CSE470 : Software Engineering
Project Report
Project Title : Shrom

Group No : 9 , Summer 2023	
ID	**Name**
19101588	Rakesh Rakshit
19201043	Ayan Roy
19101299	Md.Rubaiat Ferdous
20101492	Shad Been Sharif Sami

Table of Contents

Introduction

The Shrom platform aims to create a user-friendly digital marketplace, linking service providers and customers seamlessly. This platform simplifies the process of offering, managing, and accessing services, with features like dynamic pricing and transaction oversight. The customers also have access to features such as a digital payment system to reduce any hassle of using the site and make the method of purchasing services seamless. This document outlines the essential requirements guiding the development and implementation of the user-centered Shrom platform.

Functional Requirements

1. Admin

- Admins can view/edit all categories
- Admins can view/edit job posts
- Admins can disable and enable Workers/Customers
- Admins can disable and enable any services
- Admins can set discounts on services
- Admins can check ongoing purchases
- Admins can edit the front page to their liking

2. Workers

- Workers can view/edit their categories
- Workers can view/edit their job posts
- Workers can message admins through WhatsApp
- Workers can set discounts on services
- Workers can upload pictures for their Jobposts

3. Customers

- Customers can view all Jobposts
- Customers can search by categories
- Customers can browse Jobposts
- Customers can subscribe for notifications
- Customers can view their profile

4. Payment System

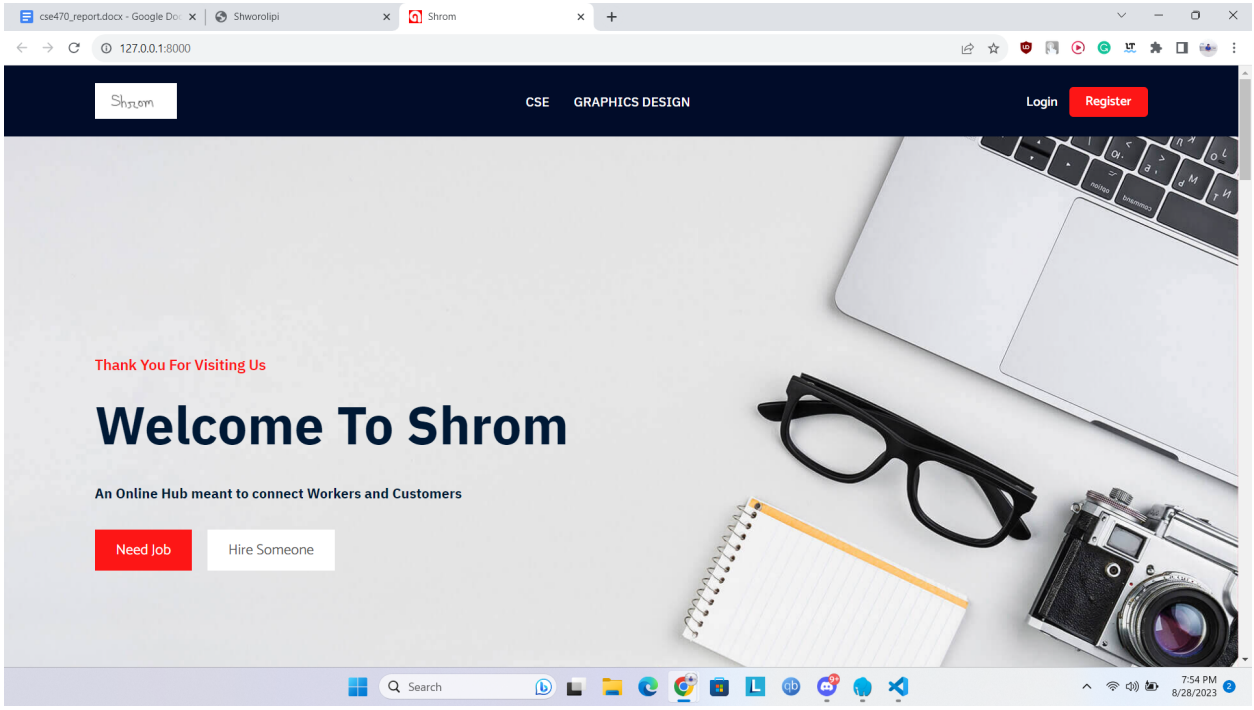
- Customers can add/remove Jobs to their cart
- Customers can view the total amount of their purchase
- Customers can use an online banking system or Bkash/Nagad

User Manual

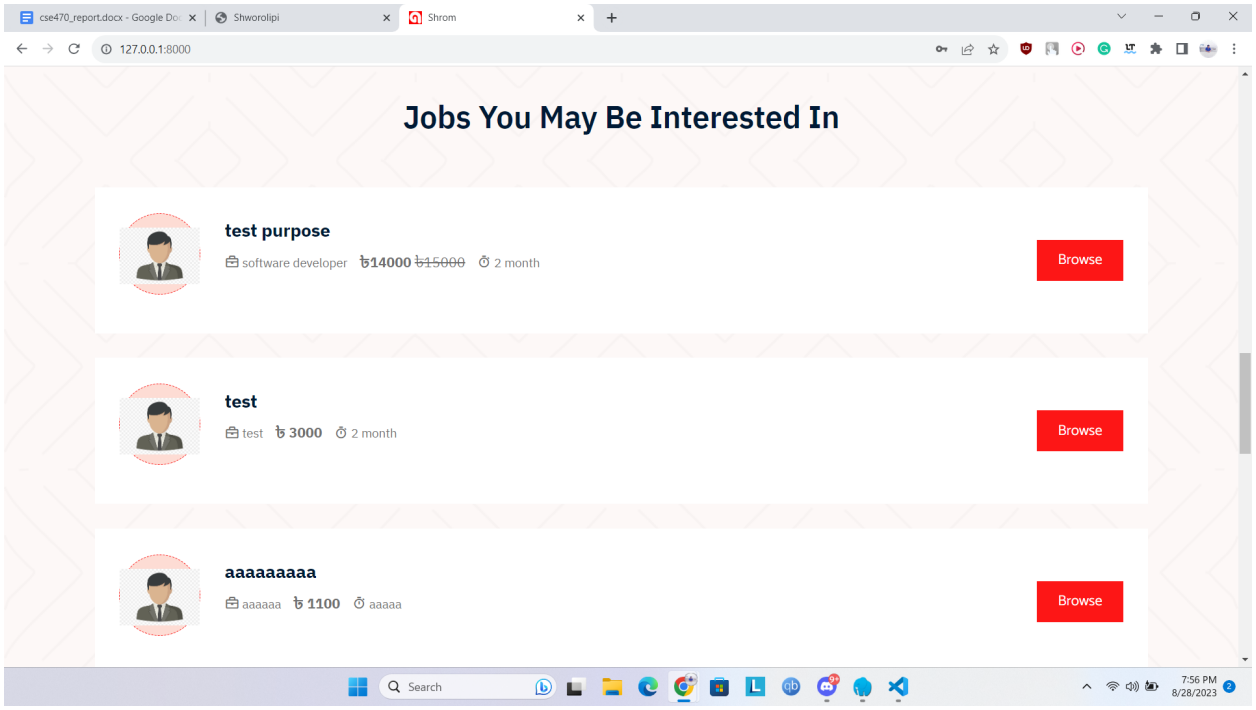
1. User Side

1.1 Homepage

- On Homepage, User can Login as User by clicking on “Login” Button and register as User by clicking on “Register” Button

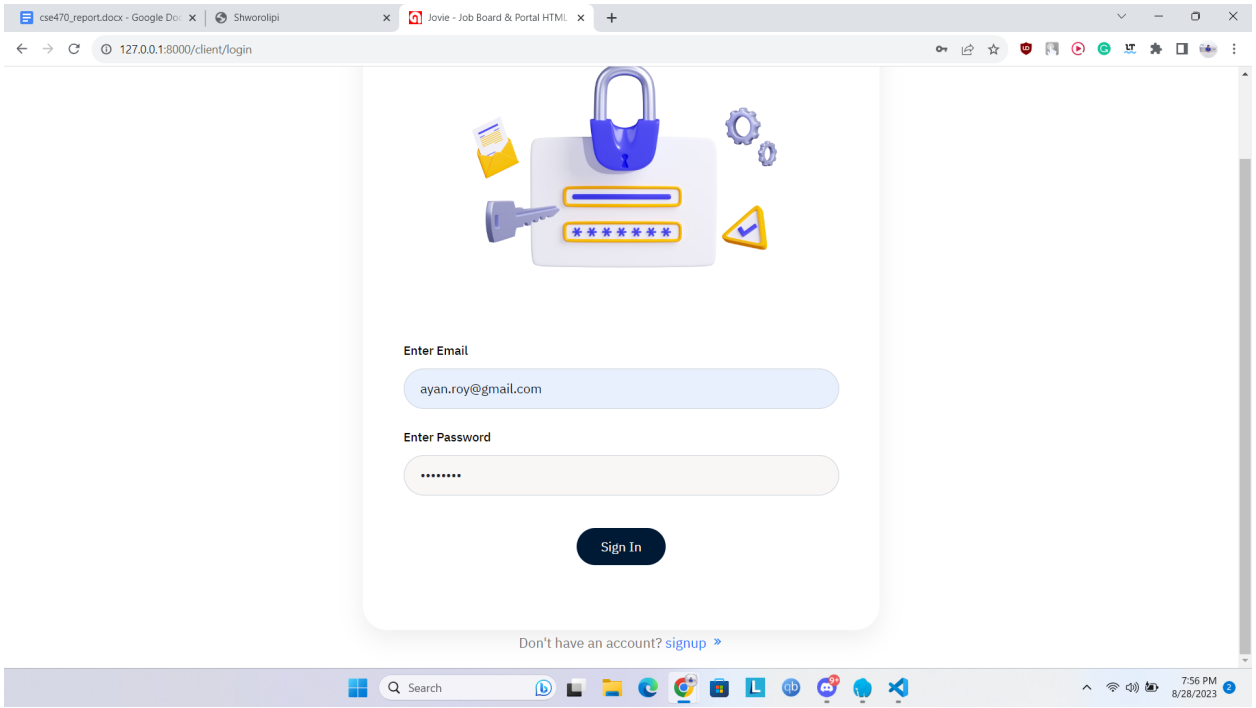


- User Can Also Browse the Jobs without Login

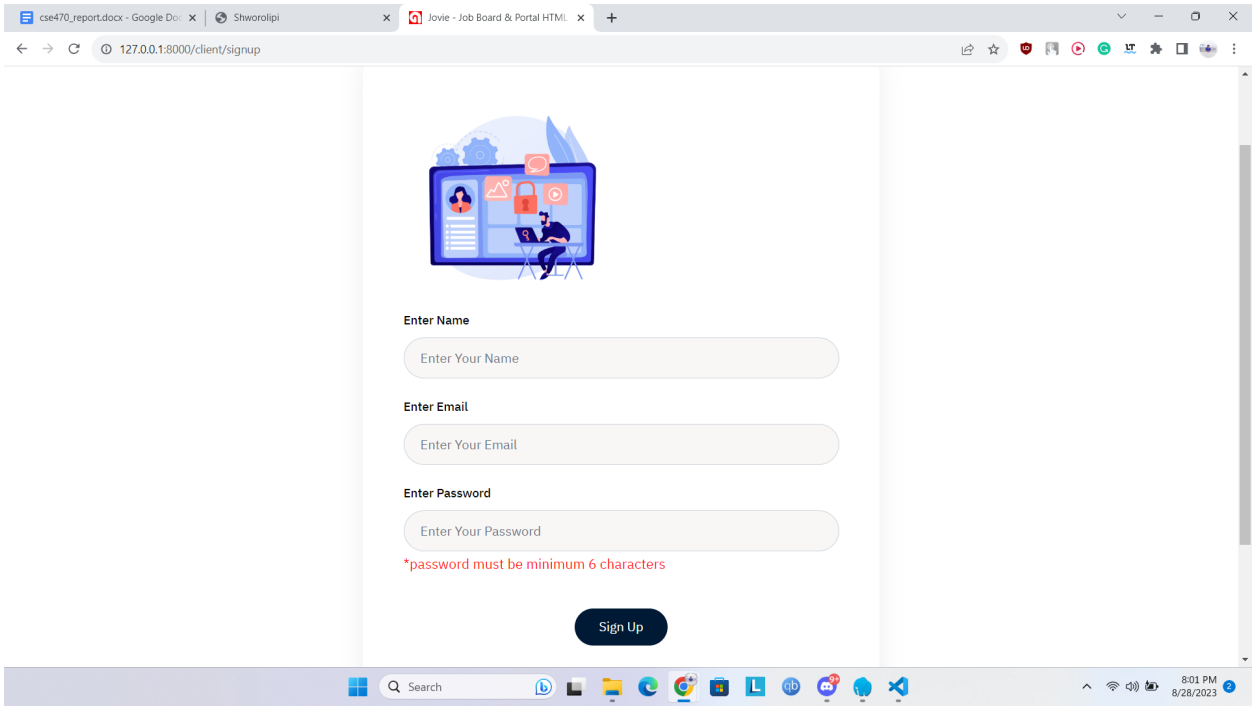


1.2 Login Page

- User can fillup the form to Login

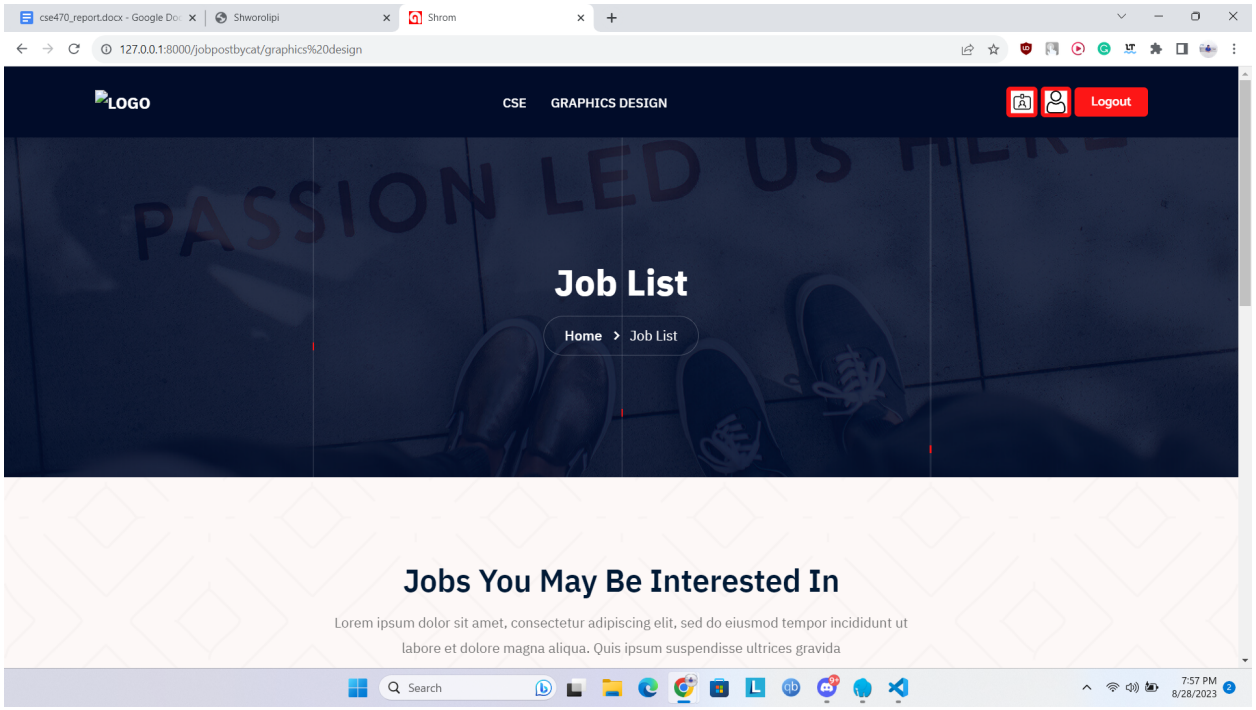


- User can Fillup the form to register



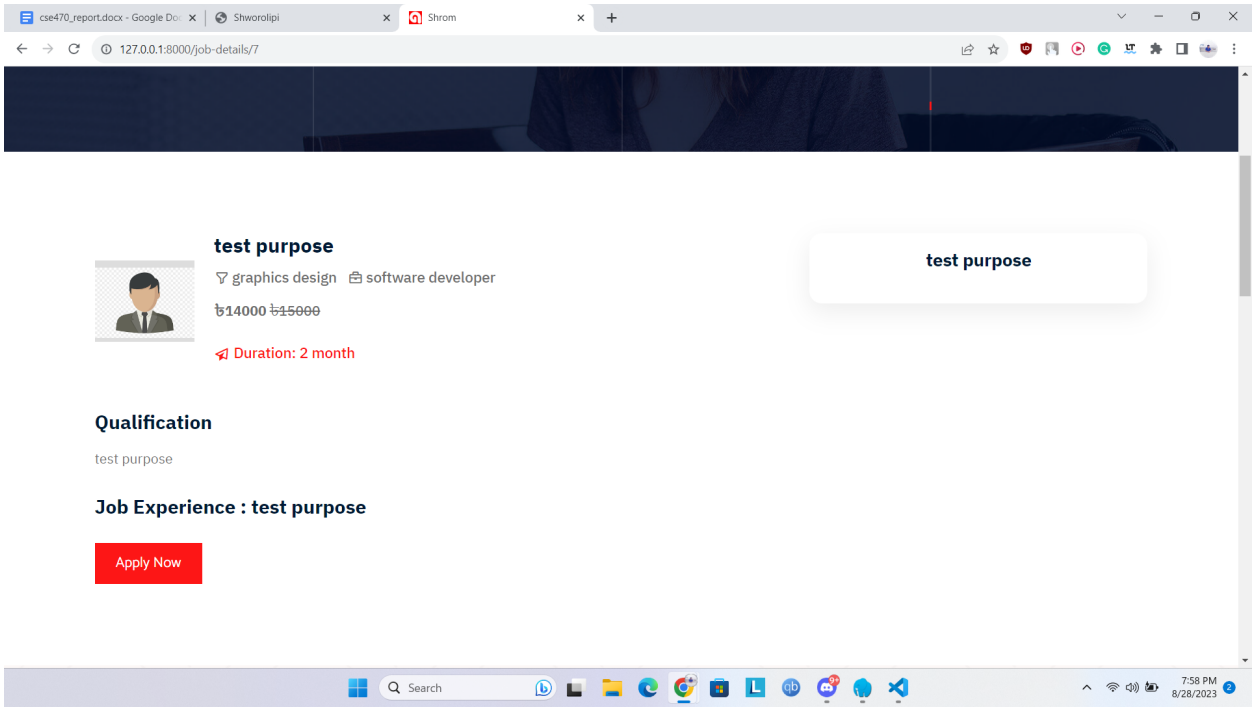
1.3 Job List

- User can View all the Jobs from this page



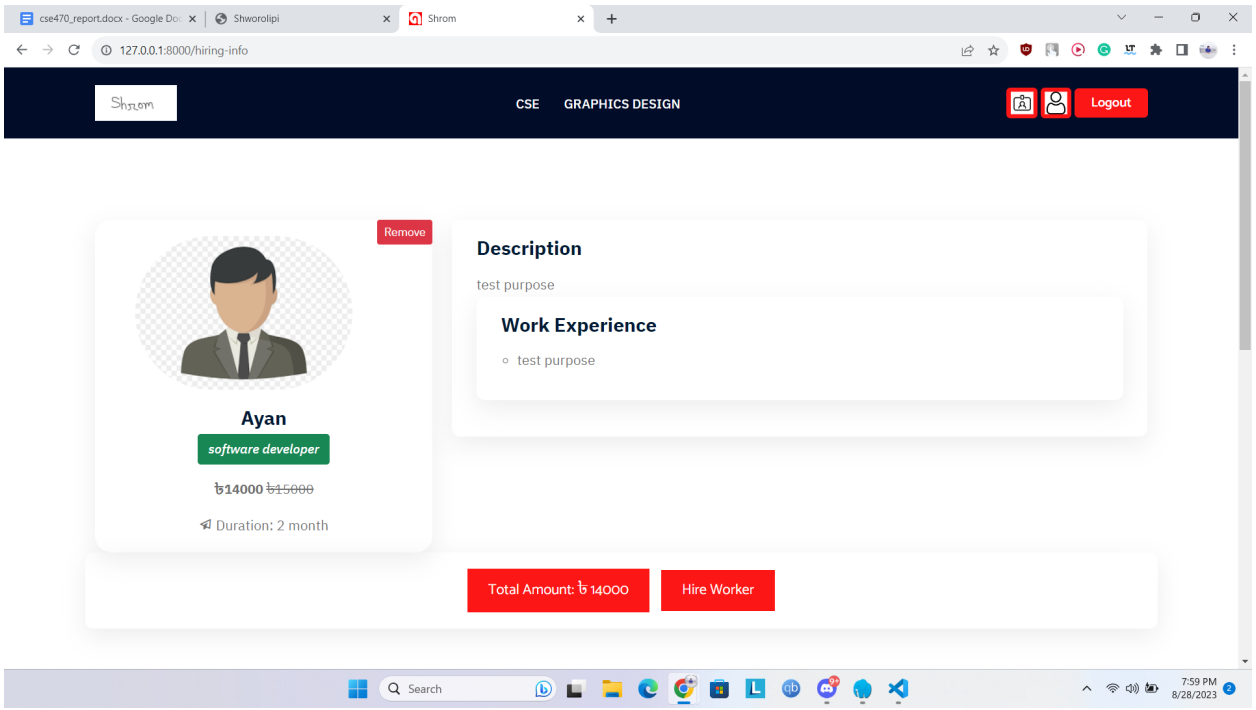
1.4 Browse Job

- User can see the Job details and Click “Apply” to add Job to Cart



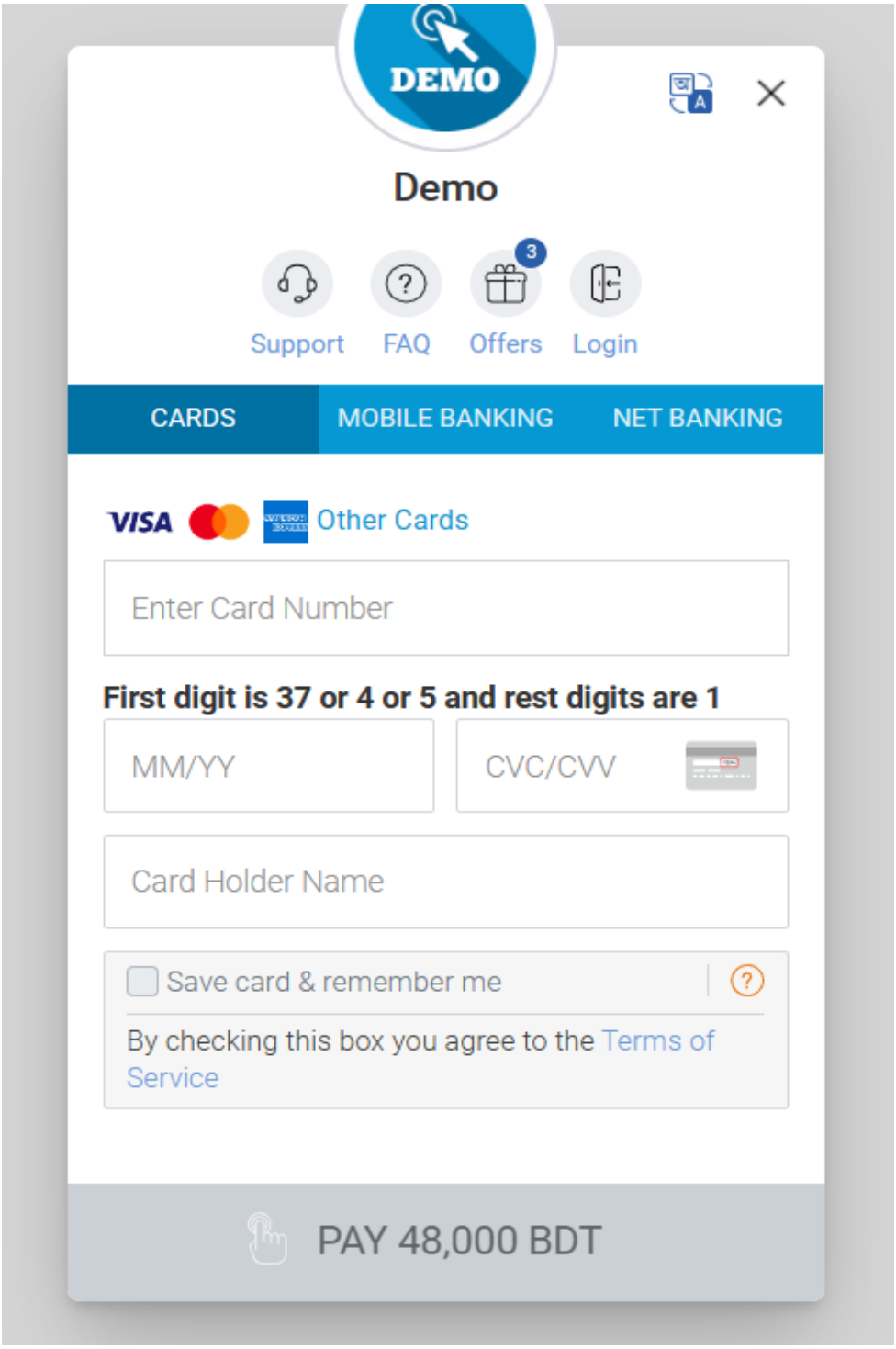
1.5 Cart

- User can click “Hire Worker” to purchase the services and See the total amount in Cart



1.6 Payment System

- Users can Press to “hire work” they will be taken to a payment system implemented by /sandbox.sslcommerz.com

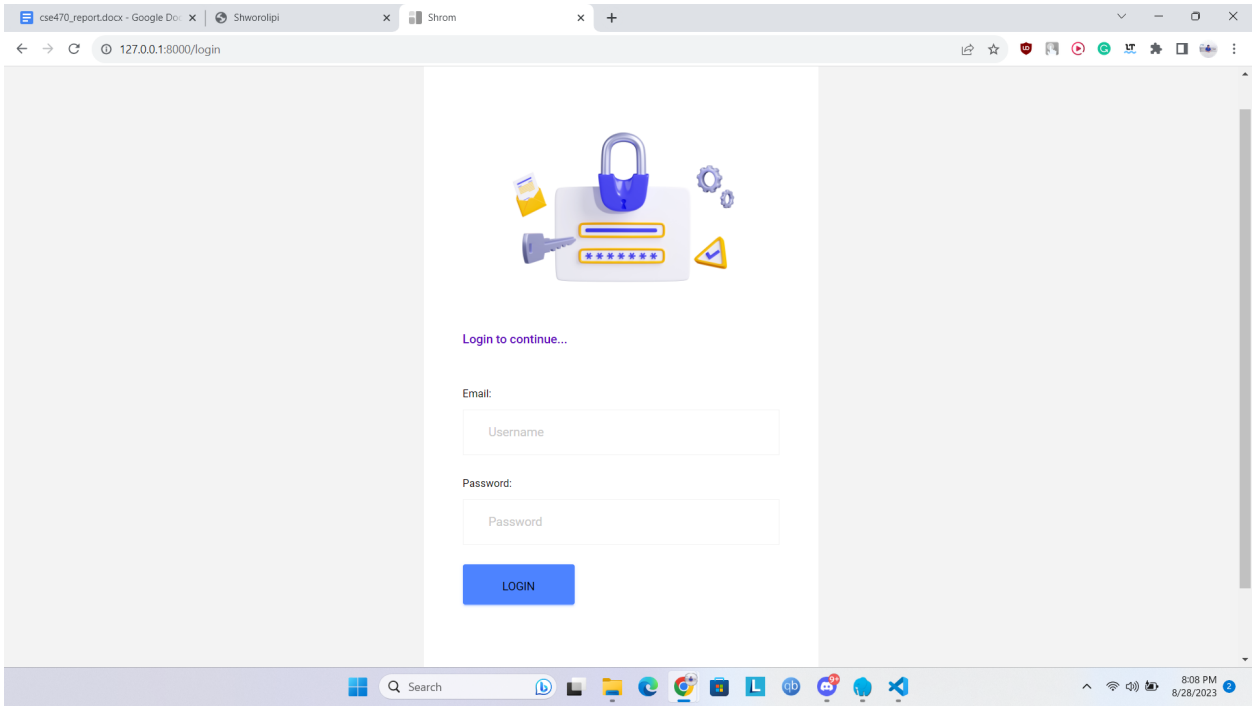


- There will be options for various mobile bankings

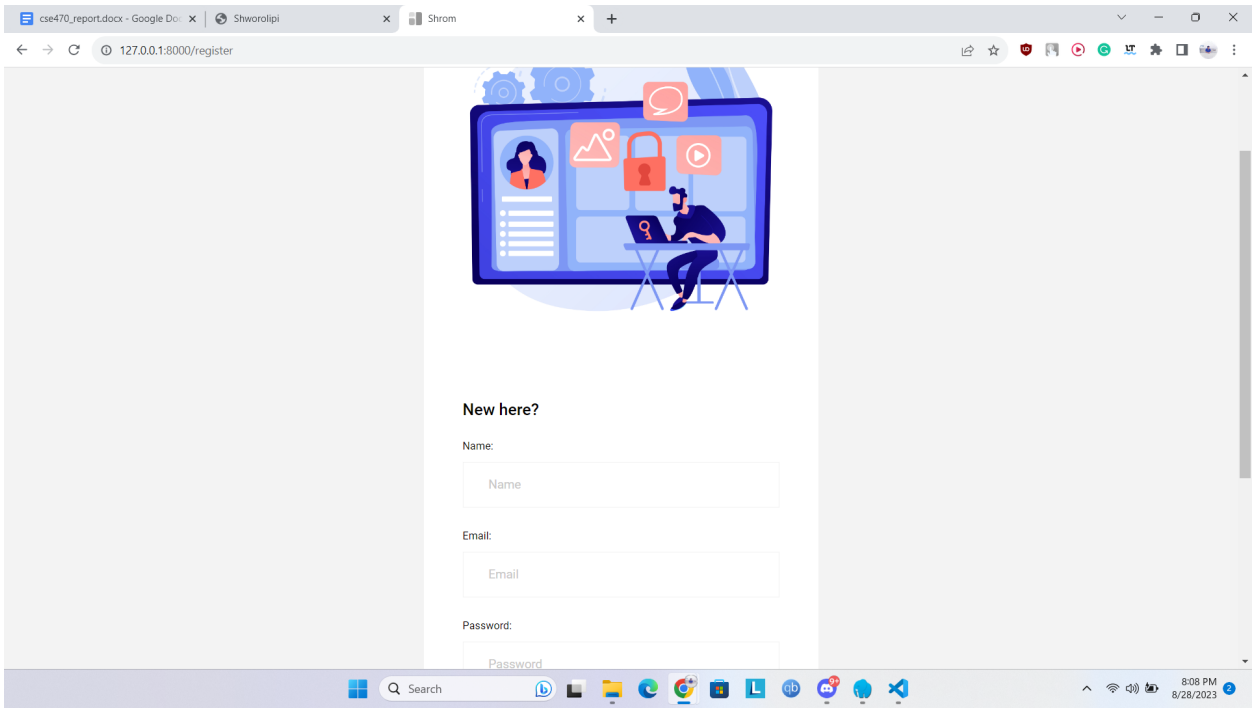
2. Worker Side

2.1 Login

- Worker can fillup the form to Login

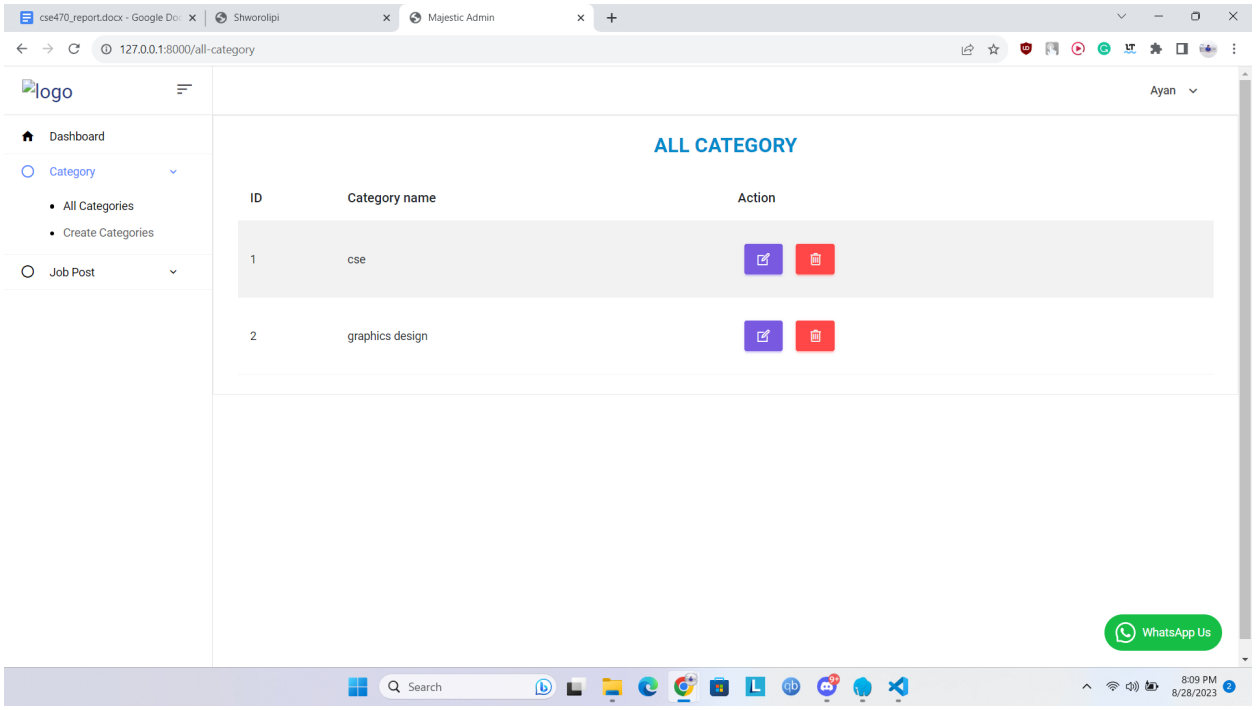


- Worker can fillup the form to Register

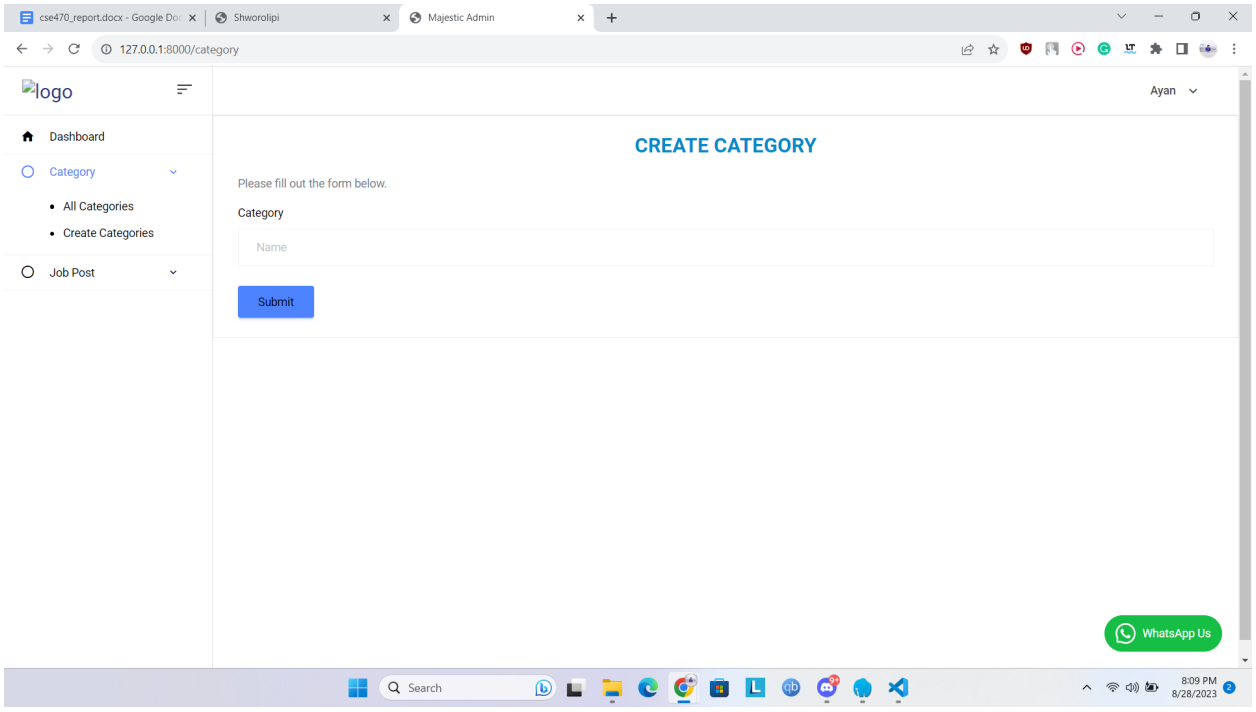


2.2 Category

- Worker can view all categories and edit them

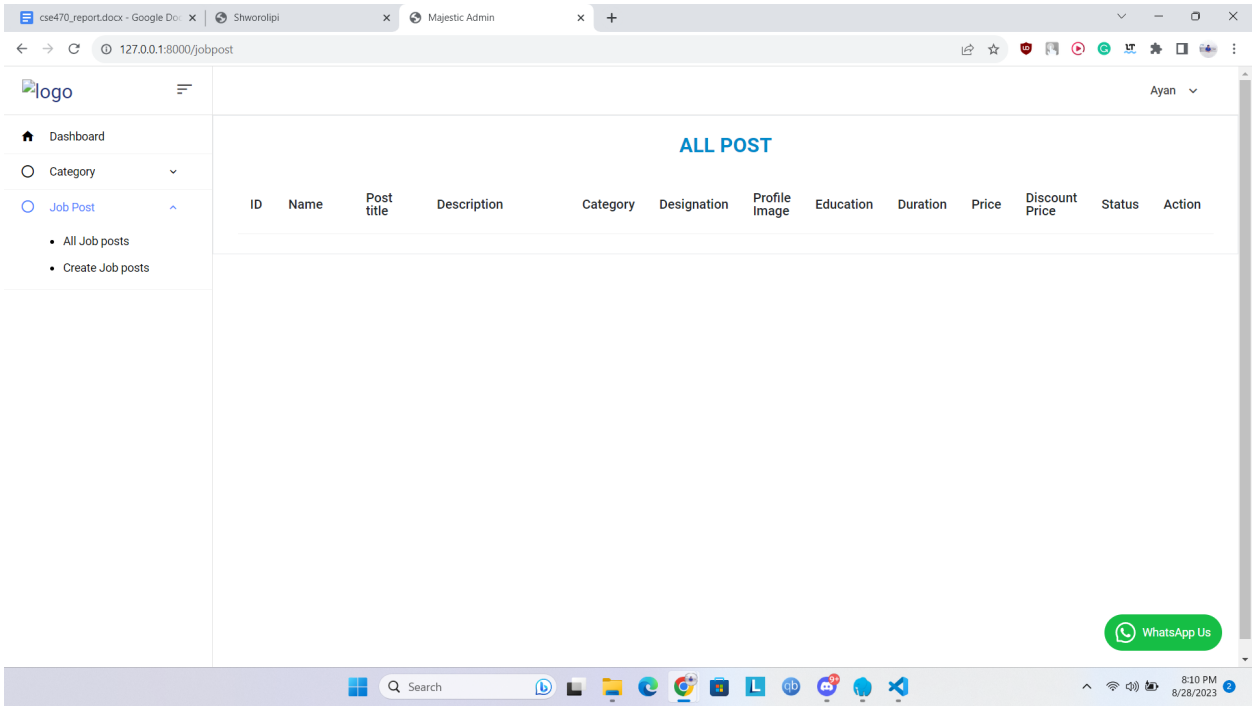


- Worker can add categories

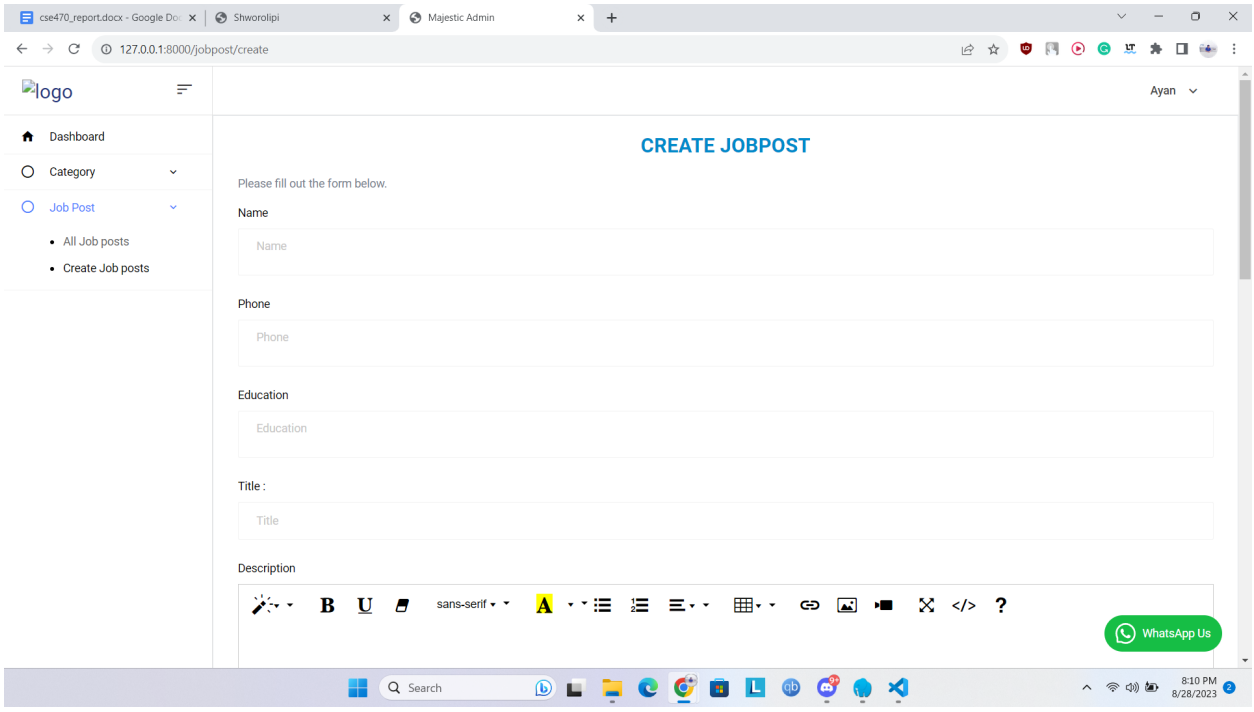


2.3 JobPost

- Worker can see their own Jobposts and edit them



- Worker can create Jobposts

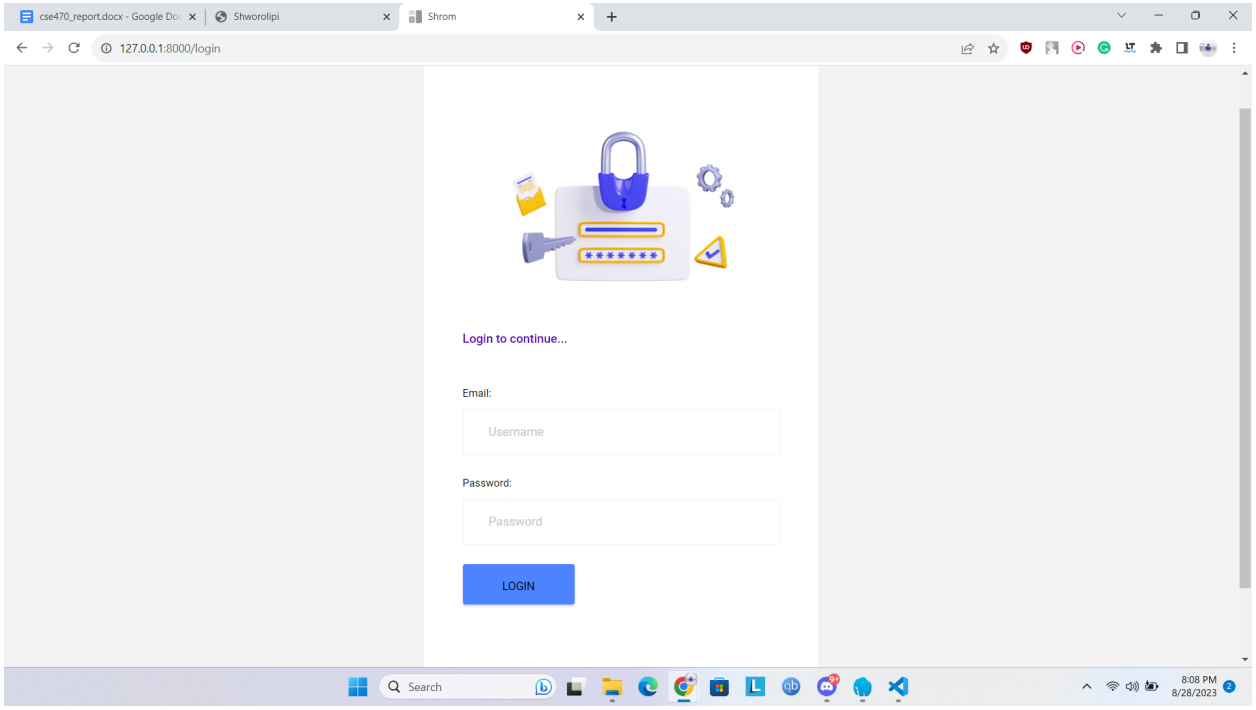


-

3. Admin Side

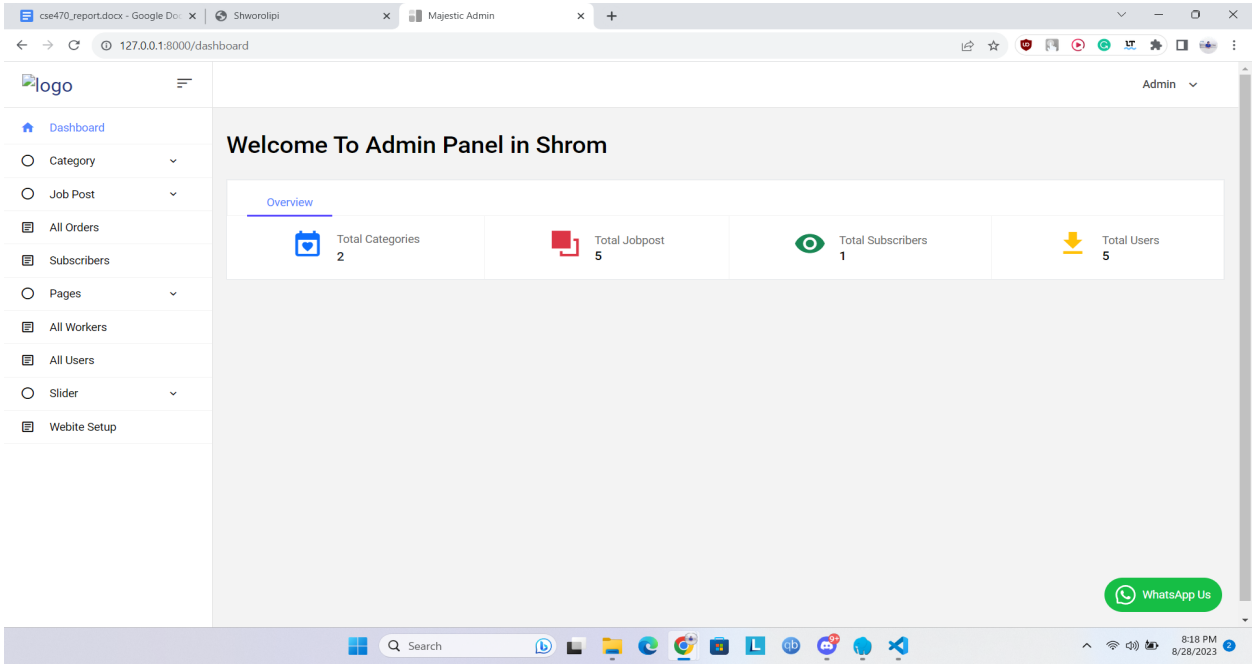
3.1 Login

- Admin can Login into the Admin Panel



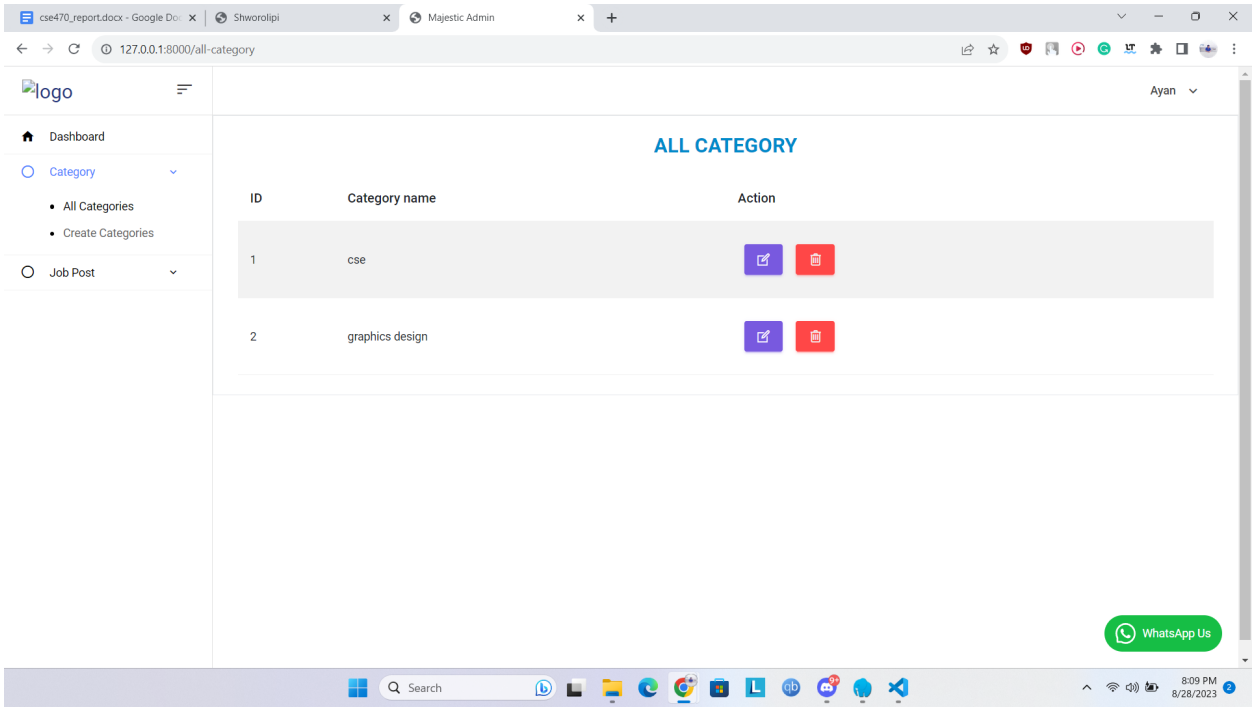
3.2 Dashboard

- Admin can View the DashBoard

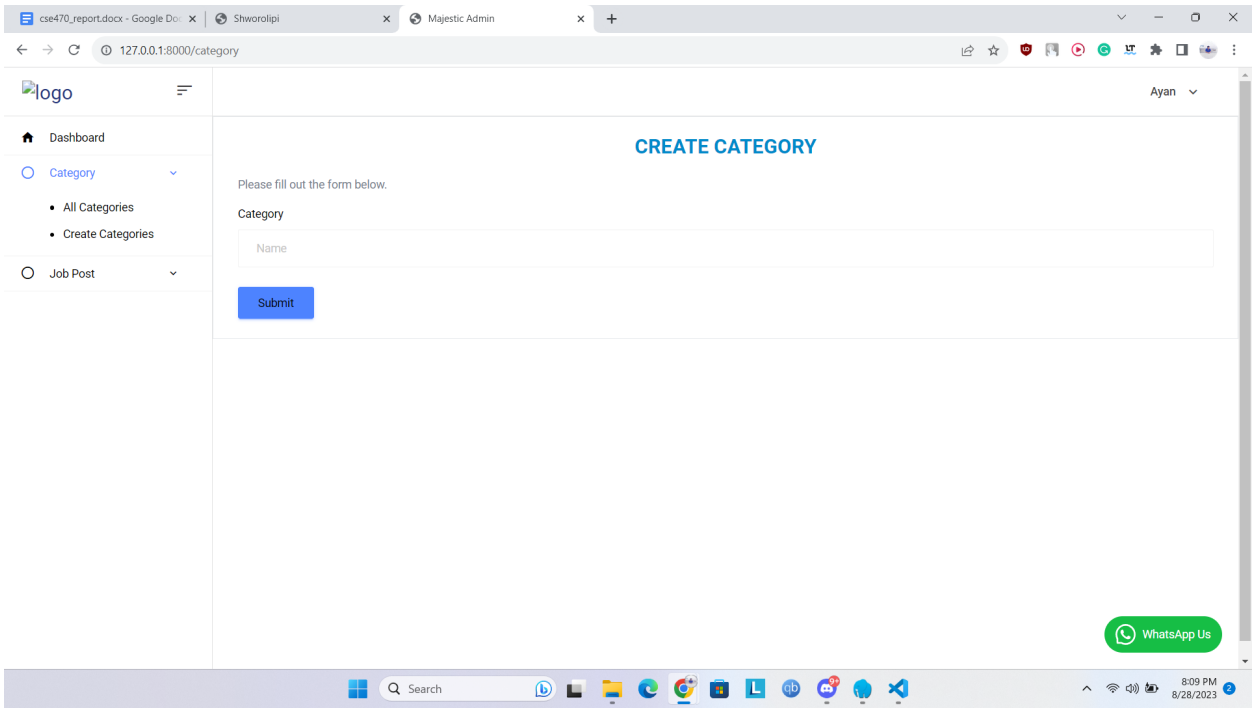


3.3 Category

- Admin can view categories and also edit them

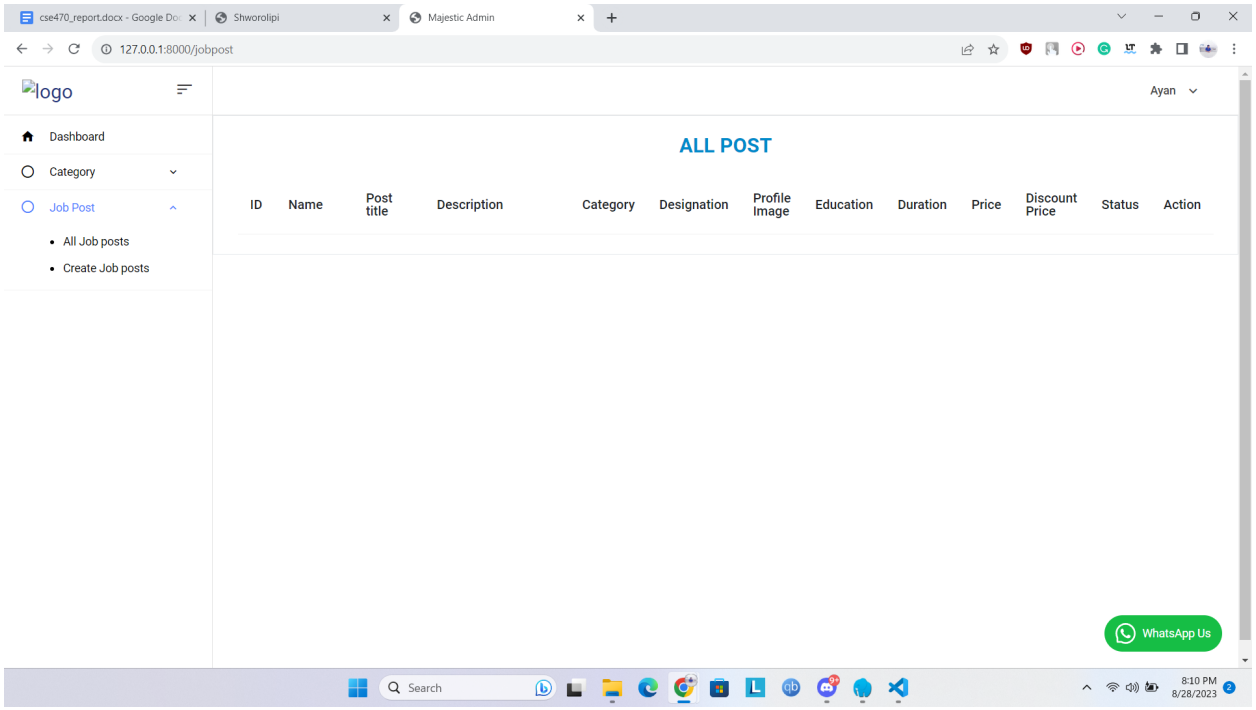


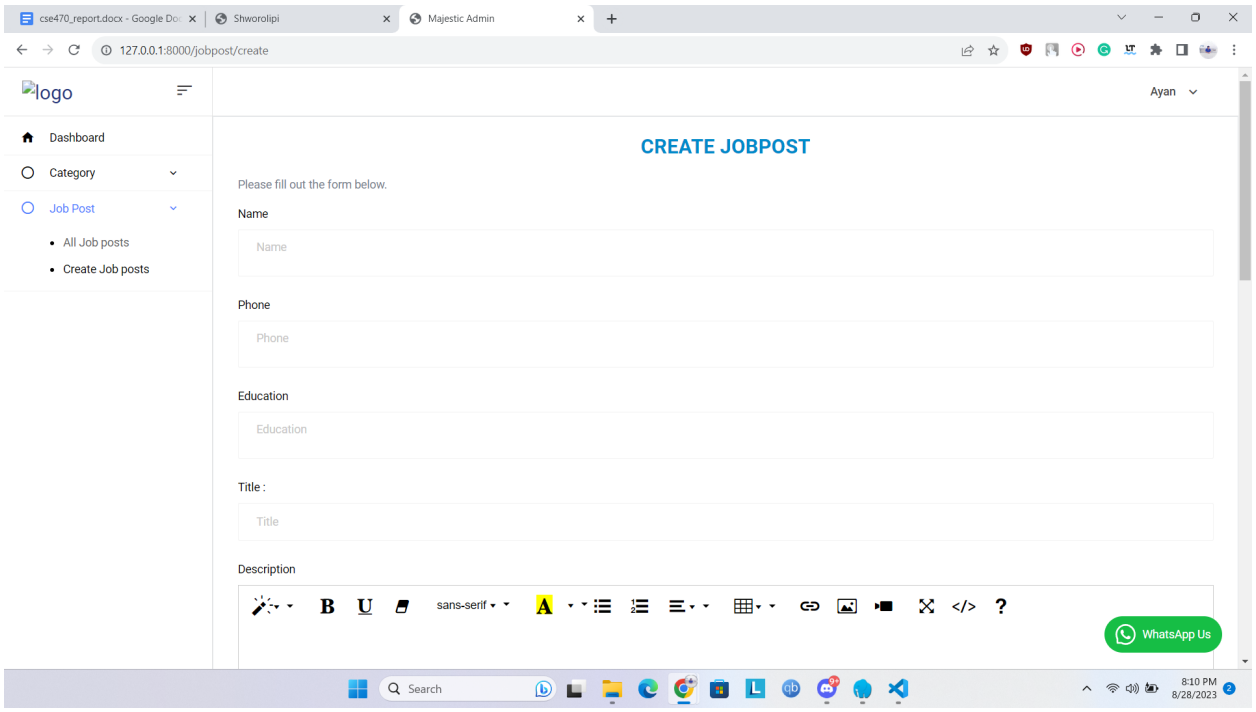
- Admin can Create categories to their liking



3.4 JobPost

- Admin can see all Jobposts And also create them

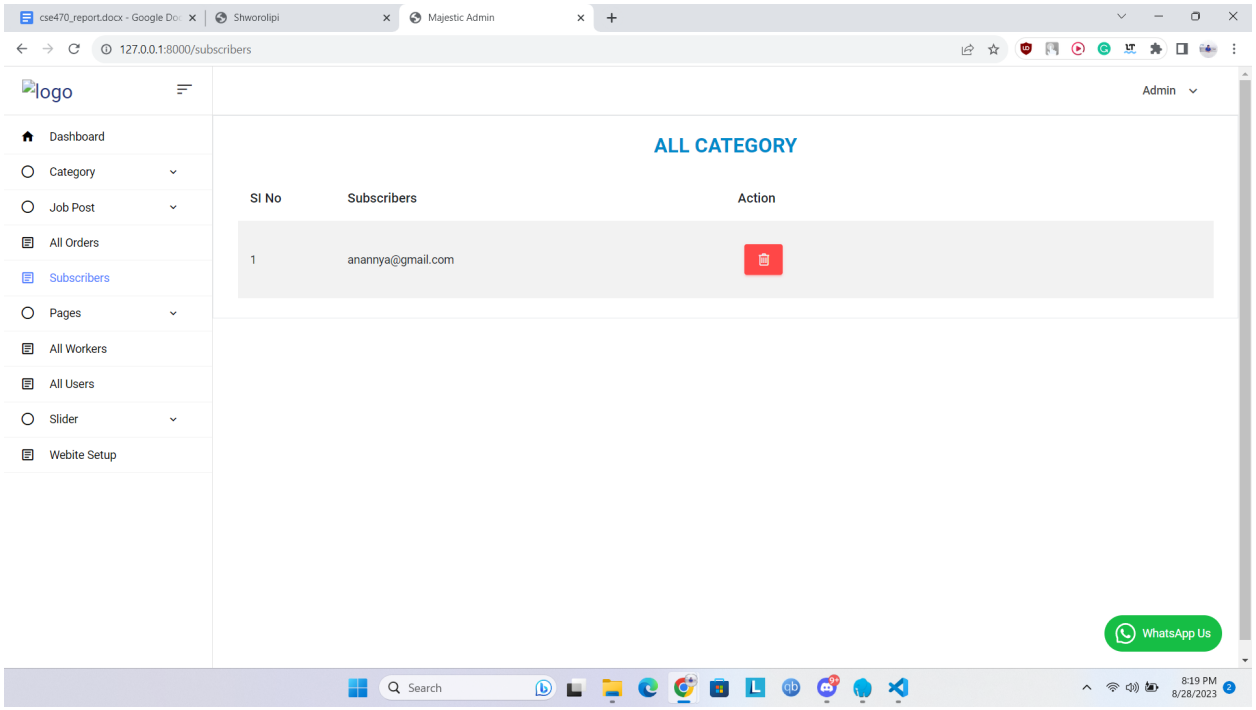




3.5 All Orders

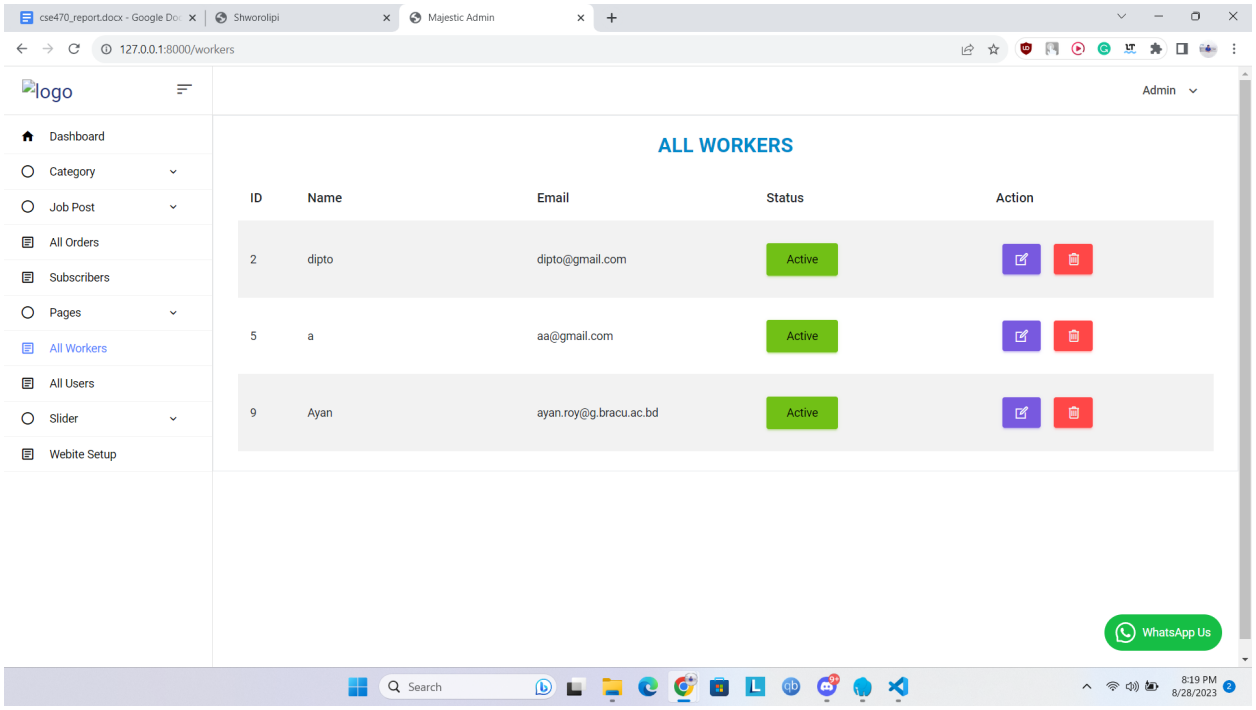
3.6 Subscribers

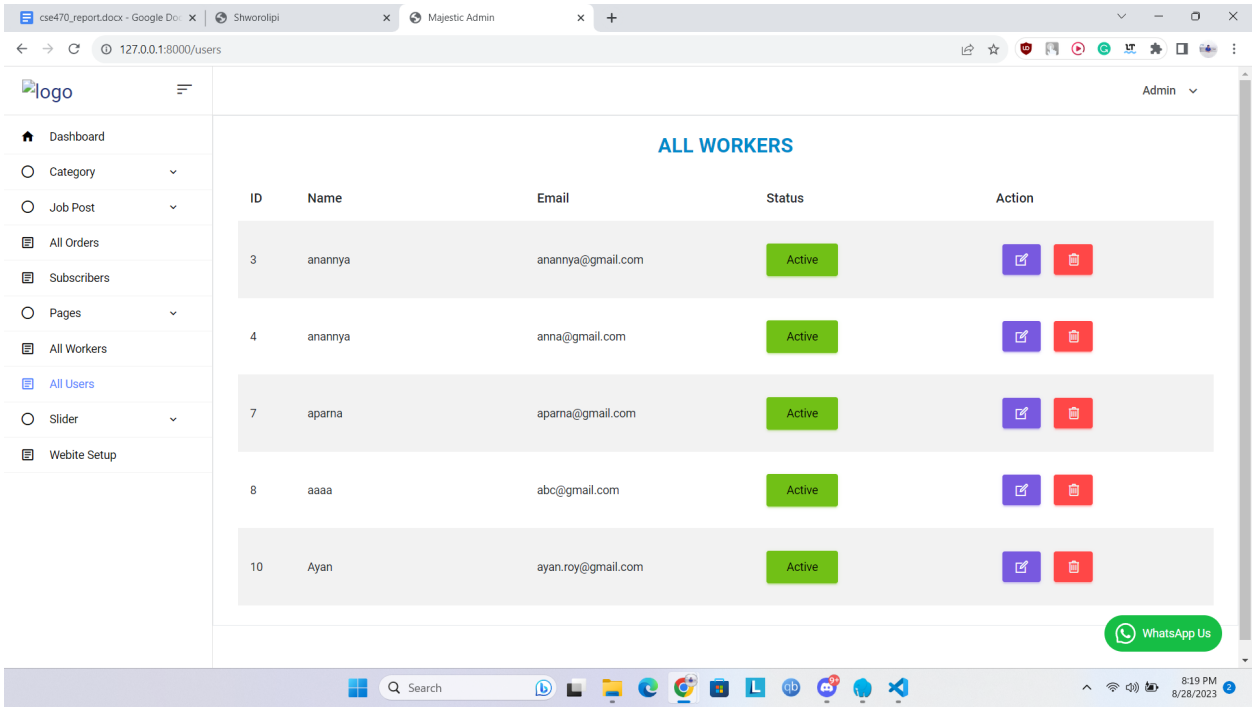
- Admin can view all Subscribers



3.7 All Workers and Users

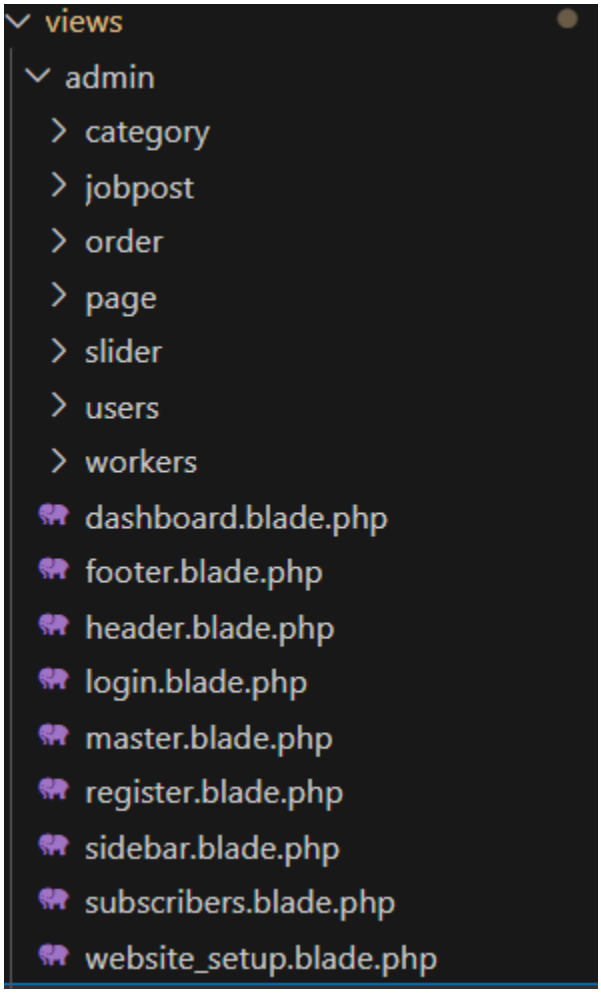
- Admin can view all Workers and Users and choose to disable or enable their account

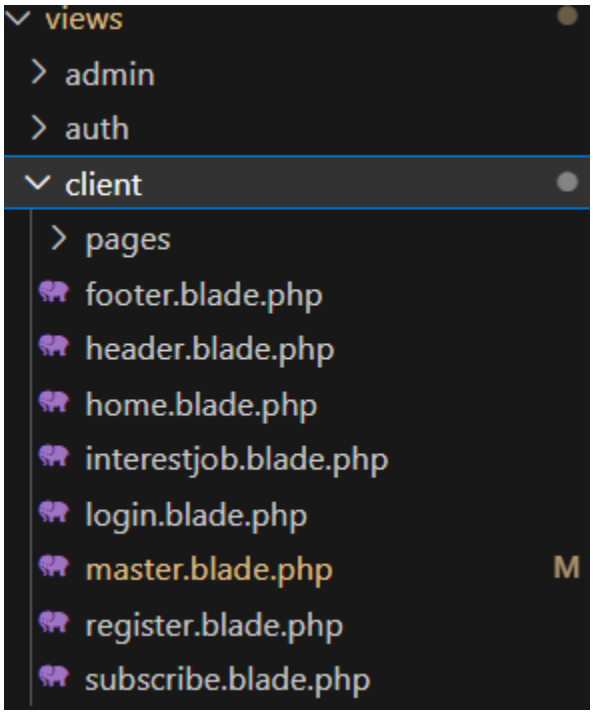

cse470_report.docx - Google Do
Shworolipi
Majestic Admin
127.0.0.1:8000/workers
logo
Admin
Dashboard
Category
Job Post
All Orders
Subscribers
Pages
All Workers
All Users
Slider
Webite Setup
ALL WORKERS
ID
Name
Email
Status
Action
2
dipto
dipto@gmail.com
Active
5
a
aa@gmail.com
Active
9
Ayan
ayan.roy@g.bracu.ac.bd
Active
WhatsApp Us
Search
8:19 PM
8/28/2023


cse470_report.docx - Google Do
Shworolipi
Majestic Admin
127.0.0.1:8000/users
logo
Admin
Dashboard
Category
Job Post
All Orders
Subscribers
Pages
All Workers
All Users
Slider
Webite Setup
ALL WORKERS
ID
Name
Email
Status
Action
3
anannya
anannya@gmail.com
Active
4
anannya
anna@gmail.com
Active
7
aparna
aparna@gmail.com
Active
8
aaaa
abc@gmail.com
Active
10
Ayan
ayan.roy@gmail.com
Active
WhatsApp Us
Search
8:19 PM
8/28/2023

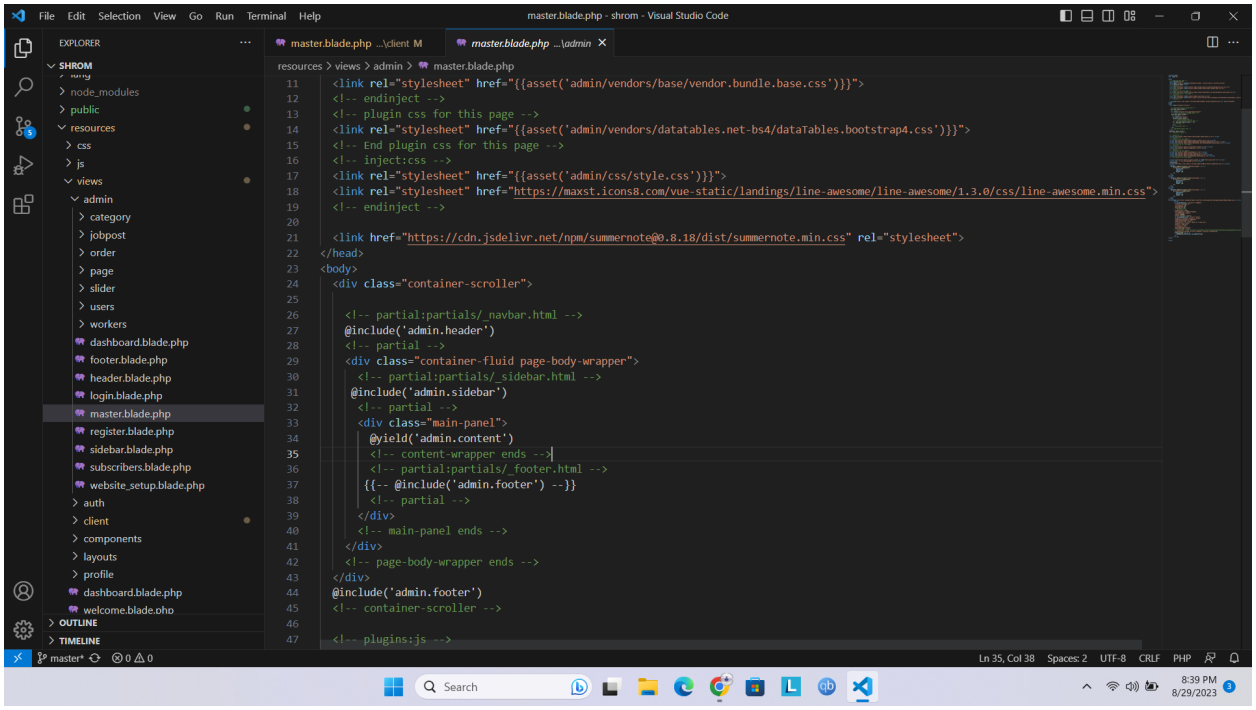
Frontend Development

In the directory named "/resources/views", we have stored the code for the template that we have mastered.





In the screenshots below, we will see that we have used Yield option in laravel in both admin and main template so that it can be used to define the section in a particular layout and is consistently used to retrieve content from the child page to a master page. So if the Laravel executes a blade file, it just verifies if the user has the extended layout and then pushes the master layout which commences from getting the @section


File Edit Selection View Go Run Terminal Help
master.blade.php - shrom - Visual Studio Code
EXPLORER
SHROM
master.blade.php ...\client M
master.blade.php ...\admin
resources > views > admin > master.blade.php
<link rel="stylesheet" href="{{asset('admin/vendors/base/vendor.bundle.base.css')}}">
<!-- endinject -->
<!-- plugin css for this page -->
<link rel="stylesheet" href="{{asset('admin/vendors/datatables.net-bs4/dataTables.bootstrap4.css')}}">
<!-- End plugin css for this page -->
<!-- inject:css -->
<link rel="stylesheet" href="{{asset('admin/css/style.css')}}">
<link rel="stylesheet" href="https://maxst.icons8.com/vue-static/landings/line-awesome/line-awesome/1.3.0/css/line-awesome.min.css">
<!-- endinject -->
<link href="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.css" rel="stylesheet">
</head>
<body>

<!-- partial:partials/_navbar.html -->
@include('admin.header')
<!-- partial -->

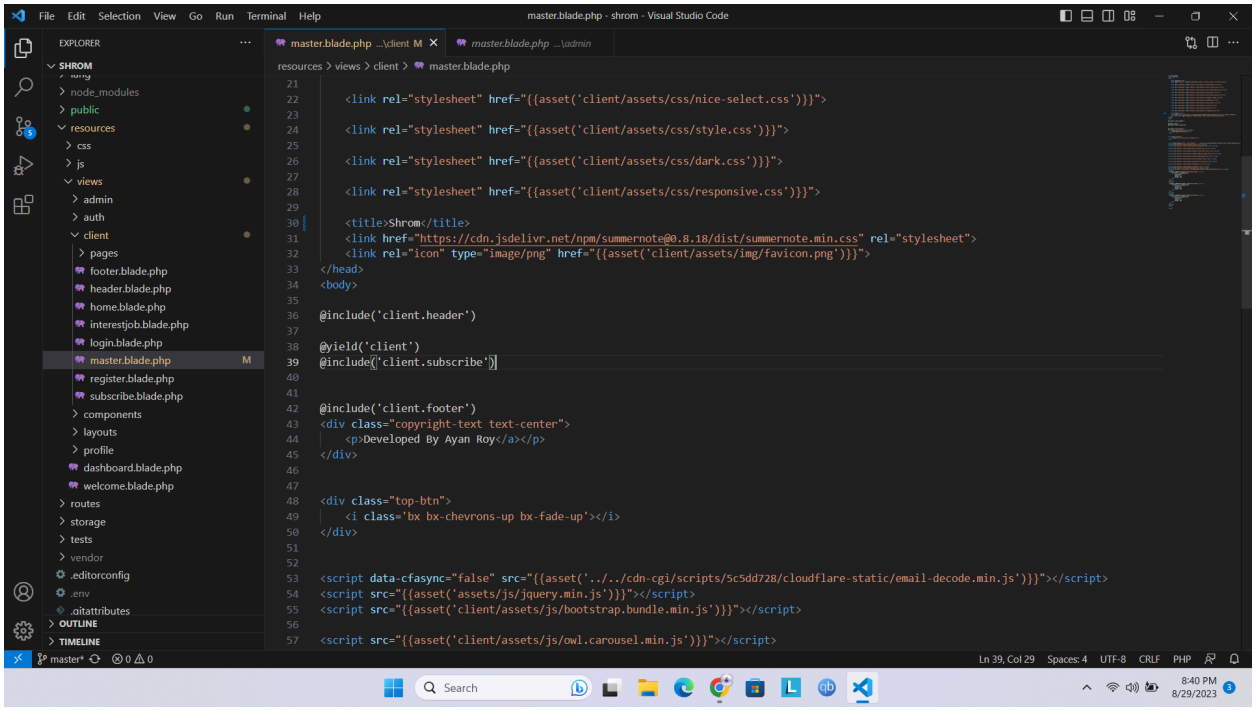
<!-- partial:partials/_sidebar.html -->
@include('admin.sidebar')
<!-- partial -->

@yield('admin.content')
<!-- content-wrapper ends -->
<!-- partial:partials/_footer.html -->
{{-- @include('admin.footer') --}}
<!-- partial -->

<!-- main-panel ends -->

<!-- page-body-wrapper ends -->

@include('admin.footer')
<!-- container-scroller -->
<!-- plugins:js -->
Ln 35, Col 38 Spaces: 2 UTF-8 CRLF PHP
8:39 PM
8/29/2023


File Edit Selection View Go Run Terminal Help
master.blade.php - shrom - Visual Studio Code
EXPLORER
SHROM
master.blade.php ...\client M
master.blade.php ...\admin
resources > views > client > master.blade.php
<link rel="stylesheet" href="{{asset('client/assets/css/nice-select.css')}}">
<link rel="stylesheet" href="{{asset('client/assets/css/style.css')}}">
<link rel="stylesheet" href="{{asset('client/assets/css/dark.css')}}">
<link rel="stylesheet" href="{{asset('client/assets/css/responsive.css')}}">
<title>Shrom</title>
<link href="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.css" rel="stylesheet">
<link rel="icon" type="image/png" href="{{asset('client/assets/img/favicon.png')}}">
</head>
<body>
@include('client.header')
@yield('client')
@include('client.subscribe')
@include('client.footer')

Developed By Ayan Roy

<script data-cfasync="false" src="{{asset('../../cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js')}}"></script>
<script src="{{asset('assets/js/jquery.min.js')}}"></script>
<script src="{{asset('client/assets/js/bootstrap.bundle.min.js')}}"></script>
<script src="{{asset('client/assets/js/owl.carousel.min.js')}}"></script>
Ln 39, Col 29 Spaces: 4 UTF-8 CRLF PHP
8:40 PM
8/29/2023

Backend Development

1. **Admin can create/remove services:**
   - admin can edit workers

```php
class DashboardController extends Controller
{
    public function dashboard()
    {
        $user= Auth::user();
        $totalcategory = DB::table('categories')->count();
        $totaljobPost = DB::table('jobposts')->count();
        $totalusers = DB::table('users')->where('role','=','client')->count();
        $totalubscribers = DB::table('subscribes')->count();

        return view('admin.dashboard', compact('user','totalcategory','totaljobPost','totalusers','totalubscribers'));
    }

    public function logout()
    {
        Session::flush();
        return redirect('/login');
    }

    public function workers()
    {
        $workers = User::where('role', '=', 'worker')->get();
//        dd($workers);
        return view('admin.workers.index', compact('workers'));
    }
    public function editworkers($id){
        $workers = User::find($id);
        return view('admin.workers.edit',compact('workers'));
    }
```

- **Admin can destroy workers , edit users , update users , destroy users , destroy subscribers**

```php
public function destroyworkers($id){
    $workers = User::find($id);
    $workers->delete();

    return redirect()->back();
}
public function users()
{
    $users = User::where('role', '=', 'client')->get();
    return view('admin.users.index', compact('users'));
}
public function editusers($id){
    $users = User::find($id);
    return view('admin.users.edit',compact('users'));
}
public function updateusers(Request $request, $id){

    $users= User::where('id',$id)->update([
        'status' => $request->status,
    ]);
    return redirect()->back();
}
public function destroyusers($id){
    $users = User::find($id);
    $users->delete();

    return redirect()->back();
}
public function subscribers(){
    $user= Auth::user();
    $subscribers = DB::table('subscribes')->get();
    return view('admin.subscribers',compact('subscribers', 'user'));
}
public function destroysubscribers($id){
    $subscribers = Subscribe::find($id);
    $subscribers->delete();

    return redirect()->back();
}
public function orders(){
    //  $orders= Order::with(['orderdetails.jobPost','orderdetails.user'])->get();
   $orders = OrderDetails::with(['jobPost', 'user', 'order'])->get();
       dd($orders->orderdetails);
    return view('admin.order.index',compact('orders'));
```

- **Admin can synchronize , update and cancel orders**

```
    public function subscribers(){
        $user= Auth::user();
        $subscribers = DB::table('subscribes')->get();
        return view('admin.subscribers',compact('subscribers', 'user'));
    }
    public function destroysubscribers($id){
        $subscribers = Subscribe::find($id);
        $subscribers->delete();

        return redirect()->back();
    }
    public function orders(){
        //  $orders= Order::with(['orderdetails.jobPost','orderdetails.user'])->get();
       $orders = OrderDetails::with(['jobPost', 'user', 'order'])->get();
//         dd($orders->orderdetails);
        return view('admin.order.index',compact('orders'));
    }

    public function sync($id)
    {
        $order = Order::find($id);
        $order-> status ='Synced';
        $order->save();

        return redirect()->back();
    }

    public function update($id)
    {

        $order = Order::find($id);
        $order-> status  = 'Delivered';
        $order->save();

        return redirect()->back();
    }
    public function cancel($id)
    {
        $order = Order::find($id);
        $order-> status  = 'Cancel';
        $order->save();

        return redirect()->back();

    }
```

2. **Workers :**

- workers can create a job offer with their specific informations and requirements

```
class JobController extends Controller
{
    public function productbycat($category){

        $categories = Category::all();

        // $products = Product::where('status',1)->where('category',$id)->limit(10)->get();

        $jobPosts = JobPost::where('category',$category)->where('status',1)->get();
        if(Auth::user()){
            $client_id = Auth::user()->id;
//            $carts = Cart::where('user_id', $user_id )->get();
        }else{
            $client_id = Auth::user();
//            $carts = Cart::where('user_id', $users_id )->get();
        }
       $settings = DB::table('settings')->get() ;
       $setting = array();
       foreach ($settings as $key => $value) {
          $setting[$value->name] = $value->value;
       }

       $result['setting'] = $setting;

       $data = [
          'setting' => $setting ,
       ] ;

//        $routeName = $name;

//        $hotdeal = Product::where('hot_deal',1)->get();
//        'carts', 'hotdeal','setting', 'routeName'
        return view('client.pages.joblist',compact('categories','jobPosts','setting'));
    }


    public function viewdetails($id){
       $settings = DB::table('settings')->get() ;
       $setting = array();
       foreach ($settings as $key => $value) {
          $setting[$value->name] = $value->value;
       }

       $result['setting'] = $setting;

       $data = [
          'setting' => $setting ,
       ] ;


        $jobPost = JobPost::findOrFail($id);
        $jobPosts = DB::table('jobposts')
            ->orderBy('jobposts.id', 'DESC')
            ->limit(4)
```

- **Worker can request for job or search for specific jobs**

```
        public function findjob(){

            if(Auth::id()){
                $category = DB::table('categories')->get();
                $settings = DB::table('settings')->get() ;
                $setting = array();
                foreach ($settings as $key => $value) {
                    $setting[$value->name] = $value->value;
                }

                $result['setting'] = $setting;

                $data = [
                    'setting' => $setting ,
                ] ;
                return view('client.pages.findjob', compact('setting', 'category'));
            }else{
                 return redirect('/client/login');
            }

            }



            public function jobrequest(Request $request)
            {
//                  dd('123');
// dd($request->all());
                $post = new JobPost;
                $post->title = $request->title;
                $post->name = $request->name;
                $post->phone = $request->phone;
                $post->description = $request->description;
                $post->category = $request->category;
                $post->image = $request->image;
                $post->url = $request->url;
                $post->price = $request->price;
                $post->discount_price = $request->discount_price;
                $post->status = $request->status;
                $post->designation = $request->designation;
                $post->duration = $request->duration;
                $post->work_experience = $request->work_experience;
                $post->qualification = $request->qualification;
                $post->education = $request->education;
                $post->post_created_by = Auth::user()->id;
                if(isset($post->discount_price)){
                    $post->total_price = $request->discount_price;

                }
                else{
                    $post->total_price = $request->price;
                }
```

-

4.

Cart class mainly creates relation between jobpost items and the user items from the database through job_id and user_id

```php
class Cart extends Model
{
    use HasFactory;
    protected $table= 'carts';
    protected $guarded = [];

    public function jobPost(){
        return $this->belongsTo(JobPost::class,'job_id','id');
    }
    public function user(){
        return $this->belongsTo(User::class,'client_id','id');
    }
}
```

Add_cart module is used to save a job_post to the cart selected by a specific client who is authorized . If the client is authorized then this process completes otherwize it redirects to login

```php
class CartController extends Controller
{
    public function add_cart(Request $request, $id)
    {
        // dd(Auth::guard('client')->user());
        if (Auth::user()) {
            $client = Auth::user();
            $jobPost = JobPost::find($id);
            $client_id = $client->id;
                $cart = new Cart;
                $cart->client_id =  $client->id;
                $cart->job_id = $jobPost->id;
                $cart->save();
                return redirect()->back();
            }
         else {

            return redirect('client/login');
        }

}
```

View cart module connects with setting database and then shows the saved items in cart table . Also delete module drops specific id from the cart table .

```
public function view_cart(){

    $categories = Category::all();
    if(Auth::user()){
        $client_id = Auth::user()->id;
        $carts = Cart::where('client_id', $client_id)->with(['jobPost.createby', 'user'])->get();

    }else{
        return redirect('client/login');
    }
   $settings = DB::table('settings')->get() ;
   $setting = array();
   foreach ($settings as $key => $value) {
       $setting[$value->name] = $value->value;
   }
    return view('client.pages.viewcart',compact('categories','carts','setting'));
}

public function delete_cart($id){
    $cart_delete = Cart::find($id);
    $cart_delete->delete();

    return redirect()->back();

}
```

Index module will calculate the total payment for selected job

```
public function index(Request $request)
{

   $client = Auth::user();
   $jobPost = Cart::where('client_id', $client->id)->get();
   $client_id = $client->id;

   $carts = Cart::where('client_id', $client_id)->with(['jobPost.createby', 'user'])->get();
   $total_cart_price = 0;
   foreach ($carts as $cart) {
       $total_cart_price = $total_cart_price + $cart->jobPost->total_price;
   }
```

$data variable stores the specific information about the client

```
$data = array();
$data['jobPost'] = Cart::where('client_id', $client->id)->get();
$data['client_id'] = $client->id;
$data['cus_email'] = $client->email;
$data['cus_phone'] = $client->phone ?? '01777762200';
$data['shipping_method'] = 'sslcommerz';
$data['currency'] = "BDT";
$data['ship_name'] = "Store Test";
$data['ship_add1'] = "Dhaka";
$data['ship_add2'] = "Dhaka";
$data['ship_city'] = "Dhaka";
$data['ship_state'] = "Dhaka";
$data['ship_postcode'] = "1000";
$data['ship_phone'] = "";
$data['ship_country'] = "Bangladesh";
$data['shipping_method'] = "NO";
$data['product_name'] = "product" ?? '';
$data['product_category'] = "category" ?? '';
$data['product_profile'] = "profile" ?? '';

$data['amount'] = $total_cart_price;
$data['total_item'] = Cart::count();
$data['transaction_id'] = rand(1000000, 999999999);
$data['tran_id'] = rand(1000000, 999999999);
```

If the payment is successful the the “Success” module is executed

```php
    public function success(Request $request)
    {
//        echo "Transaction is Successful";
        $tran_id = $request->input('tran_id');

        $amount = $request->input('amount');
        $currency = $request->input('currency');

        $sslc = new SslCommerzNotification();

        #Check order status in order tabel against the transaction id or order id.
        $order_detials = DB::table('orders')
            ->where('transaction_id', $tran_id)
            ->select('transaction_id', 'status', 'currency', 'amount')->first();

        if ($order_detials->status == 'Pending') {
            $validation = $sslc->orderValidate($request->all(), $tran_id, $amount, $currency);

            if ($validation) {
                $update_product = DB::table('orders')
                    ->where('transaction_id', $tran_id)
                    ->update(['status' => 'paid']);
                return redirect('/');

                echo "<br >Transaction is Successfully Completed";
            }
        } else if ($order_detials->status == 'Processing' || $order_detials->status == 'Complete') {
            /*
             That means through IPN Order status already updated. Now you can just show the customer that
             */
            return redirect('/');

            echo "Transaction is Successfully Completed";
        } else {

            #That means something wrong happened. You can redirect customer to your product page.
            echo "Invalid Transaction";
        }
```

Otherwise fail module is executed

```php
public function fail(Request $request)
{
    $tran_id = $request->input('tran_id');

    $order_detials = DB::table('orders')
        ->where('transaction_id', $tran_id)
        ->select('transaction_id', 'status', 'currency', 'amount')->first();

    if ($order_detials->status == 'Pending') {
        $update_product = DB::table('orders')
            ->where('transaction_id', $tran_id)
            ->update(['status' => 'Failed']);
            return redirect()->back();
        echo "Transaction is Falied";
    } else if ($order_detials->status == 'Processing' || $order_detials->status == 'Complete') {
        echo "Transaction is already Successful";
    } else {
        echo "Transaction is Invalid";
    }
```

And lastly it can be canceled using cancel module

```php
public function cancel(Request $request)
{
    $tran_id = $request->input('tran_id');

    $order_detials = DB::table('orders')
        ->where('transaction_id', $tran_id)
        ->select('transaction_id', 'status', 'currency', 'amount')->first();

    if ($order_detials->status == 'Pending') {
        $update_product = DB::table('orders')
            ->where('transaction_id', $tran_id)
            ->update(['status' => 'Canceled']);
            return redirect()->back();
        echo "Transaction is Cancel";
    } else if ($order_detials->status == 'Processing' || $order_detials->status == 'Complete') {
        echo "Transaction is already Successful";
    } else {
        echo "Transaction is Invalid";
    }
```

```php
//admin controller start...
Route::get('/dashboard', [DashboardController::class, 'dashboard'])->middleware('auth') ;
Route::get('/logout', [DashboardController::class, 'logout'])->middleware('auth') ;

//category
Route::get('/category', [CategoryController::class, 'create'])->middleware('auth') ;
Route::post('/store-category', [CategoryController::class, 'store'])->middleware('auth') ;
Route::get('/all-category', [CategoryController::class, 'index'])->middleware('auth') ;
Route::get('/edit-category/{id}', [CategoryController::class, 'edit'])->middleware('auth') ;
Route::post('/update-category/{id}', [CategoryController::class, 'update'])->middleware('auth') ;
Route::post('/delete-category/{id}', [CategoryController::class, 'destroy'])->middleware('auth') ;

//slider
Route::resource('slider',SliderController::class)->middleware('auth') ;

//all workers
Route::get('/workers', [DashboardController::class, 'workers'])->middleware('auth') ;
Route::get('/edit-workers/{id}', [DashboardController::class, 'editworkers'])->middleware('auth') ;
Route::post('/update-workers/{id}', [DashboardController::class, 'updateworkers'])->middleware('auth') ;
Route::delete('/delete-workers/{id}', [DashboardController::class, 'destroyworkers'])->middleware('auth')
```

The routes file will be helping us for admin panel navigation.For example,in this screenshot we can several routes for how an admin can check pages dashboard,category,slider and all workers.

```php
class CategoryController extends Controller
{
    public function index()
    {
        $user= Auth::user();
        $categories = Category::all();
        return view('admin.category.index', compact('categories', 'user'));
    }

    public function create()
    {
        $user= Auth::user();
        return view('admin.category.create', compact('user'));
    }

    public function store(Request $request){
        $this->validate($request,[
           'category' =>'required|unique:categories,category'
        ]);
        $category= Category::create([
            'category' => $request->category,
            'category_created_by' =>Auth::id(),
        ]);
        return redirect()->back();
    }
    public function edit($id){
        $user= Auth::user();
        $category = Category::find($id);
        return view('admin.category.edit',compact('category', 'user'));
    }
```

Controllers will help an admin with the logics behind various pages.In this above screenshot,we can see the CategoryController.php which will handle all the logics behind category page such as adding/removing/editing categories.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Subscribe extends Model
{
    use HasFactory;
    protected $table = 'subscribes';
    protected $fillable = [
        'email',
    ];
}
```

This screenshot is indicating the database for the subscribers.This will fall under the category known as model for this project.Admin will have to access this database-"subscribe.php" to get updates about subscribers.

Technology (Framework, Languages)

Framework: Laravel

Github Repository

Link: https://github.com/00rakesh/shrom_repo/tree/master

Individual Contribution

ID	Name	Contribution
19101588	Rakesh Rakshit	Customer Module
19201043	Ayan Roy	Merging all the code, Creating Worker interface, Uploading in Github, Created the SRS Documentation
19101299	Md.Rubaiat Ferdous	Covered the coding part related to the functionalities of admin such as creating/removing services,set discounts,check purchases,get total expense report etc.Created the user interfaces for admin with the help of HTML,CSS etc.Also contributed with the SRS documentation.
20101492	Shad Been Sharif Sami	Adding Job to cart , view selected job , removing job from cart , viewing total payment and proceeding for payment