contoller

{

"private": true,

"scripts": {

"dev": "npm run development",

"development": "cross-env NODE\_ENV=development node\_modules/webpack/bin/webpack.js --progress --hide-modules --config=node\_modules/laravel-mix/setup/webpack.config.js",

"watch": "npm run development -- --watch",

"watch-poll": "npm run watch -- --watch-poll",

"hot": "cross-env NODE\_ENV=development node\_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --hot --config=node\_modules/laravel-mix/setup/webpack.config.js",

"prod": "npm run production",

"production": "cross-env NODE\_ENV=production node\_modules/webpack/bin/webpack.js --no-progress --hide-modules --config=node\_modules/laravel-mix/setup/webpack.config.js"

},

"devDependencies": {

"axios": "^0.18",

"bootstrap": "^4.0.0",

"cross-env": "^5.1",

"jquery": "^3.2",

"laravel-mix": "^4.0.7",

"lodash": "^4.17.5",

"popper.js": "^1.12",

"resolve-url-loader": "^2.3.1",

"sass": "^1.15.2",

"sass-loader": "^7.1.0",

"vue": "^2.5.17",

"vue-template-compiler": "^2.6.10"

},

"dependencies": {

"emoji-mart-vue": "^2.6.4",

"laravel-echo": "^1.5.3",

"laravel-vue-pagination": "^2.1.0",

"moment": "^2.24.0",

"pusher-js": "^4.4.0",

"sweetalert2": "^8.8.0",

"vform": "^1.0.0",

"vue-fullcalendar": "^1.0.9",

"vue-progressbar": "^0.7.5",

"vue-router": "^3.0.2",

"vue-strap": "github:wffranco/vue-strap",

"vue-upload-component": "^2.8.20",

"vue2-google-maps": "^0.10.6",

"vuetify": "^1.5.8"

}

}

<?php

namespace App\Http\Controllers\API;

use App\User;

use Illuminate\Http\Request;

use Auth;

use App\Http\Controllers\Controller;

use Illuminate\Support\Facades\Hash;

class CustomerController extends Controller

{

public function index()

{

return User::where('type','Customer')->get();

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

//

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

//

}

}

x<?php

namespace App\Http\Controllers\API;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use App\User;

use Illuminate\Support\Facades\Hash;

class UserController extends Controller

{

/\*\*

\* Create a new controller instance.

\*

\* @return void

\*/

public function \_\_construct()

{

$this->middleware('auth:api');

}

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

// $this->authorize('isAdmin');

if (\Gate::allows('isAdmin') || \Gate::allows('isAuthor')) {

return User::latest()->paginate(10);

}

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

$this->validate($request,[

'name' => 'required|string|max:191',

'email' => 'required|string|email|max:191|unique:users',

'password' => 'required|string|min:6'

]);

return User::create([

'name' => $request['name'],

'email' => $request['email'],

'type' => $request['type'],

'bio' => $request['bio'],

'photo' => $request['photo'],

'password' => Hash::make($request['password']),

]);

}

public function updateProfile(Request $request)

{

$user = auth('api')->user();

$this->validate($request,[

'name' => 'required|string|max:191',

'email' => 'required|string|email|max:191|unique:users,email,'.$user->id,

'password' => 'sometimes|required|min:6'

]);

$currentPhoto = $user->photo;

if($request->photo != $currentPhoto){

$name = time().'.' . explode('/', explode(':', substr($request->photo, 0, strpos($request->photo, ';')))[1])[1];

\Image::make($request->photo)->save(public\_path('img/profile/').$name);

$request->merge(['photo' => $name]);

$userPhoto = public\_path('img/profile/').$currentPhoto;

if(file\_exists($userPhoto)){

@unlink($userPhoto);

}

}

if(!empty($request->password)){

$request->merge(['password' => Hash::make($request['password'])]);

}

$user->update($request->all());

return ['message' => "Success"];

}

public function profile()

{

return auth('api')->user();

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

$user = User::findOrFail($id);

$this->validate($request,[

'name' => 'required|string|max:191',

'email' => 'required|string|email|max:191|unique:users,email,'.$user->id,

'password' => 'sometimes|min:6'

]);

$user->update([

'name' => $request['name'],

'email' => $request['email'],

'password' => Hash::make($request['password']),

'type' => $request['type'],

'bio' => $request['bio']

]);

return ['message' => 'Updated the user info'];

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

$this->authorize('isAdmin');

$user = User::findOrFail($id);

// delete the user

$user->delete();

return ['message' => 'User Deleted'];

}

public function search(){

if ($search = \Request::get('q')) {

$users = User::where(function($query) use ($search){

$query->where('name','LIKE',"%$search%")

->orWhere('email','LIKE',"%$search%");

})->paginate(20);

}else{

$users = User::latest()->paginate(5);

}

return $users;

}

Controller.php

<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Bus\DispatchesJobs;

use Illuminate\Routing\Controller as BaseController;

use Illuminate\Foundation\Validation\ValidatesRequests;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

class Controller extends BaseController

{

use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

tt}

Councotroller.php

<?php

namespace App\Http\Controllers;

use App\Post;

use App\Transaction;

use App\User;

use Illuminate\Http\Request;

class CountController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth:api');

}

public function countPost(){

if(\Gate::allows('laundry')){

return Post::count();

}

}

public function countCustomer(){

if(\Gate::allows('laundry')){

return User::where('type','Customer')->count();

}

}

public function countTransaction(){

if(\Gate::allows('laundry')){

return Transaction::count();

}

}

}

Homecontroller.php

<?php

namespace App\Http\Controllers;

use App\User;

use Illuminate\Http\Request;

class HomeController extends Controller

{

/\*\*

\* Create a new controller instance.

\*

\* @return void

\*/

public function \_\_construct()

{

$this->middleware('auth');

}

/\*\*

\* Show the application dashboard.

\*

\* @return \Illuminate\Contracts\Support\Renderable

\*/

public function index()

{

return view('home');

}

public function shop()

{

return User::where('type','Laundry Shop')->get();

}

public function private()

{

return view('private');

}

public function users()

{

return User::all();

}

}

MessageController.php

<?php

namespace App\Http\Controllers;

use App\User;

use App\Message;

use Illuminate\Http\Request;

use App\Events\PrivateMessageSent;

class MessageController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function fetchMessages()

{

return Message::with('user')->get();

}

public function privateMessages(User $user)

{

$privateCommunication= Message::with('user')

->where(['user\_id'=> auth()->id(), 'receiver\_id'=> $user->id]) ->orWhere(function($query) use($user){

$query->where(['user\_id' => $user->id, 'receiver\_id' => auth()->id()]);

})

->get();

return $privateCommunication;

}

public function sendPrivateMessage(Request $request,User $user)

{

if(request()->has('file')){

$filename = request('file')->store('chat');

$message=Message::create([

'user\_id' => request()->user()->id,

'image' => $filename,

'receiver\_id' => $user->id

]);

}else{

$input=$request->all();

$input['receiver\_id']=$user->id;

$message=auth()->user()->messages()->create($input);

}

broadcast(new PrivateMessageSent($message->load('user')))->toOthers();

return response(['status'=>'Message private sent successfully','message'=>$message]);

}

}

PostConroller.php

<?php

namespace App\Http\Controllers;

use App\Events\PostCreated;

use App\Post;

use Illuminate\Http\Request;

use Auth;

class PostController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function \_\_construct()

{

$this->middleware('auth:api');

}

public function index()

{

return Post::latest()->get();

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

public function viewPost(){

return Post::latest()->with('user')->get();

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

$this->validate($request,[

'content' => 'required|min:10|max:250',

'title' => 'required|min:5'

]);

$post = Post::create([

'user\_id' => Auth::user()->id,

'content' => $request['content'],

'title' => $request['title']

]);

event(new PostCreated($post));

}

/\*\*

\* Display the specified resource.

\*

\* @param \App\Post $post

\* @return \Illuminate\Http\Response

\*/

public function show(Post $post)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param \App\Post $post

\* @return \Illuminate\Http\Response

\*/

public function edit(Post $post)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param \App\Post $post

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, Post $post)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param \App\Post $post

\* @return \Illuminate\Http\Response

\*/

public function destroy(Post $post)

{

//

}

}

StatusController.php

<?php

namespace App\Http\Controllers;

use App\Status;

use Illuminate\Http\Request;

class StatusController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

//

}

public function getStatus(){

return Status::latest()->get();

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

//

}

/\*\*

\* Display the specified resource.

\*

\* @param \App\Status $status

\* @return \Illuminate\Http\Response

\*/

public function show(Status $status)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param \App\Status $status

\* @return \Illuminate\Http\Response

\*/

public function edit(Status $status)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param \App\Status $status

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, Status $status)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param \App\Status $status

\* @return \Illuminate\Http\Response

\*/

public function destroy(Status $status)

{

//

}

}

TransactionController.php

<?php

namespace App\Http\Controllers;

use App\Events\StatusChanged;

use App\Transaction;

use Auth;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

class TransactionController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function \_\_construct()

{

$this->middleware('auth:api');

}

public function index()

{

if (\Gate::allows('laundry'))

{

return Transaction::latest()->with('customer','status')->get();

}

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function editStatus(Request $request, $id){

$status = Transaction::findOrFail($id);

$status->update([

'status\_id' => $request['status']

]);

return $status;

event(new StatusChanged($transaction));

}

public function store(Request $request)

{

//

$this->validate($request,[

'service' => 'required',

'date\_reserve' => 'date|after:tomorrow'

]);

$user = Auth::user()->id;

return Transaction::create([

'service' => $request['service'],

'date\_reserve' => $request['date'],

'user\_id' => $user

]);

}

/\*\*

\* Display the specified resource.

\*

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function show(Transaction $transaction)

{

//

}

public function getCustomerTransaction(){

if (\Gate::allows('customer')){

$id = Auth::user()->id;

return Transaction::where('user\_id',$id)->latest()->with('status')->get();

}

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function edit(Transaction $transaction)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

$transaction = Transaction::findOrFail($id);

$this->validate($request,[

'weight' => 'required|numeric',

'total' => 'required'

]);

$transaction->update([

'weight' => $request['weight'],

'total' => $request['total']

]);

return $transaction;

/\* $transaction->update([\*/

/\* 'user\_id' => $request['user\_id'],

'service' => $request['service'],

'weight' => $request['weight'],

'total' => $request['total'],

'status\_id' => $request['status'] \*/

/\* ]);

return ['message' => 'Updated the user info'];\*/

/\* $service = $request['service'];

$id = $request['id'];

$total = (int)$request['total'];

$status = $request['status'];

$weight =(int) $request['weight'];

$query = DB::table('transactions')

->where('id',$id)

->update(

['total' => $total],

['status\_id' => $status],

['weight' => $weight]

);

return $query;\*/

/\*return Transaction::where('id',$id)->update([

'user\_id' => $request['user\_id'],

'service' => $request['service'],

'weight' => $request['weight'],

'total' => $request['total'],

'status\_id' => $request['status']

]);\*/

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function destroy(Transaction $transaction)

{

//

}

}

Middleware

<?php

namespace App\Http\Middleware;

use Illuminate\Auth\Middleware\Authenticate as Middleware;

class Authenticate extends Middleware

{

/\*\*

\* Get the path the user should be redirected to when they are not authenticated.

\*

\* @param \Illuminate\Http\Request $request

\* @return string

\*/

protected function redirectTo($request)

{

if (! $request->expectsJson()) {

return route('login');

}

}

}

ChevkrormentenanceMode.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode as Middleware;

class CheckForMaintenanceMode extends Middleware

{

/\*\*

\* The URIs that should be reachable while maintenance mode is enabled.

\*

\* @var array

\*/

protected $except = [

//

];

}

EncryptCookies.php

<?php

namespace App\Http\Middleware;

use Illuminate\Cookie\Middleware\EncryptCookies as Middleware;

class EncryptCookies extends Middleware

{

/\*\*

\* The names of the cookies that should not be encrypted.

\*

\* @var array

\*/

protected $except = [

//

];

}

RedirectlAuthentication

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Support\Facades\Auth;

class RedirectIfAuthenticated

{

/\*\*

\* Handle an incoming request.

\*

\* @param \Illuminate\Http\Request $request

\* @param \Closure $next

\* @param string|null $guard

\* @return mixed

\*/

public function handle($request, Closure $next, $guard = null)

{

if (Auth::guard($guard)->check()) {

return redirect('/home');

}

return $next($request);

}

}

TrimStrings.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\TrimStrings as Middleware;

class TrimStrings extends Middleware

{

/\*\*

\* The names of the attributes that should not be trimmed.

\*

\* @var array

\*/

protected $except = [

'password',

'password\_confirmation',

];

}

TrustProxies.php<?php

namespace App\Http\Middleware;

use Illuminate\Http\Request;

use Fideloper\Proxy\TrustProxies as Middleware;

class TrustProxies extends Middleware

{

/\*\*

\* The trusted proxies for this application.

\*

\* @var array

\*/

protected $proxies;

/\*\*

\* The headers that should be used to detect proxies.

\*

\* @var int

\*/

protected $headers = Request::HEADER\_X\_FORWARDED\_ALL;

}

VerifyCsrfToken.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;

class VerifyCsrfToken extends Middleware

{

/\*\*

\* Indicates whether the XSRF-TOKEN cookie should be set on the response.

\*

\* @var bool

\*/

protected $addHttpCookie = true;

/\*\*

\* The URIs that should be excluded from CSRF verification.

\*

\* @var array

\*/

protected $except = [

//

];

}

Middleware

<?php

namespace App\Http\Controllers;

use App\Events\StatusChanged;

use App\Transaction;

use Auth;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

class TransactionController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function \_\_construct()

{

$this->middleware('auth:api');

}

public function index()

{

if (\Gate::allows('laundry'))

{

return Transaction::latest()->with('customer','status')->get();

}

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function editStatus(Request $request, $id){

$status = Transaction::findOrFail($id);

$status->update([

'status\_id' => $request['status']

]);

return $status;

event(new StatusChanged($transaction));

}

public function store(Request $request)

{

//

$this->validate($request,[

'service' => 'required',

'date\_reserve' => 'date|after:tomorrow'

]);

$user = Auth::user()->id;

return Transaction::create([

'service' => $request['service'],

'date\_reserve' => $request['date'],

'user\_id' => $user

]);

}

/\*\*

\* Display the specified resource.

\*

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function show(Transaction $transaction)

{

//

}

public function getCustomerTransaction(){

if (\Gate::allows('customer')){

$id = Auth::user()->id;

return Transaction::where('user\_id',$id)->latest()->with('status')->get();

}

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function edit(Transaction $transaction)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

$transaction = Transaction::findOrFail($id);

$this->validate($request,[

'weight' => 'required|numeric',

'total' => 'required'

]);

$transaction->update([

'weight' => $request['weight'],

'total' => $request['total']

]);

return $transaction;

/\* $transaction->update([\*/

/\* 'user\_id' => $request['user\_id'],

'service' => $request['service'],

'weight' => $request['weight'],

'total' => $request['total'],

'status\_id' => $request['status'] \*/

/\* ]);

return ['message' => 'Updated the user info'];\*/

/\* $service = $request['service'];

$id = $request['id'];

$total = (int)$request['total'];

$status = $request['status'];

$weight =(int) $request['weight'];

$query = DB::table('transactions')

->where('id',$id)

->update(

['total' => $total],

['status\_id' => $status],

['weight' => $weight]

);

return $query;\*/

/\*return Transaction::where('id',$id)->update([

'user\_id' => $request['user\_id'],

'service' => $request['service'],

'weight' => $request['weight'],

'total' => $request['total'],

'status\_id' => $request['status']

]);\*/

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param \App\Transaction $transaction

\* @return \Illuminate\Http\Response

\*/

public function destroy(Transaction $transaction)

{

//

}

}

Authenticate.php

<?php

namespace App\Http\Middleware;

use Illuminate\Auth\Middleware\Authenticate as Middleware;

class Authenticate extends Middleware

{

/\*\*

\* Get the path the user should be redirected to when they are not authenticated.

\*

\* @param \Illuminate\Http\Request $request

\* @return string

\*/

protected function redirectTo($request)

{

if (! $request->expectsJson()) {

return route('login');

}

}

}

CheckFormentenanceMode.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode as Middleware;

class CheckForMaintenanceMode extends Middleware

{

/\*\*

\* The URIs that should be reachable while maintenance mode is enabled.

\*

\* @var array

\*/

protected $except = [

//

];

}

EncryptCookies.php

<?php

namespace App\Http\Middleware;

use Illuminate\Cookie\Middleware\EncryptCookies as Middleware;

class EncryptCookies extends Middleware

{

/\*\*

\* The names of the cookies that should not be encrypted.

\*

\* @var array

\*/

protected $except = [

//

];

}

RedirectAuthenticated.php

<?php

namespace App\Http\Middleware;

use Illuminate\Cookie\Middleware\EncryptCookies as Middleware;

class EncryptCookies extends Middleware

{

/\*\*

\* The names of the cookies that should not be encrypted.

\*

\* @var array

\*/

protected $except = [

//

];

}

TrimStrings.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\TrimStrings as Middleware;

class TrimStrings extends Middleware

{

/\*\*

\* The names of the attributes that should not be trimmed.

\*

\* @var array

\*/

protected $except = [

'password',

'password\_confirmation',

];

}

TrustProxies.php

<?php

namespace App\Http\Middleware;

use Illuminate\Http\Request;

use Fideloper\Proxy\TrustProxies as Middleware;

class TrustProxies extends Middleware

{

/\*\*

\* The trusted proxies for this application.

\*

\* @var array

\*/

protected $proxies;

/\*\*

\* The headers that should be used to detect proxies.

\*

\* @var int

\*/

protected $headers = Request::HEADER\_X\_FORWARDED\_ALL;

}

VerifycsrfToken.php

<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;

class VerifyCsrfToken extends Middleware

{

/\*\*

\* Indicates whether the XSRF-TOKEN cookie should be set on the response.

\*

\* @var bool

\*/

protected $addHttpCookie = true;

/\*\*

\* The URIs that should be excluded from CSRF verification.

\*

\* @var array

\*/

protected $except = [

//

];

}

Kernel.php

<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel

{

/\*\*

\* The application's global HTTP middleware stack.

\*

\* These middleware are run during every request to your application.

\*

\* @var array

\*/

protected $middleware = [

\App\Http\Middleware\CheckForMaintenanceMode::class,

\Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,

\App\Http\Middleware\TrimStrings::class,

\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,

\App\Http\Middleware\TrustProxies::class,

];

/\*\*

\* The application's route middleware groups.

\*

\* @var array

\*/

protected $middlewareGroups = [

'web' => [

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

// \Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

\Laravel\Passport\Http\Middleware\CreateFreshApiToken::class,

],

'api' => [

'throttle:60,1',

'bindings',

],

];

/\*\*

\* The application's route middleware.

\*

\* These middleware may be assigned to groups or used individually.

\*

\* @var array

\*/

protected $routeMiddleware = [

'auth' => \App\Http\Middleware\Authenticate::class,

'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,

'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,

'can' => \Illuminate\Auth\Middleware\Authorize::class,

'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,

'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,

'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,

];

/\*\*

\* The priority-sorted list of middleware.

\*

\* This forces non-global middleware to always be in the given order.

\*

\* @var array

\*/

protected $middlewarePriority = [

\Illuminate\Session\Middleware\StartSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\Authenticate::class,

\Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

\Illuminate\Auth\Middleware\Authorize::class,

];

}

AppserverProvider.php

<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider

{

/\*\*

\* Register any application services.

\*

\* @return void

\*/

public function register()

{

//

}

/\*\*

\* Bootstrap any application services.

\*

\* @return void

\*/

public function boot()

{

//

}

}

AuthsServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;

use Illuminate\Support\Facades\Gate;

use Laravel\Passport\Passport;

class AuthServiceProvider extends ServiceProvider

{

/\*\*

\* The policy mappings for the application.

\*

\* @var array

\*/

protected $policies = [

// 'App\Model' => 'App\Policies\ModelPolicy',

];

/\*\*

\* Register any authentication / authorization services.

\*

\* @return void

\*/

public function boot()

{

$this->registerPolicies();

Gate::define('laundry',function($user){

return $user->type === 'Laundry Shop';

});

Gate::define('customer',function($user){

return $user->type === 'Customer';

});

Gate::define('admin',function($user){

return $user->type = 'Admin';

});

Passport::routes();

//

}

}

BroadcastServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;

use Illuminate\Support\Facades\Broadcast;

class BroadcastServiceProvider extends ServiceProvider

{

/\*\*

\* Bootstrap any application services.

\*

\* @return void

\*/

public function boot()

{

Broadcast::routes(['middleware' => ['auth:api']]);

require base\_path('routes/channels.php');

}

}

EventSeviceProvider.php

<?php

namespace App\Providers;

use Illuminate\Support\Facades\Event;

use Illuminate\Auth\Events\Registered;

use Illuminate\Auth\Listeners\SendEmailVerificationNotification;

use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;

class EventServiceProvider extends ServiceProvider

{

/\*\*

\* The event listener mappings for the application.

\*

\* @var array

\*/

protected $listen = [

Registered::class => [

SendEmailVerificationNotification::class,

],

];

/\*\*

\* Register any events for your application.

\*

\* @return void

\*/

public function boot()

{

parent::boot();

//

}

}

RouteSrviceProvider.php

<?php

namespace App\Providers;

use Illuminate\Support\Facades\Route;

use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

class RouteServiceProvider extends ServiceProvider

{

/\*\*

\* This namespace is applied to your controller routes.

\*

\* In addition, it is set as the URL generator's root namespace.

\*

\* @var string

\*/

protected $namespace = 'App\Http\Controllers';

/\*\*

\* Define your route model bindings, pattern filters, etc.

\*

\* @return void

\*/

public function boot()

{

//

parent::boot();

}

/\*\*

\* Define the routes for the application.

\*

\* @return void

\*/

public function map()

{

$this->mapApiRoutes();

$this->mapWebRoutes();

//

}

/\*\*

\* Define the "web" routes for the application.

\*

\* These routes all receive session state, CSRF protection, etc.

\*

\* @return void

\*/

protected function mapWebRoutes()

{

Route::middleware('web')

->namespace($this->namespace)

->group(base\_path('routes/web.php'));

}

/\*\*

\* Define the "api" routes for the application.

\*

\* These routes are typically stateless.

\*

\* @return void

\*/

protected function mapApiRoutes()

{

Route::prefix('api')

->middleware('api')

->namespace($this->namespace)

->group(base\_path('routes/api.php'));

}

}

Message.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Message extends Model

{

protected $guarded=[];

public function user()

{

return $this->belongsTo(User::class);

}

}

Post.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

protected $fillable = [

'content','user\_id','title'

];

public function user(){

return $this->belongsTo('App\User');

}

}

Transaction.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Transaction extends Model

{

protected $fillable = [

'user\_id','status\_id','weight','total','service','date\_reserve'

];

public function customer()

{

return $this->belongsTo('App\User', 'user\_id');

}

public function status()

{

return $this->belongsTo('App\Status');

}

}

User.php

<?php

namespace App;

use Illuminate\Notifications\Notifiable;

use Laravel\Passport\HasApiTokens;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable implements MustVerifyEmail

{

use Notifiable,HasApiTokens;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'name', 'email', 'password','phone'

];

/\*\*

\* The attributes that should be hidden for arrays.

\*

\* @var array

\*/

protected $hidden = [

'password', 'remember\_token',

];

public function transactions(){

return $this->hasMany('App\Transaction');

}

/\*\*

\* The attributes that should be cast to native types.

\*

\* @var array

\*/

protected $casts = [

'email\_verified\_at' => 'datetime',

];

public function messages()

{

return $this->hasMany(Message::class);

}

public function posts(){

return $this->hasMany('App\Post');

}

}

App.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Application Name

|--------------------------------------------------------------------------

|

| This value is the name of your application. This value is used when the

| framework needs to place the application's name in a notification or

| any other location as required by the application or its packages.

|

\*/

'name' => env('APP\_NAME', 'Laravel'),

/\*

|--------------------------------------------------------------------------

| Application Environment

|--------------------------------------------------------------------------

|

| This value determines the "environment" your application is currently

| running in. This may determine how you prefer to configure various

| services the application utilizes. Set this in your ".env" file.

|

\*/

'env' => env('APP\_ENV', 'production'),

/\*

|--------------------------------------------------------------------------

| Application Debug Mode

|--------------------------------------------------------------------------

|

| When your application is in debug mode, detailed error messages with

| stack traces will be shown on every error that occurs within your

| application. If disabled, a simple generic error page is shown.

|

\*/

'debug' => env('APP\_DEBUG', false),

/\*

|--------------------------------------------------------------------------

| Application URL

|--------------------------------------------------------------------------

|

| This URL is used by the console to properly generate URLs when using

| the Artisan command line tool. You should set this to the root of

| your application so that it is used when running Artisan tasks.

|

\*/

'url' => env('APP\_URL', 'http://localhost'),

'asset\_url' => env('ASSET\_URL', null),

/\*

|--------------------------------------------------------------------------

| Application Timezone

|--------------------------------------------------------------------------

|

| Here you may specify the default timezone for your application, which

| will be used by the PHP date and date-time functions. We have gone

| ahead and set this to a sensible default for you out of the box.

|

\*/

'timezone' => 'UTC',

/\*

|--------------------------------------------------------------------------

| Application Locale Configuration

|--------------------------------------------------------------------------

|

| The application locale determines the default locale that will be used

| by the translation service provider. You are free to set this value

| to any of the locales which will be supported by the application.

|

\*/

'locale' => 'en',

/\*

|--------------------------------------------------------------------------

| Application Fallback Locale

|--------------------------------------------------------------------------

|

| The fallback locale determines the locale to use when the current one

| is not available. You may change the value to correspond to any of

| the language folders that are provided through your application.

|

\*/

'fallback\_locale' => 'en',

/\*

|--------------------------------------------------------------------------

| Faker Locale

|--------------------------------------------------------------------------

|

| This locale will be used by the Faker PHP library when generating fake

| data for your database seeds. For example, this will be used to get

| localized telephone numbers, street address information and more.

|

\*/

'faker\_locale' => 'en\_US',

/\*

|--------------------------------------------------------------------------

| Encryption Key

|--------------------------------------------------------------------------

|

| This key is used by the Illuminate encrypter service and should be set

| to a random, 32 character string, otherwise these encrypted strings

| will not be safe. Please do this before deploying an application!

|

\*/

'key' => env('APP\_KEY'),

'cipher' => 'AES-256-CBC',

/\*

|--------------------------------------------------------------------------

| Autoloaded Service Providers

|--------------------------------------------------------------------------

|

| The service providers listed here will be automatically loaded on the

| request to your application. Feel free to add your own services to

| this array to grant expanded functionality to your applications.

|

\*/

'providers' => [

/\*

\* Laravel Framework Service Providers...

\*/

Illuminate\Auth\AuthServiceProvider::class,

Illuminate\Broadcasting\BroadcastServiceProvider::class,

Illuminate\Bus\BusServiceProvider::class,

Illuminate\Cache\CacheServiceProvider::class,

Illuminate\Foundation\Providers\ConsoleSupportServiceProvider::class,

Illuminate\Cookie\CookieServiceProvider::class,

Illuminate\Database\DatabaseServiceProvider::class,

Illuminate\Encryption\EncryptionServiceProvider::class,

Illuminate\Filesystem\FilesystemServiceProvider::class,

Illuminate\Foundation\Providers\FoundationServiceProvider::class,

Illuminate\Hashing\HashServiceProvider::class,

Illuminate\Mail\MailServiceProvider::class,

Illuminate\Notifications\NotificationServiceProvider::class,

Illuminate\Pagination\PaginationServiceProvider::class,

Illuminate\Pipeline\PipelineServiceProvider::class,

Illuminate\Queue\QueueServiceProvider::class,

Illuminate\Redis\RedisServiceProvider::class,

Illuminate\Auth\Passwords\PasswordResetServiceProvider::class,

Illuminate\Session\SessionServiceProvider::class,

Illuminate\Translation\TranslationServiceProvider::class,

Illuminate\Validation\ValidationServiceProvider::class,

Illuminate\View\ViewServiceProvider::class,

/\*

\* Package Service Providers...

\*/

/\*

\* Application Service Providers...

\*/

App\Providers\AppServiceProvider::class,

App\Providers\AuthServiceProvider::class,

App\Providers\BroadcastServiceProvider::class,

App\Providers\EventServiceProvider::class,

App\Providers\RouteServiceProvider::class,

],

/\*

|--------------------------------------------------------------------------

| Class Aliases

|--------------------------------------------------------------------------

|

| This array of class aliases will be registered when this application

| is started. However, feel free to register as many as you wish as

| the aliases are "lazy" loaded so they don't hinder performance.

|

\*/

'aliases' => [

'App' => Illuminate\Support\Facades\App::class,

'Arr' => Illuminate\Support\Arr::class,

'Artisan' => Illuminate\Support\Facades\Artisan::class,

'Auth' => Illuminate\Support\Facades\Auth::class,

'Blade' => Illuminate\Support\Facades\Blade::class,

'Broadcast' => Illuminate\Support\Facades\Broadcast::class,

'Bus' => Illuminate\Support\Facades\Bus::class,

'Cache' => Illuminate\Support\Facades\Cache::class,

'Config' => Illuminate\Support\Facades\Config::class,

'Cookie' => Illuminate\Support\Facades\Cookie::class,

'Crypt' => Illuminate\Support\Facades\Crypt::class,

'DB' => Illuminate\Support\Facades\DB::class,

'Eloquent' => Illuminate\Database\Eloquent\Model::class,

'Event' => Illuminate\Support\Facades\Event::class,

'File' => Illuminate\Support\Facades\File::class,

'Gate' => Illuminate\Support\Facades\Gate::class,

'Hash' => Illuminate\Support\Facades\Hash::class,

'Lang' => Illuminate\Support\Facades\Lang::class,

'Log' => Illuminate\Support\Facades\Log::class,

'Mail' => Illuminate\Support\Facades\Mail::class,

'Notification' => Illuminate\Support\Facades\Notification::class,

'Password' => Illuminate\Support\Facades\Password::class,

'Queue' => Illuminate\Support\Facades\Queue::class,

'Redirect' => Illuminate\Support\Facades\Redirect::class,

'Redis' => Illuminate\Support\Facades\Redis::class,

'Request' => Illuminate\Support\Facades\Request::class,

'Response' => Illuminate\Support\Facades\Response::class,

'Route' => Illuminate\Support\Facades\Route::class,

'Schema' => Illuminate\Support\Facades\Schema::class,

'Session' => Illuminate\Support\Facades\Session::class,

'Storage' => Illuminate\Support\Facades\Storage::class,

'Str' => Illuminate\Support\Str::class,

'URL' => Illuminate\Support\Facades\URL::class,

'Validator' => Illuminate\Support\Facades\Validator::class,

'View' => Illuminate\Support\Facades\View::class,

],

];

Auth.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Authentication Defaults

|--------------------------------------------------------------------------

|

| This option controls the default authentication "guard" and password

| reset options for your application. You may change these defaults

| as required, but they're a perfect start for most applications.

|

\*/

'defaults' => [

'guard' => 'web',

'passwords' => 'users',

],

/\*

|--------------------------------------------------------------------------

| Authentication Guards

|--------------------------------------------------------------------------

|

| Next, you may define every authentication guard for your application.

| Of course, a great default configuration has been defined for you

| here which uses session storage and the Eloquent user provider.

|

| All authentication drivers have a user provider. This defines how the

| users are actually retrieved out of your database or other storage

| mechanisms used by this application to persist your user's data.

|

| Supported: "session", "token"

|

\*/

'guards' => [

'web' => [

'driver' => 'session',

'provider' => 'users',

],

'api' => [

'driver' => 'passport',

'provider' => 'users',

'hash' => false,

],

],

/\*

|--------------------------------------------------------------------------

| User Providers

|--------------------------------------------------------------------------

|

| All authentication drivers have a user provider. This defines how the

| users are actually retrieved out of your database or other storage

| mechanisms used by this application to persist your user's data.

|

| If you have multiple user tables or models you may configure multiple

| sources which represent each model / table. These sources may then

| be assigned to any extra authentication guards you have defined.

|

| Supported: "database", "eloquent"

|

\*/

'providers' => [

'users' => [

'driver' => 'eloquent',

'model' => App\User::class,

],

// 'users' => [

// 'driver' => 'database',

// 'table' => 'users',

// ],

],

/\*

|--------------------------------------------------------------------------

| Resetting Passwords

|--------------------------------------------------------------------------

|

| You may specify multiple password reset configurations if you have more

| than one user table or model in the application and you want to have

| separate password reset settings based on the specific user types.

|

| The expire time is the number of minutes that the reset token should be

| considered valid. This security feature keeps tokens short-lived so

| they have less time to be guessed. You may change this as needed.

|

\*/

'passwords' => [

'users' => [

'provider' => 'users',

'table' => 'password\_resets',

'expire' => 60,

],

],

];

Broadcasting.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Broadcaster

|--------------------------------------------------------------------------

|

| This option controls the default broadcaster that will be used by the

| framework when an event needs to be broadcast. You may set this to

| any of the connections defined in the "connections" array below.

|

| Supported: "pusher", "redis", "log", "null"

|

\*/

'default' => env('BROADCAST\_DRIVER', 'null'),

/\*

|--------------------------------------------------------------------------

| Broadcast Connections

|--------------------------------------------------------------------------

|

| Here you may define all of the broadcast connections that will be used

| to broadcast events to other systems or over websockets. Samples of

| each available type of connection are provided inside this array.

|

\*/

'connections' => [

'pusher' => [

'driver' => 'pusher',

'key' => env('PUSHER\_APP\_KEY'),

'secret' => env('PUSHER\_APP\_SECRET'),

'app\_id' => env('PUSHER\_APP\_ID'),

'options' => [

'cluster' => env('PUSHER\_APP\_CLUSTER'),

'encrypted' => true,

],

],

'redis' => [

'driver' => 'redis',

'connection' => 'default',

],

'log' => [

'driver' => 'log',

],

'null' => [

'driver' => 'null',

],

],

];

Cache.php

Database’php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Database Connection Name

|--------------------------------------------------------------------------

|

| Here you may specify which of the database connections below you wish

| to use as your default connection for all database work. Of course

| you may use many connections at once using the Database library.

|

\*/

'default' => env('DB\_CONNECTION', 'mysql'),

/\*

|--------------------------------------------------------------------------

| Database Connections

|--------------------------------------------------------------------------

|

| Here are each of the database connections setup for your application.

| Of course, examples of configuring each database platform that is

| supported by Laravel is shown below to make development simple.

|

|

| All database work in Laravel is done through the PHP PDO facilities

| so make sure you have the driver for your particular database of

| choice installed on your machine before you begin development.

|

\*/

'connections' => [

'sqlite' => [

'driver' => 'sqlite',

'database' => env('DB\_DATABASE', database\_path('database.sqlite')),

'prefix' => '',

'foreign\_key\_constraints' => env('DB\_FOREIGN\_KEYS', true),

],

'mysql' => [

'driver' => 'mysql',

'host' => env('DB\_HOST', '127.0.0.1'),

'port' => env('DB\_PORT', '3306'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'unix\_socket' => env('DB\_SOCKET', ''),

'charset' => 'utf8mb4',

'collation' => 'utf8mb4\_unicode\_ci',

'prefix' => '',

'prefix\_indexes' => true,

'strict' => true,

'engine' => null,

'options' => extension\_loaded('pdo\_mysql') ? array\_filter([

PDO::MYSQL\_ATTR\_SSL\_CA => env('MYSQL\_ATTR\_SSL\_CA'),

]) : [],

],

'pgsql' => [

'driver' => 'pgsql',

'host' => env('DB\_HOST', '127.0.0.1'),

'port' => env('DB\_PORT', '5432'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'charset' => 'utf8',

'prefix' => '',

'prefix\_indexes' => true,

'schema' => 'public',

'sslmode' => 'prefer',

],

'sqlsrv' => [

'driver' => 'sqlsrv',

'host' => env('DB\_HOST', 'localhost'),

'port' => env('DB\_PORT', '1433'),

'database' => env('DB\_DATABASE', 'forge'),

'username' => env('DB\_USERNAME', 'forge'),

'password' => env('DB\_PASSWORD', ''),

'charset' => 'utf8',

'prefix' => '',

'prefix\_indexes' => true,

],

],

/\*

|--------------------------------------------------------------------------

| Migration Repository Table

|--------------------------------------------------------------------------

|

| This table keeps track of all the migrations that have already run for

| your application. Using this information, we can determine which of

| the migrations on disk haven't actually been run in the database.

|

\*/

'migrations' => 'migrations',

/\*

|--------------------------------------------------------------------------

| Redis Databases

|--------------------------------------------------------------------------

|

| Redis is an open source, fast, and advanced key-value store that also

| provides a richer body of commands than a typical key-value system

| such as APC or Memcached. Laravel makes it easy to dig right in.

|

\*/

'redis' => [

'client' => env('REDIS\_CLIENT', 'predis'),

'options' => [

'cluster' => env('REDIS\_CLUSTER', 'predis'),

],

'default' => [

'host' => env('REDIS\_HOST', '127.0.0.1'),

'password' => env('REDIS\_PASSWORD', null),

'port' => env('REDIS\_PORT', 6379),

'database' => env('REDIS\_DB', 0),

],

'cache' => [

'host' => env('REDIS\_HOST', '127.0.0.1'),

'password' => env('REDIS\_PASSWORD', null),

'port' => env('REDIS\_PORT', 6379),

'database' => env('REDIS\_CACHE\_DB', 1),

],

],

];

Filesystem.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Filesystem Disk

|--------------------------------------------------------------------------

|

| Here you may specify the default filesystem disk that should be used

| by the framework. The "local" disk, as well as a variety of cloud

| based disks are available to your application. Just store away!

|

\*/

'default' => env('FILESYSTEM\_DRIVER', 'local'),

/\*

|--------------------------------------------------------------------------

| Default Cloud Filesystem Disk

|--------------------------------------------------------------------------

|

| Many applications store files both locally and in the cloud. For this

| reason, you may specify a default "cloud" driver here. This driver

| will be bound as the Cloud disk implementation in the container.

|

\*/

'cloud' => env('FILESYSTEM\_CLOUD', 's3'),

/\*

|--------------------------------------------------------------------------

| Filesystem Disks

|--------------------------------------------------------------------------

|

| Here you may configure as many filesystem "disks" as you wish, and you

| may even configure multiple disks of the same driver. Defaults have

| been setup for each driver as an example of the required options.

|

| Supported Drivers: "local", "ftp", "sftp", "s3", "rackspace"

|

\*/

'disks' => [

'local' => [

'driver' => 'local',

'root' => storage\_path('app'),

],

'public' => [

'driver' => 'local',

'root' => storage\_path('app/public'),

'url' => env('APP\_URL').'/storage',

'visibility' => 'public',

],

's3' => [

'driver' => 's3',

'key' => env('AWS\_ACCESS\_KEY\_ID'),

'secret' => env('AWS\_SECRET\_ACCESS\_KEY'),

'region' => env('AWS\_DEFAULT\_REGION'),

'bucket' => env('AWS\_BUCKET'),

'url' => env('AWS\_URL'),

],

],

];

Hashing.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Hash Driver

|--------------------------------------------------------------------------

|

| This option controls the default hash driver that will be used to hash

| passwords for your application. By default, the bcrypt algorithm is

| used; however, you remain free to modify this option if you wish.

|

| Supported: "bcrypt", "argon", "argon2id"

|

\*/

'driver' => 'bcrypt',

/\*

|--------------------------------------------------------------------------

| Bcrypt Options

|--------------------------------------------------------------------------

|

| Here you may specify the configuration options that should be used when

| passwords are hashed using the Bcrypt algorithm. This will allow you

| to control the amount of time it takes to hash the given password.

|

\*/

'bcrypt' => [

'rounds' => env('BCRYPT\_ROUNDS', 10),

],

/\*

|--------------------------------------------------------------------------

| Argon Options

|--------------------------------------------------------------------------

|

| Here you may specify the configuration options that should be used when

| passwords are hashed using the Argon algorithm. These will allow you

| to control the amount of time it takes to hash the given password.

|

\*/

'argon' => [

'memory' => 1024,

'threads' => 2,

'time' => 2,

],

];

Logging.php

<?php

use Monolog\Handler\StreamHandler;

use Monolog\Handler\SyslogUdpHandler;

return [

/\*

|--------------------------------------------------------------------------

| Default Log Channel

|--------------------------------------------------------------------------

|

| This option defines the default log channel that gets used when writing

| messages to the logs. The name specified in this option should match

| one of the channels defined in the "channels" configuration array.

|

\*/

'default' => env('LOG\_CHANNEL', 'stack'),

/\*

|--------------------------------------------------------------------------

| Log Channels

|--------------------------------------------------------------------------

|

| Here you may configure the log channels for your application. Out of

| the box, Laravel uses the Monolog PHP logging library. This gives

| you a variety of powerful log handlers / formatters to utilize.

|

| Available Drivers: "single", "daily", "slack", "syslog",

| "errorlog", "monolog",

| "custom", "stack"

|

\*/

'channels' => [

'stack' => [

'driver' => 'stack',

'channels' => ['daily'],

'ignore\_exceptions' => false,

],

'single' => [

'driver' => 'single',

'path' => storage\_path('logs/laravel.log'),

'level' => 'debug',

],

'daily' => [

'driver' => 'daily',

'path' => storage\_path('logs/laravel.log'),

'level' => 'debug',

'days' => 14,

],

'slack' => [

'driver' => 'slack',

'url' => env('LOG\_SLACK\_WEBHOOK\_URL'),

'username' => 'Laravel Log',

'emoji' => ':boom:',

'level' => 'critical',

],

'papertrail' => [

'driver' => 'monolog',

'level' => 'debug',

'handler' => SyslogUdpHandler::class,

'handler\_with' => [

'host' => env('PAPERTRAIL\_URL'),

'port' => env('PAPERTRAIL\_PORT'),

],

],

'stderr' => [

'driver' => 'monolog',

'handler' => StreamHandler::class,

'formatter' => env('LOG\_STDERR\_FORMATTER'),

'with' => [

'stream' => 'php://stderr',

],

],

'syslog' => [

'driver' => 'syslog',

'level' => 'debug',

],

'errorlog' => [

'driver' => 'errorlog',

'level' => 'debug',

],

],

];

Mail.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Mail Driver

|--------------------------------------------------------------------------

|

| Laravel supports both SMTP and PHP's "mail" function as drivers for the

| sending of e-mail. You may specify which one you're using throughout

| your application here. By default, Laravel is setup for SMTP mail.

|

| Supported: "smtp", "sendmail", "mailgun", "mandrill", "ses",

| "sparkpost", "postmark", "log", "array"

|

\*/

'driver' => env('MAIL\_DRIVER', 'smtp'),

/\*

|------------a--------------------------------------------------------------

| SMTP Host Address

|--------------------------------------------------------------------------

|

| Here you may provide the host address of the SMTP server used by your

| applications. A default option is provided that is compatible with

| the Mailgun mail service which will provide reliable deliveries.

|

\*/

'host' => env('MAIL\_HOST', 'smtp.mailgun.org'),

/\*

|--------------------------------------------------------------------------

| SMTP Host Port

|--------------------------------------------------------------------------

|

| This is the SMTP port used by your application to deliver e-mails to

| users of the application. Like the host we have set this value to

| stay compatible with the Mailgun e-mail application by default.

|

\*/

'port' => env('MAIL\_PORT', 587),

/\*

|--------------------------------------------------------------------------

| Global "From" Address

|--------------------------------------------------------------------------

|

| You may wish for all e-mails sent by your application to be sent from

| the same address. Here, you may specify a name and address that is

| used globally for all e-mails that are sent by your application.

|

\*/

'from' => [

'address' => env('MAIL\_FROM\_ADDRESS', 'hello@example.com'),

'name' => env('MAIL\_FROM\_NAME', 'Example'),

],

/\*

|--------------------------------------------------------------------------

| E-Mail Encryption Protocol

|--------------------------------------------------------------------------

|

| Here you may specify the encryption protocol that should be used when

| the application send e-mail messages. A sensible default using the

| transport layer security protocol should provide great security.

|

\*/

'encryption' => env('MAIL\_ENCRYPTION', 'tls'),

/\*

|--------------------------------------------------------------------------

| SMTP Server Username

|--------------------------------------------------------------------------

|

| If your SMTP server requires a username for authentication, you should

| set it here. This will get used to authenticate with your server on

| connection. You may also set the "password" value below this one.

|

\*/

'username' => env('MAIL\_USERNAME'),

'password' => env('MAIL\_PASSWORD'),

/\*

|--------------------------------------------------------------------------

| Sendmail System Path

|--------------------------------------------------------------------------

|

| When using the "sendmail" driver to send e-mails, we will need to know

| the path to where Sendmail lives on this server. A default path has

| been provided here, which will work well on most of your systems.

|

\*/

'sendmail' => '/usr/sbin/sendmail -bs',

'stream' => [

'ssl' => [

'verify\_peer' => false,

'verify\_peer\_name' => false,

'allow\_self\_signed' => true,

],

],

/\*

|--------------------------------------------------------------------------

| Markdown Mail Settings

|--------------------------------------------------------------------------

|

| If you are using Markdown based email rendering, you may configure your

| theme and component paths here, allowing you to customize the design

| of the emails. Or, you may simply stick with the Laravel defaults!

|

\*/

'markdown' => [

'theme' => 'default',

'paths' => [

resource\_path('views/vendor/mail'),

],

],

/\*

|--------------------------------------------------------------------------

| Log Channel

|--------------------------------------------------------------------------

|

| If you are using the "log" driver, you may specify the logging channel

| if you prefer to keep mail messages separate from other log entries

| for simpler reading. Otherwise, the default channel will be used.

|

\*/

'log\_channel' => env('MAIL\_LOG\_CHANNEL'),

];

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Queue Connection Name

|--------------------------------------------------------------------------

|

| Laravel's queue API supports an assortment of back-ends via a single

| API, giving you convenient access to each back-end using the same

| syntax for every one. Here you may define a default connection.

|

\*/

'default' => env('QUEUE\_CONNECTION', 'sync'),

/\*

|--------------------------------------------------------------------------

| Queue Connections

|--------------------------------------------------------------------------

|

| Here you may configure the connection information for each server that

| is used by your application. A default configuration has been added

| for each back-end shipped with Laravel. You are free to add more.

|

| Drivers: "sync", "database", "beanstalkd", "sqs", "redis", "null"

|

\*/

'connections' => [

'sync' => [

'driver' => 'sync',

],

'database' => [

'driver' => 'database',

'table' => 'jobs',

'queue' => 'default',

'retry\_after' => 90,

],

'beanstalkd' => [

'driver' => 'beanstalkd',

'host' => 'localhost',

'queue' => 'default',

'retry\_after' => 90,

'block\_for' => 0,

],

'sqs' => [

'driver' => 'sqs',

'key' => env('AWS\_ACCESS\_KEY\_ID'),

'secret' => env('AWS\_SECRET\_ACCESS\_KEY'),

'prefix' => env('SQS\_PREFIX', 'https://sqs.us-east-1.amazonaws.com/your-account-id'),

'queue' => env('SQS\_QUEUE', 'your-queue-name'),

'region' => env('AWS\_DEFAULT\_REGION', 'us-east-1'),

],

'redis' => [

'driver' => 'redis',

'connection' => 'default',

'queue' => env('REDIS\_QUEUE', 'default'),

'retry\_after' => 90,

'block\_for' => null,

],

],

/\*

|--------------------------------------------------------------------------

| Failed Queue Jobs

|--------------------------------------------------------------------------

|

| These options configure the behavior of failed queue job logging so you

| can control which database and table are used to store the jobs that

| have failed. You may change them to any database / table you wish.

|

\*/

'failed' => [

'database' => env('DB\_CONNECTION', 'mysql'),

'table' => 'failed\_jobs',

],

];

Queue.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Queue Connection Name

|--------------------------------------------------------------------------

|

| Laravel's queue API supports an assortment of back-ends via a single

| API, giving you convenient access to each back-end using the same

| syntax for every one. Here you may define a default connection.

|

\*/

'default' => env('QUEUE\_CONNECTION', 'sync'),

/\*

|--------------------------------------------------------------------------

| Queue Connections

|--------------------------------------------------------------------------

|

| Here you may configure the connection information for each server that

| is used by your application. A default configuration has been added

| for each back-end shipped with Laravel. You are free to add more.

|

| Drivers: "sync", "database", "beanstalkd", "sqs", "redis", "null"

|

\*/

'connections' => [

'sync' => [

'driver' => 'sync',

],

'database' => [

'driver' => 'database',

'table' => 'jobs',

'queue' => 'default',

'retry\_after' => 90,

],

'beanstalkd' => [

'driver' => 'beanstalkd',

'host' => 'localhost',

'queue' => 'default',

'retry\_after' => 90,

'block\_for' => 0,

],

'sqs' => [

'driver' => 'sqs',

'key' => env('AWS\_ACCESS\_KEY\_ID'),

'secret' => env('AWS\_SECRET\_ACCESS\_KEY'),

'prefix' => env('SQS\_PREFIX', 'https://sqs.us-east-1.amazonaws.com/your-account-id'),

'queue' => env('SQS\_QUEUE', 'your-queue-name'),

'region' => env('AWS\_DEFAULT\_REGION', 'us-east-1'),

],

'redis' => [

'driver' => 'redis',

'connection' => 'default',

'queue' => env('REDIS\_QUEUE', 'default'),

'retry\_after' => 90,

'block\_for' => null,

],

],

/\*

|--------------------------------------------------------------------------

| Failed Queue Jobs

|--------------------------------------------------------------------------

|

| These options configure the behavior of failed queue job logging so you

| can control which database and table are used to store the jobs that

| have failed. You may change them to any database / table you wish.

|

\*/

'failed' => [

'database' => env('DB\_CONNECTION', 'mysql'),

'table' => 'failed\_jobs',

],

];

Service’php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Third Party Services

|--------------------------------------------------------------------------

|

| This file is for storing the credentials for third party services such

| as Stripe, Mailgun, SparkPost and others. This file provides a sane

| default location for this type of information, allowing packages

| to have a conventional place to find your various credentials.

|

\*/

'mailgun' => [

'domain' => env('MAILGUN\_DOMAIN'),

'secret' => env('MAILGUN\_SECRET'),

'endpoint' => env('MAILGUN\_ENDPOINT', 'api.mailgun.net'),

],

'postmark' => [

'token' => env('POSTMARK\_TOKEN'),

],

'ses' => [

'key' => env('AWS\_ACCESS\_KEY\_ID'),

'secret' => env('AWS\_SECRET\_ACCESS\_KEY'),

'region' => env('AWS\_DEFAULT\_REGION', 'us-east-1'),

],

'sparkpost' => [

'secret' => env('SPARKPOST\_SECRET'),

],

'stripe' => [

'model' => App\User::class,

'key' => env('STRIPE\_KEY'),

'secret' => env('STRIPE\_SECRET'),

'webhook' => [

'secret' => env('STRIPE\_WEBHOOK\_SECRET'),

'tolerance' => env('STRIPE\_WEBHOOK\_TOLERANCE', 300),

],

],

];

UserFactory.php

<?php

use App\User;

use Illuminate\Support\Str;

use Faker\Generator as Faker;

/\*

|--------------------------------------------------------------------------

| Model Factories

|--------------------------------------------------------------------------

|

| This directory should contain each of the model factory definitions for

| your application. Factories provide a convenient way to generate new

| model instances for testing / seeding your application's database.

|

\*/

$factory->define(User::class, function (Faker $faker) {

return [

'name' => $faker->name,

'email' => $faker->unique()->safeEmail,

'email\_verified\_at' => now(),

'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password

'remember\_token' => Str::random(10),

];

});

Migrations

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('users', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('name');

$table->string('email')->unique();

$table->string('type')->default('Customer');

$table->string('phone');

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('users');

}

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreatePasswordResetsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('password\_resets', function (Blueprint $table) {

$table->string('email')->index();

$table->string('token');

$table->timestamp('created\_at')->nullable();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('password\_resets');

}

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateMessagesTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('messages', function (Blueprint $table) {

$table->bigIncrements('id');

$table->unsignedInteger('user\_id');

$table->unsignedInteger('receiver\_id')->nullable();

$table->text('message')->nullable();

$table->string('image')->nullable();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('messages');

}

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreatePostsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('posts', function (Blueprint $table) {

$table->bigIncrements('id');

$table->unsignedInteger('user\_id');

$table->string('image')->nullable();

$table->string('title');

$table->longText('content');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('posts');

}

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreatePostsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('posts', function (Blueprint $table) {

$table->bigIncrements('id');

$table->unsignedInteger('user\_id');

$table->string('image')->nullable();

$table->string('title');

$table->longText('content');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('posts');

}

}

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateStatusesTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('statuses', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('name')->default('Not Approved');

$table->unsignedInteger('percent')->default(0);

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('statuses');

}

}

<?php

use App\User;

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class SeedUsersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

User::create([

'name' => 'Lavada Queen',

'type' => 'Laundry Shop',

'email' => 'lavada@gmail.com',

'phone' => '09472195796',

'email\_verified\_at' => NOW(),

'password' => Hash::make('admin123')

]);

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

//

}

}

<?php

use App\Status;

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class SeedStatusesTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Status::create([

'name' => 'Pending',

'percent' => 0,

]);

Status::create([

'name' => 'Approved',

'percent' => 1,

]);

Status::create([

'name' => 'Order Placed',

'percent' => 10,

]);

Status::create([

'name' => 'Washing',

'percent' => 30,

]);

Status::create([

'name' => 'Drying',

'percent' => 50,

]);

Status::create([

'name' => 'Quality Check',

'percent' => 70,

]);

Status::create([

'name' => 'Out for Delivery',

'percent' => 100,

]);

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

//

}

}

Databaseseeder.php

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

/\*\*

\* Seed the application's database.

\*

\* @return void

\*/

public function run()

{

// $this->call(UsersTableSeeder::class);

}

}

Components

Message.listvue

<template>

<div>

<v-list

class="p-3"

v-for="(message, index) in allMessages"

:key="index"

>

<div class="message-wrapper">

<v-flex>

<span class="caption font-bold">{{message.user.name}}</span>

</v-flex>

<div v-if="message.message" class="text-message-container">

<v-chip :color="(user.id===message.user\_id)?'green':'red'" text-color="white">

{{message.message}}

</v-chip>

</div>

<div class="image-container">

<img v-if="message.image" :src="'/storage/'+message.image" alt="">

</div>

<v-flex class="small font-italic">

{{message.created\_at}}

</v-flex>

</div>

</v-list>

</div>

</template>

<script>

export default {

props:['user','allMessages'],

};

</script>

<style scoped>

.chat-card{

margin-bottom:140px;

}

.floating-div{

position: fixed;

}

.chat-card img {

max-width: 300px;

max-height: 200px;

}

</style>

Admin.vue

<template>

<v-app id="inspire" dark>

<v-navigation-drawer

v-model="drawer"

clipped

fixed

app

>

<v-list dense>

<v-list-tile @click="">

<v-list-tile-action>

<v-icon>dashboard</v-icon>

</v-list-tile-action>

<v-list-tile-content>

<v-list-tile-title>Dashboard</v-list-tile-title>

</v-list-tile-content>

</v-list-tile>

<v-list-tile @click="">

<v-list-tile-action>

<v-icon>settings</v-icon>

</v-list-tile-action>

<v-list-tile-content>

<v-list-tile-title>Settings</v-list-tile-title>

</v-list-tile-content>

</v-list-tile>

</v-list>

</v-navigation-drawer>

<v-toolbar app fixed clipped-left>

<v-toolbar-side-icon @click.stop="drawer = !drawer"></v-toolbar-side-icon>

<v-toolbar-title>Application</v-toolbar-title>

</v-toolbar>

<v-content>

<v-container fluid fill-height>

<v-layout justify-center align-center>

<v-flex shrink>

<v-tooltip right>

<template v-slot:activator="{ on }">

<v-btn :href="source" icon large target="\_blank" v-on="on">

<v-icon large>code</v-icon>

</v-btn>

</template>

<span>Source</span>

</v-tooltip>

<v-tooltip right>

<template v-slot:activator="{ on }">

<v-btn icon large href="https://codepen.io/johnjleider/pen/qxQWda" target="\_blank" v-on="on">

<v-icon large>mdi-codepen</v-icon>

</v-btn>

</template>

<span>Codepen</span>

</v-tooltip>

</v-flex>

</v-layout>

</v-container>

</v-content>

<v-footer app fixed>

<span>&copy; 2017</span>

</v-footer>

</v-app>

</template>

<script>

export default {

data: () => ({

drawer: null

}),

props: {

source: String

}

};

</script>

Appointmeny.vue

<template>

<div class="container-fluid">

<div class="row pt-5" v-if="$gate.laundry()">

<div class="col-md-12">

<div class="card">

<div class="card-header">

<h3 class="card-title">Transaction Table</h3>

<div class="card-tools">

</div>

</div>

<v-card>

<v-card-title>

Transaction Data

<v-spacer></v-spacer>

<v-text-field

v-model="search"

append-icon="search"

label="Search"

single-line

hide-details

></v-text-field>

</v-card-title>

<v-data-table

:headers="headers"

:items="transactions"

:search="search"

>

<template v-slot:items="props">

<td>{{ props.item.customer.name }}</td>

<td class="text-xs-left">{{ props.item.service}}</td>

<td class="text-xs-left">{{ props.item.weight}}</td>

<td class="text-xs-left"><span class="label label-warning">{{ props.item.status.name}}</span></td>

<td class="text-xs-left" >Php{{ props.item.total}}</td>

<td class="text-xs-left">{{ props.item.created\_at |myDate }}</td>

<td class="text-xs-left">

<a href="#" class="btn btn-primary" @click="editModal(props.item)">

Transact

</a>

<a href="#" class="btn btn-warning" @click="editStatus(props.item)">

Status change

</a>

<a href="#" class="btn btn-danger" @click="editStatus(props.item)">

<i class="fa fa-trash"> </i>

</a>

</td>

</template>

<v-alert v-slot:no-results :value="true" color="error" icon="warning">

Your search for "{{ search }}" found no results.

</v-alert>

</v-data-table>

</v-card>

</div>

</div>

</div>

<div class="modal fade" id="addNew" tabindex="-1" role="dialog" aria-labelledby="addNewLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="addNewLabel">Update Transaction Info</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<form @submit.prevent=" updateAppointment()">

<div class="modal-body">

<input type="text" :value="form.name" class="form-control" disabled name="user"><br><br>

<input type="service" :value="form.service" class="form-control" disabled name="service" >

<br>

<!-- <select name="status" class="form-control" v-model="form.status">

<option v-for="status in statuses" :key="status.id" :value="status.id">{{status.name}}</option>

</select> -->

<br><br>

<input type="number" class="form-control" v-model="form.weight" name="weight" placeholder="Enter kilogram weight" style="width:100%;">

<br><br>

<div v-if="form.service === 'Fullservice'">

<h3>PHP: </h3>

<input type="service" v-model="form.total = form.weight\*23" class="form-control" disabled name="total" >

</div>

<div v-if="form.service === 'Vintage'">

<h3>PHP: </h3>

<input type="service" v-model="form.total = form.weight\*32" class="form-control" disabled name="total" >

</div>

<div v-if="form.service === 'Finelinen'">

<h3>PHP: </h3>

<input type="service" v-model="form.total = form.weight\*28" class="form-control" disabled name="total" >

</div>

<div v-if="form.service === 'Garment'">

<h3>PHP: </h3>

<input type="service" v-model="form.total = form.weight\*38" class="form-control" disabled name="total" >

</div>

<div v-if="form.service === 'Carpet'">

<h3>PHP: </h3>

<input type="service" v-model="form.total = form.weight\*45" class="form-control" disabled name="total" >

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Submit</button>

</div>

</form>

</div>

</div>

</div>

<div class="modal fade" id="changeStatus" tabindex="-1" role="dialog" aria-labelledby="addNewLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="addNewLabel">Update Transaction Info</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<form @submit.prevent="statusUpdate()">

<div class="modal-body">

<h3 style="display:inline;"> Current Status:</h3> &nbsp;<span style="display:inline; color:blue;"><b>{{ this.form.current}}</b></span>

<br>

<select name="status" class="form-control" v-model="form.status">

<option v-for="status in statuses" :key="status.id" :value="status.id">{{status.name}}</option>

</select>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Submit</button>`

</div>

</form>

</div>

</div>

</div>

</div>

</template>

<script>

export default{

data(){

return{

transactions:[],

search:'',

price:0,

statuses:[],

headers:

[

{

text: 'Name',

align: 'left',

sortable:true,

value: 'customer.name'

},

{text:'Service', value:'service'},

{text:'Weight' ,value:'weight'},

{text:'Status' ,value:'status'},

{text:'Total',value:'total'},

{ text: 'Created At', value: 'created\_at' },

{text:'Modify' , value:'modify'}

],

form: new Form({

id: '',

name:'',

service:'',

weight:'',

total:'',

status:'',

current:''

})

}

},

mounted(){

if(this.$gate.laundry()){

this.loadTransact();

this.loadStatus();

this.statusChanged();

}

},

methods:{

/\*newModal(){

this.editmode = false;

this.form.reset();

$('#addNew').modal('show');

},\*/

statusUpdate(){

let status = this.form.status

this.form.put('api/statuschange/'+this.form.id)

.then(()=>{

this.$Progress.start();

$('#changeStatus').modal('hide')

toast.fire(

'Updated!',

'Status has been updated.',

'success'

)

this.$Progress,finish();

Fire.$emit('statusChanged')

})

},

editStatus(changestatus){

this.form.id = changestatus.id

this.form.current = changestatus.status.name

this.form.status = changestatus.status

$('#changeStatus').modal('show')

},

editModal(transaction){

this.form.id = transaction.id

this.form.user\_id = transaction.user\_id

this.form.name = transaction.customer.name

this.form.service = transaction.service

this.form.total = transaction.total

this.form.weight = transaction.weight

$('#addNew').modal('show')

},

updateAppointment(){

this.form.put('api/transactions/'+this.form.id)

.then(()=>{

this.$Progress.start();

$('#addNew').modal('hide')

toast.fire(

'Updated!',

'Transaction has been updated.',

'success'

)

this.$Progress,finish();

Fire.$emit('statusChanged')

}).catch((err)=>{

console.log(err);

})

},

loadTransact(){

axios.get('api/transactions').then(({data}) => this.transactions = data)

},

statusChanged(){

this.loadTransact();

Fire.$on('statusChanged',()=>{

this.loadTransact();

})

},

loadStatus(){

axios.get('api/getstatus').then(({data}) => this.statuses = data)

}

},

};

</script>

Chat.vue

<template>

<v-layout row>

<v-flex class="online-users" xs3>

<v-list v-if="user.type ==='Laundry Shop'">

<v-list-tile

v-for="friend in friends"

:color="(friend.id==activeFriend)?'green':''"

:key="friend.id"

@click="activeFriend=friend.id"

>

<v-list-tile-action>

<v-icon :color="(onlineFriends.find(user=>user.id===friend.id))?'green':'red'">account\_circle</v-icon>

</v-list-tile-action>

<v-list-tile-content>

<v-list-tile-title>{{friend.name}}</v-list-tile-title>

</v-list-tile-content>

<!-- <v-list-tile-avatar>

<img :src="item.avatar">

</v-list-tile-avatar> -->

</v-list-tile>

</v-list>

<v-list v-if="user.type ==='Customer'">

<v-list-tile

v-for="shop in shops"

:color="(friend.id==activeFriend)?'green':''"

:key="friend.id"

@click="activeFriend=friend.id"

>

<v-list-tile-action>

<v-icon :color="(onlineFriends.find(user=>user.id===friend.id))?'green':'red'">account\_circle</v-icon>

</v-list-tile-action>

<v-list-tile-content>

<v-list-tile-title>{{shop.name}}</v-list-tile-title>

</v-list-tile-content>

<!-- <v-list-tile-avatar>

<img :src="item.avatar">

</v-list-tile-avatar> -->

</v-list-tile>

</v-list>

</v-flex>

<v-flex id="privateMessageBox" class="messages mb-5" xs9>

<message-list :user="user" :all-messages="allMessages"></message-list>

<div class="floating-div">

<picker v-if="emoStatus" set="emojione" @select="onInput" title="Pick your emoji…" />

</div>

<v-footer

height="auto"

fixed

color="grey"

>

<v-layout row >

<v-flex class="ml-2 text-right" xs1>

<v-btn @click="toggleEmo" fab dark small color="pink">

<v-icon>insert\_emoticon </v-icon>

</v-btn>

</v-flex>

<v-flex xs1 class="text-center">

<file-upload

:post-action="'/private-messages/'+activeFriend"

ref='upload'

v-model="files"

@input-file="$refs.upload.active = true"

:headers="{'X-CSRF-TOKEN': token}"

>

<v-icon class="mt-3">attach\_file</v-icon>

</file-upload>

</v-flex>

<v-flex xs6 >

<v-text-field

rows=2

v-model="message"

label="Enter Message"

single-line

@keyup.enter="sendMessage"

></v-text-field>

</v-flex>

<v-flex xs4>

<v-btn

@click="sendMessage"

dark class="mt-3 ml-2 white--text" small color="green">send</v-btn>

</v-flex>

</v-layout>

</v-footer>

</v-flex>

</v-layout>

</template>

<script>

import MessageList from './\_message-list'

import { Picker } from 'emoji-mart-vue'

export default {

props:['user'],

components:{

Picker,

MessageList

},

data () {

return {

message:null,

files:[],

activeFriend:null,

typingFriend:{},

onlineFriends:[],

allMessages:[],

typingClock:null,

emoStatus:false,

users:[],

shops:[],

token:document.head.querySelector('meta[name="csrf-token"]').content

}

},

computed:{

friends(){

return this.users.filter((user)=>{

return user.id !==this.user.id;

})

}

},

watch:{

files:{

deep:true,

handler(){

let success=this.files[0].success;

if(success){

this.fetchMessages();

}

}

},

activeFriend(val){

this.fetchMessages();

},

'$refs.upload'(val){

console.log(val);

}

},

methods:{

onTyping(){

Echo.private('privatechat.'+this.activeFriend).whisper('typing',{

user:this.user

});

},

sendMessage(){

//check if there message

if(!this.message){

return alert('Please enter message');

}

if(!this.activeFriend){

return alert('Please select friend');

}

axios.post('/private-messages/'+this.activeFriend, {message: this.message}).then(response => {

this.message=null;

this.allMessages.push(response.data.message)

setTimeout(this.scrollToEnd,100);

});

},

fetchMessages() {

if(!this.activeFriend){

return alert('Please select friend');

}

axios.get('/private-messages/'+this.activeFriend).then(response => {

this.allMessages = response.data;

setTimeout(this.scrollToEnd,100);

});

},

fetchUsers() {

axios.get('/users').then(response => {

this.users = response.data;

if(this.friends.length>0){

this.activeFriend=this.friends[0].id;

}

});

},

fetchShop(){

axios.get('/shop').then(response => {

this.shop = response.data;

if(this.friends.length>0){

this.activeFriend=this.friends[0].id;

}

})

},

scrollToEnd(){

document.getElementById('privateMessageBox').scrollTo(0,99999);

},

toggleEmo(){

this.emoStatus= !this.emoStatus;

},

onInput(e){

if(!e){

return false;

}

if(!this.message){

this.message=e.native;

}else{

this.message=this.message + e.native;

}

this.emoStatus=false;

},

onResponse(e){

console.log('onrespnse file up',e);

}

},

mounted(){

},

created(){

this.fetchUsers();

this.fetchShop();

Echo.join('plchat')

.here((users) => {

console.log('online',users);

this.onlineFriends=users;

})

.joining((user) => {

this.onlineFriends.push(user);

console.log('joining',user.name);

})

.leaving((user) => {

this.onlineFriends.splice(this.onlineFriends.indexOf(user),1);

console.log('leaving',user.name);

});

Echo.private('privatechat.'+this.user.id)

.listen('PrivateMessageSent',(e)=>{

console.log('pmessage sent')

this.activeFriend=e.message.user\_id;

this.allMessages.push(e.message)

setTimeout(this.scrollToEnd,100);

})

.listenForWhisper('typing', (e) => {

if(e.user.id==this.activeFriend){

this.typingFriend=e.user;

if(this.typingClock) clearTimeout();

this.typingClock=setTimeout(()=>{

this.typingFriend={};

},9000);

}

});

}

};

</script>

<style scoped>

.online-users,.messages{

overflow-y:scroll;

height:100vh;

}

</style>

Customer.vue

<template>

<div class="container-fluid pt-5">

<v-card>

<v-card-title>

Customers Data

<v-spacer></v-spacer>

<v-text-field

v-model="search"

append-icon="search"

label="Search"

single-line

hide-details

></v-text-field>

</v-card-title>

<v-data-table

:headers="headers"

:items="users"

:search="search"

>

<template v-slot:items="props">

<td>{{ props.item.name }}</td>

<td class="text-xs-left">{{ props.item.email}}</td>

<td class="text-xs-left">{{ props.item.type }}</td>

<td class="text-xs-left">{{ props.item.created\_at }}</td>

</template>

<v-alert v-slot:no-results :value="true" color="error" icon="warning">

Your search for "{{ search }}" found no results.

</v-alert>

</v-data-table>

</v-card>

</div>

</template>

<script>

export default {

data(){

return{

users:[],

search: '',

headers: [

{

text: 'Name',

align: 'left',

sortable: true,

value: 'name'

},

{ text: 'Email', value: 'email' },

{ text: 'Type', value: 'type' },

{ text: 'Created At', value: 'created' }

]

}

},

created(){

this.loadCustomers();

},

methods: {

loadCustomers(){

if(this.$gate.laundry()){

axios.get('api/customers').then(({data})=>(this.users = data));

}

}

}

};

</script>

Dashboard.vue

<template>

<div class="container-fluid">

<div class="pt-5" v-if="this.$gate.laundry()">

<div class="row">

<!-- Column -->

<div class="col-lg-4 col-md-4">

<div class="card card-inverse card-primary">

<div class="card-body">

<div class="d-flex">

<div class="m-r-20 align-self-center">

<h1 class="text-white"><i class="ti-user"></i></h1>

</div>

<div>

<h3 class="card-title">Customer</h3>

<h6 class="card-subtitle">2019</h6>

</div>

</div>

<div class="row">

<div class="col-4 align-self-center">

<h2 class="font-light text-white">{{customers}}</h2>

</div>

<div class="col-8 p-t-10 p-b-20 align-self-center">

<div class="usage chartist-chart" style="height:65px"></div>

</div>

</div>

</div>

</div>

</div>

<!-- Column -->

<!-- Column -->

<div class="col-lg-4 col-md-4">

<div class="card card-inverse card-success">

<div class="card-body">

<div class="d-flex">

<div class="m-r-20 align-self-center">

<h1 class="text-white"><i class="ti-thought"></i></h1>

</div>

<div>

<h3 class="card-title">Post</h3>

<h6 class="card-subtitle">2019</h6>

</div>

</div>

<div class="row">

<div class="col-4 align-self-center">

<h2 class="font-light text-white">{{posts}}</h2>

</div>

<div class="col-8 p-t-10 p-b-20 text-right">

<div class="spark-count" style="height:65px"></div>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-4 col-md-4">

<div class="card card-inverse card-warning">

<div class="card-body">

<div class="d-flex">

<div class="m-r-20 align-self-center">

<h1 class="text-white"><i class="ti-thought"></i></h1>

</div>

<div>

<h3 class="card-title">Transaction</h3>

<h6 class="card-subtitle">2019</h6>

</div>

</div>

<div class="row">

<div class="col-4 align-self-center">

<h2 class="font-light text-white">{{transactions}}</h2>

</div>

<div class="col-8 p-t-10 p-b-20 text-right">

<div class="spark-count" style="height:65px"></div>

</div>

</div>

</div>

</div>

</div>

<!-- Column -->

<!-- Column -->

<!-- Column -->

</div>

</div>

<full-calendar :events="fcEvents" locale="en"></full-calendar>

</div>

</template>

<script type="text/javascript">

var demoEvents = [

{

title : 'Laurence Anthony',

start : '2019-04-25',

end : '2019-04-27'

},

{

title: 'Leodivino Lawas',

start : '2019-04-05',

end : '2019-04-15'

}

]

export default {

data () {

return {

fcEvents : demoEvents,

customers: [],

posts:[],

transactions:[]

}

},

mounted(){

if(this.$gate.laundry()){

this.countCustomer();

this.countPost();

this.countTransaction();

}

},

components : {

'full-calendar': require('vue-fullcalendar')

},

methods:{

countCustomer(){

axios.get('api/countCustomer').then(({data}) => {

this.customers = data;

})

},

countPost(){

axios.get('api/countPost').then(({data}) => {

this.posts = data;

})

},

countTransaction(){

axios.get('api/countTransaction').then(({data}) => {

this.transactions = data;

})

},

}

};

</script>

ExampleComponent.vue

<template>

<v-app id="inspire">

<v-navigation-drawer

v-model="drawer"

:clipped="$vuetify.breakpoint.lgAndUp"

fixed

app

>

<v-list dense>

<template v-for="item in items">

<v-layout

v-if="item.heading"

:key="item.heading"

row

align-center

>

<v-flex xs6>

<v-subheader v-if="item.heading">

{{ item.heading }}

</v-subheader>

</v-flex>

<v-flex xs6 class="text-xs-center">

<a href="#!" class="body-2 black--text">EDIT</a>

</v-flex>

</v-layout>

<v-list-group

v-else-if="item.children"

:key="item.text"

v-model="item.model"

:prepend-icon="item.model ? item.icon : item['icon-alt']"

append-icon=""

>

<template v-slot:activator>

<v-list-tile>

<v-list-tile-content>

<v-list-tile-title>

{{ item.text }}

</v-list-tile-title>

</v-list-tile-content>

</v-list-tile>

</template>

<v-list-tile

v-for="(child, i) in item.children"

:key="i"

@click=""

>

<v-list-tile-action v-if="child.icon">

<v-icon>{{ child.icon }}</v-icon>

</v-list-tile-action>

<v-list-tile-content>

<v-list-tile-title>

{{ child.text }}

</v-list-tile-title>

</v-list-tile-content>

</v-list-tile>

</v-list-group>

<v-list-tile v-else :key="item.text" @click="">

<v-list-tile-action>

<v-icon>{{ item.icon }}</v-icon>

</v-list-tile-action>

<v-list-tile-content>

<v-list-tile-title>

{{ item.text }}

</v-list-tile-title>

</v-list-tile-content>

</v-list-tile>

</template>

</v-list>

</v-navigation-drawer>

<v-toolbar

:clipped-left="$vuetify.breakpoint.lgAndUp"

color="blue darken-3"

dark

app

fixed

>

<v-toolbar-title style="width: 300px" class="ml-0 pl-3">

<v-toolbar-side-icon @click.stop="drawer = !drawer"></v-toolbar-side-icon>

<span class="hidden-sm-and-down">Google Contacts</span>

</v-toolbar-title>

<v-text-field

flat

solo-inverted

hide-details

prepend-inner-icon="search"

label="Search"

class="hidden-sm-and-down"

></v-text-field>

<v-spacer></v-spacer>

<v-btn icon>

<v-icon>apps</v-icon>

</v-btn>

<v-btn icon>

<v-icon>notifications</v-icon>

</v-btn>

<v-btn icon large>

<v-avatar size="32px" tile>

<img

src="https://cdn.vuetifyjs.com/images/logos/logo.svg"

alt="Vuetify"

>

</v-avatar>

</v-btn>

</v-toolbar>

<v-content>

<v-container fluid fill-height>

<v-layout justify-center align-center>

<v-tooltip right>

<template v-slot:activator="{ on }">

<v-btn :href="source" icon large target="\_blank" v-on="on">

<v-icon large>code</v-icon>

</v-btn>

</template>

<span>Source</span>

</v-tooltip>

<v-tooltip right>

<template v-slot:activator="{ on }">

<v-btn icon large href="https://codepen.io/johnjleider/pen/EQOYVV" target="\_blank" v-on="on">

<v-icon large>mdi-codepen</v-icon>

</v-btn>

</template>

<span>Codepen</span>

</v-tooltip>

</v-layout>

</v-container>

</v-content>

<v-btn

fab

bottom

right

color="pink"

dark

fixed

@click="dialog = !dialog"

>

<v-icon>add</v-icon>

</v-btn>

<v-dialog v-model="dialog" width="800px">

<v-card>

<v-card-title

class="grey lighten-4 py-4 title"

>

Create contact

</v-card-title>

<v-container grid-list-sm class="pa-4">

<v-layout row wrap>

<v-flex xs12 align-center justify-space-between>

<v-layout align-center>

<v-avatar size="40px" class="mr-3">

<img

src="//ssl.gstatic.com/s2/oz/images/sge/grey\_silhouette.png"

alt=""

>

</v-avatar>

<v-text-field

placeholder="Name"

></v-text-field>

</v-layout>

</v-flex>

<v-flex xs6>

<v-text-field

prepend-icon="business"

placeholder="Company"

></v-text-field>

</v-flex>

<v-flex xs6>

<v-text-field

placeholder="Job title"

></v-text-field>

</v-flex>

<v-flex xs12>

<v-text-field

prepend-icon="mail"

placeholder="Email"

></v-text-field>

</v-flex>

<v-flex xs12>

<v-text-field

type="tel"

prepend-icon="phone"

placeholder="(000) 000 - 0000"

mask="phone"

></v-text-field>

</v-flex>

<v-flex xs12>

<v-text-field

prepend-icon="notes"

placeholder="Notes"

></v-text-field>

</v-flex>

</v-layout>

</v-container>

<v-card-actions>

<v-btn flat color="primary">More</v-btn>

<v-spacer></v-spacer>

<v-btn flat color="primary" @click="dialog = false">Cancel</v-btn>

<v-btn flat @click="dialog = false">Save</v-btn>

</v-card-actions>

</v-card>

</v-dialog>

</v-app>

</template>

<script>

export default {

data: () => ({

dialog: false,

drawer: null,

items: [

{ icon: 'contacts', text: 'Contacts' },

{ icon: 'history', text: 'Frequently contacted' },

{ icon: 'content\_copy', text: 'Duplicates' },

{

icon: 'keyboard\_arrow\_up',

'icon-alt': 'keyboard\_arrow\_down',

text: 'Labels',

model: true,

children: [

{ icon: 'add', text: 'Create label' }

]

},

{

icon: 'keyboard\_arrow\_up',

'icon-alt': 'keyboard\_arrow\_down',

text: 'More',

model: false,

children: [

{ text: 'Import' },

{ text: 'Export' },

{ text: 'Print' },

{ text: 'Undo changes' },

{ text: 'Other contacts' }

]

},

{ icon: 'settings', text: 'Settings' },

{ icon: 'chat\_bubble', text: 'Send feedback' },

{ icon: 'help', text: 'Help' },

{ icon: 'phonelink', text: 'App downloads' },

{ icon: 'keyboard', text: 'Go to the old version' }

]

}),

props: {

source: String

}

};

</script>

Home.vue

<template>

<div class="container" v-if="$gate.customer()">

<div class="card-body mt-5" v-for="post in posts">

<div class="blog-image"><img src="/assets/images/lavada.jpg" alt="img" class="img-responsive"/></div>

<h2>{{post.title}}</h2>

<label class="label label-rounded label-success">{{post.user.name}}</label>

<p class="m-t-20 m-b-20">

{{post.content}}

</p>

<div class="d-flex">

<div class="read"><a href="javascript:void(0)" class="link font-medium">View Profile</a></div>

<div class="ml-auto">

<a href="javascript:void(0)" class="link m-r-10 " data-toggle="tooltip" title="Like"><i class="mdi mdi-heart-outline"></i></a>

</div>

</div>

</div>

</div>

</template>

<script>

export default{

data(){

return{

posts: []

}

},

created(){

if(this.$gate.customer()){

this.viewPost();

window.Echo.channel('posts').listen('PostCreated',({post}) => this.addPost(post));

}

},

methods:{

viewPost(){

axios.get('api/viewpost').then(({data}) => this.posts = data)

}

}

};

</script>

Maps.vue

<template>

<div id="container-fluid">

<div id="floating-panel">

<!-- <input id="latlng" type="text" value="40.714224,-73.961452"> -->

<select id="latlng">

<option value="40.714224,-73.961452"> 1F SOTTO BRANCH</option>

<option value="10.3139,123.9012"> AYALA BRANCH </option>

</select>

<input id="submit" type="button" value="FIND NOW">

</div>

<div id="map"></div>

</div>

</template>

<script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCB5UntvXRlD5c0WGM1qCffiEVP1cE0w6M&callback=initMap">

</script>

<script type="text/javascript">

function initMap() {

var map = new google.maps.Map(document.getElementById('map'), {

zoom: 12,

center: {lat:10.3157, lng: 123.8854}

});

var geocoder = new google.maps.Geocoder;

var infowindow = new google.maps.InfoWindow;

document.getElementById('submit').addEventListener('click', function() {

geocodeLatLng(geocoder, map, infowindow);

});

}

function geocodeLatLng(geocoder, map, infowindow) {

var input = document.getElementById('latlng').value;

var latlngStr = input.split(',', 2);

var latlng = {lat: parseFloat(latlngStr[0]), lng: parseFloat(latlngStr[1])};

geocoder.geocode({'location': latlng}, function(results, status) {

if (status === 'OK') {

if (results[0]) {

map.setZoom(20);

var marker = new google.maps.Marker({

position: latlng,

map: map

});

infowindow.setContent(results[0].formatted\_address);

infowindow.open(map, marker);

} else {

window.alert('No results found');

}

} else {

window.alert('Geocoder failed due to: ' + status);

}

});

}

</script>

Post.view

<template>

<div class="container-fluid">

<div class="row pt-5" v-if="$gate.laundry()">

<div class="col-md-12">

<div class="card">

<div class="card-header">

<h3 class="card-title">Post Table</h3>

<div class="card-tools">

<button class="btn btn-warning" @click="newModal">Add Post <v-icon color="#fff">add\_box</v-icon></button>

</div>

</div>

</div>

</div>

</div>

<v-card>

<v-card-title>

Customers Data

<v-spacer></v-spacer>

<v-text-field

v-model="search"

append-icon="search"

label="Search"

single-line

hide-details

></v-text-field>

</v-card-title>

<v-data-table

:headers="headers"

:items="posts"

:search="search"

>

<template v-slot:items="props">

<td>{{ props.item.title }}</td>

<td class="text-xs-left">{{ props.item.content}}</td>

<td class="text-xs-left">{{ props.item.created\_at }}</td>

<td class="text-xs-left">

<a href="#" @click="editModal(user)">

<i class="fa fa-edit" style="color:blue;"></i>

</a>

/

<a href="#" @click="deletePost(props.item.id)">

<i class="fa fa-trash" style="color:red;"></i>

</a>

</td>

</template>

<v-alert v-slot:no-results :value="true" color="error" icon="warning">

Your search for "{{ search }}" found no results.

</v-alert>

</v-data-table>

</v-card>

<div class="modal fade" id="addNew" tabindex="-1" role="dialog" aria-labelledby="addNewLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" v-show="!editmode" id="addNewLabel">Add Post</h5>

<h5 class="modal-title" v-show="editmode" id="addNewLabel">Update User's Info</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<form @submit.prevent="editmode ? updateUser() :createPost()">

<div class="modal-body">

<!-- <select class="form-control" name="transaction\_id" v-model="form.transaction\_id">

<option value="">---Select Account Name---</option>

<option v-for="account in accounts" :key="account.id" :value="account.id">{{account.account\_name}}</option>

</select>

<br>

<input type="text" class="form-control" placeholder="Enter Passenger Name" name="passenger\_name" v-model="form.passenger\_name"><br> -->

<input type="text" name="title" placeholder="Enter title of post" :class="{'is-invalid': form.errors.has('content') }" v-model="form.title" class="form-control">

<has-error :form="form" field="title"></has-error>

<br><br>

<textarea class="form-control" placeholder="Whats on your mind ?" :class="{'is-invalid': form.errors.has('content') }" style="height:200px;" name="content" v-model="form.content">

</textarea>

<has-error :form="form" field="content"></has-error>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

<button v-show="editmode" type="submit" class="btn btn-success">Update</button>

<button v-show="!editmode" type="submit" class="btn btn-primary">Post</button>

</div>

</form>

</div>

</div>

</div>

</div>

</template>

<script>

export default{

data(){

return{

posts:[],

search:'',

headers:

[

{

text: 'Title',

align: 'left',

sortable: true,

value: 'title'

},

{text:'Content', value:'content'},

{ text: 'Created At', value: 'created' },

{text:'Modify' , value:'modify'}

],

editmode: false,

form: new Form({

id: '',

content:'',

title:'',

image:'',

})

}

},

mounted(){

if(this.$gate.laundry()){

this.getPost();

this.createdPost();

window.Echo.channel('posts').listen('PostCreated',({post}) => {

this.posts.push(post)

console.log(posts)

});

}

},

methods:{

newModal(){

this.editmode = false;

this.form.reset();

$('#addNew').modal('show');

},

createPost(){

this.form.post('api/posts').then((response) => {

this.$Progress.start();

Fire.$emit('createdPost');

$('#addNew').modal('hide')

toast.fire({

type: 'success',

title: 'Post added into account successfully'

})

this.$Progress.finish();

})

},

deletePost(id){

swal({

title: 'Are you sure?',

text: "You won't be able to revert this!",

type: 'warning',

showCancelButton: true,

confirmButtonColor: '#3085d6',

cancelButtonColor: '#d33',

confirmButtonText: 'Yes, delete it!'

})

},

getPost(){

axios.get('api/posts').then(({data}) => this.posts = data);

},

createdPost(){

this.getPost();

Fire.$on('createdPost',()=>{

this.getPost();

})

}

}

};

</script>

<style type="text/css" scoped>

.text-xs-left .fa {

font-size: 20px;

}

</style>

Reservation.vue

<template>

<div class="container-fluid">

<div class="row pt-5" v-if="$gate.customer()">

<div class="col-md-12">

<div class="card">

<div class="card-header">

<h3 class="card-title">Transaction</h3>

<div class="card-tools">

<button class="btn btn-success" @click="newModal">Add Transaction <v-icon color="#fff">add\_box</v-icon></button>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-3 col-md-6" v-for="transact in transacts" :key="transact.id" >

<div class="card">

<div class="card-body">

<div class="row p-t-10 p-b-10">

<!-- Column -->

<div class="col p-r-0">

<h3 class="font-light">Php{{transact.total}}</h3>

<h6 class="text-muted">{{transact.service}}</h6>

<span class="label label-primary">{{transact.status.name}}</span>

</div>

<!-- Column -->

<div class="col text-right align-self-center">

<div data-label="100%" class="css-bar m-b-0 css-bar-danger css-bar-100"><i class="mdi mdi-briefcase-check"></i></div>

<br><br>

<button class="btn btn-danger" @click= "viewTransact(transact)">Full Details </button>

</div>

</div>

</div>

</div>

</div>

<div class="modal fade" id="addNew" tabindex="-1" role="dialog" aria-labelledby="addNewLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="addNewLabel">Add Transaction</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<form @submit.prevent="createTransaction()">

<div class="modal-body">

<input type="date" name="date" v-model="form.date" class="form-control" :class="{'is-invalid': form.errors.has('date') }">

<has-error :form="form" field="date"></has-error>

<br><br>

<select name="service" v-model="form.service" class="form-control" required :class="{'is-invalid': form.errors.has('service') }">

<option value="" disabled selected>------------ Choose service ---------------</option>

<option value="Fullservice">Full Service</option>

<option value="Finelinen">Fine Linen & Table Clothes</option>

<option value="Vintage">Bridal and Vintage Items</option>

<option value="Garment">Garment Repair & Alteration</option>

<option value="Carpet">Carpet & Upholster Cleaning</option>

</select>

<has-error :form="form" field="service"></has-error>

<br><br>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Submit</button>

</div>

</form>

</div>

</div>

</div>

<div class="modal fade" id="viewdetails" tabindex="-1" role="dialog" aria-labelledby="addNewLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="addNewLabel">View Transaction</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<div>

<h5 style="display:inline;"><b>Date Reservation :</b> </h5><span style="display:inline;color:blue;"> <b>{{form.date |myDate}} </b></span>

</div>

<div>

<h5 style="display:inline;"><b>Weight :</b> </h5><span style="display:inline;color:green;"><b> {{form.weight}}kg </b></span>

</div>

<div>

<h5 style="display:inline;"><b>Service :</b> </h5><span style="display:inline;color:indigo;"><b> {{form.service}}</b></span>

</div>

<br><br>

<div class="progress">

<progressbar :now="form.percent" label type="success" striped animated></progressbar>

</div>

<br>

<h3 style="display:inline;"> <b>Status: </b></h3> <span style="display:inline;color:red;">{{form.status}}</span>

<img src="/images/tenor.gif" alt="delivery" v-if="form.percent >= 100" style="width:100%;">

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

</div>

</div>

</div>

</div>

</div>

</template>

<script>

import { progressbar } from 'vue-strap'

export default{

components: {

progressbar

},

props: ['status', 'initial', 'transaction\_id'],

data(){

return{

customers:[],

transacts:'',

statusNew : this.status,

progress: this.initial,

editmode: false,

form: new Form({

id: '',

service:'',

status:'',

percent:'',

date:'',

weight:''

})

}

},

mounted(){

if(this.$gate.customer()){

this.getTransaction();

/\* window.Echo.private('transaction-tracker.' + this.transaction\_id)

.listen('StatusChanged', (order) => {

this.statusNew = order.status\_name

this.progress = order.status\_percent

});\*/

}

},

methods:{

newModal(){

this.editmode = false;

this.form.reset();

$('#addNew').modal('show');

},

viewTransact(transact){

this.form.service = transact.service

this.form.status = transact.status.name;

this.form.percent = transact.status.percent

this.form.date = transact.date\_reserve

this.form.weight = transact.weight

$('#viewdetails').modal('show')

},

getTransaction(){

axios.get('api/mytransaction').then(({data}) => this.transacts = data)

},

createTransaction(){

this.form.post('api/transactions').then((response) => {

this.$Progress.start();

$('#addNew').modal('hide')

toast.fire({

type: 'success',

title: 'Partial Reservation added into shop successfully'

})

this.$Progress.finish();

})

}

}

};

</script>

Tesr.vue

<template>

<h1> Test </h1>

</template>

<script>

export default{

};

</script>