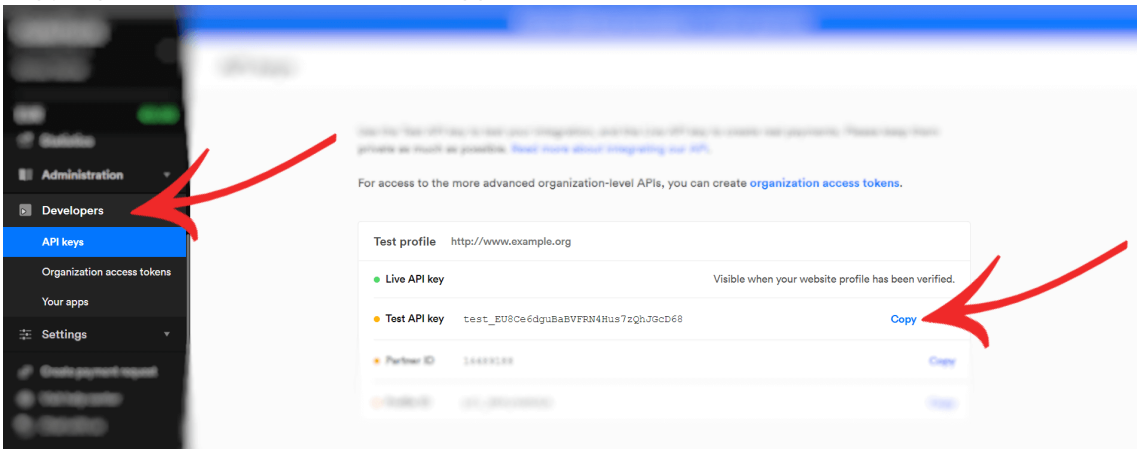
- Copy & paste those inside PickBazar app .env file like mentioned above.
- If you want to use webhook during development on localhost, you must use a tool like ngrok to have the webhooks delivered to your local machine.

- Copy the forwarding https link & paste .env file after link add line- /webhooks/mollie

```
# Payment -> Mollie
MOLLIE_KEY=test_EU8Ce6dguBaBVFRN4Hus7zQhJGcD68
MOLLIE_WEBHOOK_URL=https://c852-37-111-243-195.in.ngrok.io/webhooks/mollie
```

How to create & setup Mollie information properly?

- Go to the Mollie website & login https://www.mollie.com If you aren't a registered user, the complete the Mollie registration first.
- After logged in into Mollie dashboard, follow the developer link to copy the API keys.
- Choose Test API key your are testing your development environment.
- Copy & paste those inside PickBazar app .env file like mentioned above.



Special Notes for Mollie.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

- Mollie DOCS -- https://docs.mollie.com/
- Mollie Webhooks -- https://docs.mollie.com/overview/webhooks

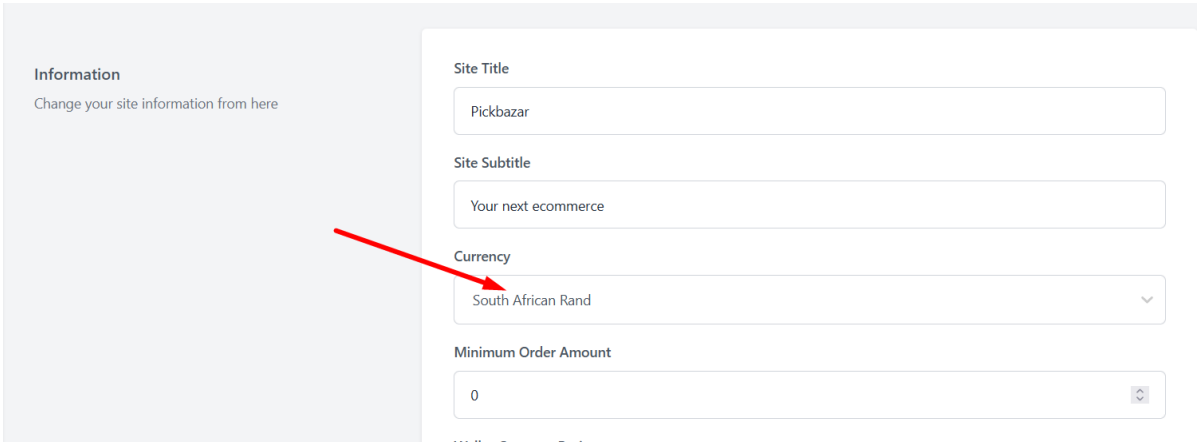
Paystack

Paystack is a Nigerian fintech company that provides online payment solutions for businesses. It offers a range of services, including online and offline payments, subscriptions, and invoicing. Paystack's platform enables businesses to easily and securely accept payments from customers via credit card, debit card, and mobile money.

Paystack integrate inside PickBazar.

Please follow & complete this steps for Paystack integration for your e-commerce system.

- First go to settings from PickBazar admin dashboard. Inside settings you will find Currency option.Select your Currency. If you want to Test development on localhost.then Select South African Rand(ZAR).



Here is a breakdown of countries and the currencies they can accept payments in.

Country	Available Currencies
Ghana	GHS
Nigeria	NGN, USD
South Africa	ZAR

For more information: Paystack currency is available.

- Inside api there is `.env` file. You have to copy & paste this line of codes inside the `.env` file. We will discuss later about how to create those API keys in Paystack official website.

```
PAYSTACK_PUBLIC_KEY=[YOUR_PAYSTACK_PUBLIC_KEY]
PAYSTACK_SECRET_KEY=[YOUR_PAYSTACK_SECRET_KEY]
PAYSTACK_PAYMENT_URL=https://api.paystack.co
MERCHANT_EMAIL=[YOUR_PAYSTACK_MERCHANT_EMAIL]
```

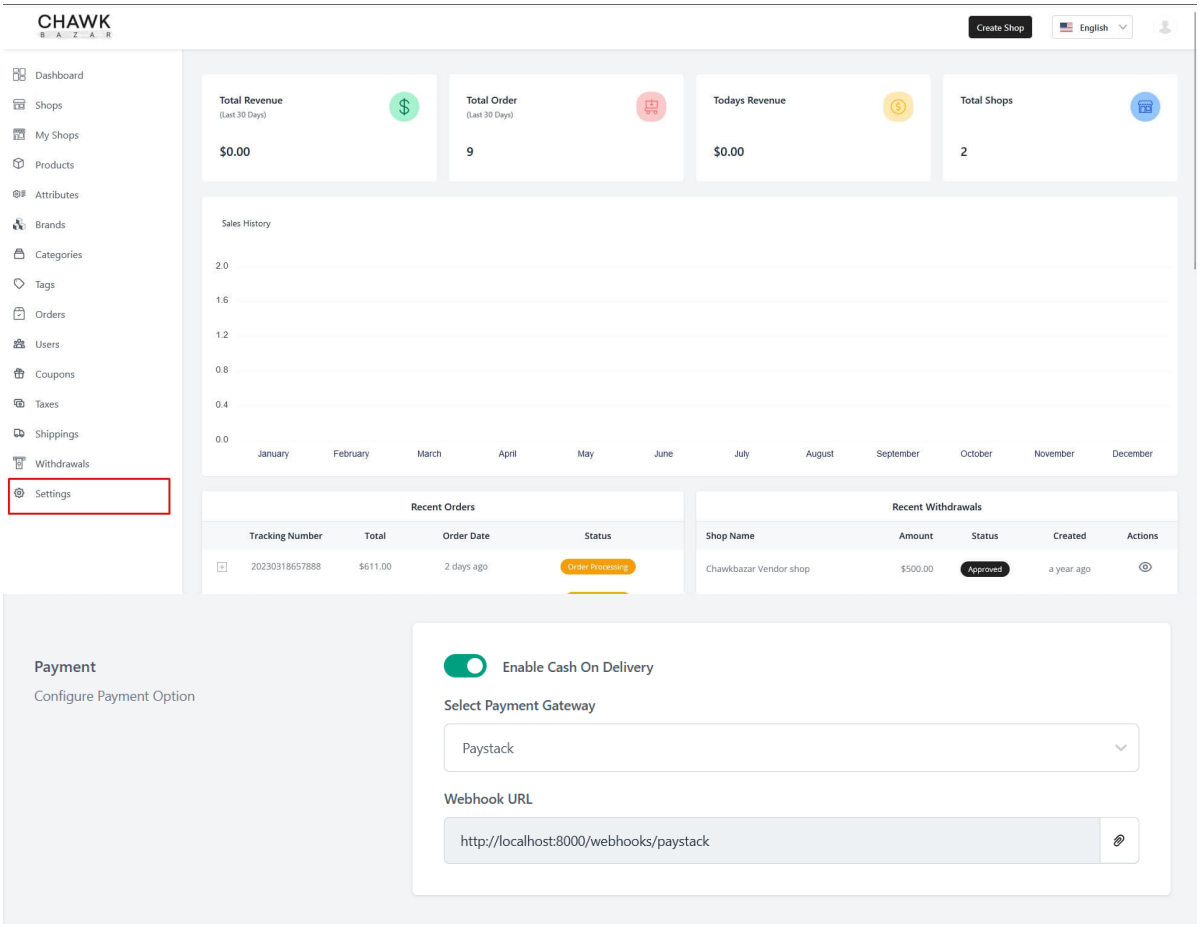
- Add Paystack publishable key inside pickbazar shop.

```
NEXT_PAYSTACK_PUBLIC_KEY=[YOUR_YOUR_PAYSTACK_PUBLIC_KEY]
```

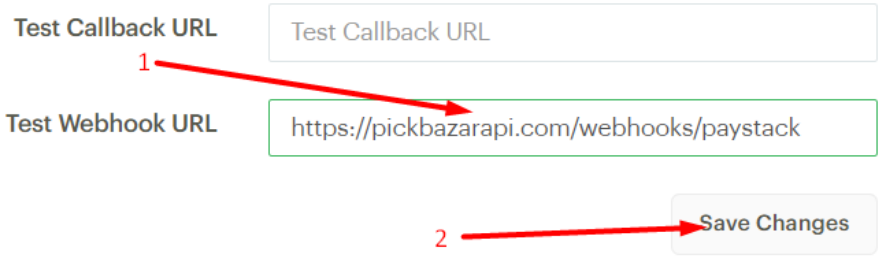
SHOP_URL=[YOUR_SHOP_URL] This parameter is must have in .env file when Paystack is using. Otherwise the payment redirection will be broken.

Paystack Webhook settings

- To activate Paystack go to settings from PickBazar admin dashboard. Inside settings you will find configure payment option. (e.g. Paystack Webhook URL is coming from local development. This static link will dynamically generated in live environment)

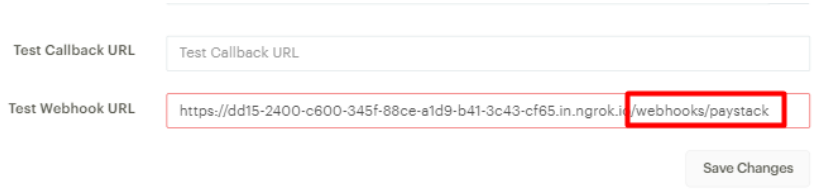


- Copy the webhook url & registered in Paystack Dashboard.



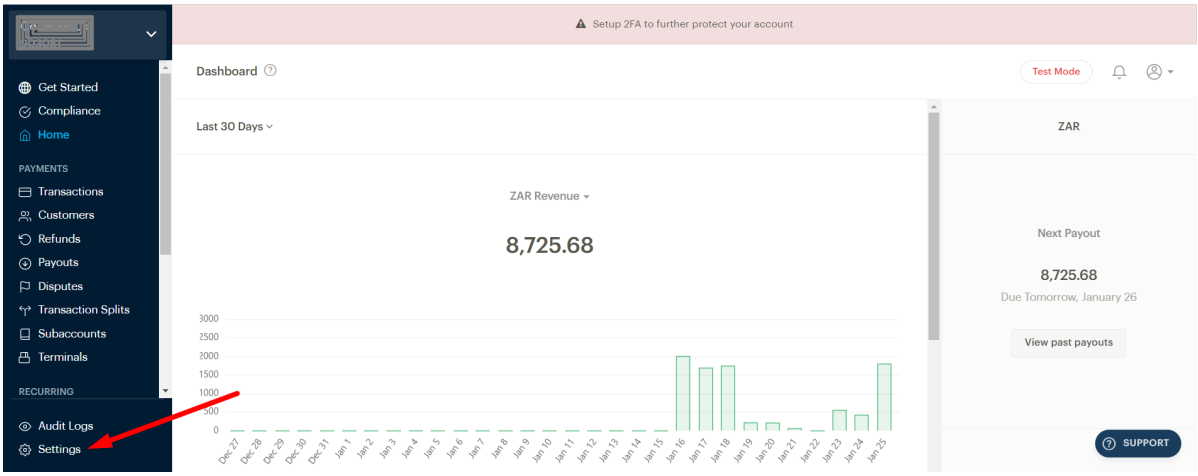
- If you want to use webhook during development on localhost, you must use a tools like ngrok to have the webhooks delivered to your local machine.

- Copy the forwarding https link & paste on Test Webhook URL, after paste forwarding https link, add line- /webhooks/paystack.

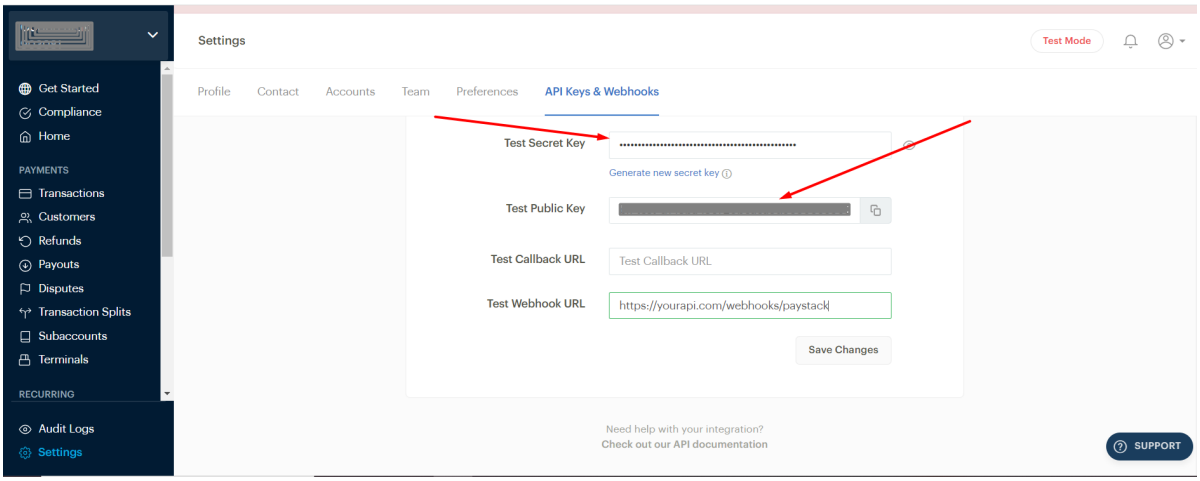


How to create & setup Paystack information properly?

- Go to the [Paystack official website] & login If you aren't a registered user, Complete the Paystack registration first.
- After logged in into Paystack dashboard, Click on Settings.



- After That Click on API Keys & Webhooks for Paystack Secret Key & Paystack Public Key.



- Copy & paste those inside PickBazar API and PickBazar Shop `.env` file like mentioned above.

Special Notes for Paystack.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

- Paystack official documentation
- Paystack Webhooks documentation
- Accept Payments in US Dollars (USD)

Integratioin of new payment gateway

To integrate a payment gateway in ChawkBazar-Laravel, you will need to follow these general steps:

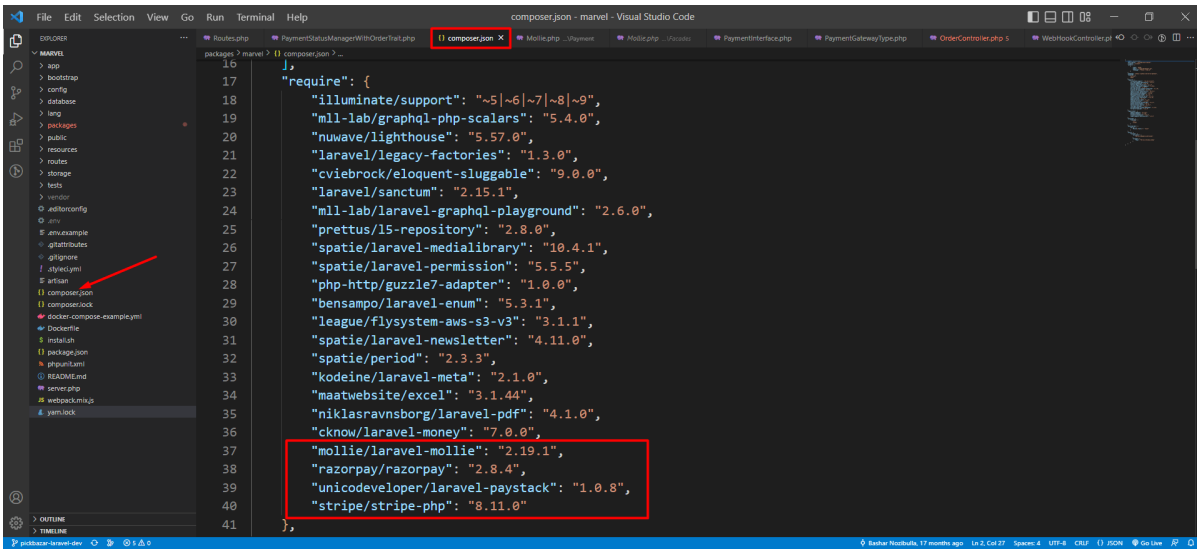
Getting Started with API

Step 1: Install and configure the payment gateway package

First, you will need to install the payment gateway package for Laravel (if there are any available package). There are several packages available that provide integration with various payment gateways. Example is given in the screenshot.

Once you have chosen and installed a package, you will need to configure it by adding your payment gateway credentials and any other required settings.

To check the installed dependencies of payment gateway in ChawkBazar-Laravel, you can open the composer.json file in a text editor and find that.



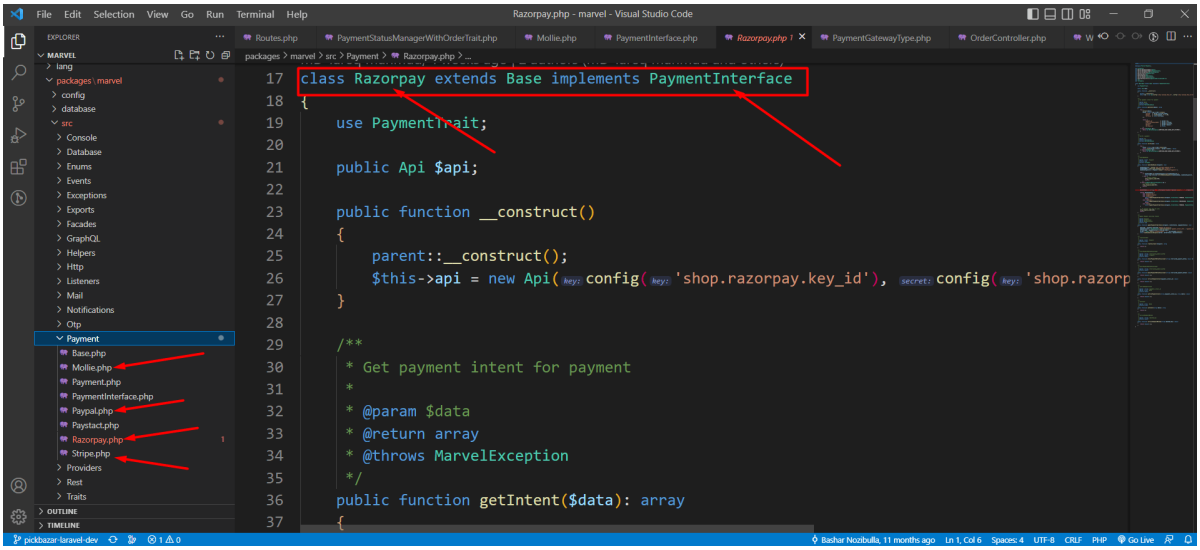
Step 2: Add payment gateway name in the Enum.

Add PaymentGateway Enum `API -> package -> marvel -> src -> Enums -> PaymentGatewayType.php`

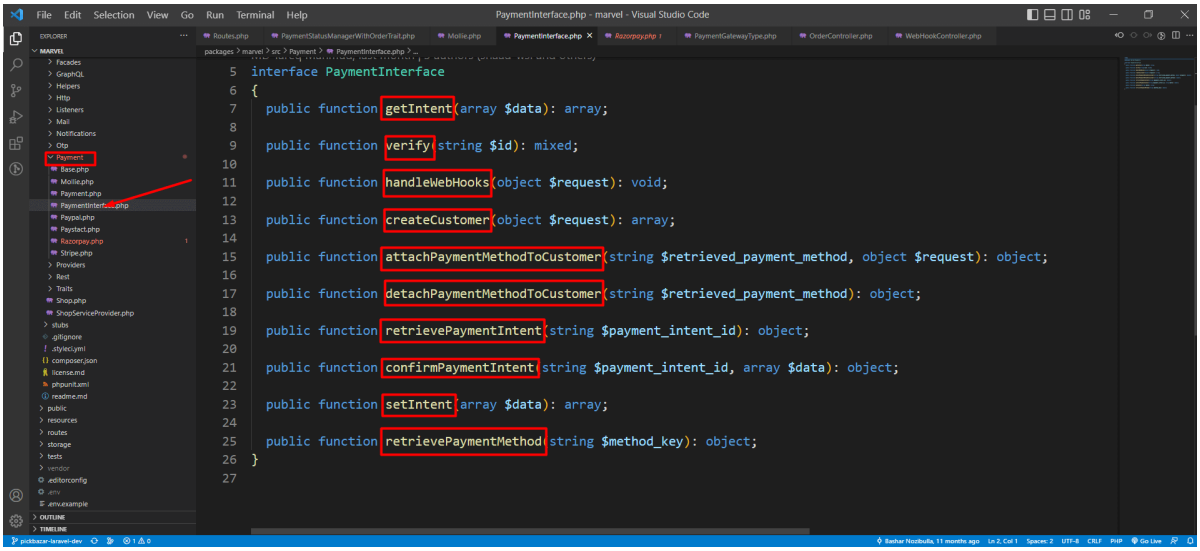
```php
final class PaymentGatewayType extends Enum
{
    public const STRIPE = 'STRIPE';
    public const CASH_ON_DELIVERY = 'CASH_ON_DELIVERY';
    public const MOLLIE = 'MOLLIE';
    public const CASH = 'CASH';
    public const RAZORPAY = 'RAZORPAY';
    public const FULL_WALLET_PAYMENT = 'FULL_WALLET_PAYMENT';
    public const PAYPAL = 'PAYPAL';
}
```

Step 3: Configure the Payment Facade for the new payment gateway.

Now go to `API -> package -> marvel -> src -> Payment` then create new payment Class (e.g. Stripe, PayPal, Razorpay, Mollie) for implements the PaymentInterface.The Class must implements all of the methods defined in the PaymentInterface.



Methods defined in the PaymentInterface



Here's an example of a class that implements that PaymentInterface:

```php
<?php

namespace Marvel\Payments;

use Exception;
use Marvel\Database\Models\Order;
use Marvel\Database\Models\PaymentIntent;
use Marvel\Exceptions\MarvelException;
use Marvel\Traits\PaymentTrait;
use Razorpay\Api\Api;
use Marvel\Enums\OrderStatus;
use Marvel\Enums\PaymentStatus;
use Razorpay\Api\Errors\SignatureVerificationError;
use Str;
use Throwable;

class Razorpay extends Base implements PaymentInterface
{
    use PaymentTrait;

    public Api $api;

    public function __construct()
    {
        parent::__construct();
        $this->api = new Api(config('shop.razorpay.key_id'), config('shop.razorpay.key_secret'));
    }

    /**
     * Get payment intent for payment
     *
     * @param $data
     * @return array
     * @throws MarvelException
     */
    public function getIntent($data): array
    {
        try {
            extract($data);
            $order = $this->api->order->create([
                'receipt'  => $order_tracking_number,
                'amount'   => round($amount, 2) * 100,
                'currency' => $this->currency,
            ]);

            return [
                'payment_id'            => $order->id,
                'order_tracking_number' => $order->receipt,
                'currency'              => $order->currency,
                'amount'                => $order->amount,
                'is_redirect'           => false,
            ];
        } catch (Exception $e) {
            throw new MarvelException(SOMETHING_WENT_WRONG_WITH_PAYMENT);
        }
    }

    /**
     * Verify a payment
     *
     * @param $id
     * @return false|mixed
     * @throws MarvelException
     */
    public function verify($id): mixed
    {
        try {
            $order = $this->api->order->fetch($id);
            return isset($order->status) ? $order->status : false;
        } catch (Exception $e) {
            throw new MarvelException(SOMETHING_WENT_WRONG_WITH_PAYMENT);
        }
    }

    /**
     * handleWebHooks
     *
     * @param  mixed  $request
     * @return void
     * @throws Throwable
     */
    public function handleWebHooks($request): void
    {
        $webhookSecret = config('shop.razorpay.webhook_secret');
        $webhookBody = @file_get_contents('php://input');
        $webhookSignature = $request->header('X-Razorpay-Signature');

        try {
            if ($webhookBody && $webhookSignature && $webhookSecret) {
                $this->api->utility->verifyWebhookSignature($webhookBody, $webhookSignature, $webhookSecret);
            } else {
                // Invalid request
                http_response_code(400);
                exit();
            }
        } catch (SignatureVerificationError $e) {
            // Invalid signature
            http_response_code(400);
            exit();
```

```
        }

        $eventStatus = (string) Str::of($request->event)->replace('payment.', '', $request->event);

        switch ($eventStatus) {
            case 'dispute.won':
            case 'dispute.created':
            case 'authorized':
                $this->updatePaymentOrderStatus($request, OrderStatus::PENDING, PaymentStatus::PROCESSING);
                break;
            case 'captured':
                $this->updatePaymentOrderStatus($request, OrderStatus::PROCESSING, PaymentStatus::SUCCESS);
                break;
            case 'failed':
                $this->updatePaymentOrderStatus($request, OrderStatus::PENDING, PaymentStatus::FAILED);
        }

        http_response_code(200);
    }

    /**
     * Update Payment and Order Status
```

*** Note : *** It is important to note that each payment gateway has its own set of requirements and may have different methods for processing payments. You will need to follow the official documentation for the specific payment gateway.

Step 4: Use the payment gateway for submitting the order.

Go to `API -> package -> marvel -> src -> Http -> Controllers -> OrderController.php -> submitPayment` then you've to add your PaymentGatewayType and function name in switch case.

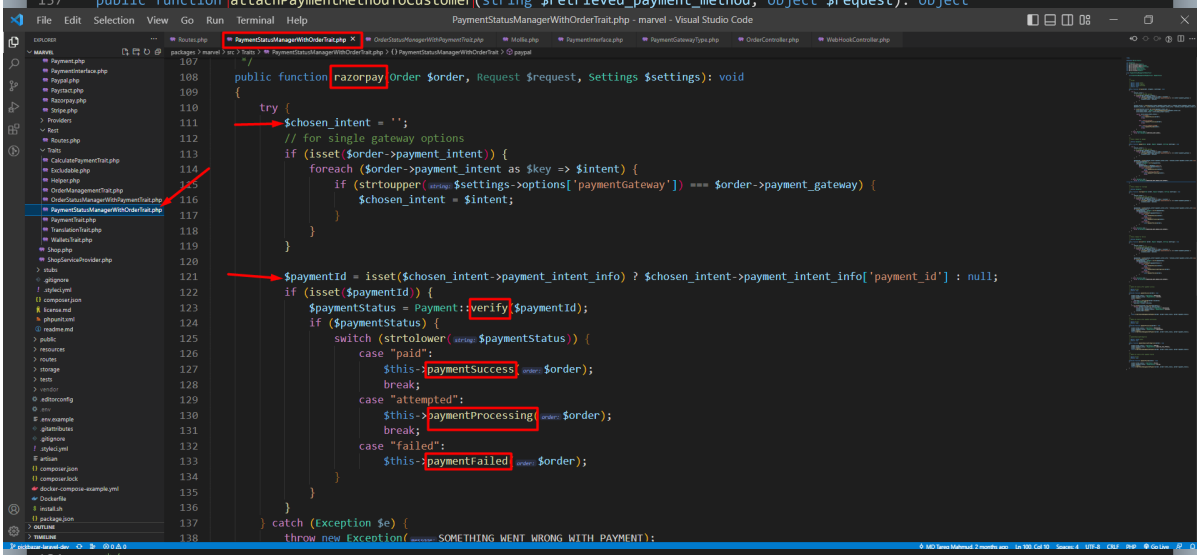
```
packages > marvel > src > Http > Controllers > OrderController.php > OrderController > submitPayment
     * @throws Exception
     */
    public function submitPayment(Request $request): void
    {
        $tracking_number = $request->tracking_number ?? null;
        try {
            $order = $this->repository->with(relations: ['products', 'children.shop', 'wallet_point', 'payment_intent'])
                ->findOneByFieldOrFail(field: 'tracking_number', value: $tracking_number);

            switch ($order->payment_gateway) {
                case PaymentGatewayType::STRIPE:
                    $this->stripe(order: $order, request: $request, settings: $this->settings);
                    break;
                case PaymentGatewayType::PAYPAL:
                    $this->paypal(order: $order, request: $request, settings: $this->settings);
                    break;
                case PaymentGatewayType::MOLLIE:
                    $this->mollie(order: $order, request: $request, settings: $this->settings);
                    break;
                case PaymentGatewayType::RAZORPAY:
                    $this->razorpay(order: $order, request: $request, settings: $this->settings);
                    break;
```

Gateway Type

function

After That go to `API -> package -> marvel -> src -> Traits -> PaymentStatusManagerWithOrderTrait.php` for verify your payment status update, you've to add function like Stripe, PayPal, Razorpay or Mollie.

```
    /**
     * attachPaymentMethodToCustomer
     *
     * @param string $retrieved_payment_method
     * @param object $request
     * @return object
     */
    public function attachPaymentMethodToCustomer(string $retrieved_payment_method, object $request): object
```




```
     */
    public function razorpay(Order $order, Request $request, Settings $settings): void
    {
        try {
            $chosen_intent = '';
            // for single gateway options
            if (isset($order->payment_intent)) {
                foreach ($order->payment_intent as $key => $intent) {
                    if (strtoupper(string: $settings->options['paymentGateway']) === $order->payment_gateway) {
                        $chosen_intent = $intent;
                    }
                }
            }

            $paymentId = isset($chosen_intent->payment_intent_info) ? $chosen_intent->payment_intent_info['payment_id'] : null;
            if (isset($paymentId)) {
                $paymentStatus = Payment::verify($paymentId);
                if ($paymentStatus) {
                    switch (strtolower(string: $paymentStatus)) {
                        case "paid":
                            $this->paymentSuccess(order: $order);
                            break;
                        case "attempted":
                            $this->paymentProcessing(order: $order);
                            break;
                        case "failed":
                            $this->paymentFailed(order: $order);
                    }
                }
            }
        } catch (Exception $e) {
            throw new Exception(message: SOMETHING_WENT_WRONG_WITH_PAYMENT);
```

Step 5: How to setup webhook in ChawkBazar-laravel follow the steps

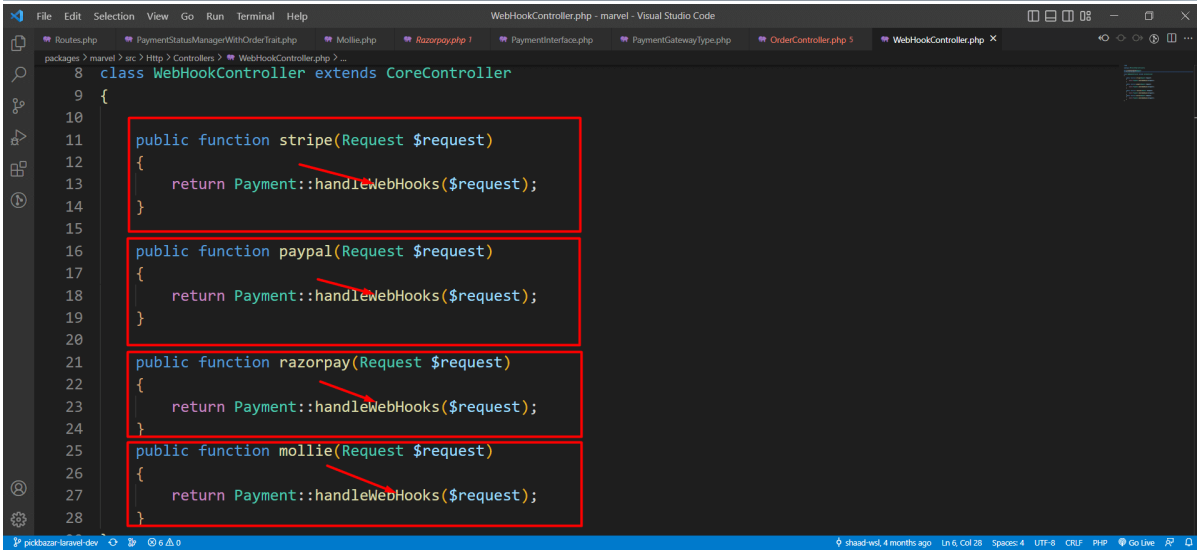
To use a payment gateway webhook, you would first configure the webhook URL in your payment gateway account. Then, whenever the specified events occur, the payment gateway will send an HTTP POST request to the webhook URL with a payload of data about the event.

```
     */
    public function confirmPaymentIntent(string $payment_intent_id, array $data): object
    {
        return (object) [];
    }

    /**
     * setIntent
     *
     * @param  array  $data
     * @return array
     */
    public function setIntent(array $data): array
    {
        return [];
    }
```

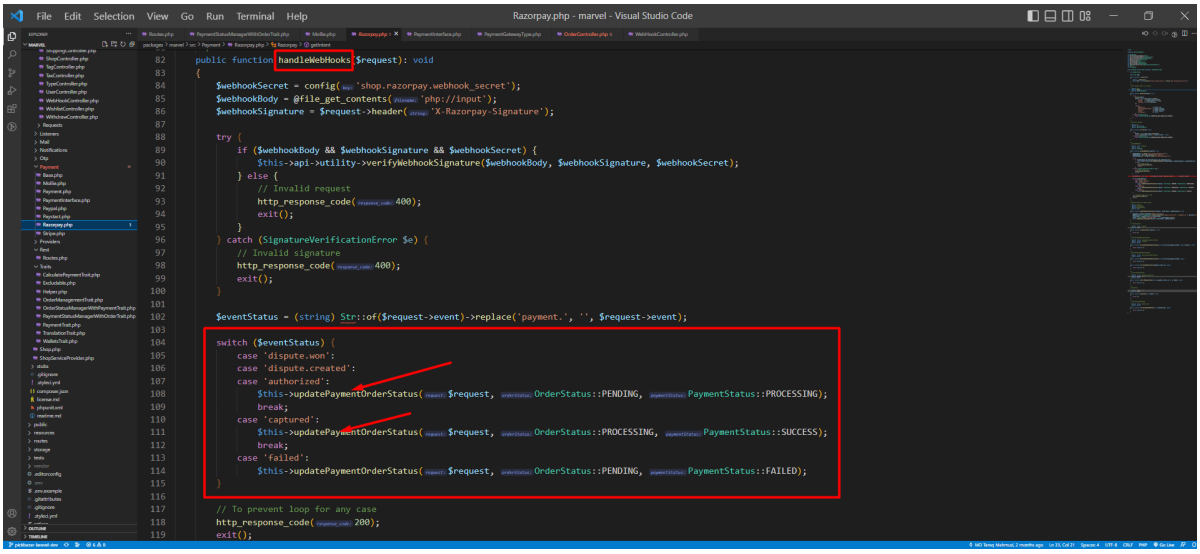
Fi go to API -> package -> marvel -> src -> Routes then add a post route.

```php
/**
 *
 *
 */
Route::post('webhooks/razorpay', [WebHookController::class, 'razorpay']);
Route::post('webhooks/stripe', [WebHookController::class, 'stripe']);
Route::post('webhooks/paypal', [WebHookController::class, 'paypal']);
Route::post('webhooks/mollie', [WebHookController::class, 'mollie']);
}
```

Add your function in WebHookController



To handle ***webhook events*** follow your payment gateway offical webhook documentation



*** Note : *** For locally webhook testing you can use ngrok tools for that. Please follow their official documentation.

Getting Started with admin dashboard.

First go to `marvel-admin -> rest -> types -> index.ts -> export enum PaymentGateway` then add PaymentGateway Name.

```
export enum PaymentGateway {
  STRIPE = 'STRIPE',
  COD = 'CASH_ON_DELIVERY',
  CASH = 'CASH',
  FULL_WALLET_PAYMENT = 'FULL_WALLET_PAYMENT',
  PAYPAL = 'PAYPAL',
  MOLLIE = 'MOLLIE',
  RAZORPAY = 'RAZORPAY',
}
```

after that go to `marvel-admin -> rest -> src ->components -> setings -> payment.ts` add name & Title.

```
export const PAYMENT_GATEWAY = [
  { name: 'stripe', title: 'Stripe' },
  { name: 'paypal', title: 'Paypal' },
  { name: 'razorpay', title: 'RazorPay' },
  { name: 'mollie', title: 'Mollie' },
];
```

Then it will automatically add that payment gateway in the marvel-admin settings.

Getting Started with shop front.

There are two types of payment gateway system can be integrated here.

- Redirect based payment gateway (e.g PayPal). Where the customer will redirect to that payment gateway site during order checkout. Complete the payment there. And then comeback to the application.
- Non redirect based payment gateway. Where the customer will stay on the application and complete the whole payment process here. Here we consider Stripe as a non-redirect based payment gateway. Though Stripe has features too similar to redirect based payment gateway.

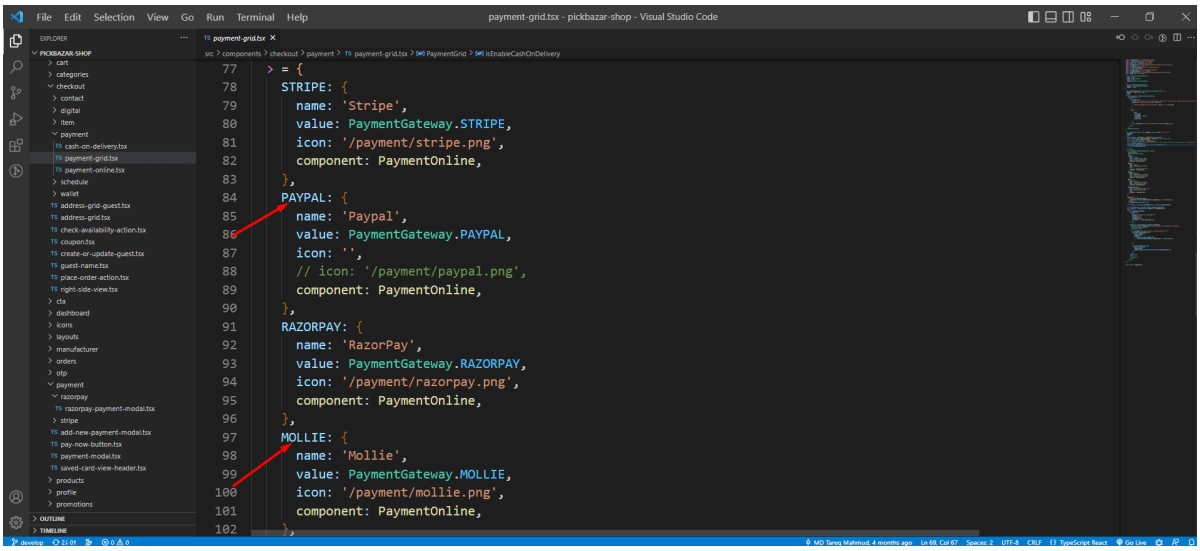
Redirect-base Payment Gateway

if you want to integrate redirect based payment gateway (e.g. PayPal, Mollie). follow the steps

First go to `shop -> src -> types -> index.ts -> export enum PaymentGateway` then add PaymentGateway Name.

```
export enum PaymentGateway {
  STRIPE = 'STRIPE',
  COD = 'CASH_ON_DELIVERY',
  CASH = 'CASH',
  FULL_WALLET_PAYMENT = 'FULL_WALLET_PAYMENT',
  PAYPAL = 'PAYPAL',
  MOLLIE = 'MOLLIE',
  RAZORPAY = 'RAZORPAY',
}
```

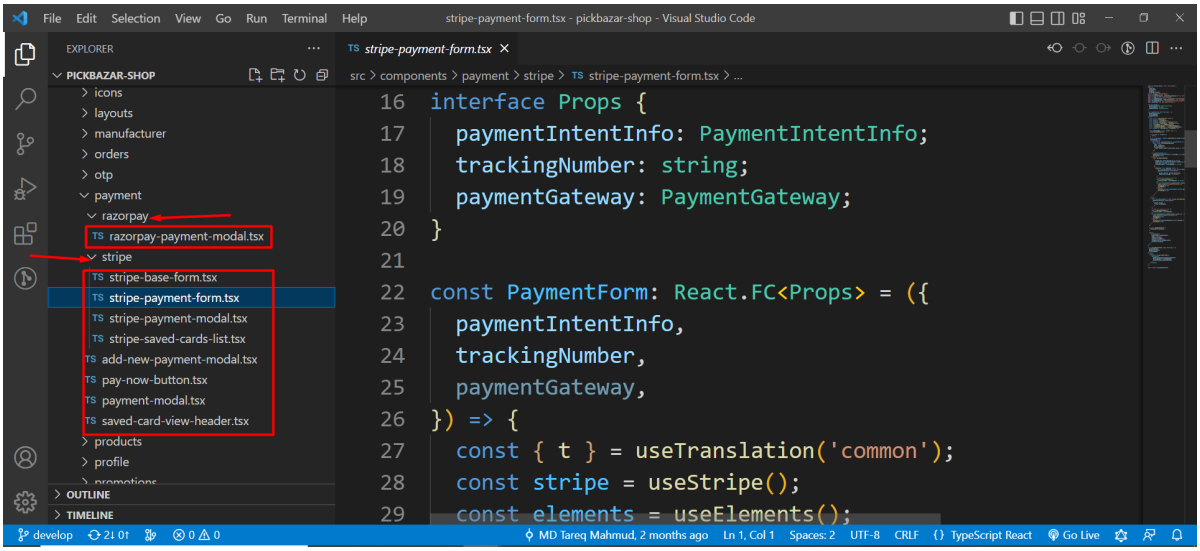
after that go to `shop -> src -> components -> checkout -> payment -> Payment-grid.tsx` then add your PaymentGateway object.



Non-redirect based payment gateway

First, complete the redirect-based payment gateway steps mentioned above. Because that two steps is universal for all payment gateway to apply.

After That go to `shop -> src -> components -> payment` then add a folder like Stripe, Razorpay. Inside your payment folder you can create your required typescript files with related functionalities for your Payment Gateway. For example, you can checkout the Stripe folder. You can find all the necessary indication, components guide, payment-method (card) saving options etc in the Stripe folder.



That's all for today. If you need any more help you can always contact with our support agents in our support portal.

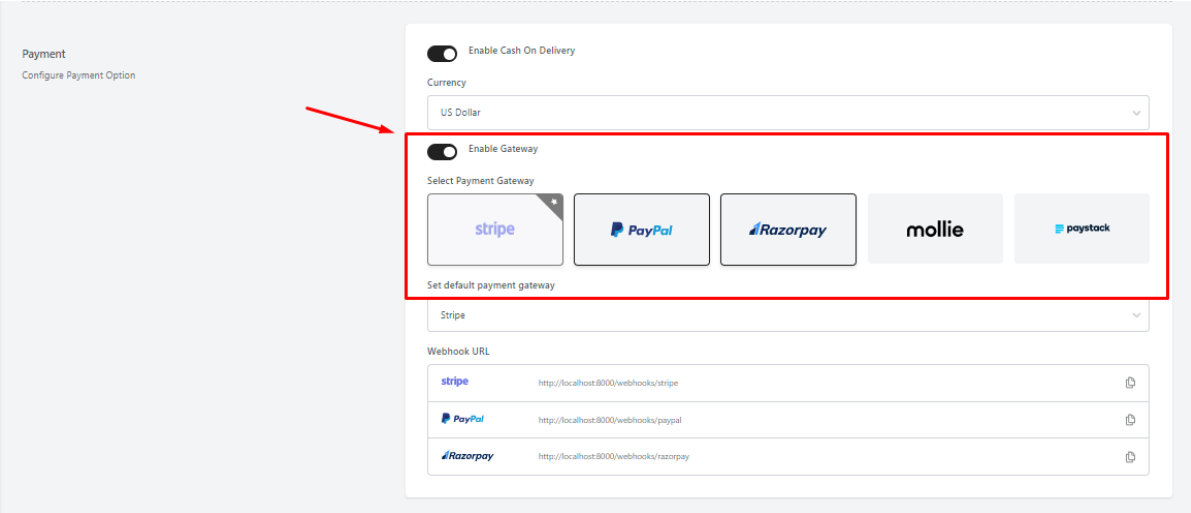
Multiple payment gateway

In this chapter we are going to discuss about how you can use multiple payment gateway in our application. But before jumping into this feature user needs to keep some points in mind first.

```
1. Choose your desired payment gateway based on availablity in your region.
2. Choose your currency correctly before activating that payment gateway.
3. Set up all the necessary .env keys and payment gateway internal settings
properly before starting operations.
4. Configure webhooks properly if needed.
```

How to activate multiple payment gateways [in Admin] ?

In admin dashboard there are new settings introduced for this feature. Please check this screenshot first,



Description [sequence by number in the screenshot]

1. Choose the currency first
2. Enable gateway if you want to use any
3. Select the payment gateway based on your region and supported currency.
4. Select default payment gateway. Default gateway only available from selected ones.
5. Webhooks URL will be generated. You can use this URL for webhooks setting in payment gateways.

How to use multiple payment gateways from customer end [in Shop] ?

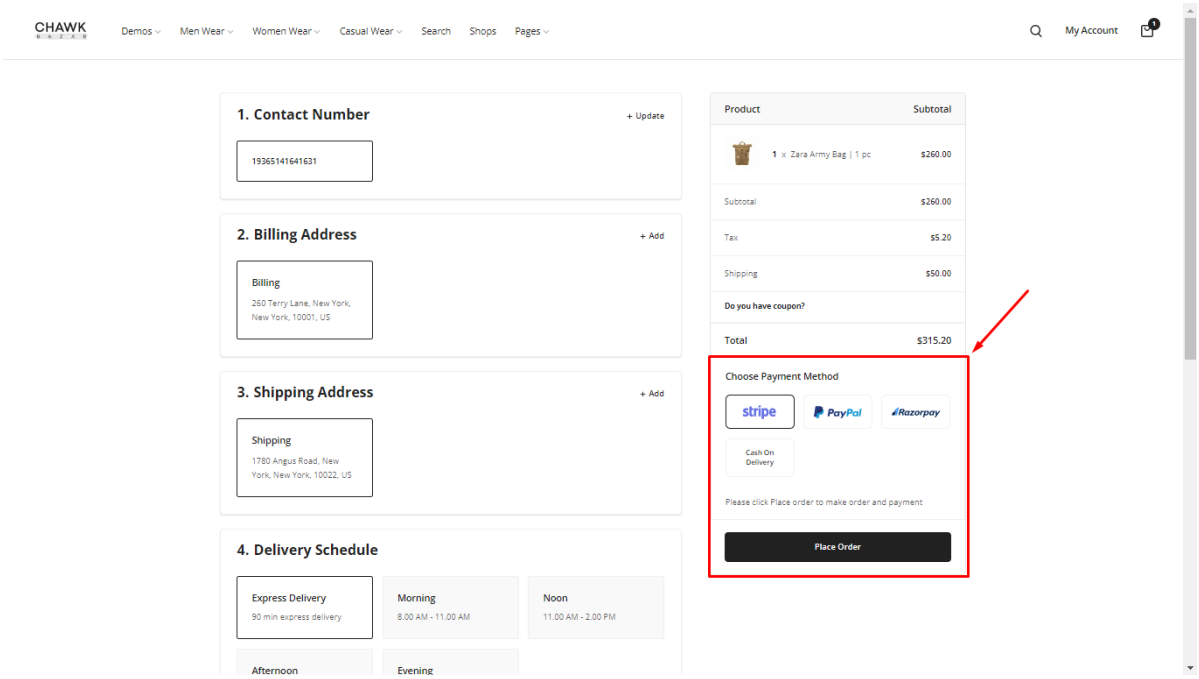
Payment gateways activated by super-admin will be available in the checkout process. There are two steps of operation,

1. Select a payment gateway during place order
2. Select different payment gateway (if needed), after the place order.

This two points are cleared below with necessary screenshot.

1. Select a payment gateway during place order

Please check this screenshot first,



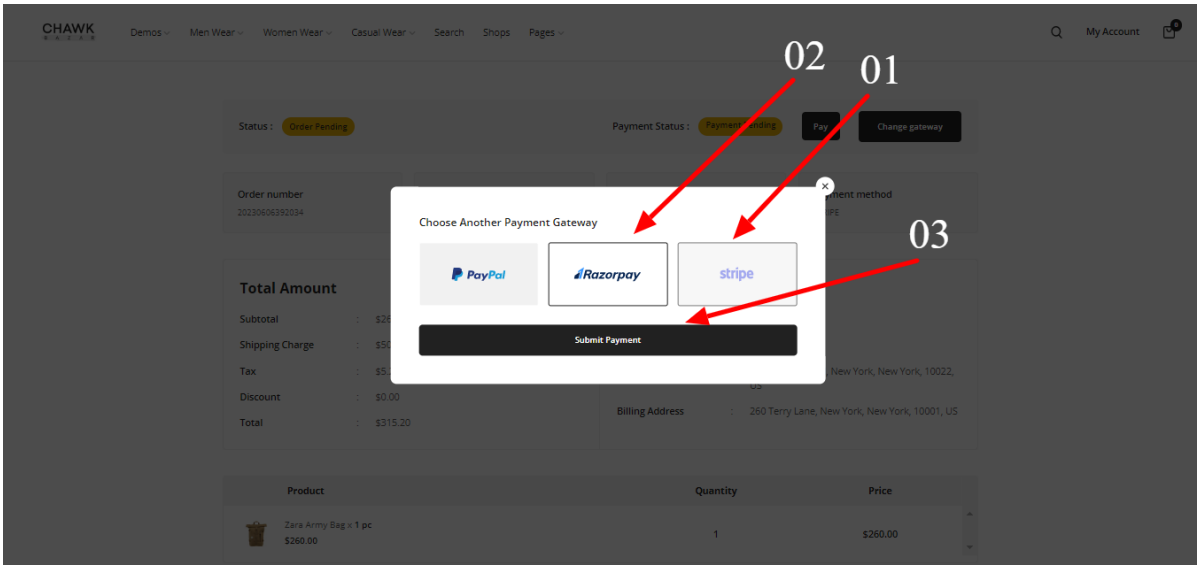
Select a payment gateway just before placing order.

** Notes for customer : ** Please choose this carefully. Otherwise you may need to re-select the payment procedure again.

2. Select different payment gateway (if needed)

In order details page there is a option to re-select the payment gateway. If any customer select a gateway and placeorder but found out any issue of his/her selected gateway, then he/she will have a opportunity to re-select the payment gateway again and complete the order process.

Please check the screenshot below carefully,



Description [sequence by number in the screenshot]

1. Stripe is the previous selected payment gateway. Which do not intend to use now. So he/she might change the gateway and submit the payment.
2. PayPal was selected payment gateway now.
3. Submit payment button need to be clicked, otherwise the operation will not be continued.

Translation

If you're using `chawkbazar-laravel` after `v2.0.0` then you can translate static content for multilanguage. It doesn't matter if the language is RTL or not, you can do both. For translation we use this next-i18next package.

Existing Language

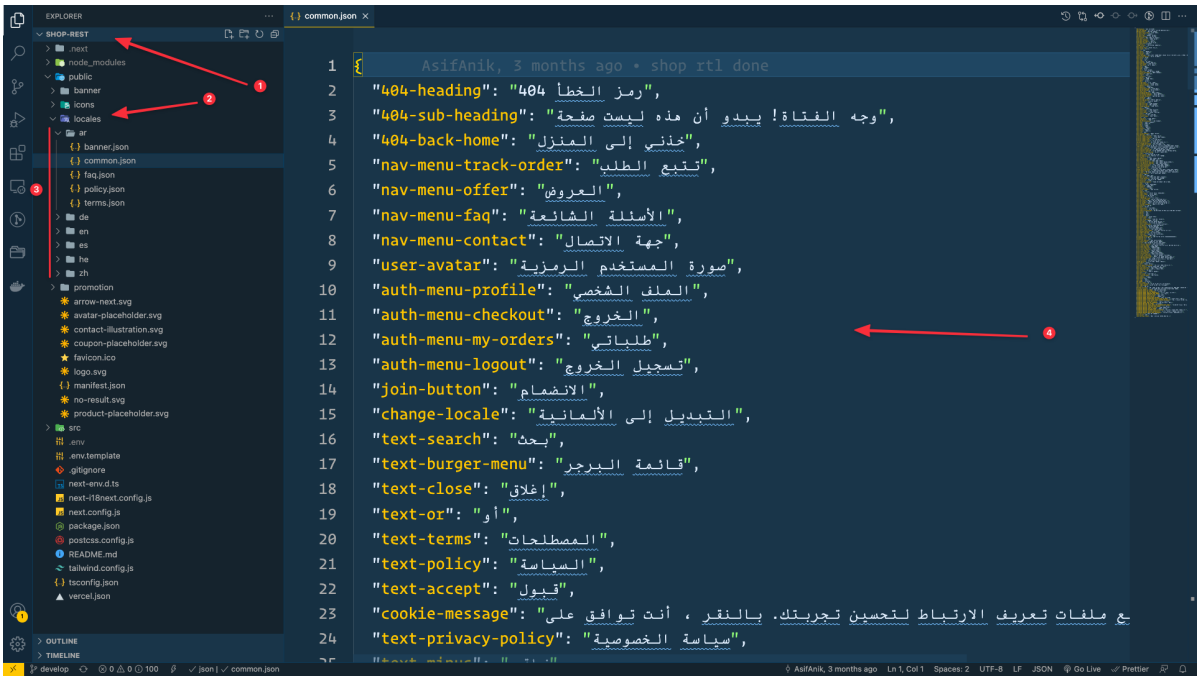
Right now, we provide translation files for `ar`, `de`, `en`, `es`, `he`, and `zh.`

If your translation language is from them then just edit the translation string from,

For Rest

```
chawkbazar-laravel -> shop -> public -> locales -> YOUR LANGUAGE
```

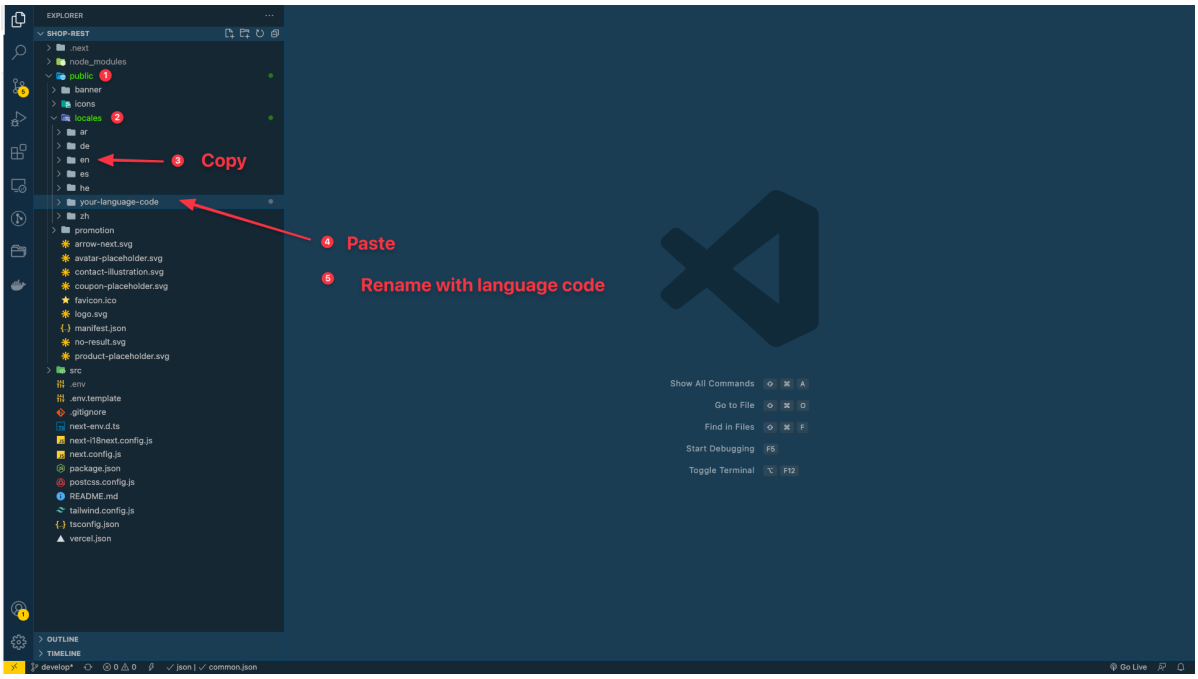
```
chawkbazar-laravel -> admin -> rest -> public -> locales -> YOUR LANGUAGE
```



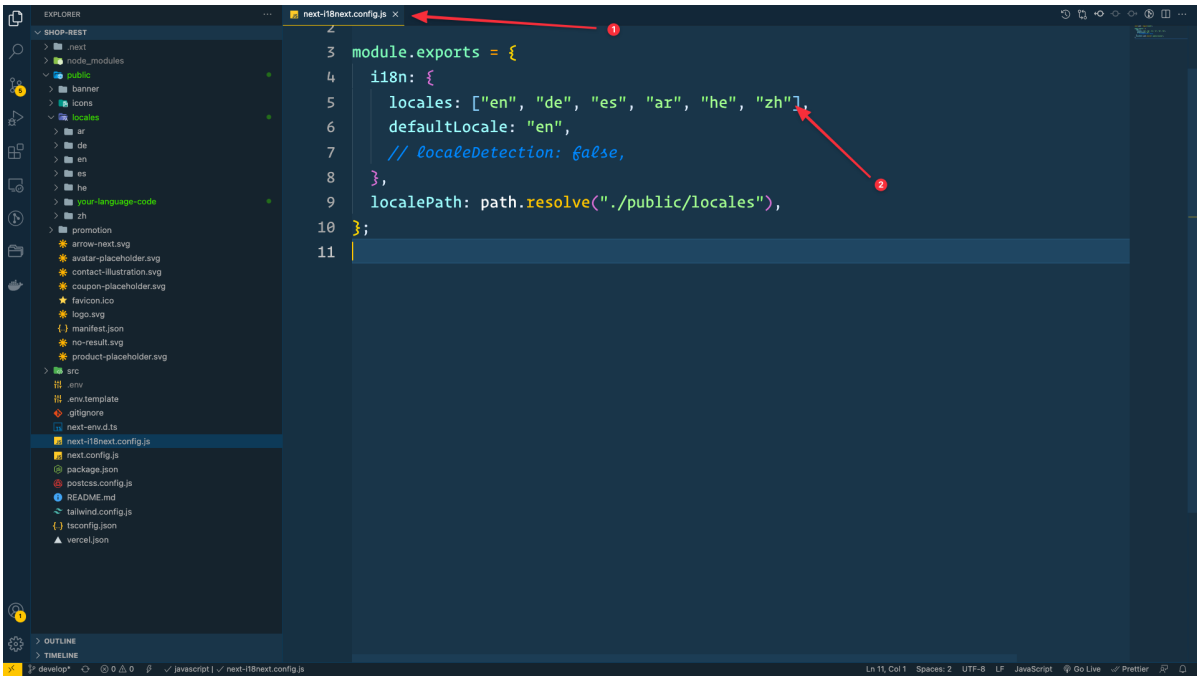
After translating `string` from `JSON` re-run `or` rebuild ` project it'll translate the static string.

New Language

If your language is not in the lists then just copy the `en` folder from `locale` and paste it in the same folder by rename your language code.



After creating that folder edit `next-i18next.config.js` and add the `language-code` at `locales` object.



Default Language

By default, English is the default language. To change that edit,

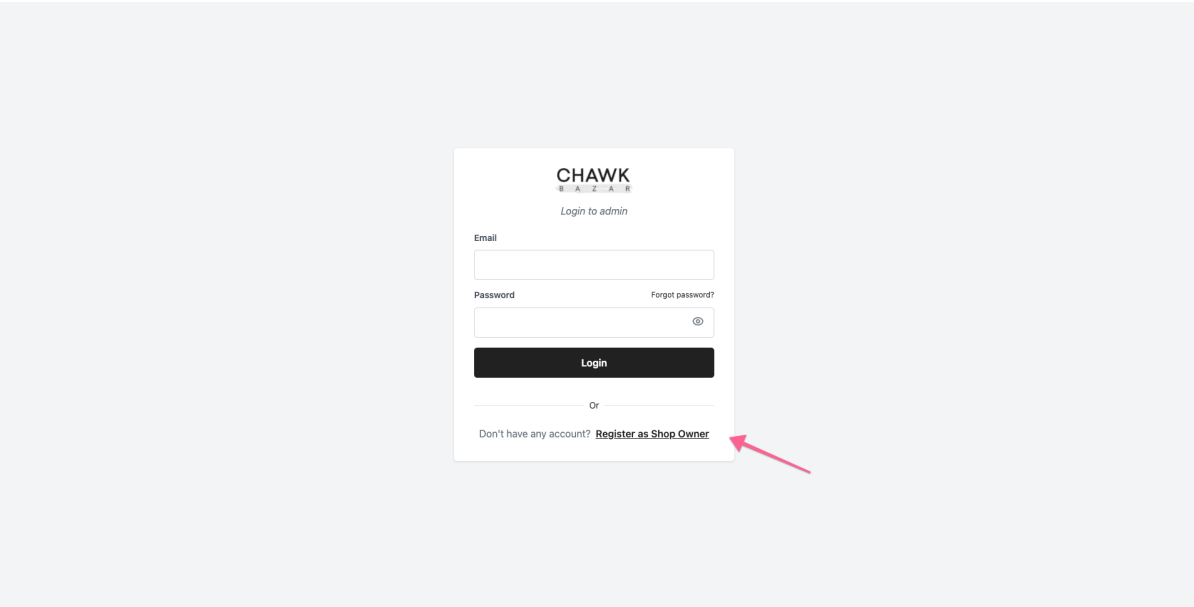
```
chawkbazar-laravel -> shop -> public -> locales -> YOUR LANGUAGE -> next-i18next.config.js
```

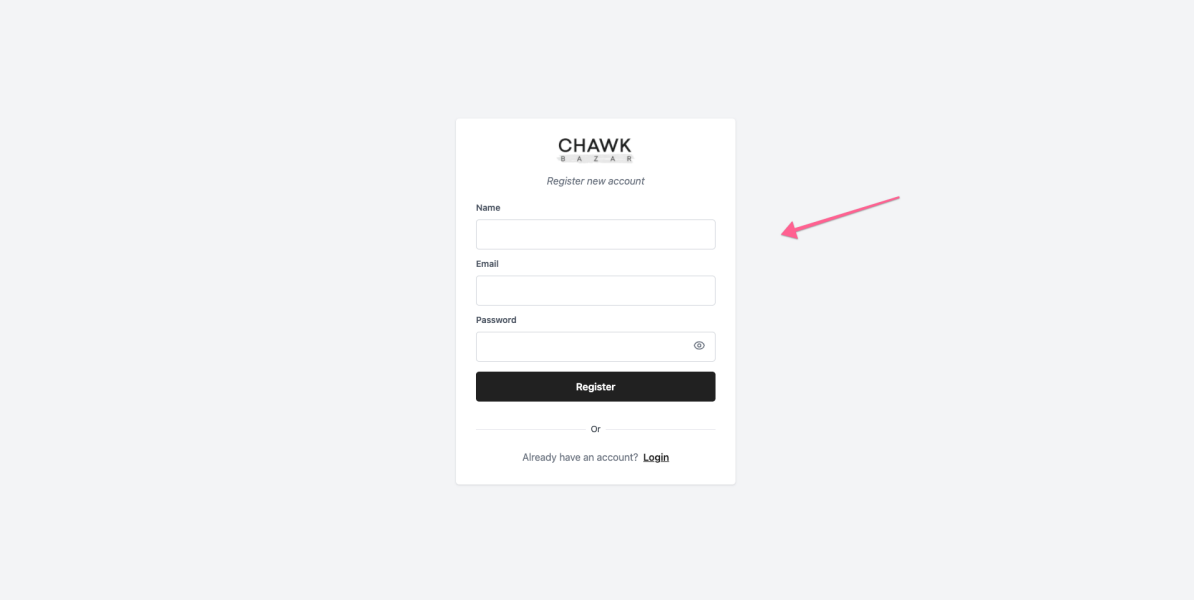
```
chawkbazar-laravel -> admin -> rest -> public -> locales -> YOUR LANGUAGE -> next-i18next.config.js
```

Multivendor

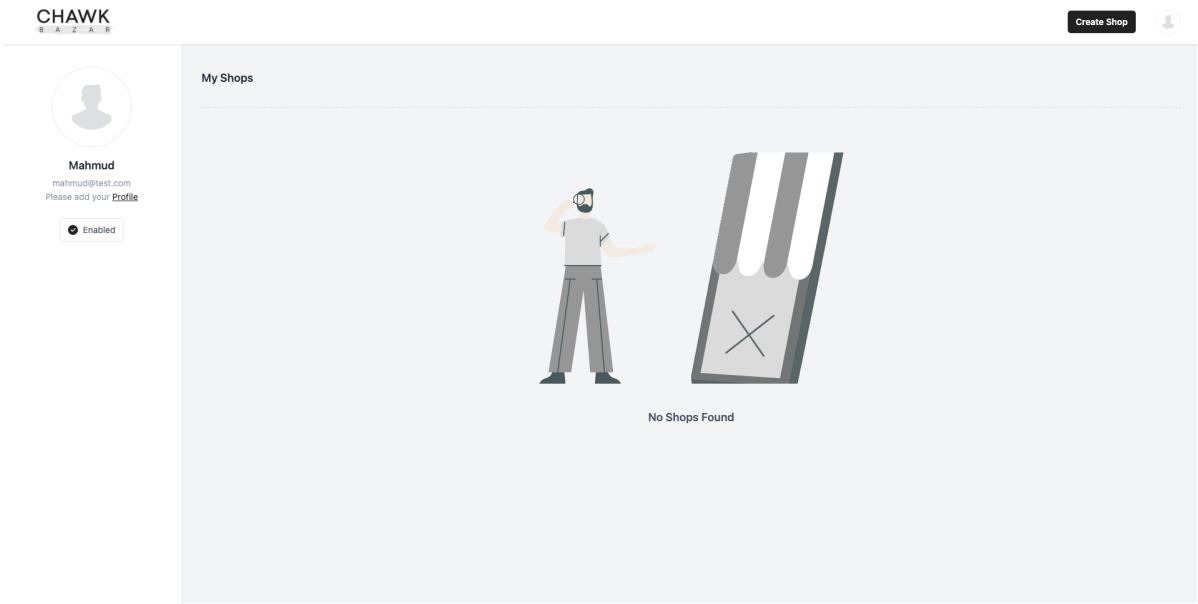
Create New Shop

To create new shop login as an `administrator` or create a new account for creating `shop`

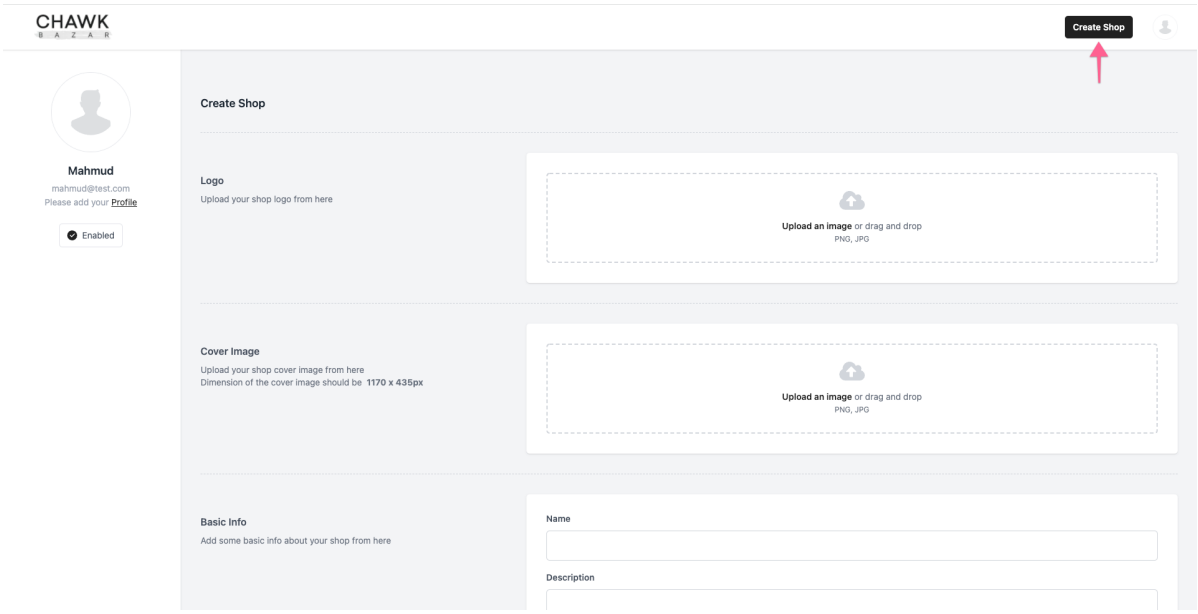




After creating the account you'll be redirected to this page,

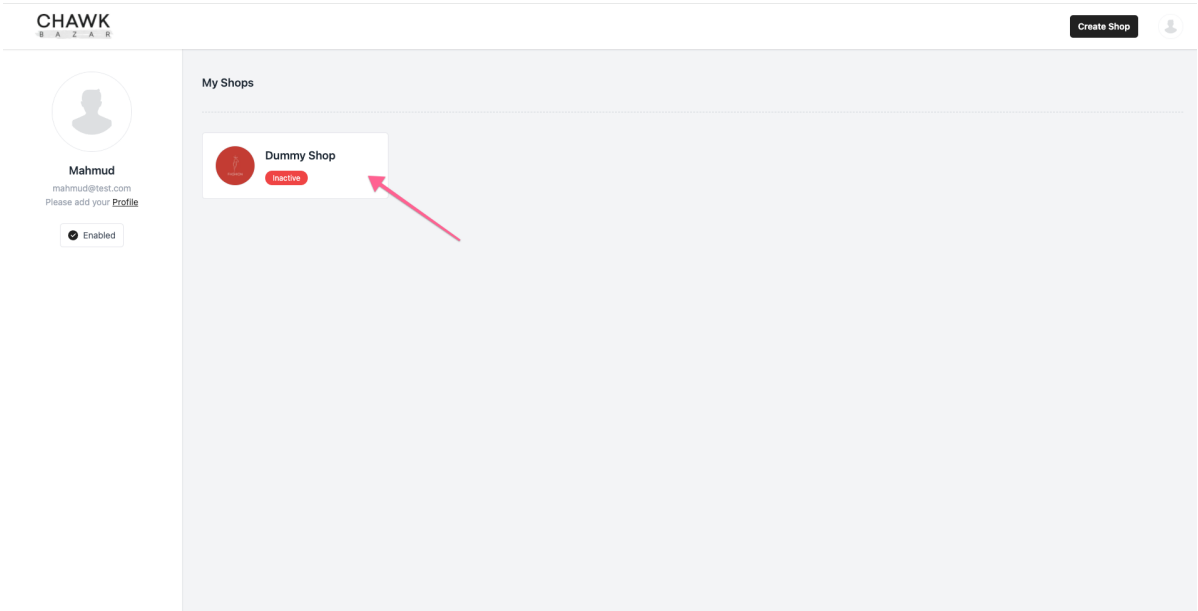


After that, click `Create Shop`



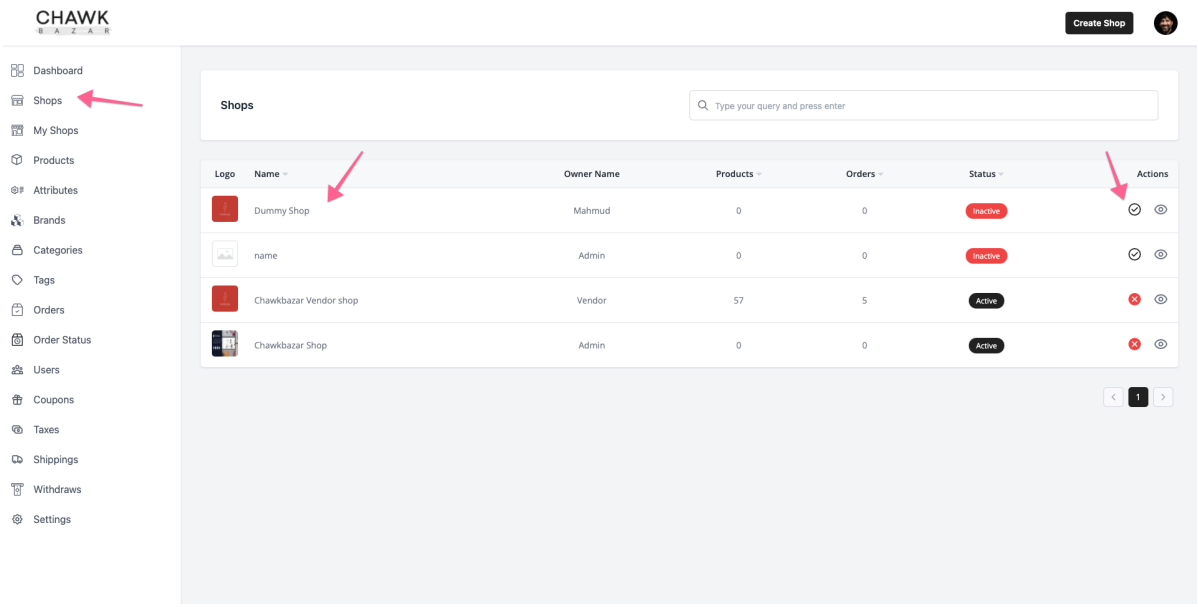
And provide all the information for the `store`.

After creating the shop you'll redirect to this page,

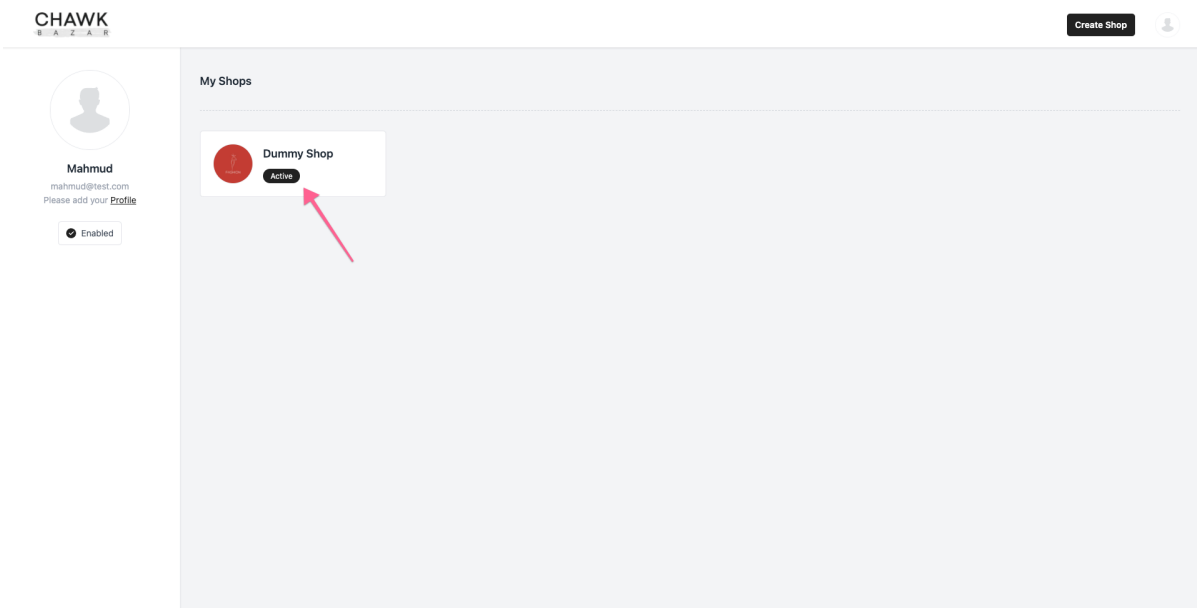


By default, the shop will be inactive. Only `administrator` can active a shop.

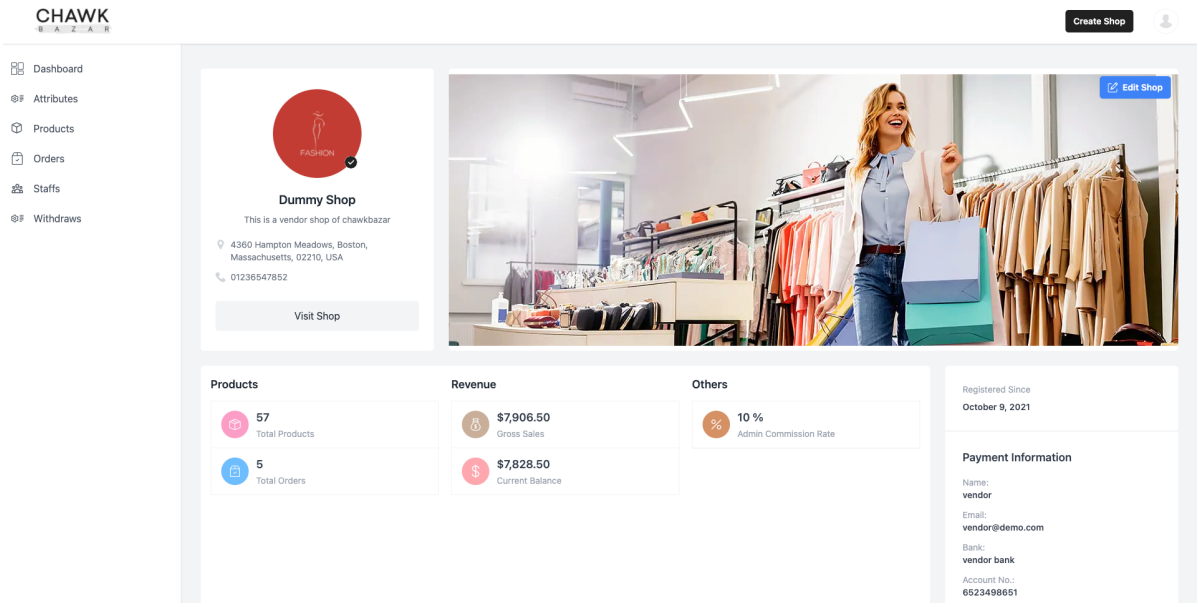
From `administrator account go to` shop `and click` tick `mark to` activate `or` deactivate ` a shop.



After `activate` the shop by `administrator` the `vendor` dashboard will be like this,

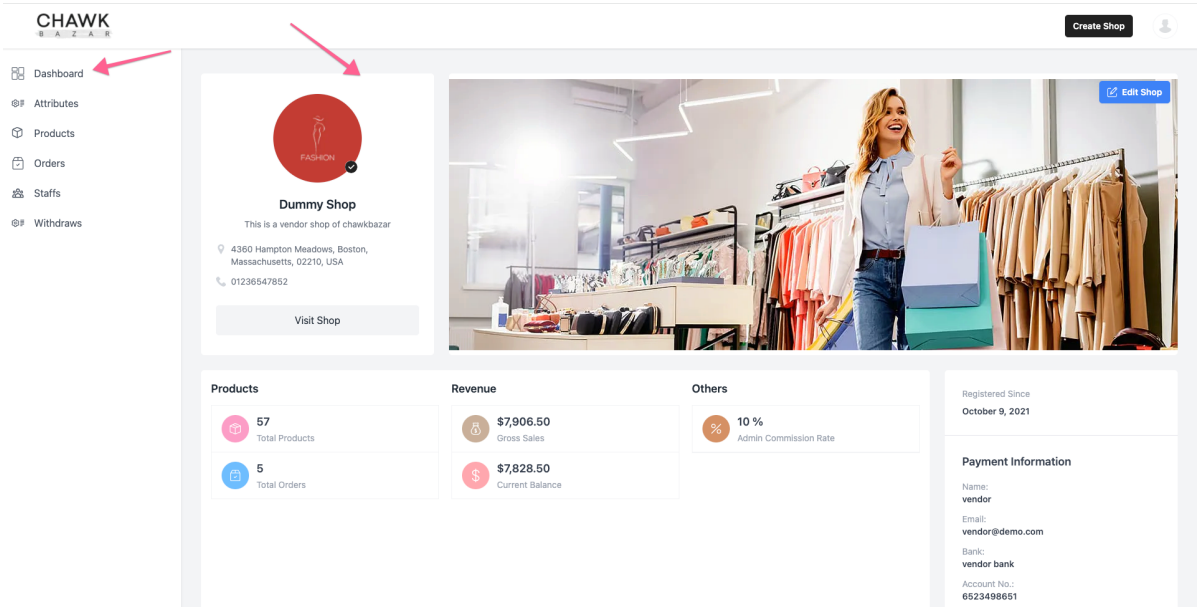


After click on `shop`, you'll be redirected to dashboard page,

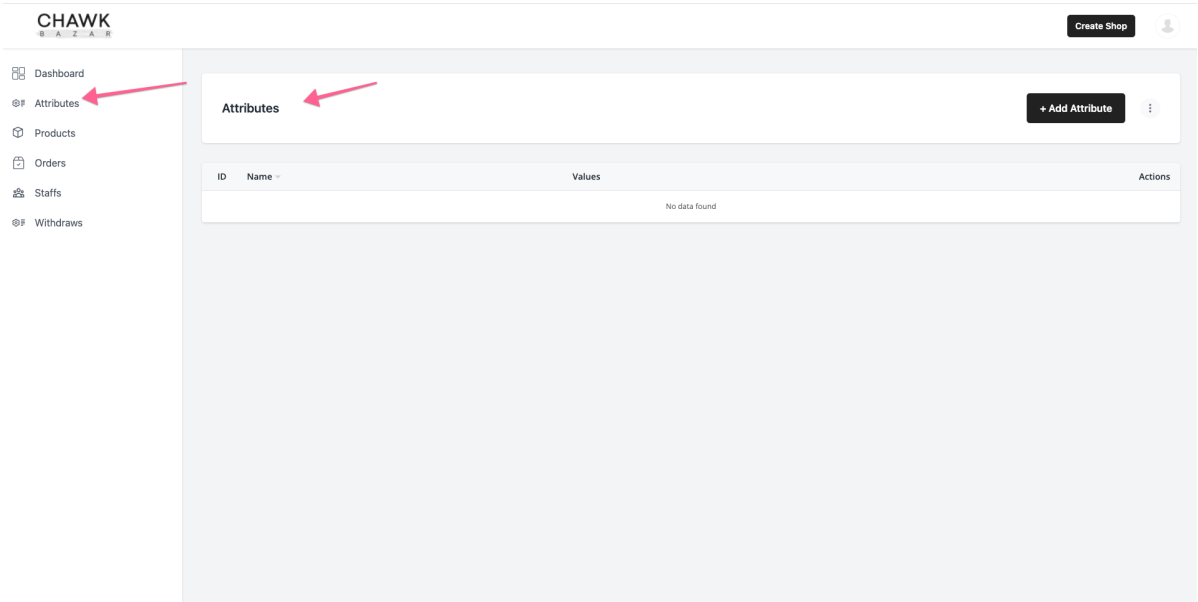


From this dashboard, you can maintain you shop,

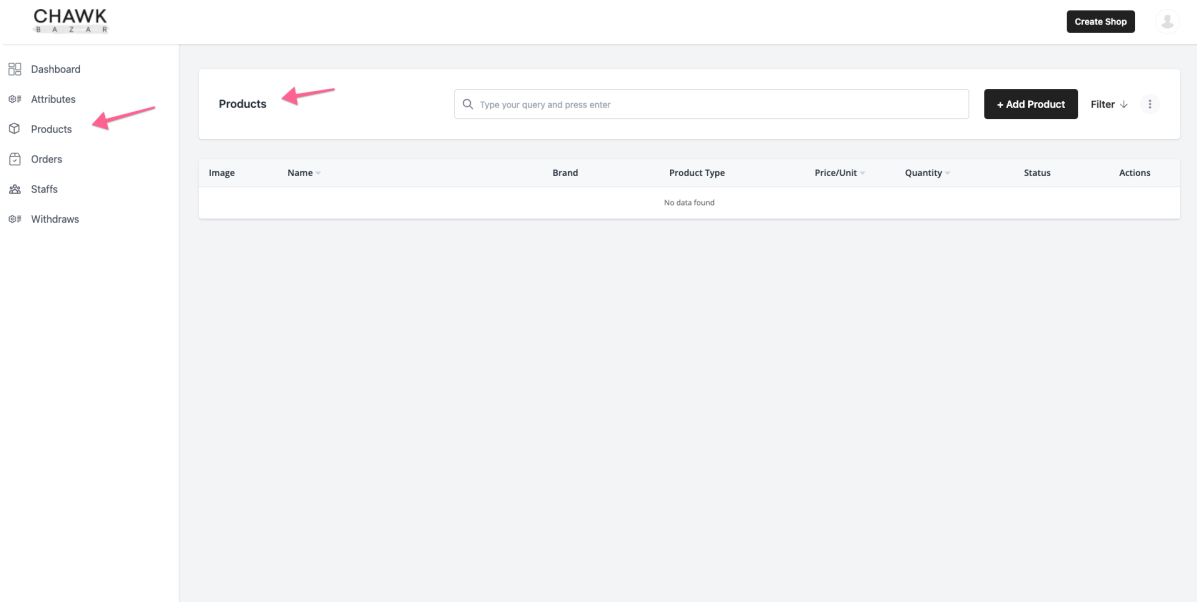
Dashboard:



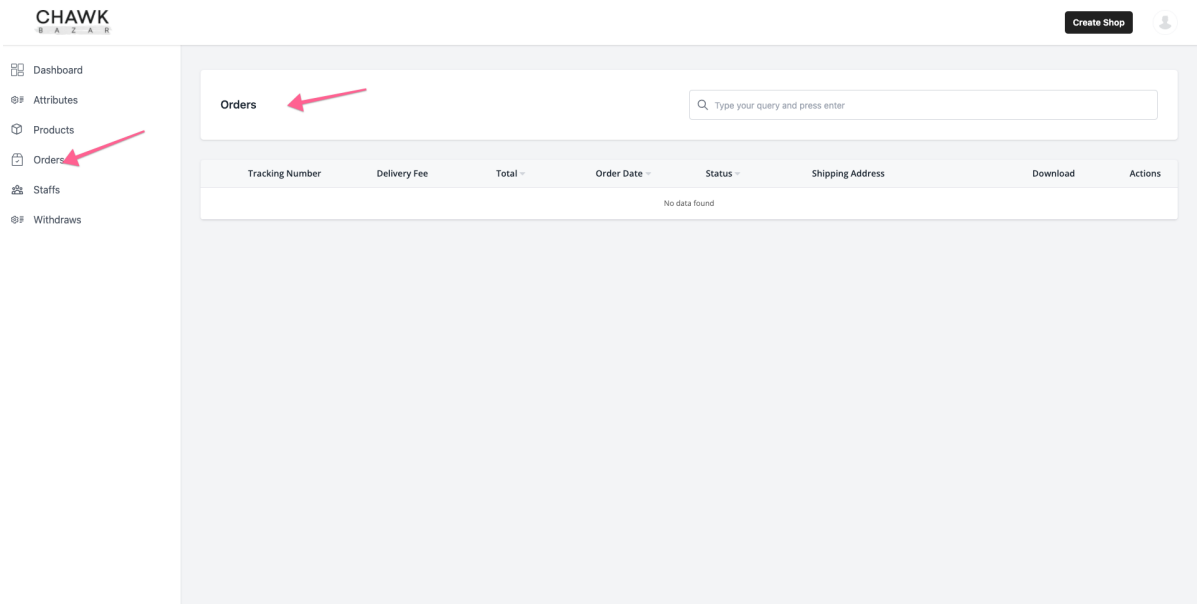
Attributes:



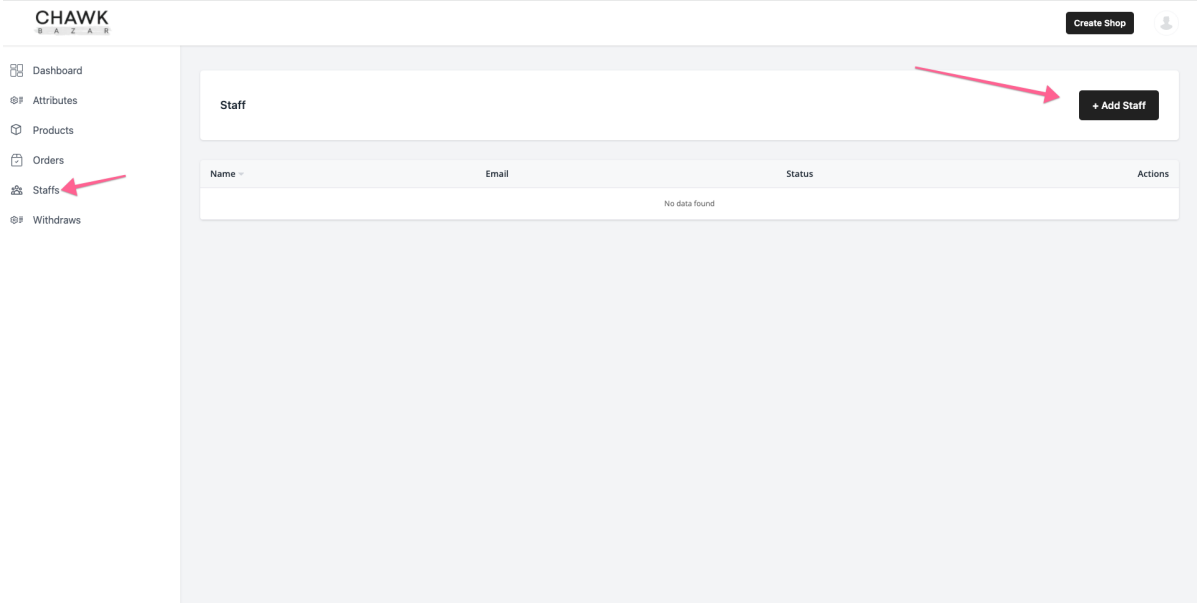
Products:



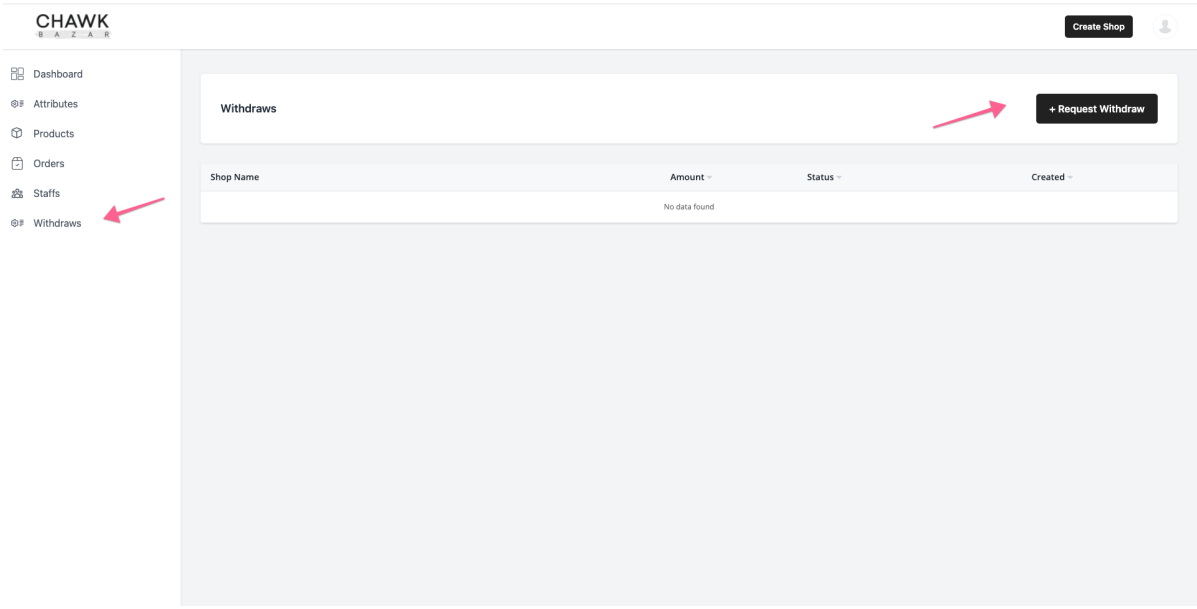
Order:



Staff:



WithDraw:



User Roles:

Super Admin:

Super admin can do everything. the admin can maintain and edit every store on the site.

Store Owner:

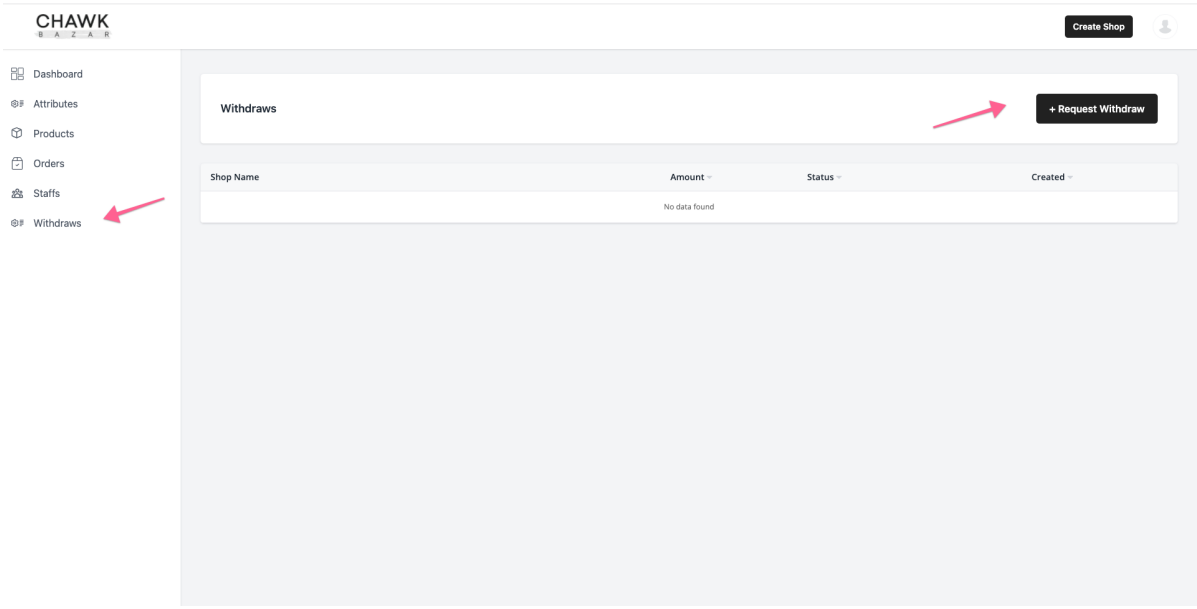
Store owner can edit or main it's store, staff or payment.

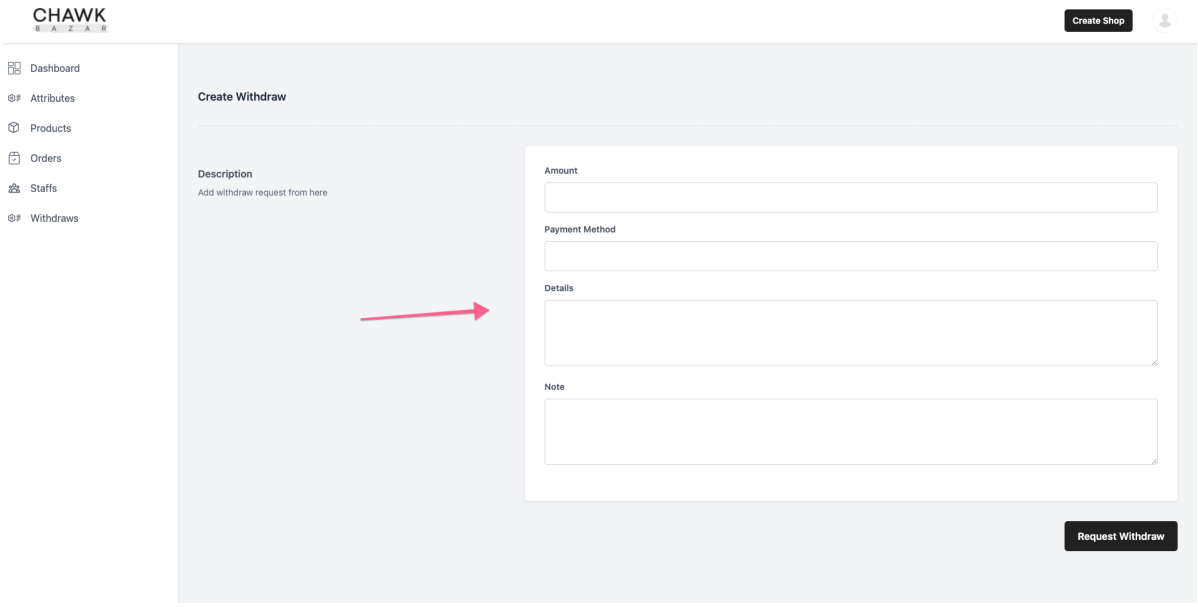
Staff:

The staff of a store has similar permission as store owner, but the staff can't update the store or withdraw payment.

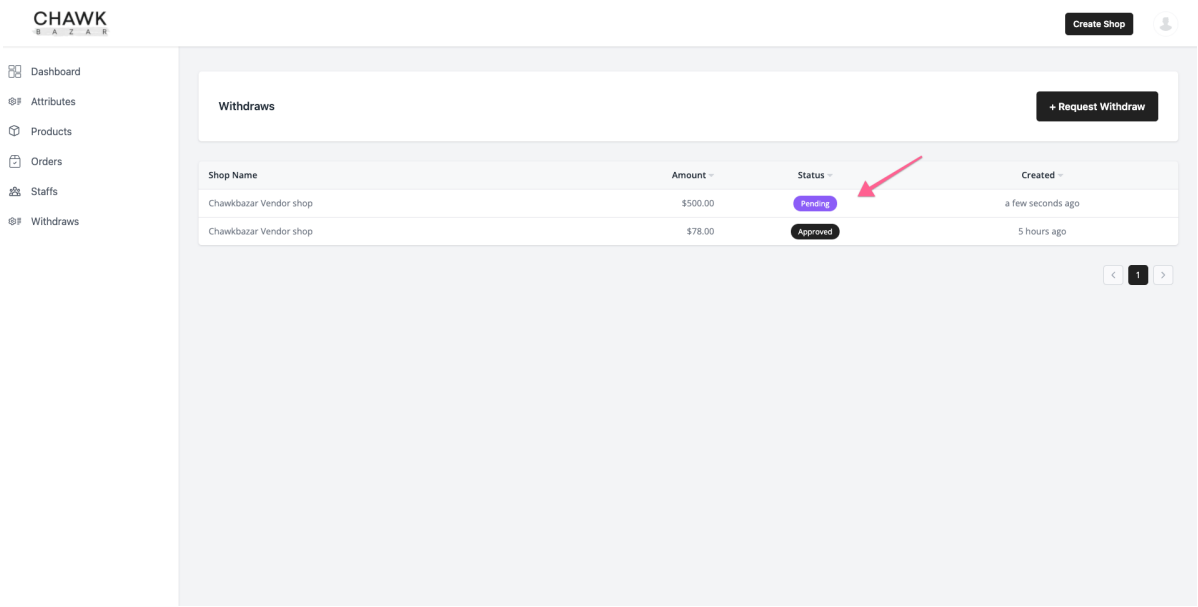
Withdraw Payment:

Only the store owner can withdraw its payment. To do that go to your shop dashboard -> withdraws -> Request a withdraw,

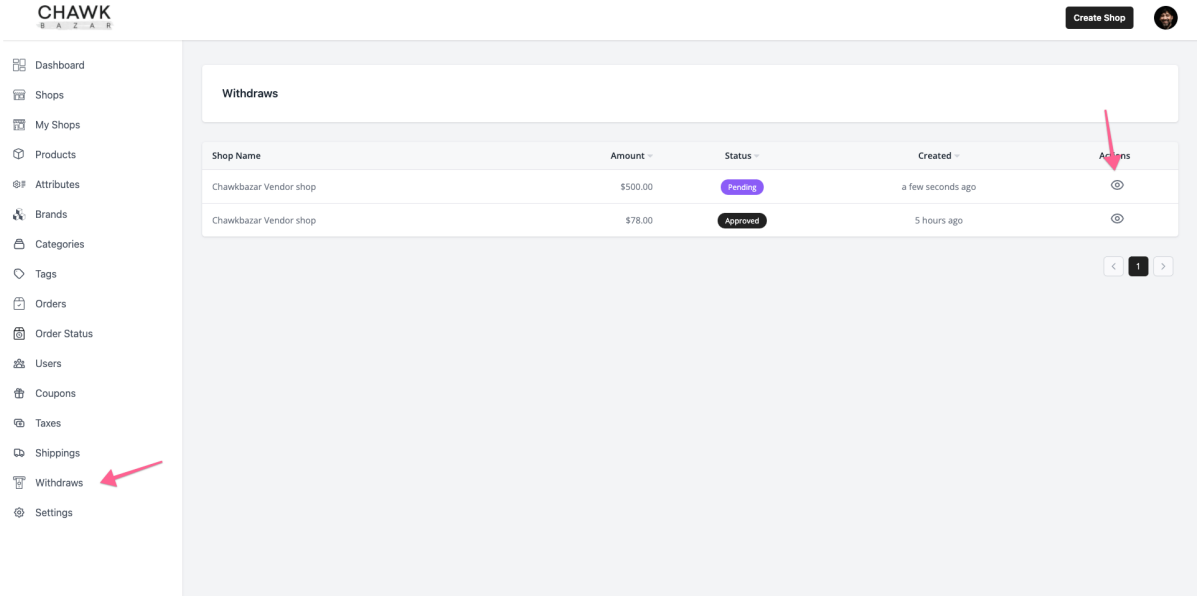


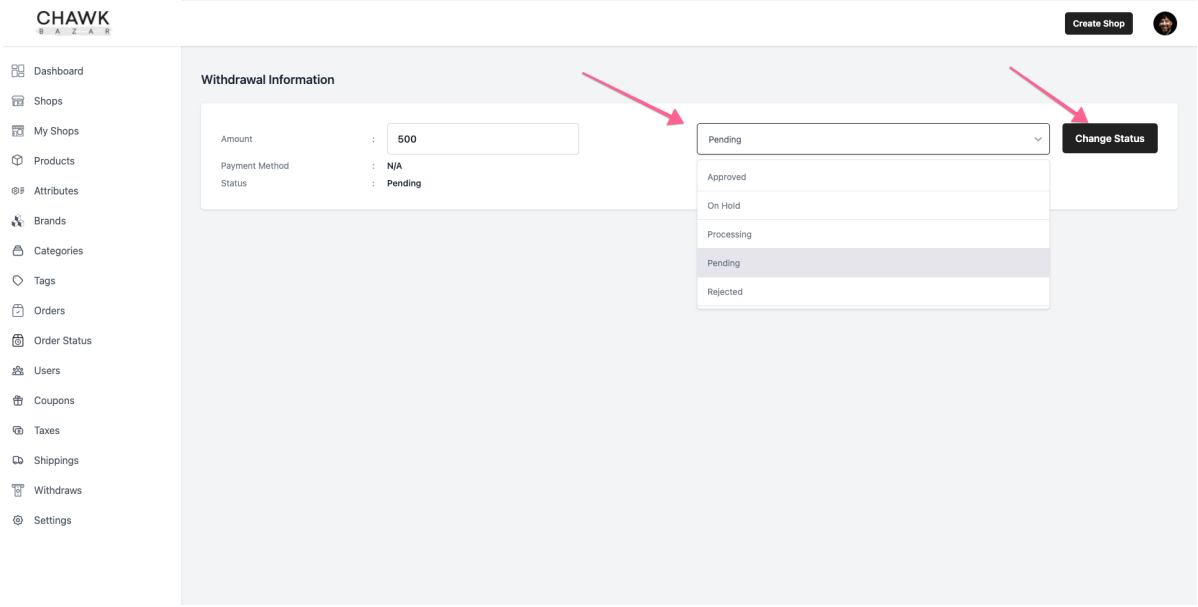


After request payment, the dashboard will be like this,

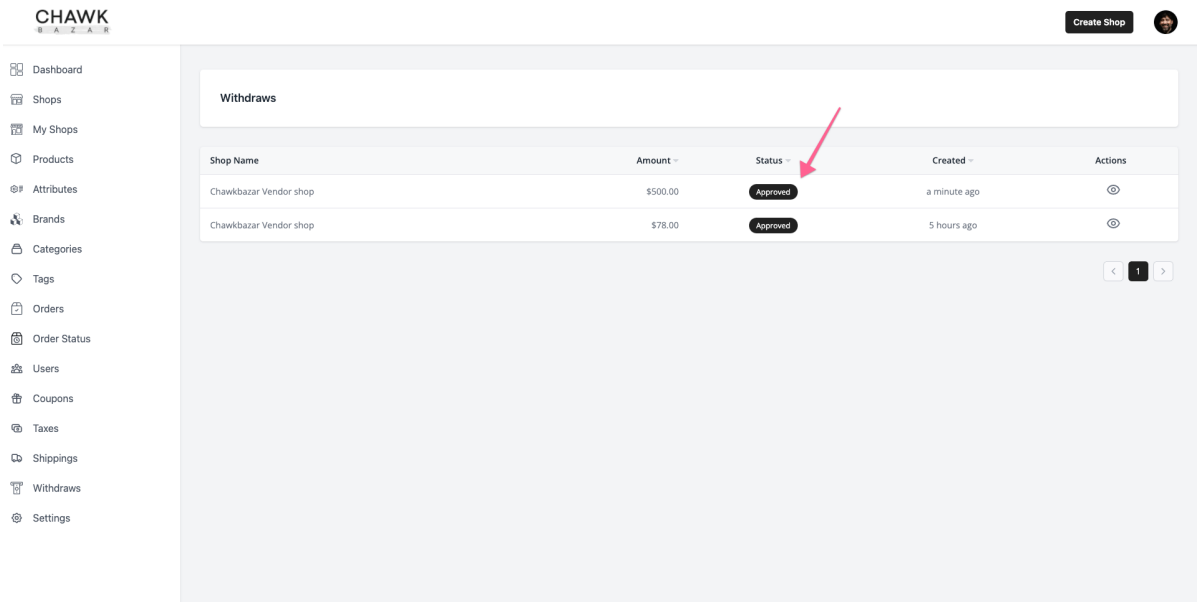


After request, the admin has to be approved the admin.



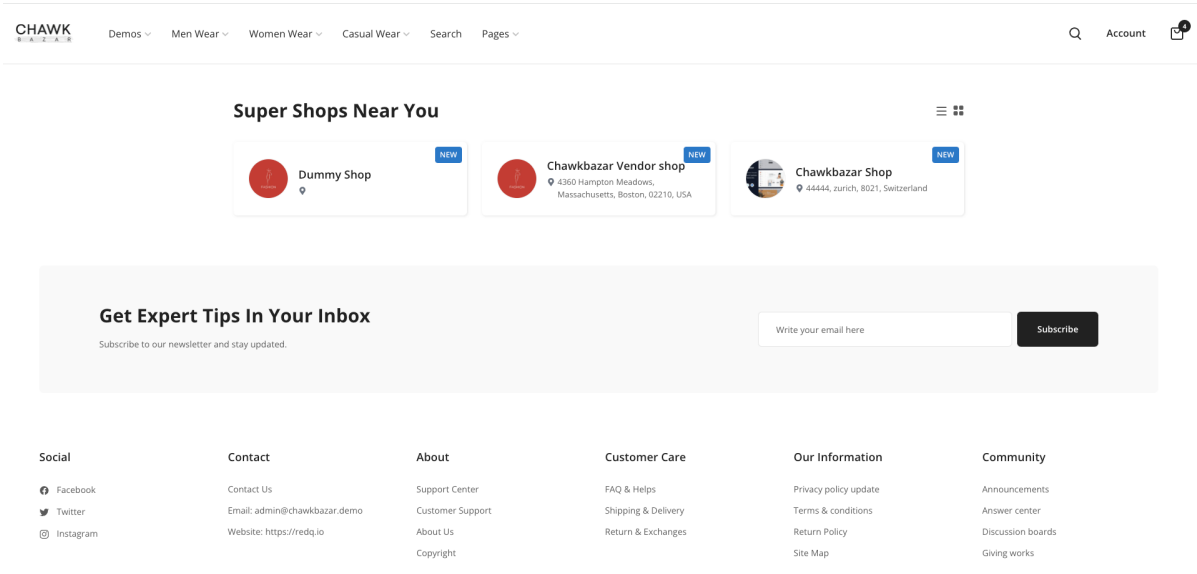


After approved,

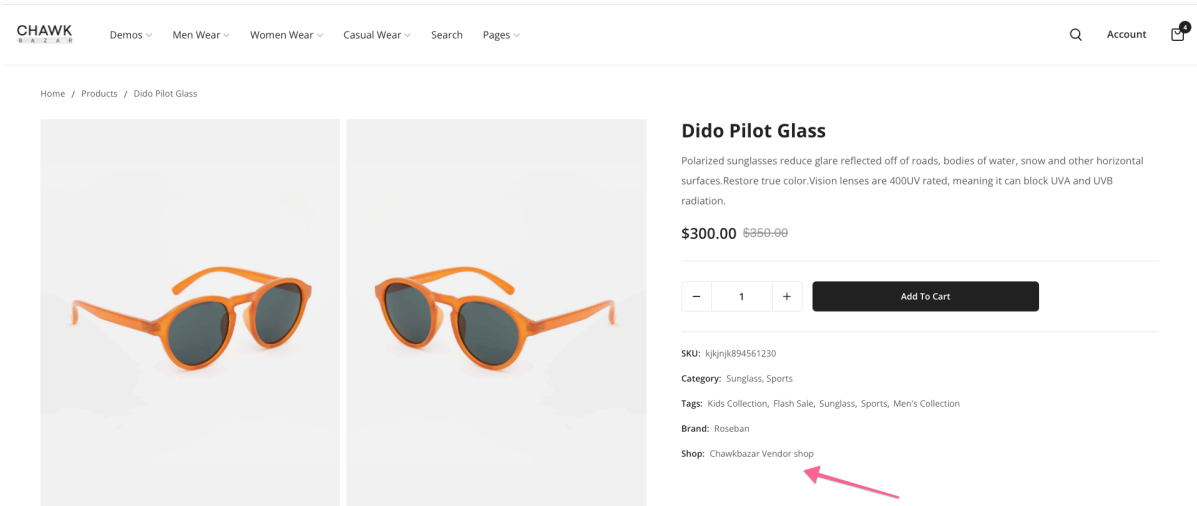


FrontEnd Shop

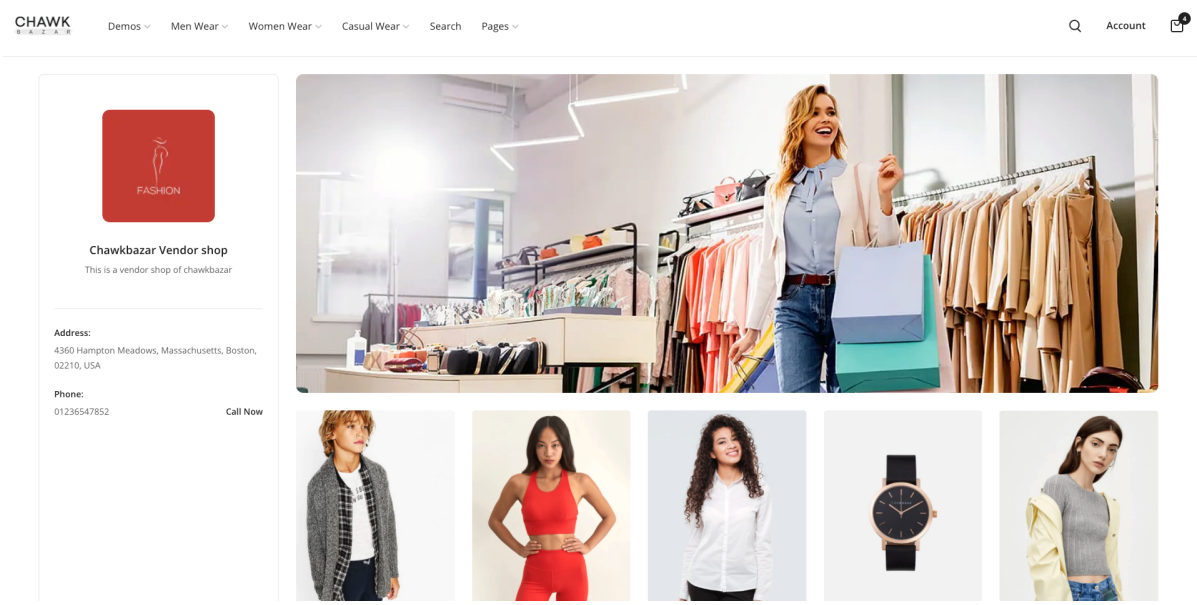
From frontend when customers click on the `shop` page they'll get all the shops as a card list,



Also when customers click on a product they will get the seller link,

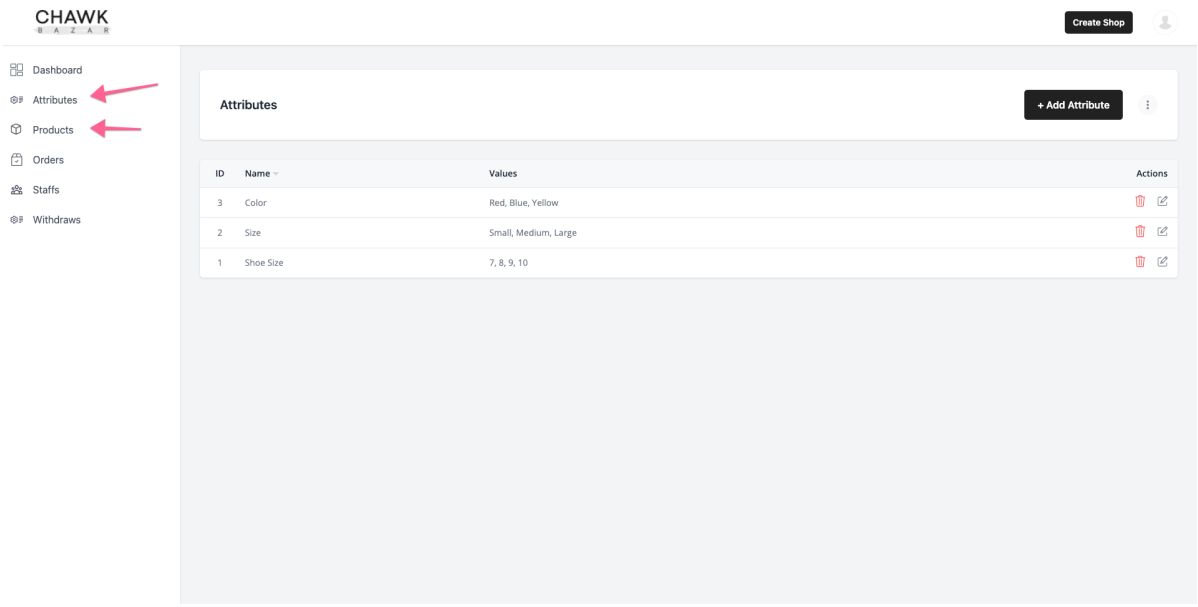


After clicking the `seller link` they will redirect to the seller shop page,



Export Import

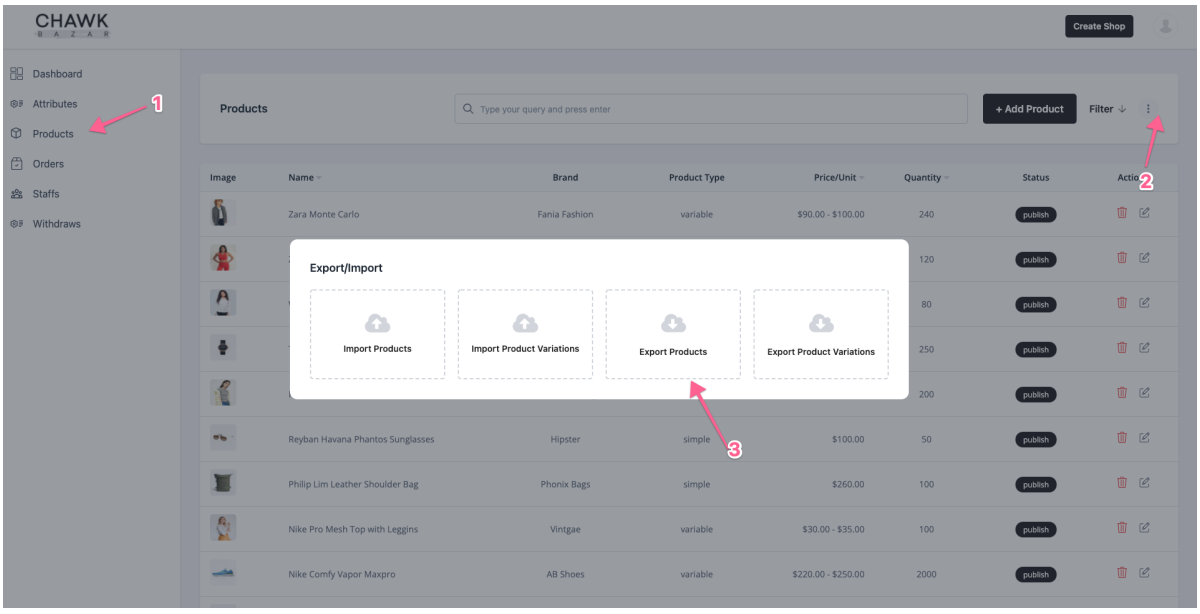
Export Import works under a shop. So, You have to go to the `shop page -> products menu` for product or `shop page -> attributes menu` to import or export attributes.



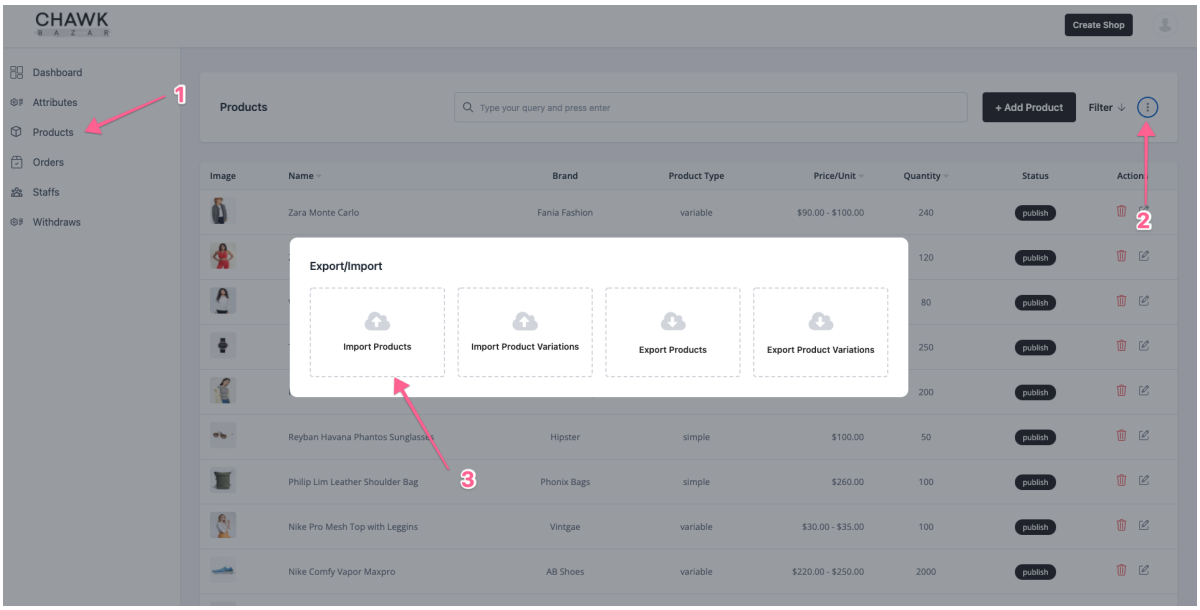
Export Import work different way for simple and variable product

Simple Product

For simple products, you've to export only products data. To do that, go to `Your Shop -> products`. Then click on `three dots` and export the simple products.



Similarly, for import `csv,` go to the same option and import exported `csv`

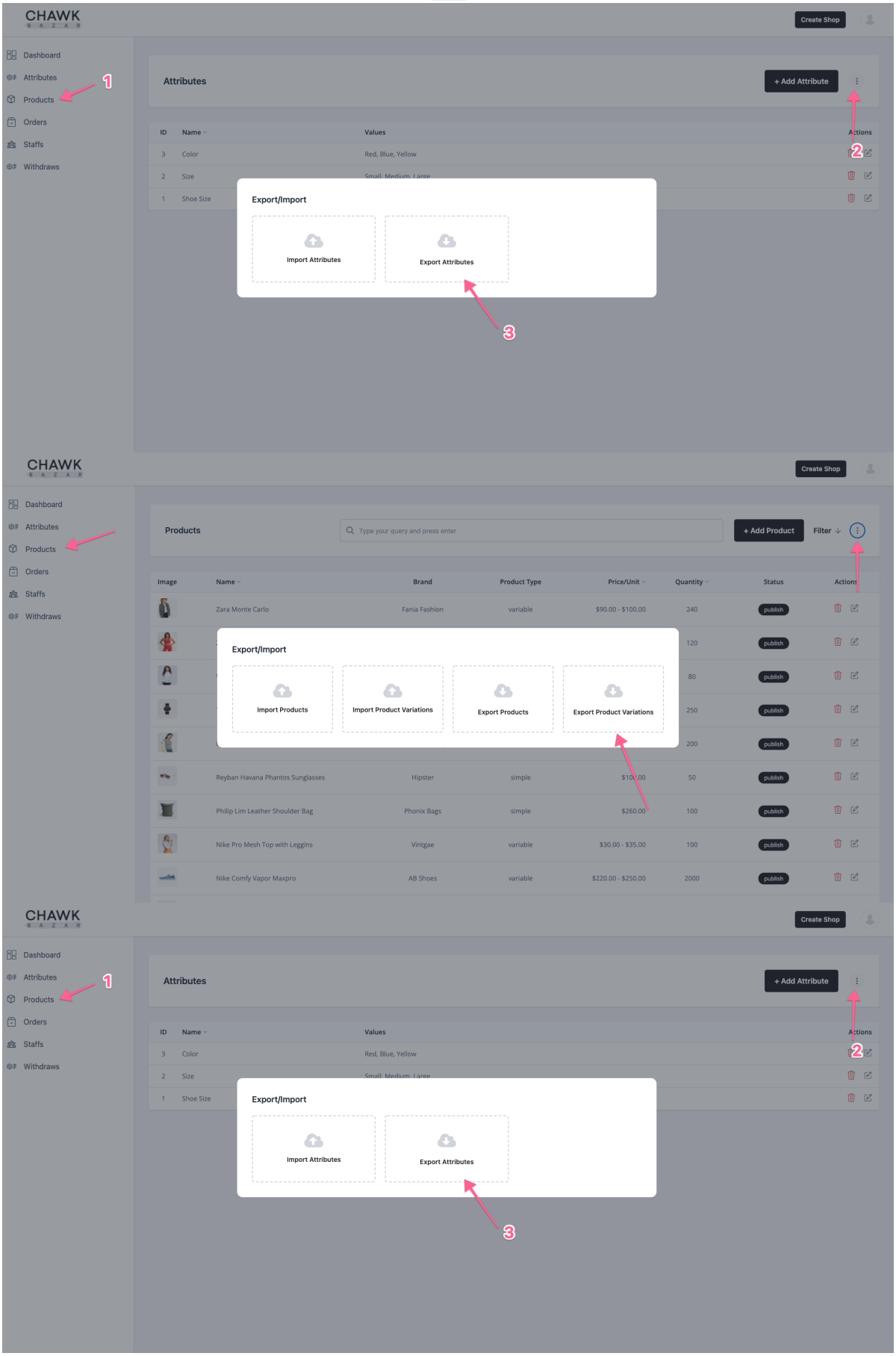


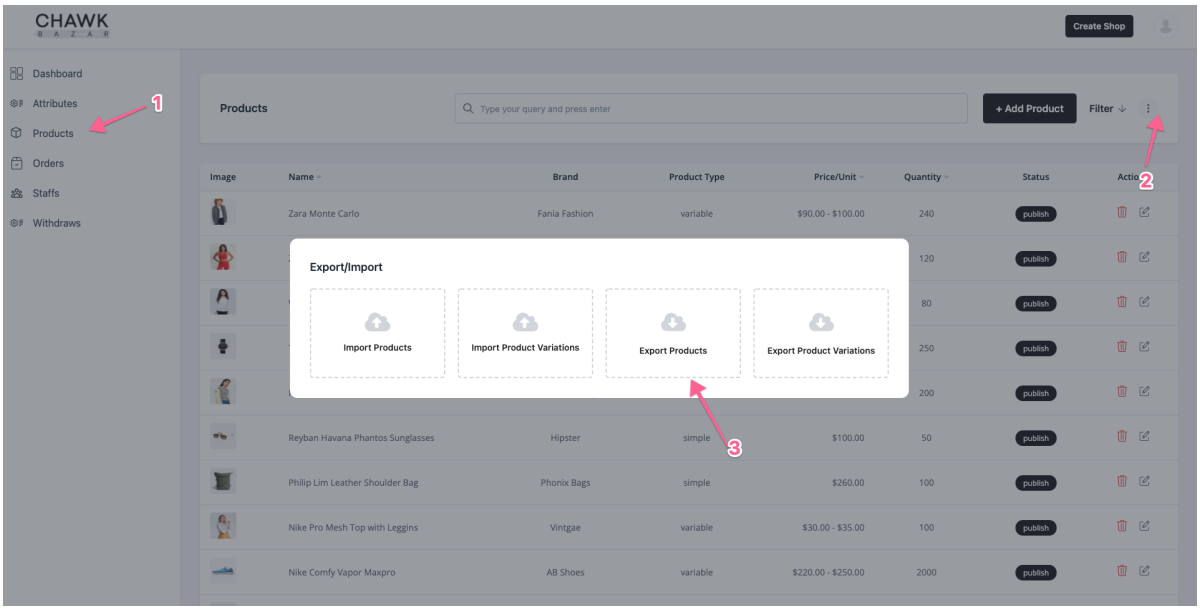
variable products

For variable products, you will need three different `csv.`

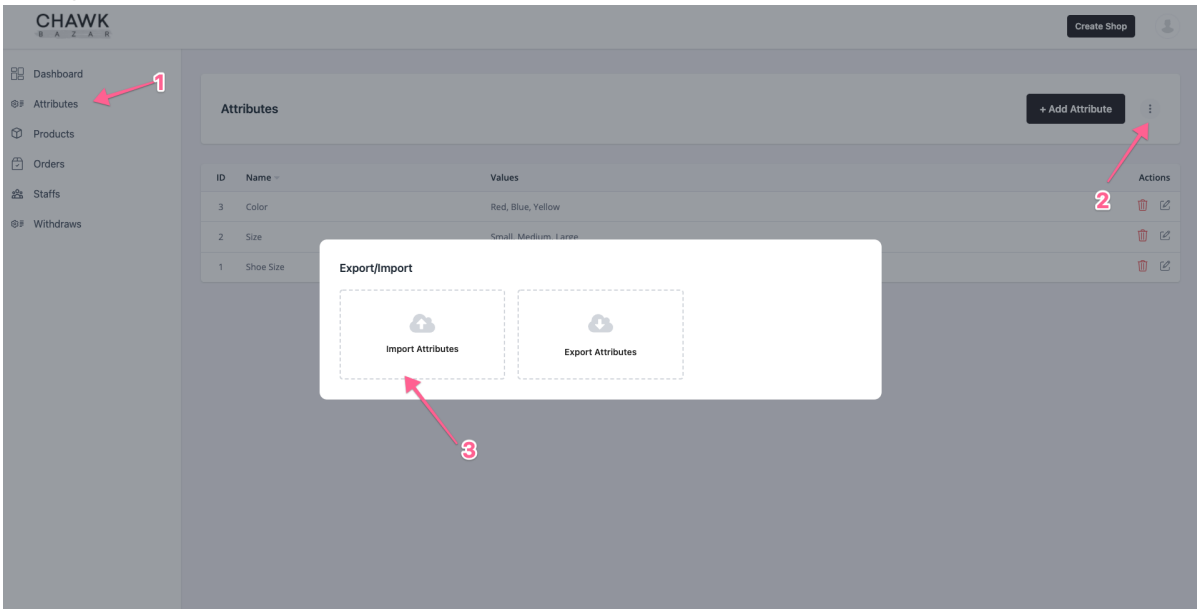
- Attributes `csv`
- products `csv`
- variation options `csv`

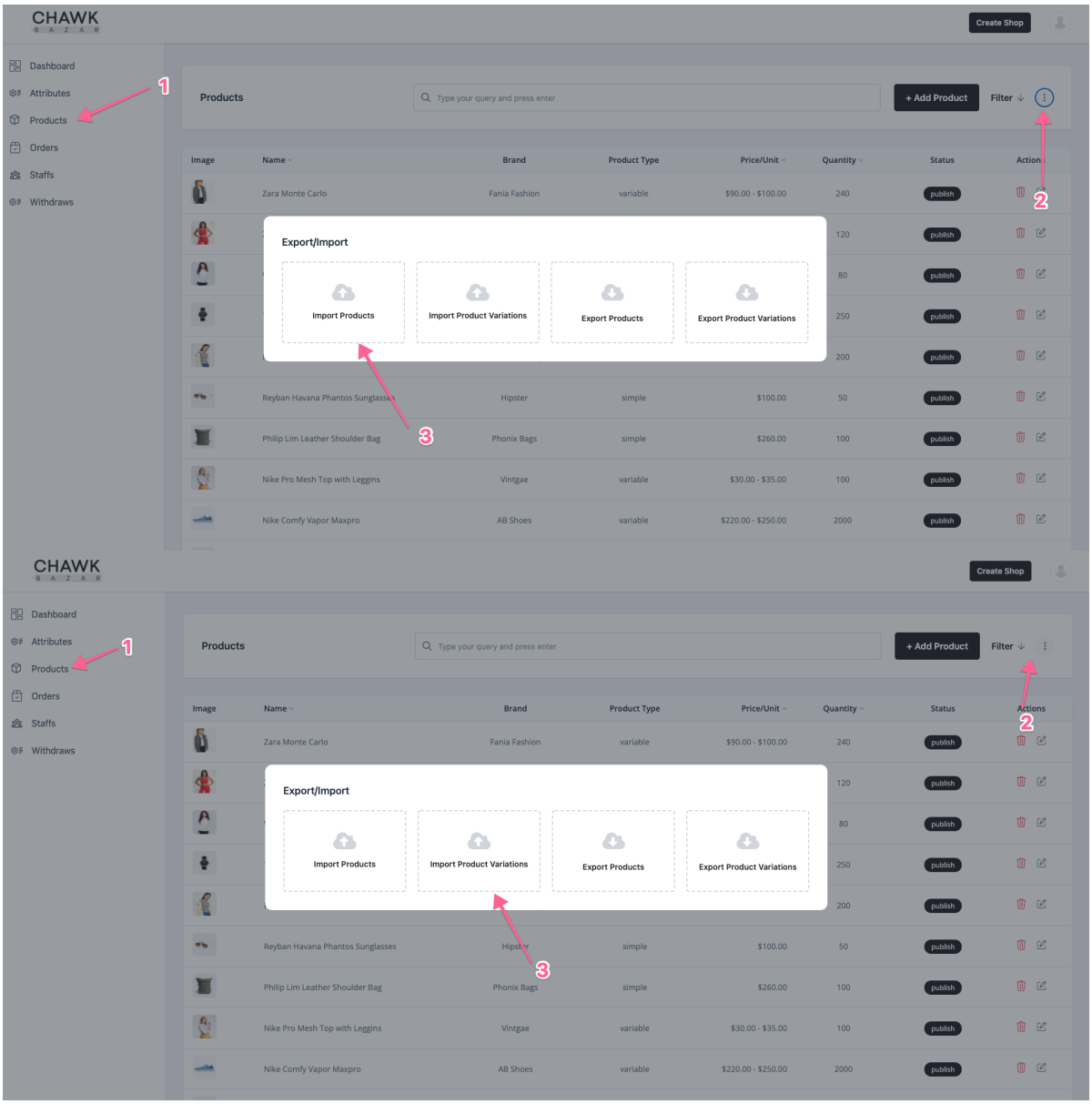
Make sure to re-import you export these three `csv`.





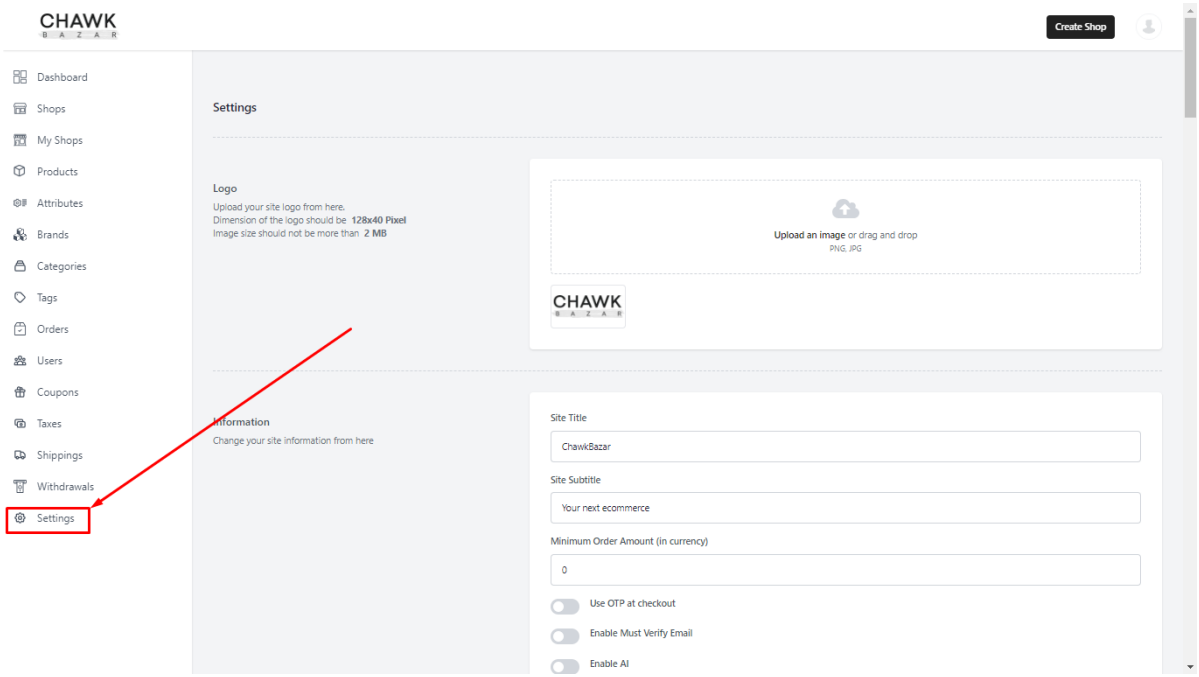
Now, during import variations product,
At First, you have to import Attributes
then Products and
Then products variation.



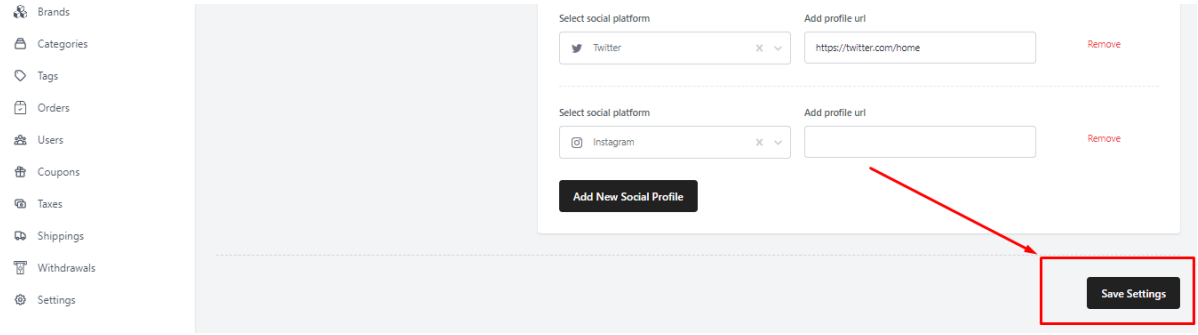


Settings Management

In `settings` menu you will get the settings management form there.You can modify the basic information about your E-Commerce System, such as the Change Logo, Site Title, Site Information , Payment Information, SEO, Delivery Schedule, Shop Settings, etc.

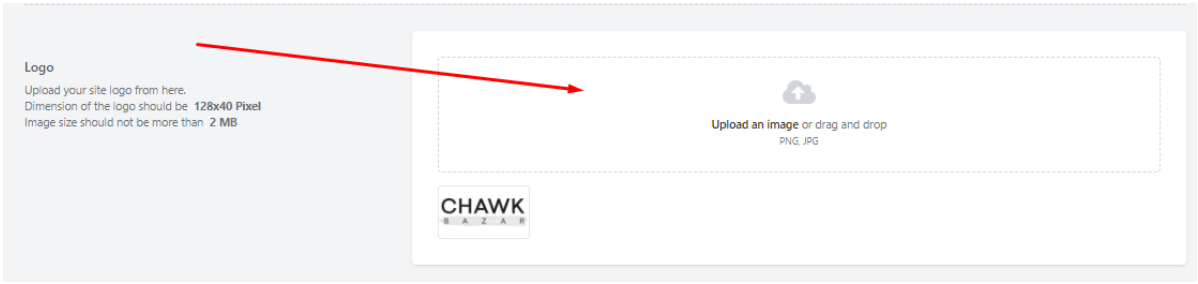


`Note:` If you make changes or update any fields, it's important to save those changes by clicking on the `Save Settings` button bottom of the page.



Change Logo

Click on Upload an image and select the file for your new logo from your Device. or Drag and Drop your new logo. (`Supported Formats JPG or PNG`)



We support two types of file uploads; one is `local,` and another is `AWS S3`.

Both uploading systems follow this procedure,

API

At first, add your API URL to your `api -> .env.`

```
APP_NAME=ChawkBazar
APP_ENV=production
APP_KEY=base64:dladdfdsfdfdsfssdf/sdfdsfer
APP_DEBUG=true
APP_URL=https://YOUR_DOMAIN.COM/backend
APP_SERVICE=chawkbazar.test
APP_NOTICE_DOMAIN=CHAWKBAZAR_

DUMMY_DATA_PATH=chawkbazar

LOG_CHANNEL=stack
LOG_LEVEL=debug

```

> If you're using windows and your API is running with ports like `localhost:8000` or `127.0.0.1:8000`, then make sure you add the domain with port to `APP_URL.` Like this `API_URL=http://localhost:8000`

After that, clear your API cache by using this command from the `api` folder,

```
php artisan optimize:clear
```

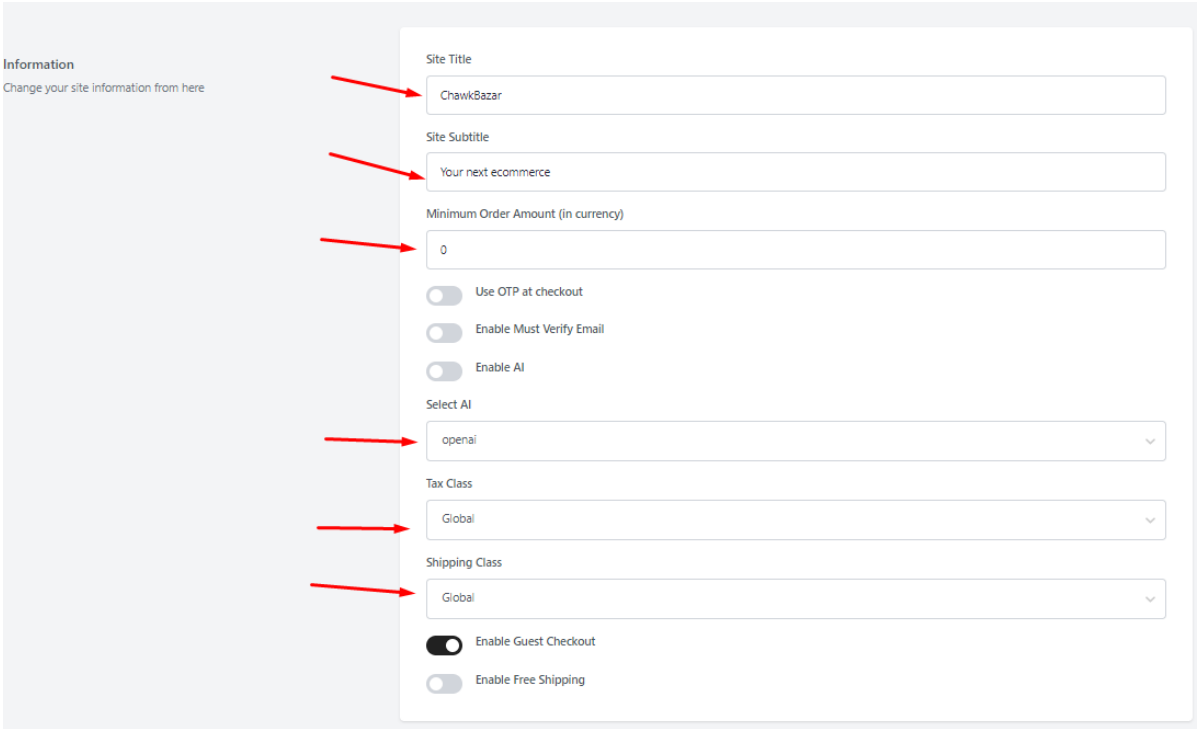
Then link the storage system by using this command,

```
php artisan storage:link
```

> If you're using AWS S3, then update `MEDIA_DISK=s3` and add all `AWS_` credentials
> For AWS s3, make sure your properly setup permission of the bucket with ACL enable by follow this link -> https://stackoverflow.com/a/70603995/2158023

Change Site information

This is the Site `Information` area of the admin settings. Here you can change your Site Title, Site Subtitle, Currency, Minimum Order Amount, Wallet Currency Ratio, Sign Up Points, Maximum Question Limit, OTP checkout, Tax Class, Shipping Class and handle Free Shipping.

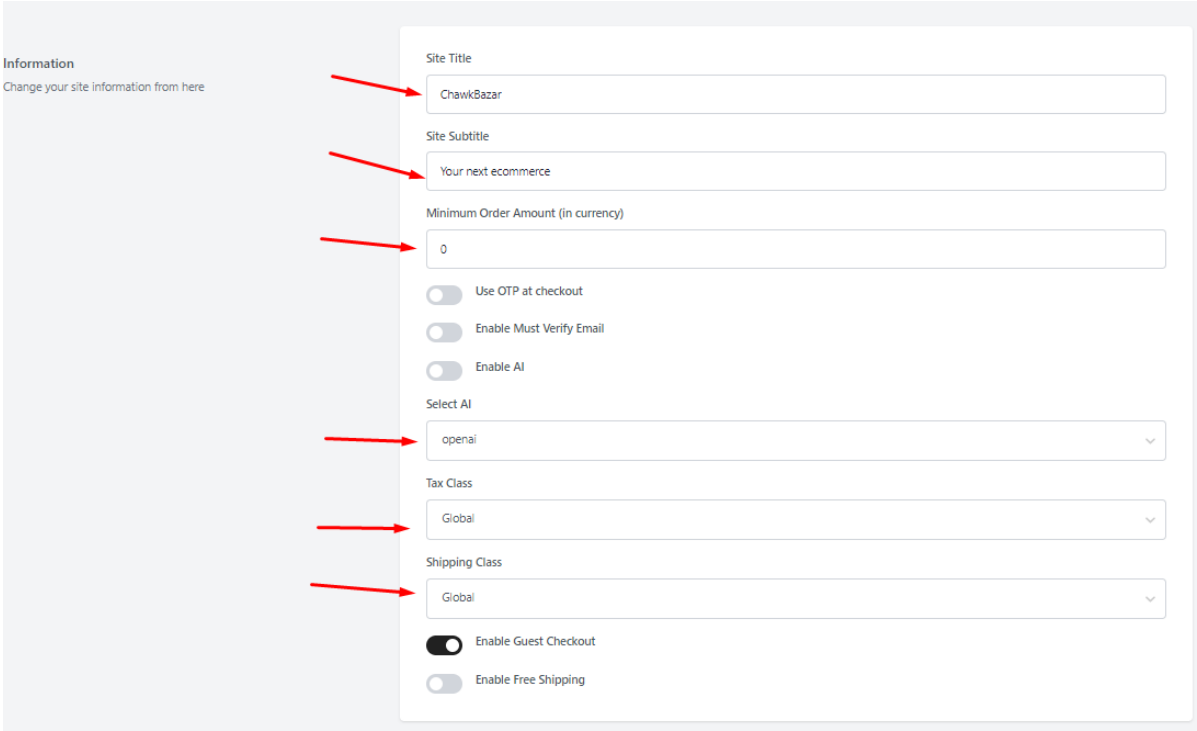


Shipping class, Tax class & Free Shipping

You may find the shipping class at the conclusion of the information section. You may choose the precise class you wish to utilize from this menu. These classes were developed within the shipping category. The classes include a value for shipping that should be added to the total price bought.

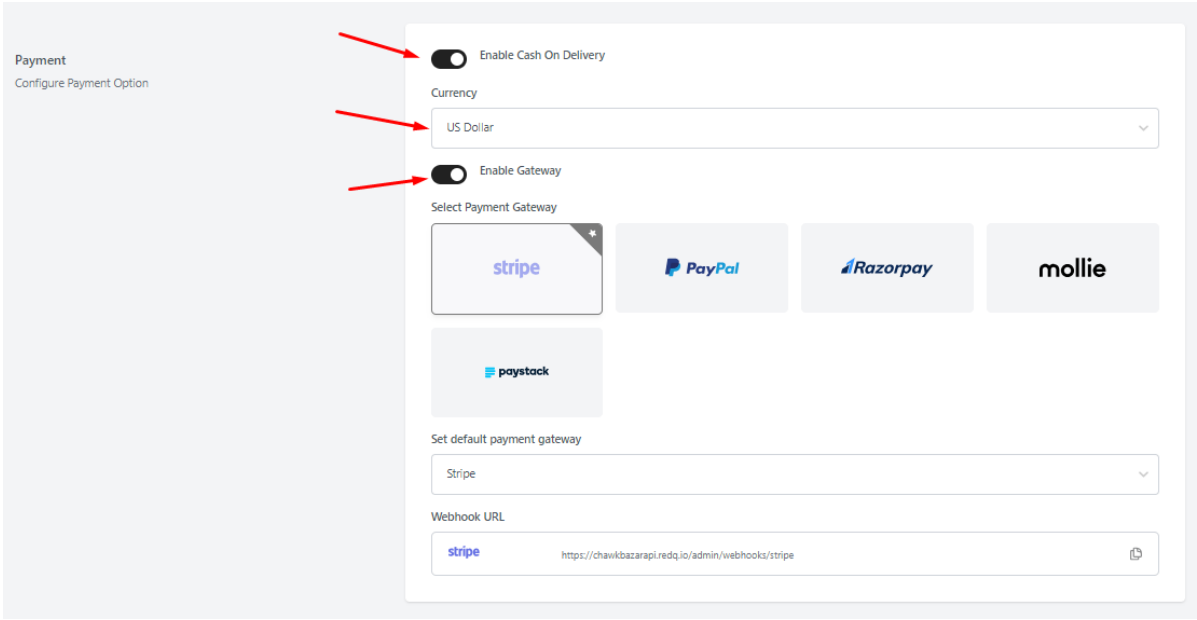
For tax classes, the same applies. You may create tax classes in the admin settings' tax area by entering a dollar amount or percentage that will be added to the total amount of your order.

The option to allow free delivery is another one. It may be turned off to stop offering free delivery. Customers will receive free delivery if it is activated. If not, the cost of shipment must be paid.



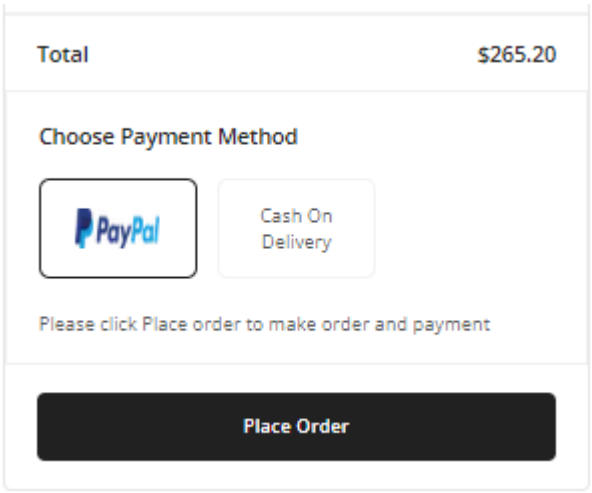
Payment Gateway Management

This is the `payment` area of the admin settings. This will allow you to choose a payment gateway such as Stripe, Rezorpay, Paypal, Mollie Paystack etc. You can also enable or disable Cash On Delivery.



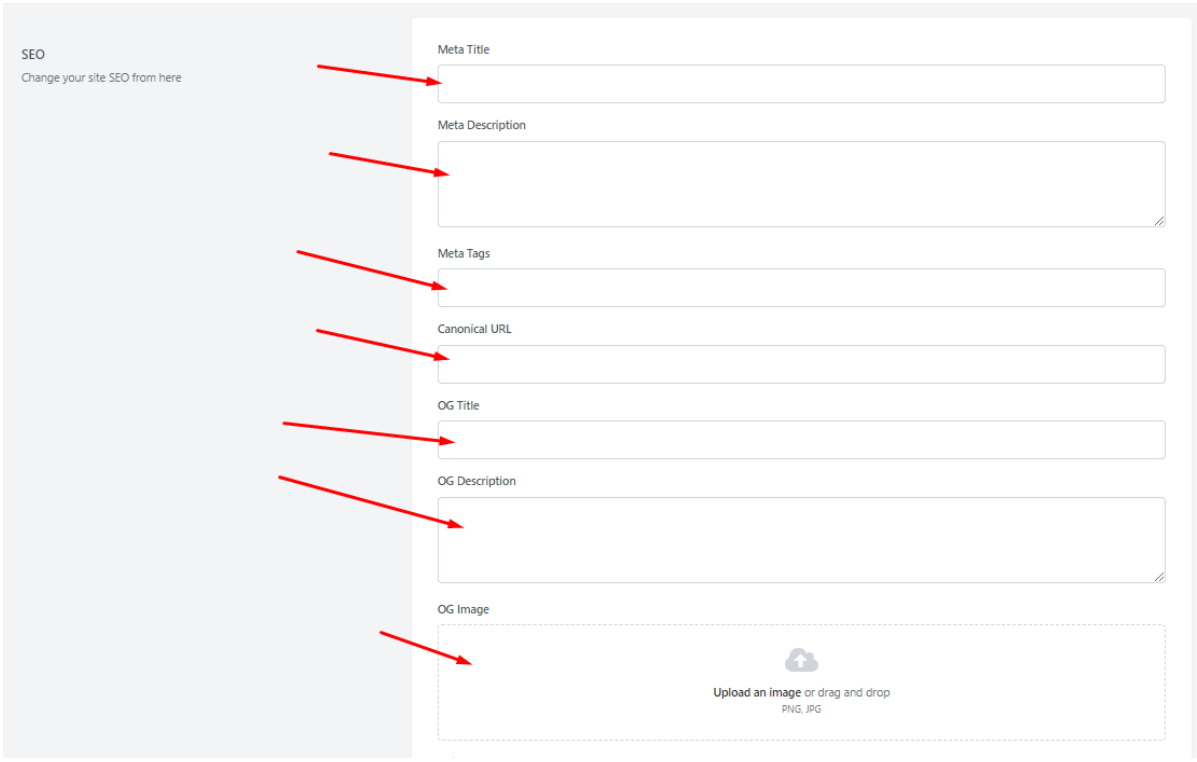
Customer

When a client goes to a store to buy anything, he adds it to his cart. When he goes to pick a payment option, he will see `Cash on delivery` as well as payment gateways such as Stripe, Paypal, Razorpay, and Mollie, which are configured in the payment settings area of the dashboard.



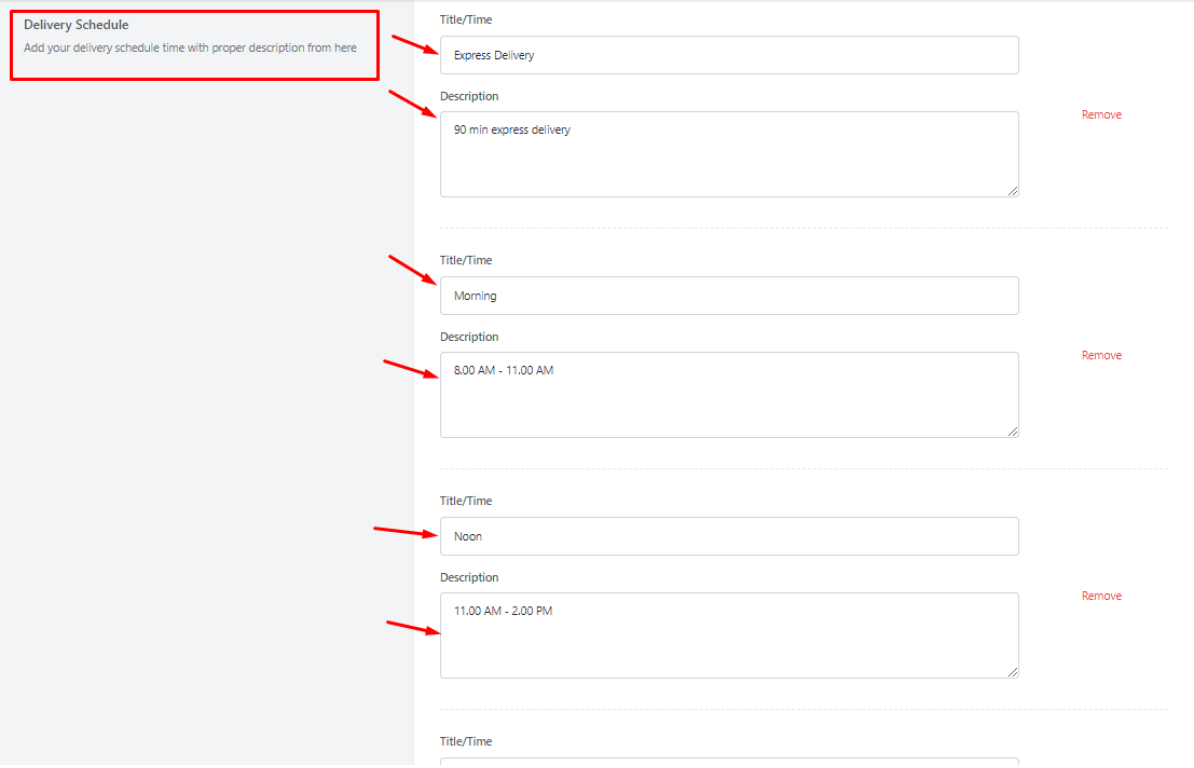
SEO

Search Engine Optimization (`SEO`) is a process of optimizing your website to rank higher in search engine results pages (SERPs) for specific keywords and phrases. you can add Meta Title, Meta Description, Meta Tags, Canonical URL, OG Title, OG Description etc.



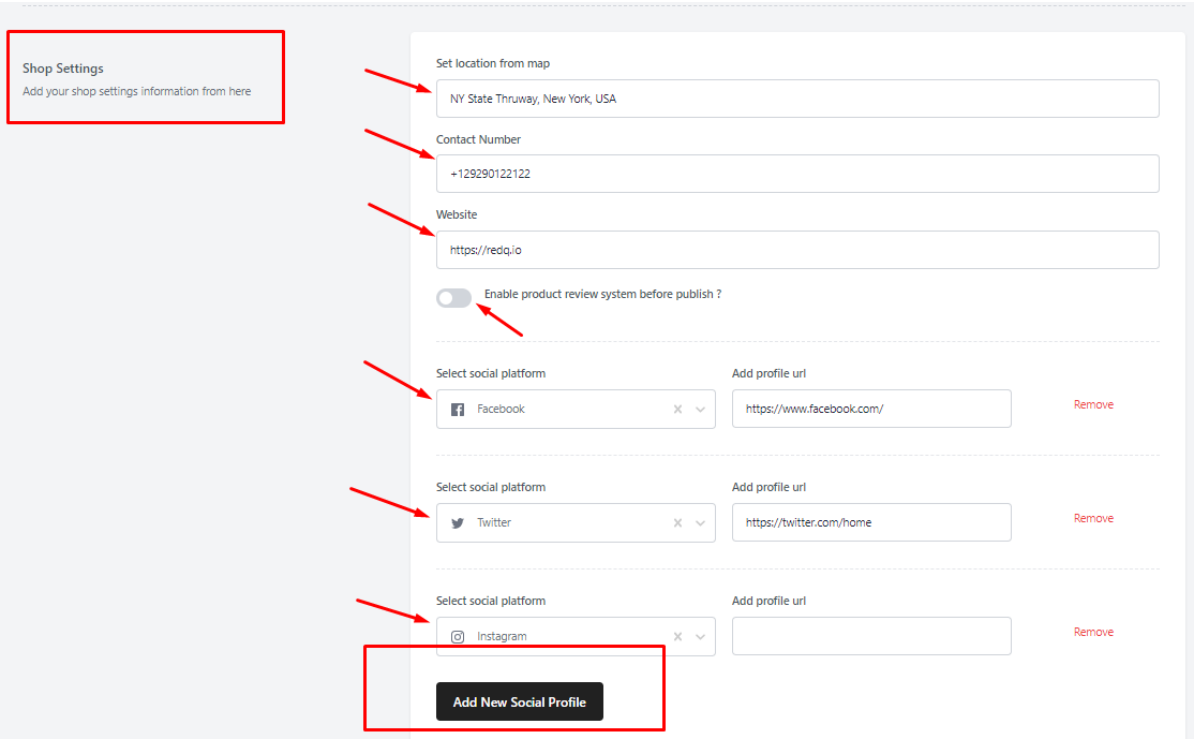
Add or Remove Delivery Schedule

This is the `Delivery Schedule` area of the settings. This will allow you to add new Delivery Schedule such as Morning, Noon, Afternoon, Evening etc. You can also remove any existing Delivery Schedule.



Shop Settings & Google Map

This is the `Shop Settings` area of the settings. This will allow you to change location from map, Contact Number, Website, Enable or Disable Google Map Address, or add new social platform account, etc.



New Static Page

Both shop and admin are built using React NextJS framework. So all the existing pages are available to this location,

Shop,

```
/chawkbazar-laravel/shop/src/pages
```

Admin,

```
/chawkbazar-laravel/admin/rest/src/pages
```

You can use the NextJS routing feature for the new pages. Check these official NextJS docs for pages and routing,

https://nextjs.org/docs/basic-features/pages

https://nextjs.org/docs/routing/introduction

FAQ:

How to change the default layout?

Please go to,

```
chawkbazar-laravel/shop/src/pages
```

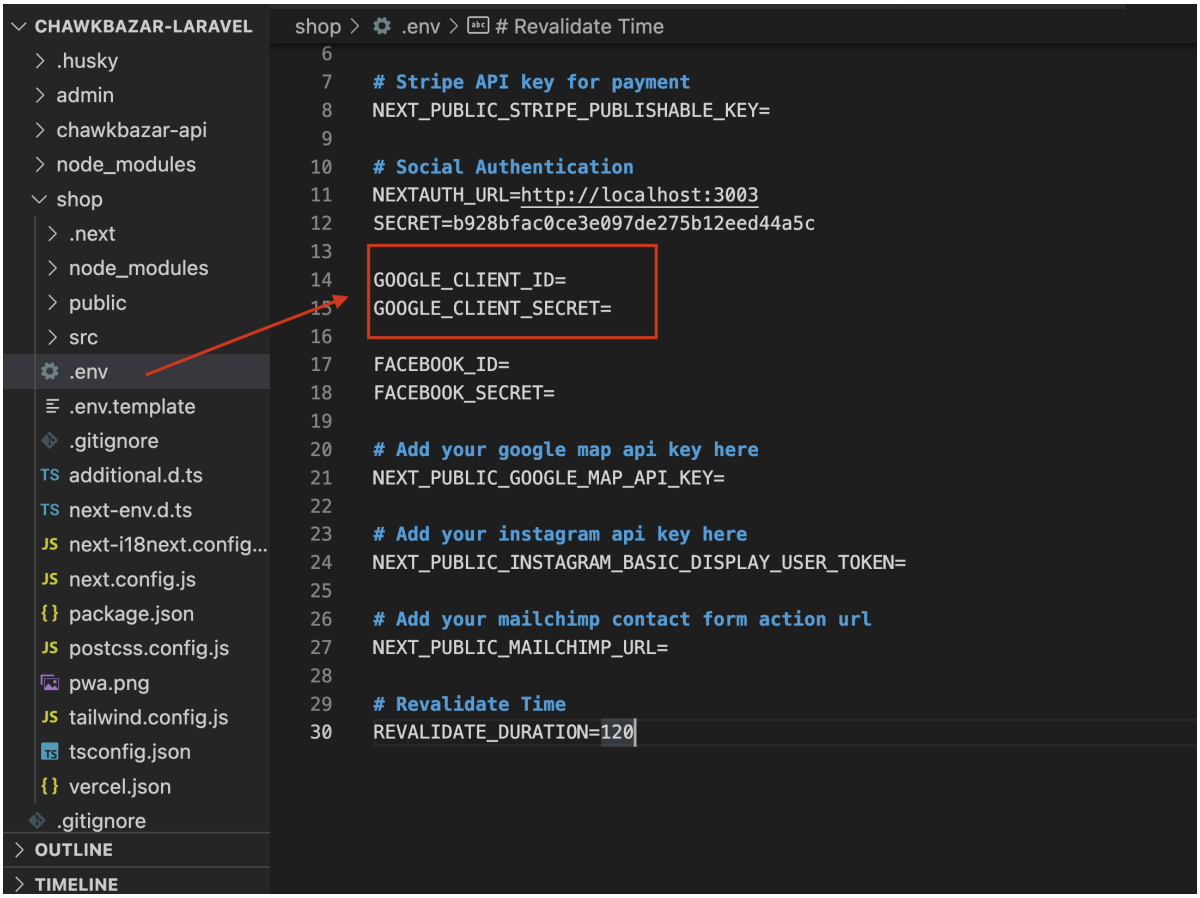
And delete or change the name of `index.tsx`

Then rename `your specific demo.tsx` to `index.tsx`

After that, make sure you rebuild your project to run in production.

Why showing client_id error while Google social login

Ans: Go to `.env` file and change the variables from `GOOGLE_ID=` to `GOOGLE_CLIENT_ID=` and `GOOGLE_SECRET=` to `GOOGLE_CLIENT_SECRET=`. Make this changes in the `.env` file of chawkbazar-api and shop directory.

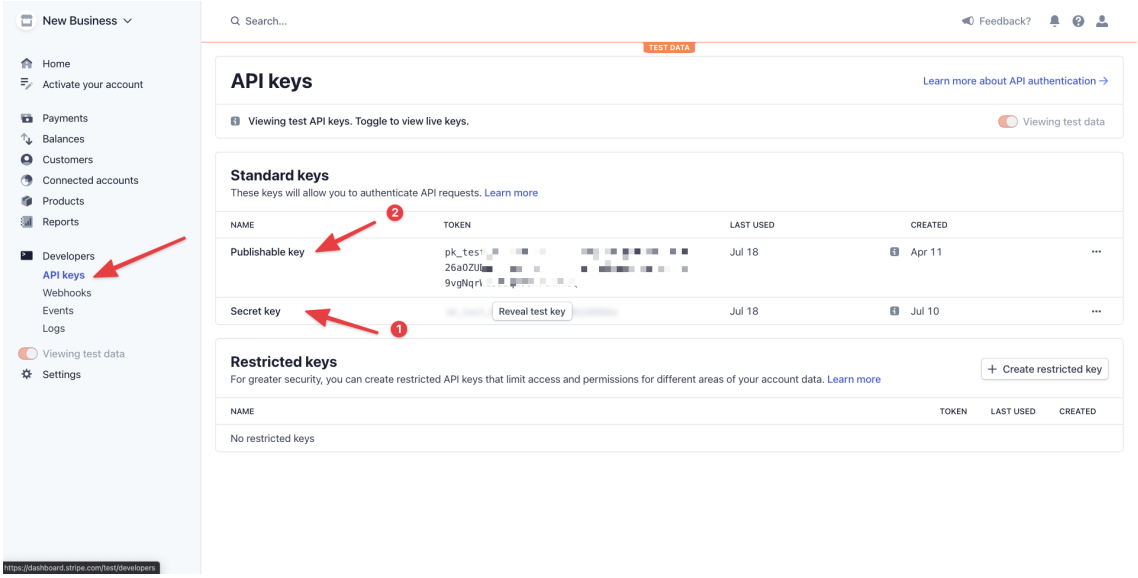


How to configure Stripe payment gateway?

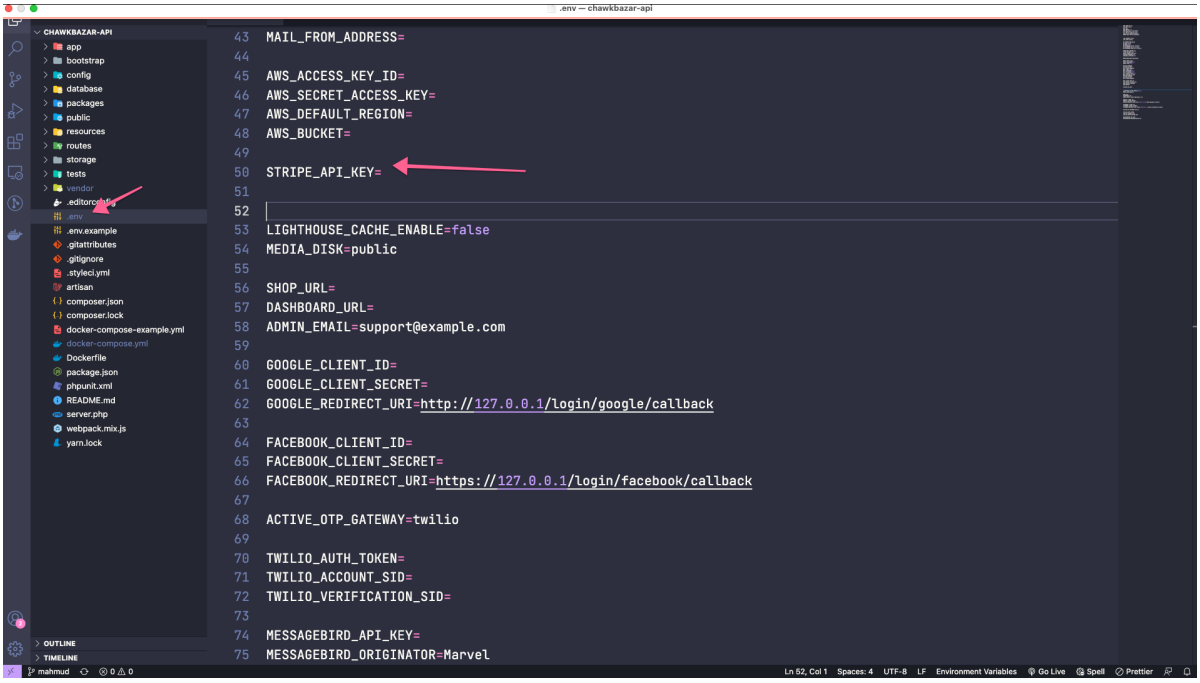
To configure `Stripe` payment gateway,

1. At first, create a developer account from stripe developer dashboard (https://dashboard.stripe.com/)
2. log in to that account

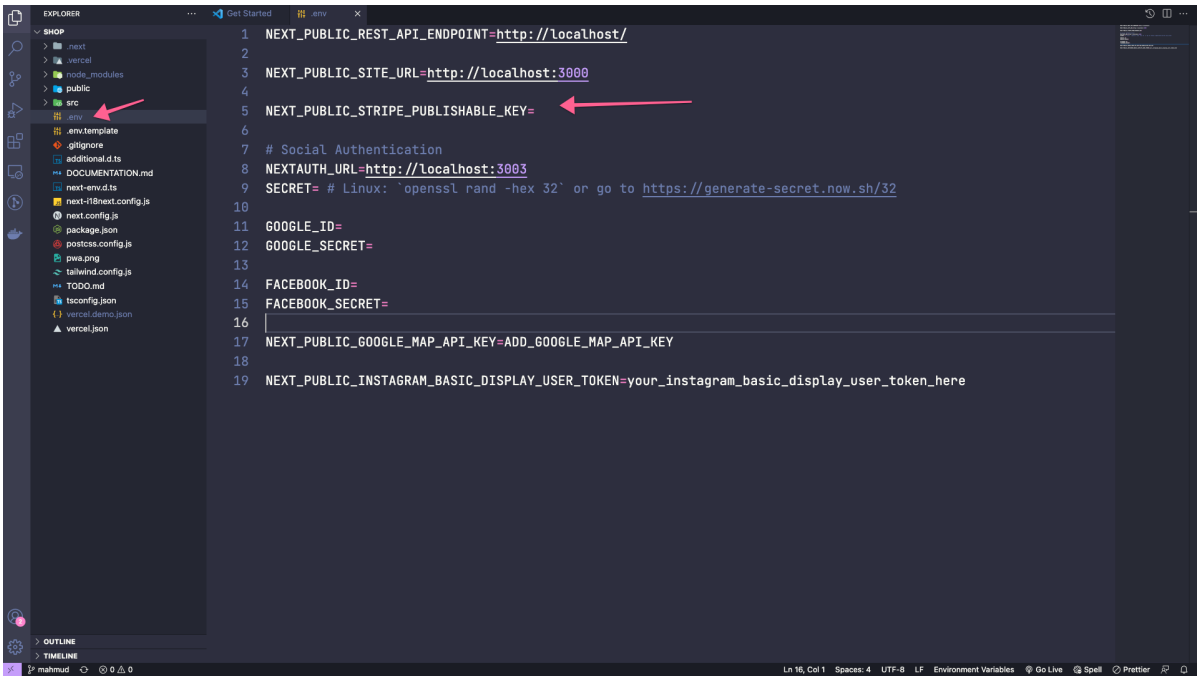
3. Then go to the `Developer -> API Keys` section and on that section create an API key. After creating, you'll get one `Secret key` and one `Publishable key.`



4. Then copy `Secret key` to your `chawkbazar` `api->.env`



5. And similarly, add `Publishable key` to your `frontend/shop/.env`



After configuration, make sure you rebuild your project using this command,

For REST API

```
yarn build:shop-rest
yarn build:admin-rest
```

Why am I facing "You may need an appropriate loader to handle this file type" during running shop rest?

Ans: Please run `yarn clean` then `yarn` then run `yarn dev:shop-rest`

I am changing schema files but changes is not working

Your changes might not work because schema is cached. SO you might need to clear schema cache using the below command `php artisan lighthouse:clear-cache`.

Changing .env files but not getting the changes

Run Below command `php artisan optimize:clear`

Changing route but not getting the changes.

Run `php artisan optimize:clear` or `php artisan route:clear`

I have set `STRIPE_API_KEY` in .env but still getting error.

In some cases `STRIPE_API_KEY` value can't read from .env in those cases you have to put the key in the config file directly in `api/packages/marvel/src/Config/laravel-omnipay.php`

6. Getting error on forget password email sending
   Make sure you have run the `php artisan marvel:install` commands successfully and copied the necessary email templates to your resources folder. You can also do it by `php artisan marvel:copy-files` command.

   > NB: This same issue can occur during order creation.

Can I use it with my existing laravel?

Yes, you can. Follow the below steps.

- Make sure you are using laravel 8.
- Copy `api/packages` folder from the downloaded files into your laravel root folder.
- Put below code in your laravel `composer.json` file into `require` section.

```
"ignited/laravel-omnipay": "dev-master",
"omnipay/common": "dev-master",
"omnipay/stripe": "dev-master",
"chawkbazar/shop": "dev-master"
```

- Put below code in bottom of your `composer.json`. If you already have an `repositories` section then put code
  inside `repositories` to your existing `repositories`

```
"repositories": {
        "chawkbazar/shop": {
            "type": "path",
            "url": "packages/marvel"
        },
        "ignited/laravel-omnipay": {
            "type": "path",
            "url": "packages/laravel-omnipay"
        },
        "omnipay/common": {
            "type": "path",
            "url": "packages/omnipay-common"
        },
        "omnipay/stripe": {
            "type": "path",
            "url": "packages/omnipay-stripe"
        }
    }
```

- Now run `composer install`
- Copy necessary env variables from .env.example to you env file.
- Run `php artisan marvel:install` and follow necessary steps.
- To run server `php artisan serve`
- For image upload to work properly you need to run `php artisan storage:link`.

Why am I getting `Access denied for user`?

navigate to `api` then run `./vendor/bin/php down -v`. It will delete any of your existing mysql volumes. Now run `./vendor/bin/php up -d` on same directory or run `bash install.sh` on root directory

Why am I getting permission issue during deployment?

Run below commands for fixing permission issue in your laravel app during deployment

```
sudo chown -R $USER:www-data storage
sudo chown -R $USER:www-data bootstrap/cache

sudo chown -R www-data:www-data storage
sudo chown -R www-data:www-data bootstrap/cache
```

Why am I getting "The GET method is not supported for this route. Supported methods: HEAD"?

Run `php artisan optimize:clear`

I am getting "Cannot use 'Mixed' as class name as it is reserved" after access graphql endpoint.

You are using the application in PHP 8. Downgrade it to PHP 7.4. Mixed is a reserved word in PHP 8.

How to resolve the `Load More` `Infinity` loading issue?

If you click on the `Load More` button and the button is spinning continuously, then check API request from the network tab. If it is HTTP and your API is hosted on HTTPS, then open,

```
/Users/tareqmahmud/tmbox/tmserver/larabox/work/redq/laravel-
ecommerce/api/app/Providers/AppServiceProvider.php
```

And add this code to `boot` method,

```
if (!\App::environment('local')) {
    $this->app['request']->server->set('HTTPS', true);
}
```

I'm trying to upload images, but the images are not displayed on the frontend?

We support two types of file uploads; one is `local,` and another is `AWS S3`.

Both uploading systems follow this procedure,

API

At first, add your API URL to your `chawkbazar-api -> .env.`

```
APP_NAME=ChawkBazar
APP_ENV=production
APP_KEY=base64:dladdfdsfdfdsfssdf/sdfdsfer
APP_DEBUG=true
APP_URL=https://YOUR_DOMAIN.COM/backend
APP_SERVICE=chawkbazar.test
APP_NOTICE_DOMAIN=CHAWKBAZAR_

DUMMY_DATA_PATH=chawkbazar

LOG_CHANNEL=stack
LOG_LEVEL=debug

```

> If you're using windows and your API is running with ports like `localhost:8000` or `127.0.0.1:8000`, then make sure you add the domain with port to `APP_URL.` Like this `API_URL=http://localhost:8000`

After that, clear your API cache by using this command from the `chawkbazar-api` folder,

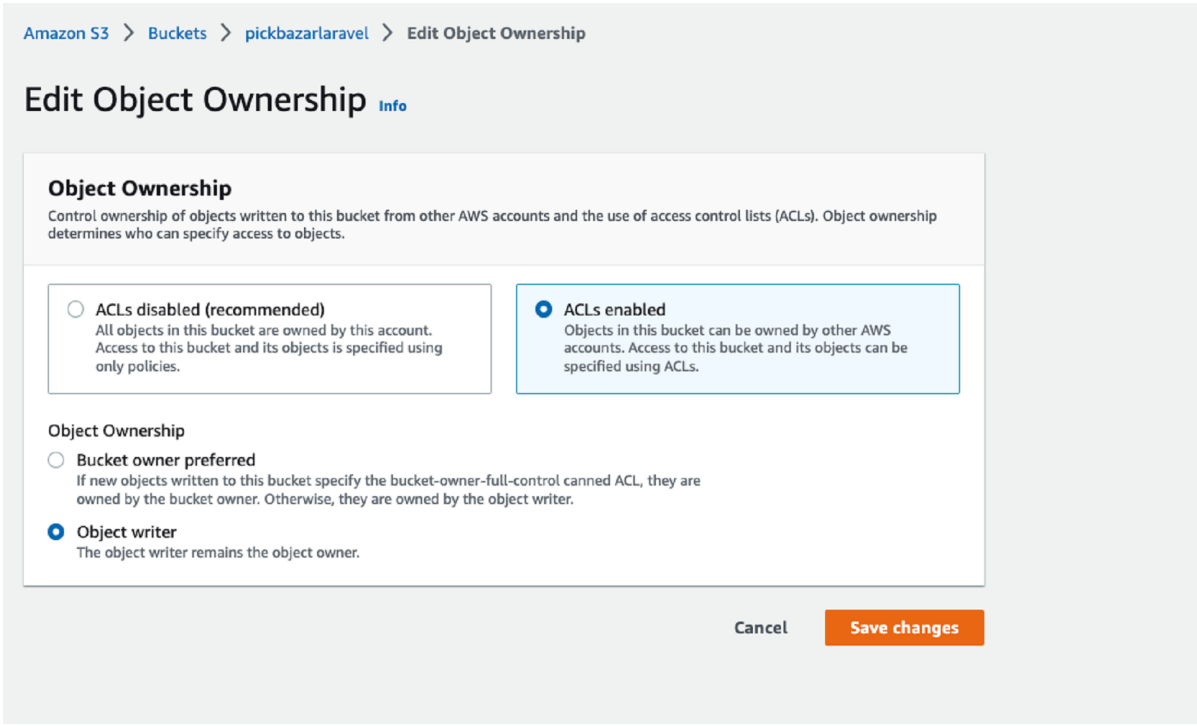
```
php artisan optimize:clear
```

Then link the storage system by using this command,

```
php artisan storage:link
```

> If you're using AWS S3, then update `MEDIA_DISK=s3` and add all `AWS_` credentials

> For AWS s3, make sure your properly setup permission of the bucket with ACL enable by follow this link -> https://stackoverflow.com/a/70603995/2158023



Admin

Then add your domain to your `chawkbazar-laravel -> admin -> rest -> next.config.js`

```
images: {
  domains: [
    "YOUR_DOMAIN.COM",
    "via.placeholder.com",
    "res.cloudinary.com",
    "s3.amazonaws.com",
    "18.141.64.26",
    "127.0.0.1",
    "localhost",
    "picsum.photos",
    "pickbazar-sail.test",
    "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "lh3.googleusercontent.com",
  ],
},
```

> If you're using `AWS S3`, then also add your root S3 domain like this,
>
> 
>

Shop

Similarly, add your domain to your `chawkbazar-laravel -> shop -> next.config.js`

```
images: {
  domains: [
    "YOUR_DOMAIN.COM",
    "via.placeholder.com",
    "res.cloudinary.com",
    "s3.amazonaws.com",
    "18.141.64.26",
    "127.0.0.1",
    "localhost",
    "picsum.photos",
    "pickbazar-sail.test",
    "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "lh3.googleusercontent.com",
  ],
},
```

> If you're using `AWS S3`, then also add your root S3 domain like this,
>
> 
>

Build Frontend,

If you run in dev mode, then re-run will affect your update. But if you're running as a production mode, then you've to rebuild the project and re-run again. Otherwise, image upload will not work.

https://chawkbazar-laravel-doc.vercel.app/vps-server#build-project

After building the project, replace your server shop and admin folder using this rebuild folder and re-run the app using this command,

```
pm2 restart all
```

How to resolve the `502 Bad Gateway` error for frontend/admin?

There can be several reasons that throw 502 errors for the frontend in the production server.

Reason 1:

On your `Nginx file, a specific port is set for shop or the admin, so when you run the script as a` PM2` instance, the script has to be run that specific port; otherwise, your domain/subdomain will throw a 502 error.

PORT:

```
3003 -> Shop Rest
3002 -> Admin Rest
```

To check that at the first stop the PM2 instance by this command,

```
pm2 stop 'all'
```

then go to the `chawkbazar` folder from your server and try to run the script using yarn and check which port it is running,

`Rest:`

```
yarn start:admin-rest
```

```
yarn shop-rest
```

And, check if the port matched with the Nginx port or not? If not matched, then change the port to `Nginx` config.

Reason 2:

To check that at the first stop the PM2 instance by this command,

> pm2 stop 'all'

then go to the `chawkbazar` folder from your server and try to run the script using yarn and check which port it is running,

`Rest:`

`yarn start:admin-rest`

`yarn shop-rest`

With this command, it'll give you an error that no build file is found. To resolve that, rebuild your project and then rerun the scripts.

https://chawkbazar-doc.vercel.app/vps-server#frontend-project-build

How to resolve `javascript heap out of memory` issue?

If you're using cPanel based server, then you maybe get an error like "call_and_retry_last allocation failed - javascript heap out of memory".

To resolve that issue, you've to add a limit of the memory bound,

Please check this,

https://stackoverflow.com/questions/38558989/node-js-heap-out-of-memory

https://support.snyk.io/hc/en-us/articles/360002046418-JavaScript-heap-out-of-memory

https://blog.openreplay.com/javascript-heap-out-of-memory-error

Image upload throw Internal Server Error; how to resolve that?

For uploading, the php8.1-gd library is required. So make sure you install the php8.1-gd library on your server.

Sometimes my server shutdown or halts automatically. After restart, it works fine again; how to resolve that?

In general, this happens mainly when your upstream server fails to handle its processing power when there is lots of traffic or requires lots of operation. Please upgrade your server configuration to double to check the issue. Please check this video,

https://www.youtube.com/watch?v=4pIqAz5bzX0

If that doesn't solve the issue, then you've to redeploy again with a new ec2 instance.

S3 uploading doesn't work after adding credentials to .env; how to resolve that?

For AWS s3, make sure your properly setup permission of the bucket with ACL enable by follow this link

https://stackoverflow.com/a/70603995/2158023

How to rebuild the project?

When you upload the code to the server as a production, NextJS fetch its code from the build file. So after deployed, when you update or change code on scripts, the updated scripts don't work at the front end until you rebuild the project again. NextJs works this way.

Now there are two ways you can follow to rebuild the project,

If your server has more than 2+ CPU core and 6GB+ of memory, then you can directly edit code at your server then rebuild the project from the server directly using this command,

At first go to `/var/www/chawkbazar-laravel`

For `shop,`

```
yarn build:shop-rest
```

And for `admin,`

```
yarn build:admin-rest
```

After completing the build, restart the PM2 process using this command,

```
pm2 restart 'all'
```

But suppose your server doesn't have that processing power, and you try to build your scripts directly from the server. In that case, your server will be shut down during the build, and you've to restart the server.

In that case, you can rebuild the scripts at your local computer and upload them to your server. If you follow our VPS Server Deployment guide, then you already know the process. To build the scripts on your pc, follow this procedure,

At first, download the `chawkbazar-laravel` folder from the server to your computer, and customize the code at your computer. Then use this command from the `root` folder,

To install all the packages,

```
yarn
```

Then build the shop,

```
yarn build:shop-rest
```

Then build the admin,

```
yarn build:admin-rest
```

Then delete the shop and admin folder from the server and upload your build shop and admin folder to the server at the exact same location and use this command to restart the PM2,

```
pm2 restart 'all'
```

How to increase upload size?

The API supports `10MB` of upload limit for a single file by default. If your upload file is more than `10MB`, then please follow this,

At first, login to the server using ssh, and then,

Edit,

```
/etc/php/1/fpm/php.ini
```

Then update,

```
upload_max_filesize = 256M
post_max_size = 256M
max_execution_time = 300
```

Then restart the PHP,

```
sudo service php8.1-fpm restart
```

Then update,

```
/etc/nginx/sites-enabled/chawkbazar
```

and Add `client_max_body_size 256M`

```
client_max_body_size 256M
```

```
server {

    server_name [redacted].
        client_max_body_size 256M;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.html index.htm index.php;

    charset utf-8;
```

Then restart the server,

```
sudo nginx -t
sudo service nginx restart
```

After that update,

```
chawkbazar-api/packages/marvel/config/media-library.php
```

```
'max_file_size' => 1024 * 1024 * 256,
```

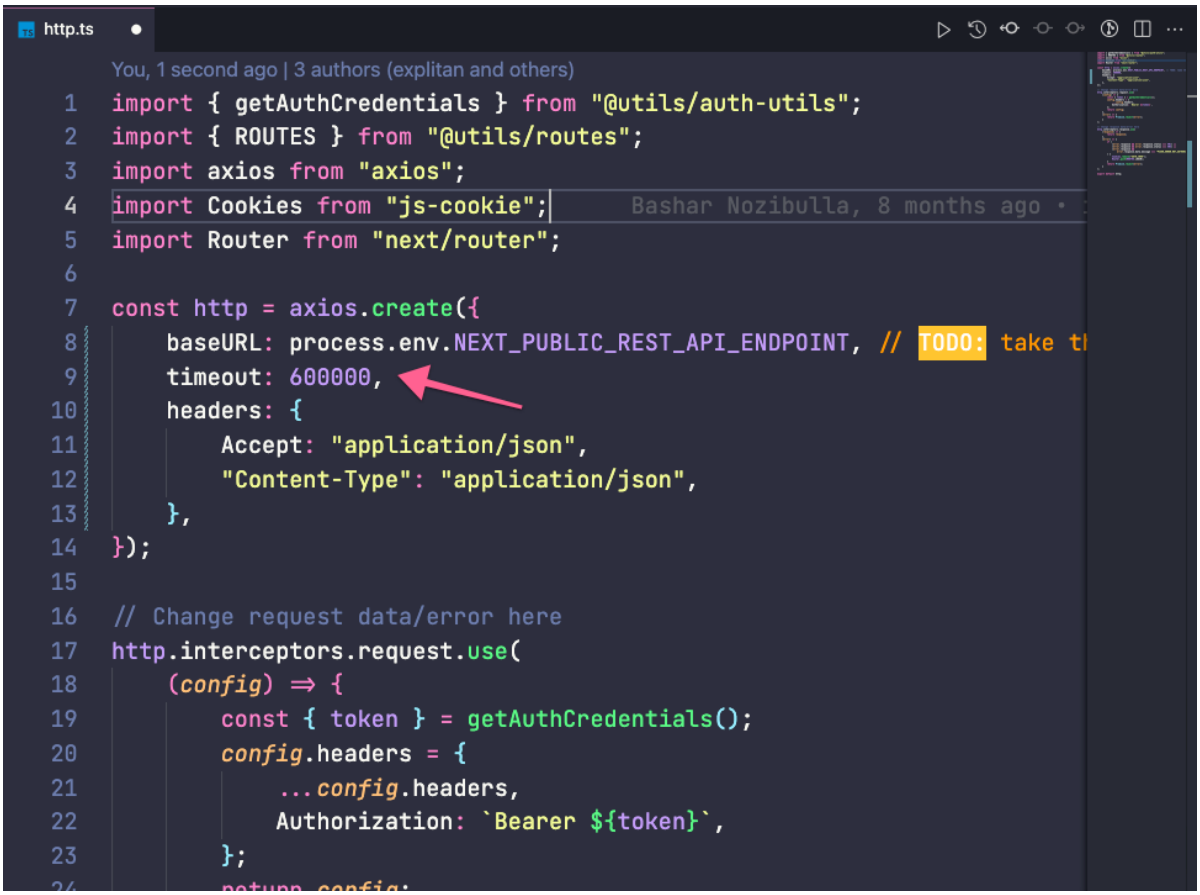
```
...
<?php

return [

    /*
     * The disk on which to store added files and derived images by defa
     * one or more of the disks you've configured in config/filesystems
     */
    'disk_name' => env('MEDIA_DISK', config('shop.media_disk')),

    /*
     * The maximum file size of an item in bytes.
     * Adding a larger file will result in an exception.
     */
    'max_file_size' => 1024 * 1024 * 256,

    /*
     * This queue will be used to generate derived and responsive images
     * Leave empty to use the default queue.
     */
    'queue_name' => '',

    /*
```

After that, run this command from the `chawkbazar-api.` folder,

```
php artisan optimize:clear
```

And for `admin`, increase `timeout` from,

```
admin/rest/src/utils/api/http.ts
```




```
timeout: 600000
```

After that Rebuild Your Project and restart again.

How to resolve `docker: invalid reference format: repository name must be lowercase.`?

Please open your terminal and use this command,

```
brew services stop mysql
```

It'll stop any MySQL instance that is installed using brew.

Then use this command to check is there MySQL running or not,

```
mysql -v
```

If no MySQL is running then, go to,

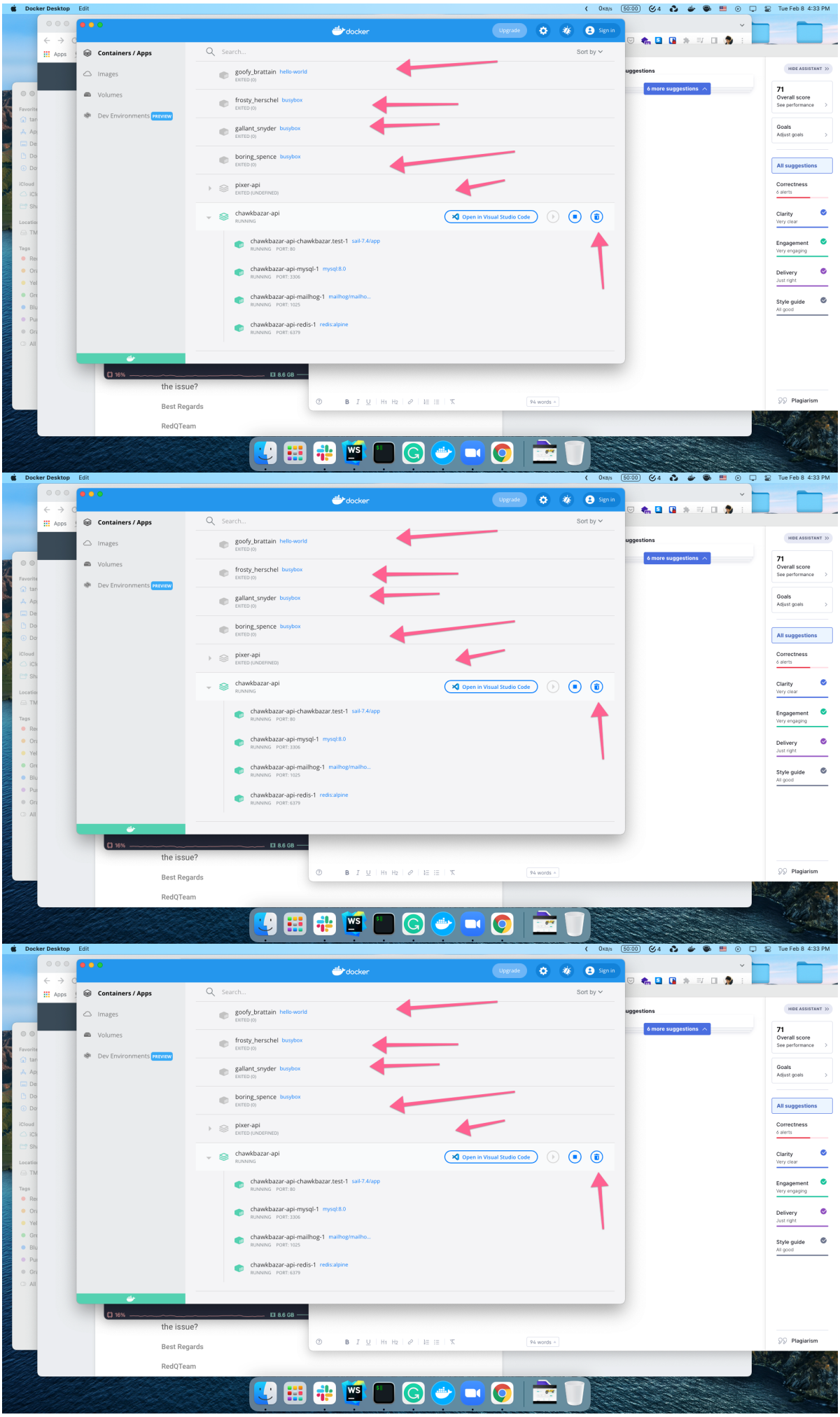
```
chawkbazar-laravel -> chawkbazar-api
```

and use this command,

```
./vendor/bin/sail down -v
```

After that,

Open docker dashboard and delete all the existing container and images,



And after that remove,

```
chawkbazar-laravel -> chawkbazar-api -> .env
```

And then go to your root `chawkbazar-laravel` folder and use this command again,

```
chmod +x install.sh
bash install.sh
```

How to remove the existing payment gateway?

You can remove the Stripe Payment from the available payment array,

```
shop/src/components/checkout/payment/payment-grid.tsx
```

Why checkout `Place Order` button is disabled?

The place order button will be activated only when you fill in this information,

- OTP (Mobile number)
- Billing Address
- Shipping Address
- Delivery Schedule
- Payment Gateway

How do I log in to the admin panel as the main administrator?

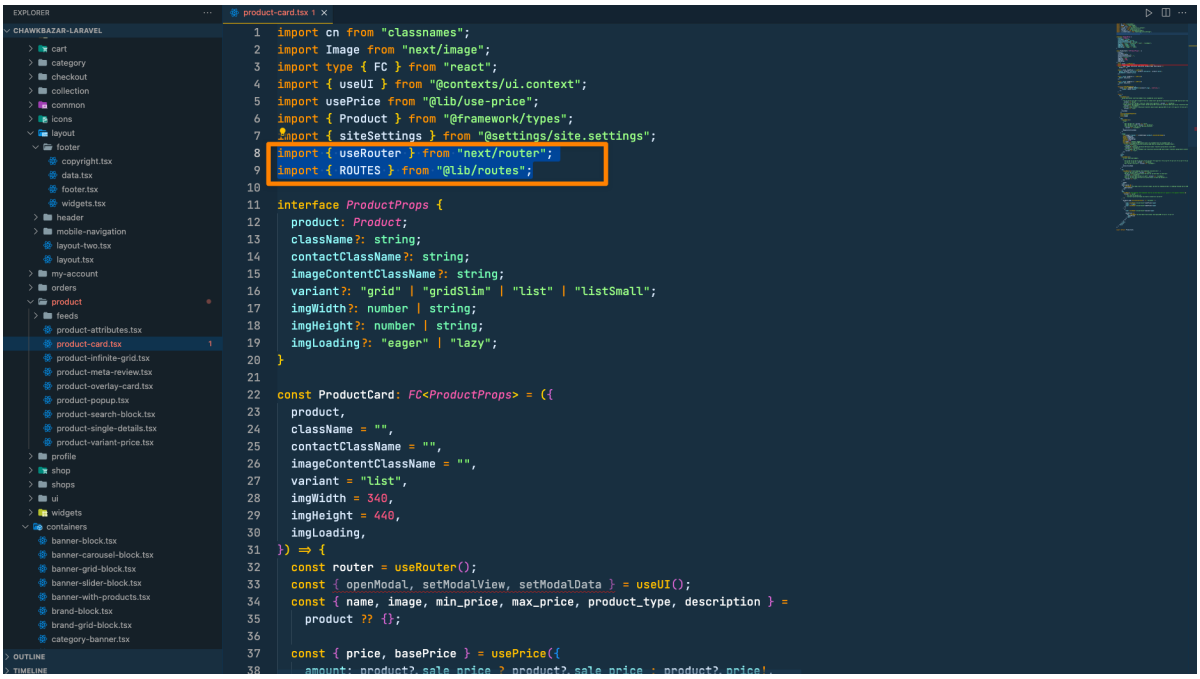
When you install the API using `php artisan marvel:install` or using `bash install.sh` (`docker`), you'll get a prompt to create an admin account, and you've to create admin credentials that time. And then, you can use that credentials for the main admin account.

How to disable product popup view and redirect to a single product page?

At first edit `shop/src/components/product/product-card.tsx`

Import these two lines,

```
import { useRouter } from "next/router";
import { ROUTES } from "@lib/routes";
```

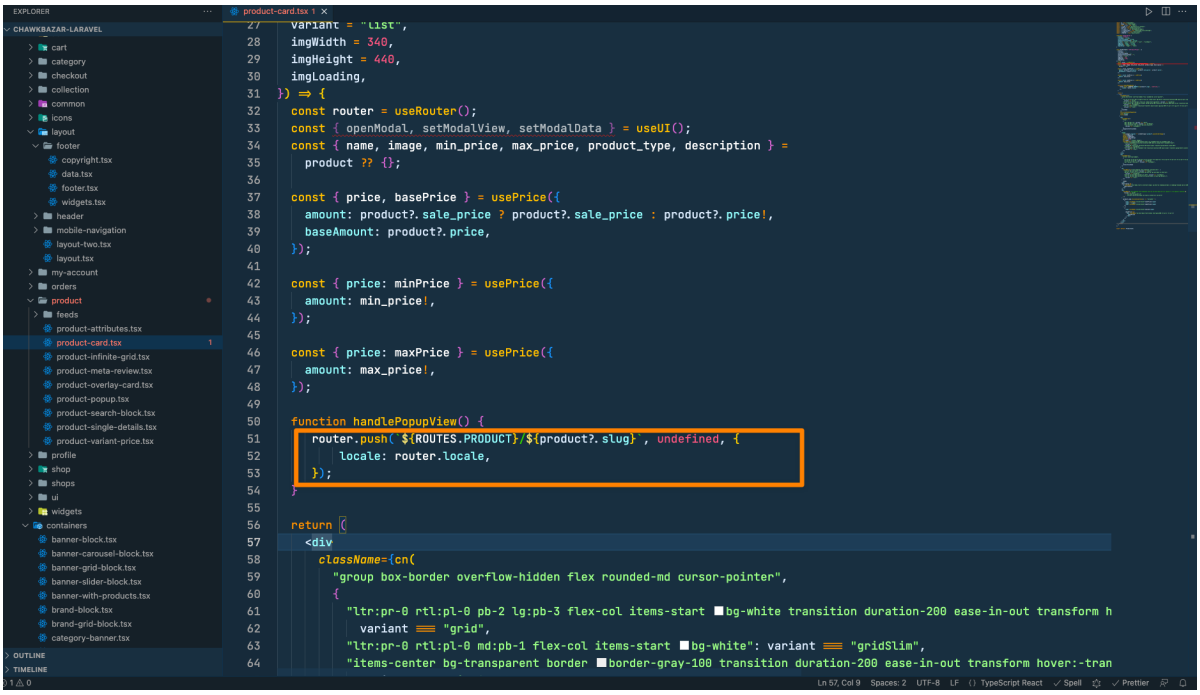


Then call useRouter,

```
const router = useRouter();
```

Then update `handlePopupView` with this code,

```
router.push(`${ROUTES.PRODUCT}/${product?.slug}`, undefined, {
    locale: router.locale,
});
```



Support

For support, Please open a ticket in the support forum

```
https://redqsupport.ticksy.com/
```

> There are some basic requirements for our application so that support team can help you quick such as,

1. Asking queries regarding feature that is already implemented in the application
2. Following recommended configuration, environments & server which you need to met first before you proceed with installation, deployment in your server to receive support.
3. Support query need to be within Envato item support policy. (https://themeforest.net/page/item_support_policy)
4. You should maintain only one support ticket at a time. Creating multiple ticket can cause unexpected delays.
5. Ticket should provide as much details as possible related to the issue such as screenshot, video explanation, access, how to reproduce, environments, if any changes or customisations are made etc

> During working days our ticket response can take 24 hours to 48 hours depending on the volume of tickets pending prior to your ticket . We follow Envato Item support policy https://themeforest.net/page/item_support_policy to provide standard support for our items

Thank You!