

SimplySubs Documentation

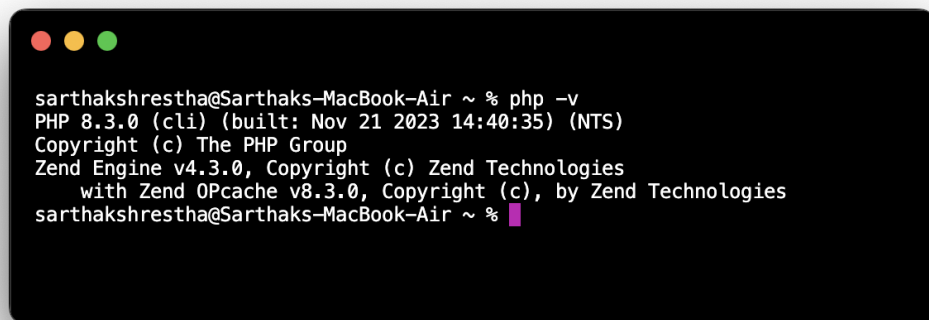
SimplySubs is a **Laravel based e-commerce subscription website** designed and developed by **Sarthak Shrestha**. All the development has been done on branch '**prototype**' in '**comp-bkt**' instead of '**main**' to resolve conflicts and maintain the template. The commit history is present in the '**prototype**' branch of **comp-bkt** and in the '**master**' branch of '**tbc-bsc-l6**'

The GitHub URLs are: **(Com-bkt contains the latest code and commits)**
[comp-bkt/component-2-sarthakshrestha at prototype \(github.com\)](#) **(Branch: prototype)**

[tbc-bsc-l6/component-2-sarthakshrestha: component-2-sarthakshrestha created by GitHub Classroom](#)

PHP Version used: **8.3.0**

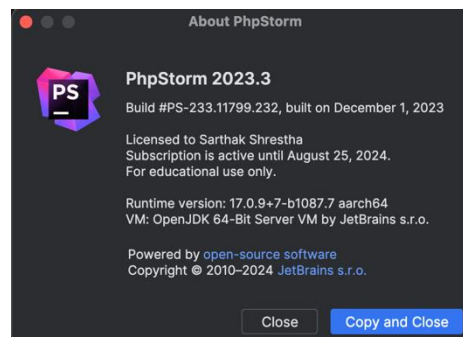
Laravel Version used: **10**



```
sarthakshrestha@Sarthaks-MacBook-Air ~ % php -v
PHP 8.3.0 (cli) (built: Nov 21 2023 14:40:35) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.3.0, Copyright (c) Zend Technologies
    with Zend OPcache v8.3.0, Copyright (c), by Zend Technologies
sarthakshrestha@Sarthaks-MacBook-Air ~ %
```

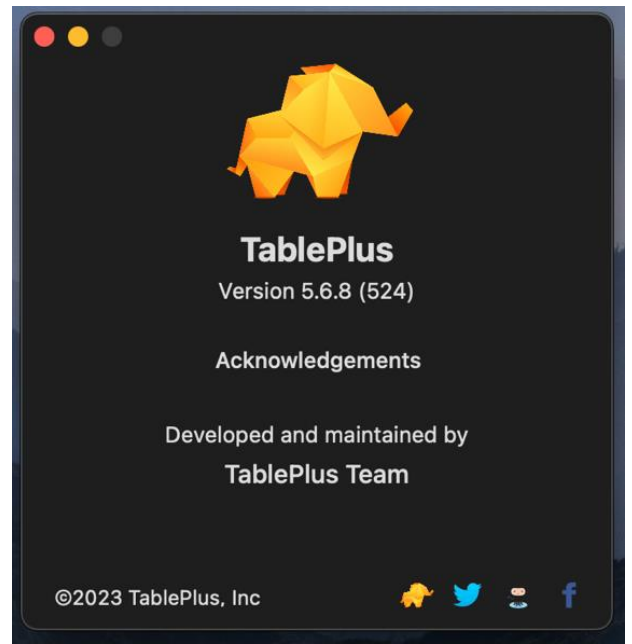
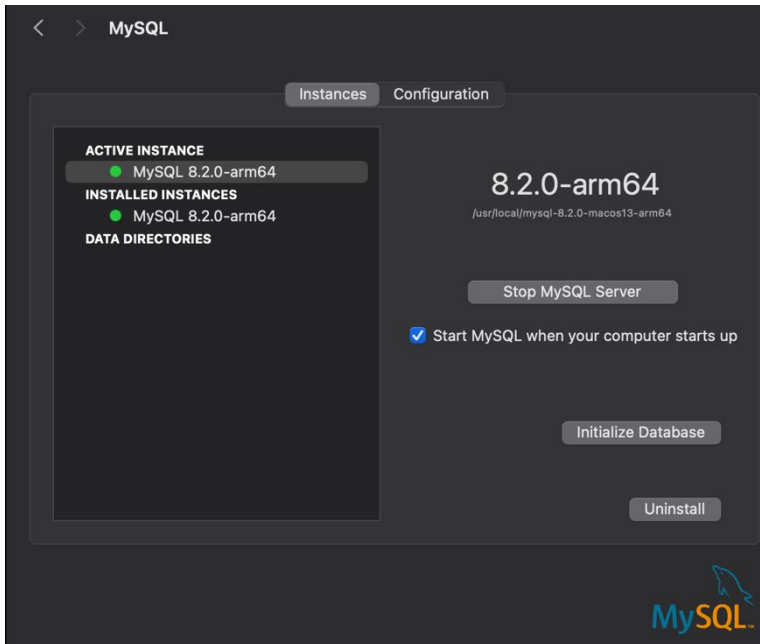
IDE Utilized for the development of **SimplySubs**:

PhpStorm 2023.3

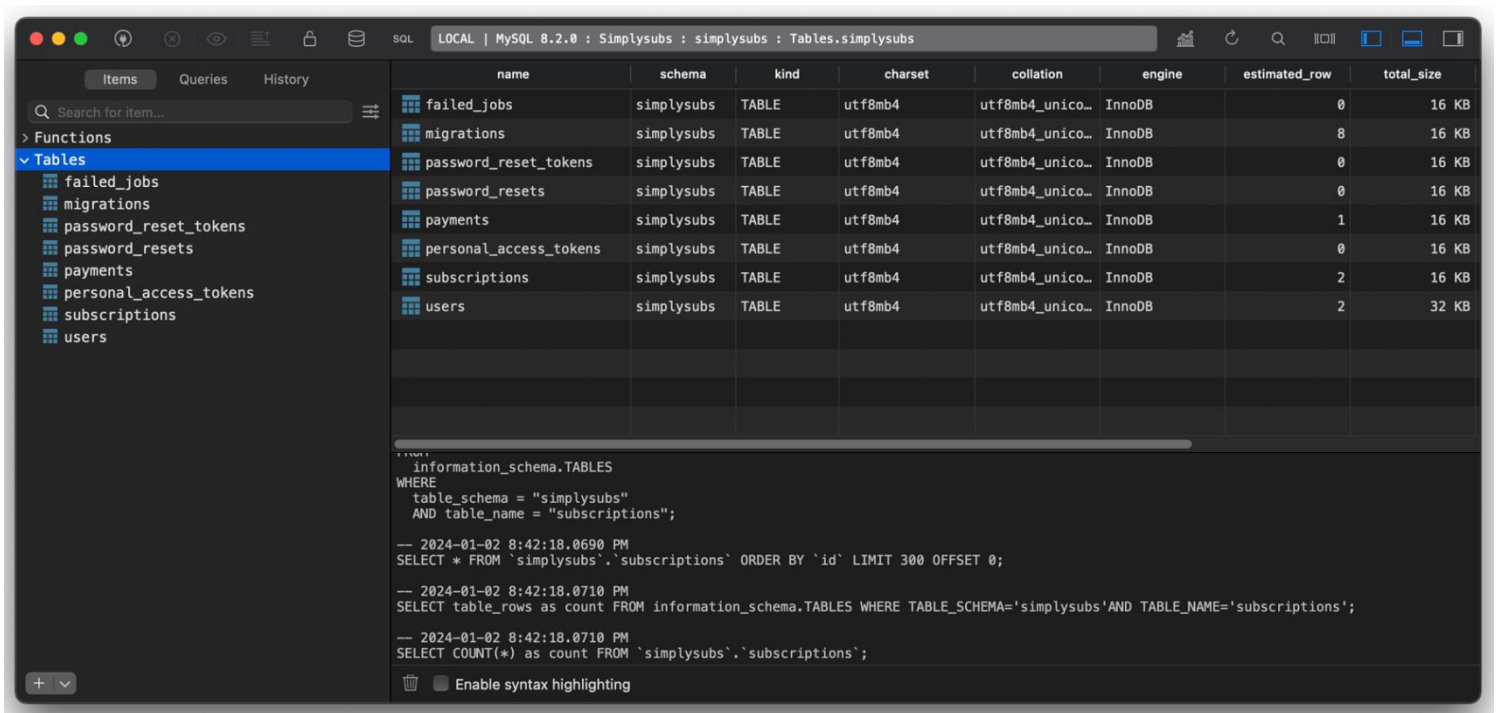


Database Applications utilized:

MySQL 8.2.0 & TablePlus



Database Table



Instructions for Installation

1. Clone the directory through the GitHub Repository: [comp-bkt/component-2-sarthakshrestha at prototype \(github.com\)](https://github.com/comp-bkt/component-2-sarthakshrestha)
2. Make sure that Composer (Package Manager System) is installed on the system, the guideline for installing composer can be found on the website: <https://getcomposer.org>
3. Install MySQL on your system and initialize the database and configure correctly.
4. Create the database through SQL command and configure the .env file as of your configurations.
5. Run the command on the terminal that are `.env.example .env` if you are on Windows or `cp .env.example .env` if you are running on macOS
6. Create the database tables through the command using `php artisan migrate`
7. Seed the database as well for the administrator be created using the command `php artisan db:seed`
 - a. The credential for the admin is as follows
 - i. Email: sarthakstha24@gmail.com
 - ii. Password: theadminofsimplysubs
8. Run commands `composer install`, `npm install`, and then `php artisan storage:link` within the command line that helps in installing necessary packages and the storage link command helps to store the images when uploaded by the admin.
9. For starting the web app, run the command `npm run dev` and then `php artisan serve`
10. Now browse through the website SimplySubs that will be launched through your default browser usually in the URL: 127.0.0.1:8000

Main working functions // Flow of the program

The website functions as a simple e-commerce website but for subscription-based items that are digitally accessible to users.

For the files within the project, **Web.php** contains all the important routes of the program. While the **views** folder contains all the front-end-blade files. **Authentication** and **Authorization** is implemented through **Laravel Breeze**. Simple roles have been given where the administrator is only able to perform CRUD operations. The User is only able to purchase the subscription uploaded by the administrator.

If any user is not signed in at all then a 'Login to Buy' on the **/subs** page. After the user is signed in and selects the required subscription for purchasing, they are redirected to the checkout page and can pay easily through **PayPal**.

It is a simple **CRUD** based application where the administrator can create subscriptions and update or delete them. Meanwhile, the user can then read the data that had been added by the administrator.

There are a lot of **Controllers** within the project that handles tasks. The main ones are the Admin, Checkout, Subscription controllers. These controllers help the admin to perform CRUD operations and the checkout controller linked with PayPal controller allows the user to pay the certain amount of the subscription with their PayPal account.

For the **extra features** I have implemented **PayPal** for purchasing the desired items, and there is also the **searching and sorting** present in the subscriptions page. I have implemented **2FA** only on local machine.

I have utilized **Authentication** and **Gates** so that only the administrator is allowed to view certain pages such as for the CRUD pages. While the user is only allowed to checkout with their selected subscription item.

These are the basic major functions of the website.

Code snippets of the **Laravel web-app**

(Note: Snippets does not contain the entire code, simply for documentation and presentation purposes only)

1. Few Routes and Restricted routes in **web.php** that restricts pages for users that do not have the role of **administrator**.

```
web.php

Route::get('/', function () {
    return view('welcome');
});

Route::post('/add-product', [\App\Http\Controllers\SubscriptionController::class, 'store'])->name('admin');

Route::get('/subs', [SubscriptionController::class, 'index'])->name('subscriptions.index');

Route::get('/about-us', function () {
    return view('aboutus');
});

Route::middleware(['auth', 'verified'])->group(function () {
    // Common routes for authenticated users

    Route::get('/admin/create', function () {

        // Only allow access to '/admin' if the user is an admin

        if (Gate::allows('isAdmin', auth()->user())) {
            return view('admin-create');
        }

        abort(403, 'Unauthorized action.');
```

```
    })->name('admin');

    // Admin Route, Prefix adds admin in the starting of the URL
    Route::prefix('admin')->group(function () {

        Route::get('/updatesub', [SubscriptionController::class, 'showUpdateList'])->name('admin.updatesub.show');
        Route::get('/updatesub/{id}', [SubscriptionController::class, 'showUpdateForm'])->name('admin.updatesub.show');
        Route::put('/updatesub/{id}', [SubscriptionController::class, 'updateSubscription'])->name('admin.updatesub.update');

        Route::delete('/subscriptions/{id}', [SubscriptionController::class, 'deleteSubscription'])->name('admin.subscriptions.delete-subscription');
    });

    Route::get('/updatesub/{id}', [SubscriptionController::class, 'showUpdateForm'])->name('updatesub');
});
```

2. Controller responsible for searching and sorting in the subscriptions page.

```
SearchController.php

class SearchController extends Controller
{
    public function search(Request $request)
    {
        $query = $request->input('query');

        $subscriptions = Subscription::where('title', 'like', "%$query%")
            ->orWhere('description', 'like', "%$query%")
            ->get();

        return view('subs', compact('subscriptions'));
    }

    public function sort(Request $request)
    {
        $field = $request->input('field');
        $direction = $request->input('direction');

        $subscriptions = Subscription::orderBy($field, $direction)->get();

        return view('subs', compact('subscriptions'));
    }
}
```

3. PayPal function within PayPal controller that helps with creating order using PayPal Sandbox.

```
PayPalController.php

public function paypal(Request $request)
{
    $provider = new PayPalClient;
    $provider->setApiCredentials(config('paypal'));
    $paypalToken = $provider->getAccessToken();

    $response = $provider->createOrder([
        "intent" => "CAPTURE",
        "application_context" => [
            "return_url" => route('success'),
            "cancel_url" => route('cancel')
        ],
        "purchase_units" => [
            [
                "amount" => [
                    "currency_code" => config('paypal.currency'),
                    "value" => $request->price
                ]
            ]
        ]
    ]);
    // dd($response);
    if (isset($response['id']) && $response['id'] != null) {
        foreach ($response['links'] as $link) {
            if ($link['rel'] == 'approve') {
                session()->put('product_name', $request->product_name);
                session()->put('price', $request->price);
                session()->put('payer_name', $request->name);
                return redirect()->away($link['href']);
            }
        }
    } else {
        return redirect()->back()->with('error', 'Something went wrong!');
    }
}
```

4. Blade file for the reusable component

```
topbar.blade.php

<body>
<div class="navigation">
  <nav id="main-navbar">
    <ul>
      <li><a href="/">Home

      </a></li>
      <li><a href="/subs">Subscriptions
        <svg xmlns="http://www.w3.org/2000/svg" height="16" width="12" viewBox="0
0 384 512">
          <!--!Font Awesome Free 6.5.0 by @fontawesome - https://fontawesome.com
License - https://fontawesome.com/license/free Copyright 2023 Fonticons, Inc.-->
          <path fill="#ffffff"
            d="M80 0C44.7 0 16 28.7 16 64V448c0 35.3 28.7 64 64 64H304c35.3
0 64-28.7 64-64V64c0-35.3-28.7-64-64-64H80zm80 432h64c8.8 0 16 7.2 16 16s-7.2 16-16 16H160c-
8.8 0-16-7.2-16-16s7.2-16 16-16z"/>
          </svg>

        </a></li>
      @auth
        <li class="signin-nav"><a href="/dashboard">Dashboard
          <svg xmlns="http://www.w3.org/2000/svg" height="16" width="14"
viewBox="0 0 448 512">
            <!--!Font Awesome Free 6.5.0 by @fontawesome -
https://fontawesome.com License - https://fontawesome.com/license/free Copyright 2023
Fonticons, Inc.-->
            <path fill="#ffffff"
              d="M224 256A128 128 0 1 0 224 0a128 128 0 1 0 0 256zm-45.7
48C79.8 304 0 383.8 0 482.3C0 498.7 13.3 512 29.7 512H418.3c16.4 0 29.7-13.3 29.7-29.7C448
383.8 368.2 304 269.7 304H178.3z"/>
            </svg>
          </a></li>
        <li class="register-nav"><a href="{{ route('profile.edit') }}">Edit Profile
Info</a></li>
      @else
        <li class="signin-nav"><a href="/login">Sign in
          <svg xmlns="http://www.w3.org/2000/svg" height="16" width="14"
viewBox="0 0 448 512">
            <!--!Font Awesome Free 6.5.0 by @fontawesome -
https://fontawesome.com License - https://fontawesome.com/license/free Copyright 2023
Fonticons, Inc.-->
            <path fill="#ffffff"
              d="M224 256A128 128 0 1 0 224 0a128 128 0 1 0 0 256zm-45.7
48C79.8 304 0 383.8 0 482.3C0 498.7 13.3 512 29.7 512H418.3c16.4 0 29.7-13.3 29.7-29.7C448
383.8 368.2 304 269.7 304H178.3z"/>
            </svg>
          </a></li>
        <li class="register-nav"><a href="{{ route('register') }}">Register</a></li>
      @endauth
    </ul>
  </nav>
</div>
@yield('banner')
@yield('banner-1')
@yield('banner-2')
@yield('main-body')
@yield('footer')
</body>
```

Guide for browsing **SimplySubs** as the User

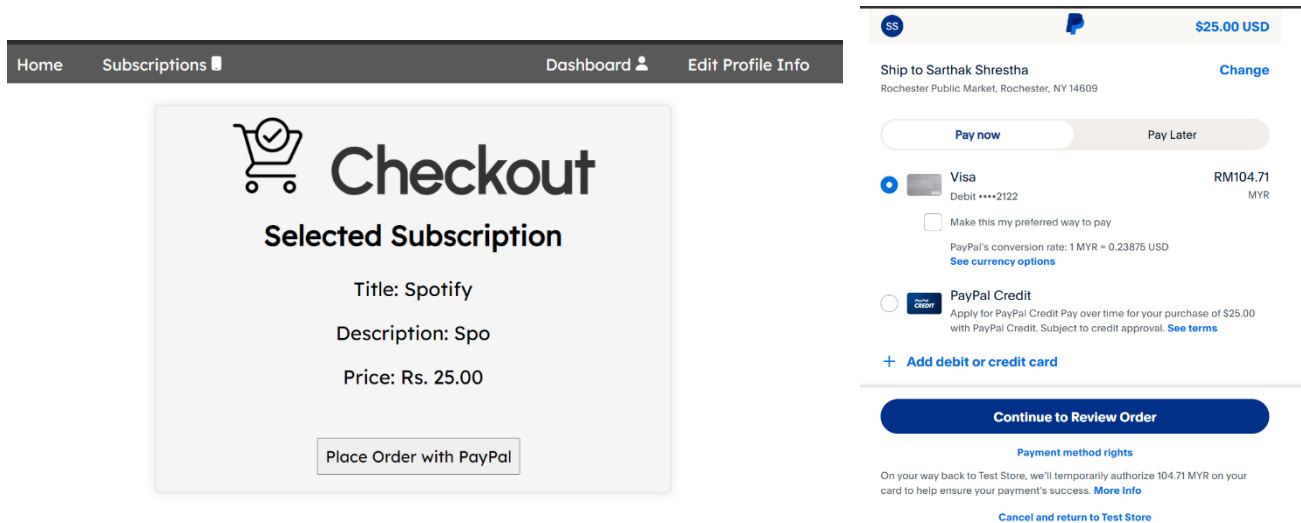
1. When the application is run the user will be presented with the home page
2. By hovering over the Subscription link in the top bar and clicking it, a list of subscriptions will appear.
3. For purchasing the subscription, the user must sign up.
4. After filling up the required fields, they will receive a 2FA link that will confirm that they are registering to be a member of SimplySubs.
5. While the registering user goes through their inbox they can then click on the mail and then the link.
6. The 2FA machine can only be verified when it is on the local machine.
7. Now the user is then able to purchase items by going through the subscriptions page.

Guide for browsing **SimplySubs** as the Administrator

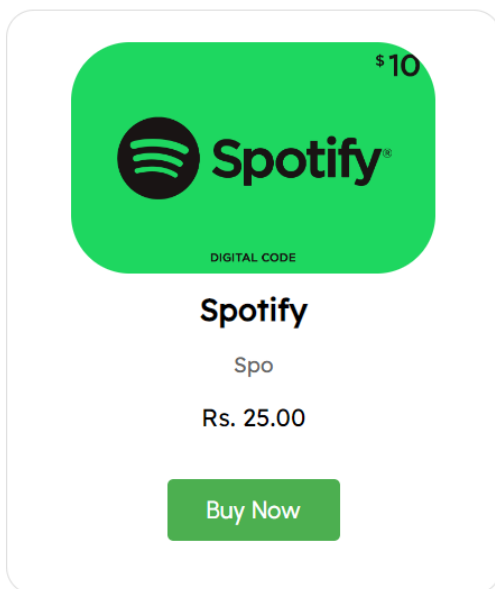
1. Firstly, before logging in as administrator, make sure that the command 'php artisan db:seed' is entered and the DB table gets filled with the credentials of the admin.
2. The administrator's credential is as follows:
 - i. Email: sarthakstha24@gmail.com
 - ii. Password: theadminofsimplysubs
3. After the administrator has logged in, scrolling below will appear the Create Product Item and Update Product Item.
4. The administrator can add products to the subscriptions page along with the required criteria. They can upload the image of the subscription as well.
5. The administrator can basically perform **CRUD** operations.
6. For updating or deleting the subscription the administrator can click the Update Product Item and navigate to the product that needs to be updated / deleted.

Addition features of **SimplySubs**

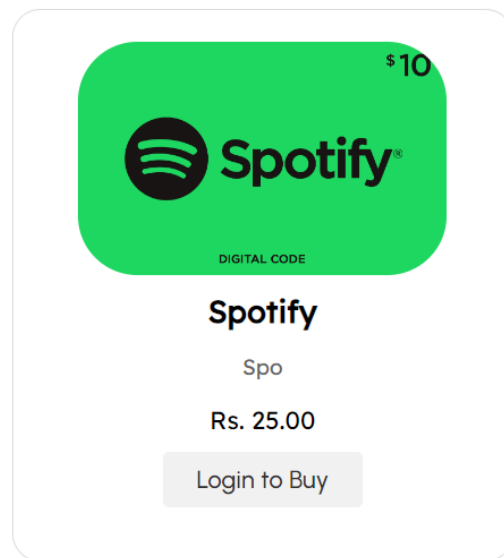
1. Ability to pay with **PayPal** through PayPal Sandbox (as it is for development)



2. Authentication **middleware** where if signed out the user is unable to purchase item. The user must be registered and signed in to purchase any item.

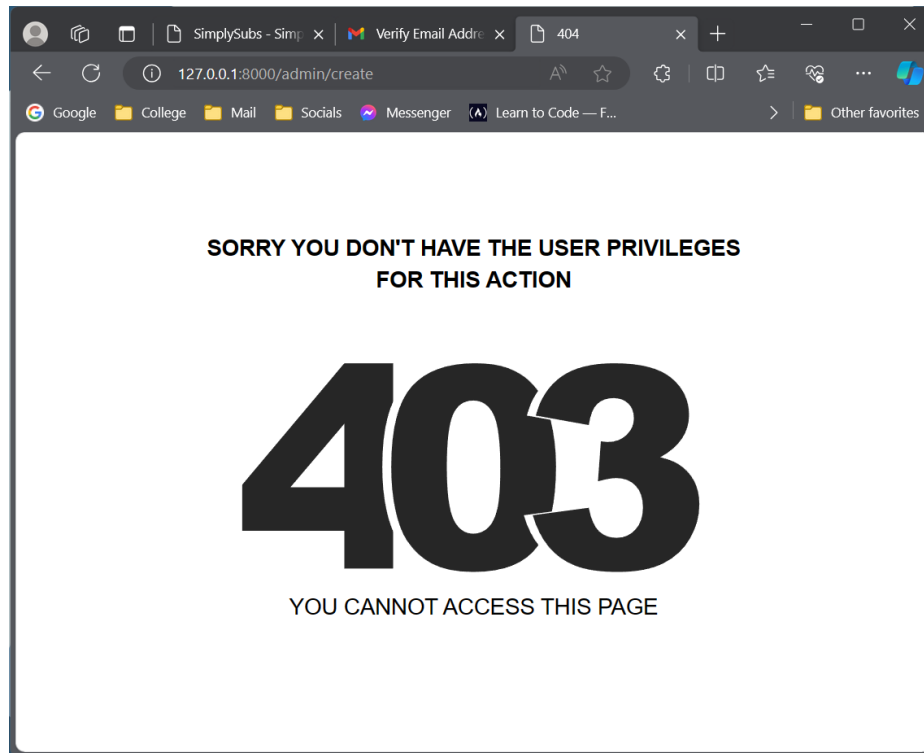


If signed in

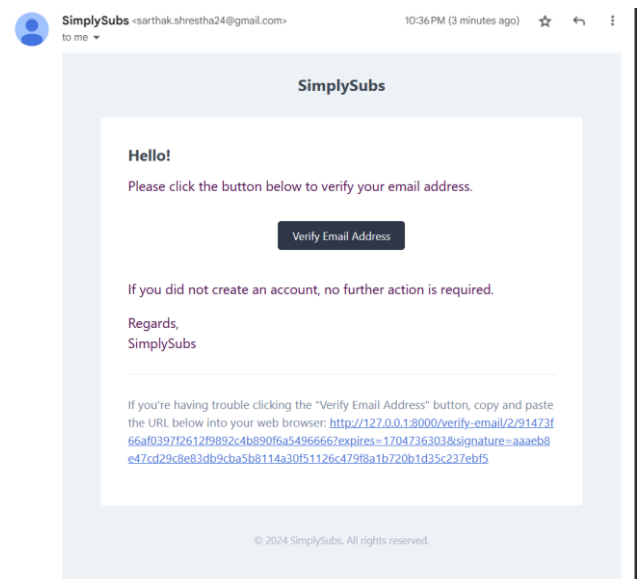


If signed out

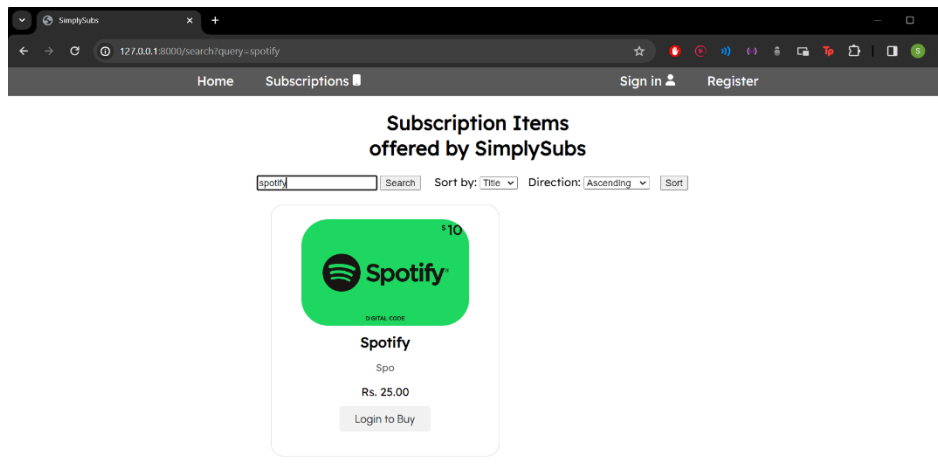
3. Normal signed-in users can't access protected routes of the Administrator.
Example: if the user that is not the administrator is logged in, they cannot access administrative functions. Error 403 will pop up to the user's view.



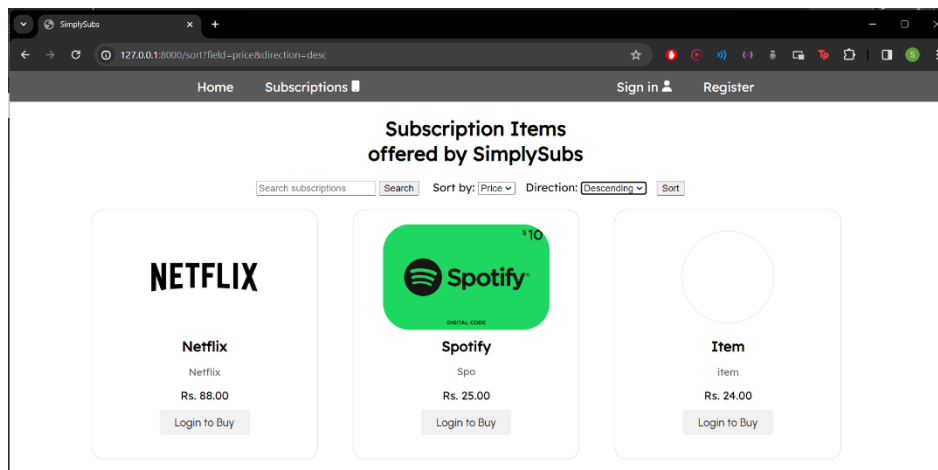
4. Email verification method has been implemented where the user gets sent an email when they are trying to register to SimplySubs into their email account.



5. Searching and sorting method has been implemented where the user is able to sort items alphabetically / price in an ascending order. The user is also able to search for the required item.

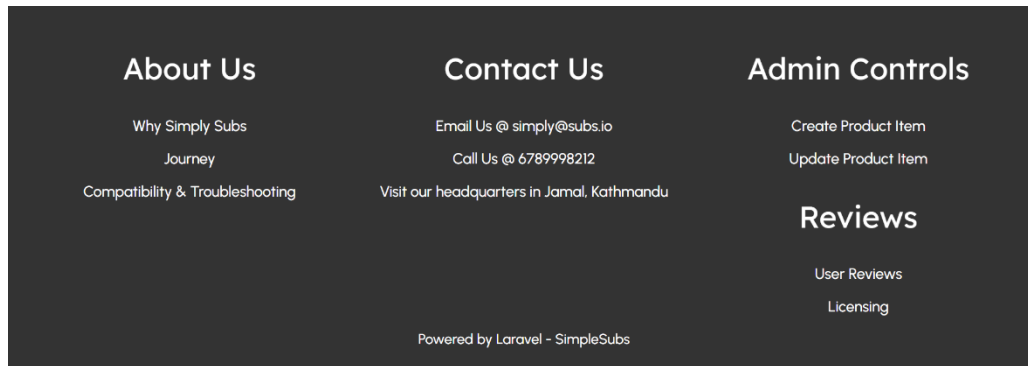


User searching input

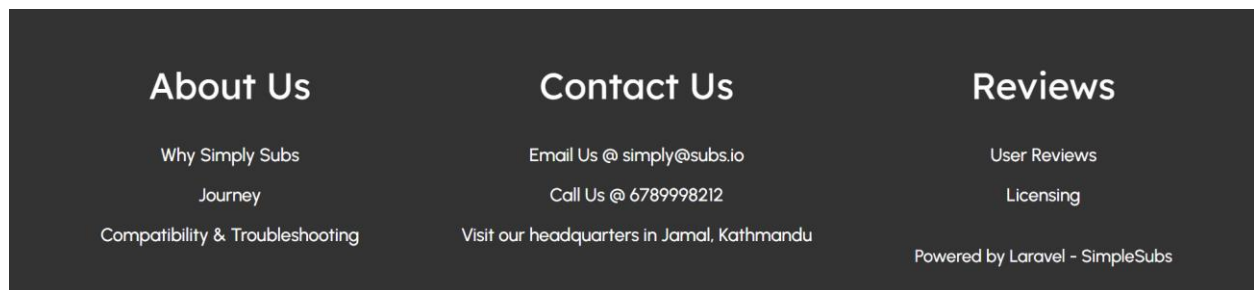


User sorting price by descending order

6. When the administrator is logged in, the administrator can only create and update subscription items. When the administrator is logged in a newly created section gets created in the footer of the website.



If logged in as Administrator



If logged in as User / User is Signed Out

7. The administrator can upload images of the subscription as well.

Subscription Items offered by SimplySubs

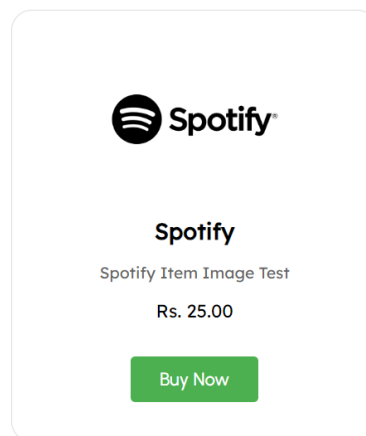
Image: S...ng

Title:

Description:

Price:

Please add hi-res images of the subscription



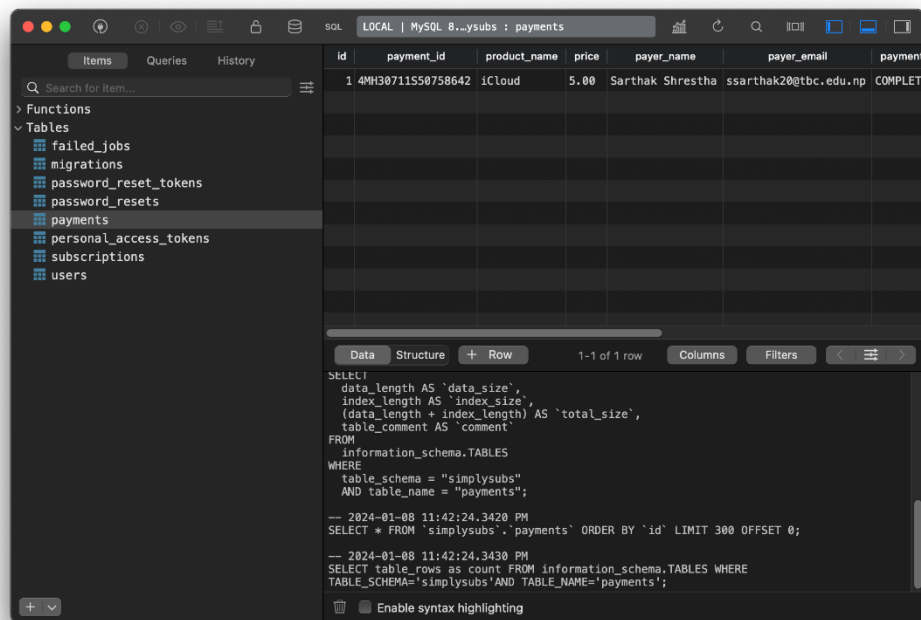
8. Custom Error Page if the URL does not exist.

WHOOOPS! PAGE IS UNAVAILABLE

404

PAGE REQUESTED NOT FOUND, PLEASE TRY AGAIN

9. After subscription has been paid by the user through PayPal the purchased item can be seen in the database table.



Thank you for
going through
SimplySubs's
documentation!