



Inspiring Excellence

**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

Section No	Content	Page No
1	Introduction	2
2	Functional Requirements	2
3	User Manual	3
4	Frontend Development	17
5	Backend Development	19
6	Technology (Framework, Languages)	31
7	Github Repo Link	31
8	Individual Contribution	32

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

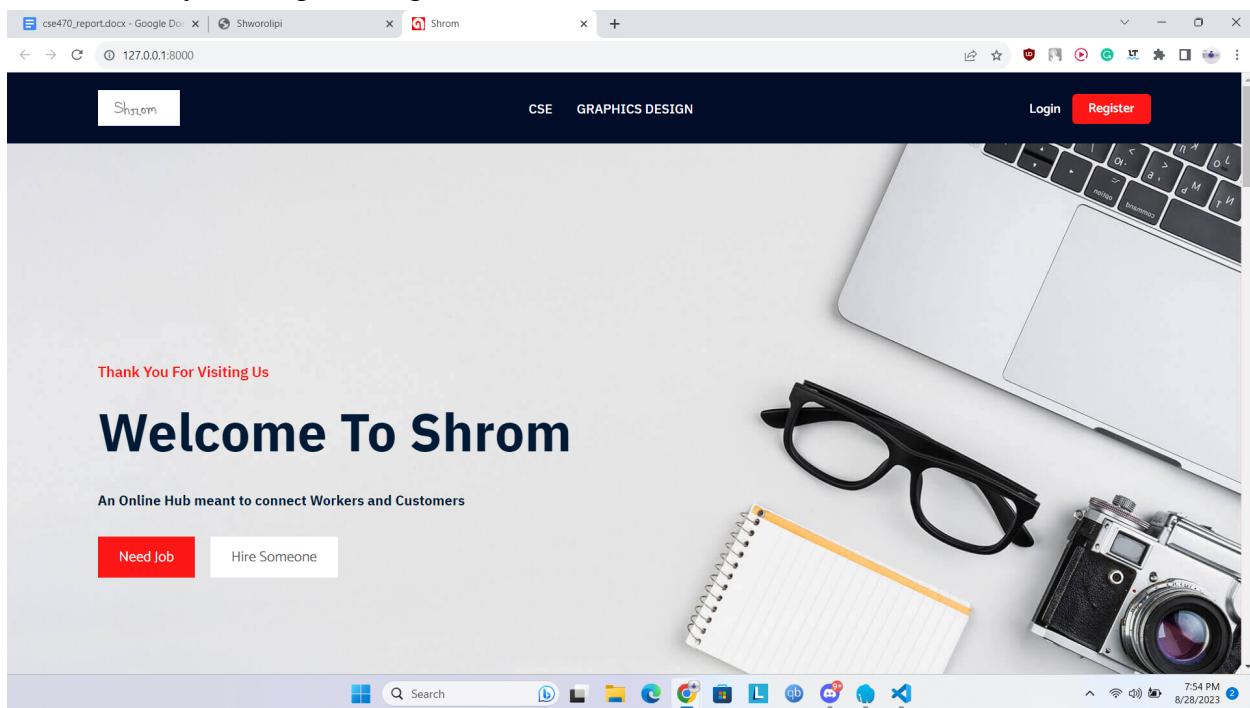
- Customers can add/remove Jobs to their cart
- Customers can view the total amount of their purchase
- Customers can use an online banking system or Bkash/Nagad

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

Jobs You May Be Interested In

- test purpose**
software developer ₦14000 ₦15000 2 month [Browse](#)
- test**
test ₦ 3000 2 month [Browse](#)
- aaaaaaaaaa**
aaaaaaaa ₦ 1100 2 aaaaa [Browse](#)

1.2 Login Page

- User can fillup the form to Login

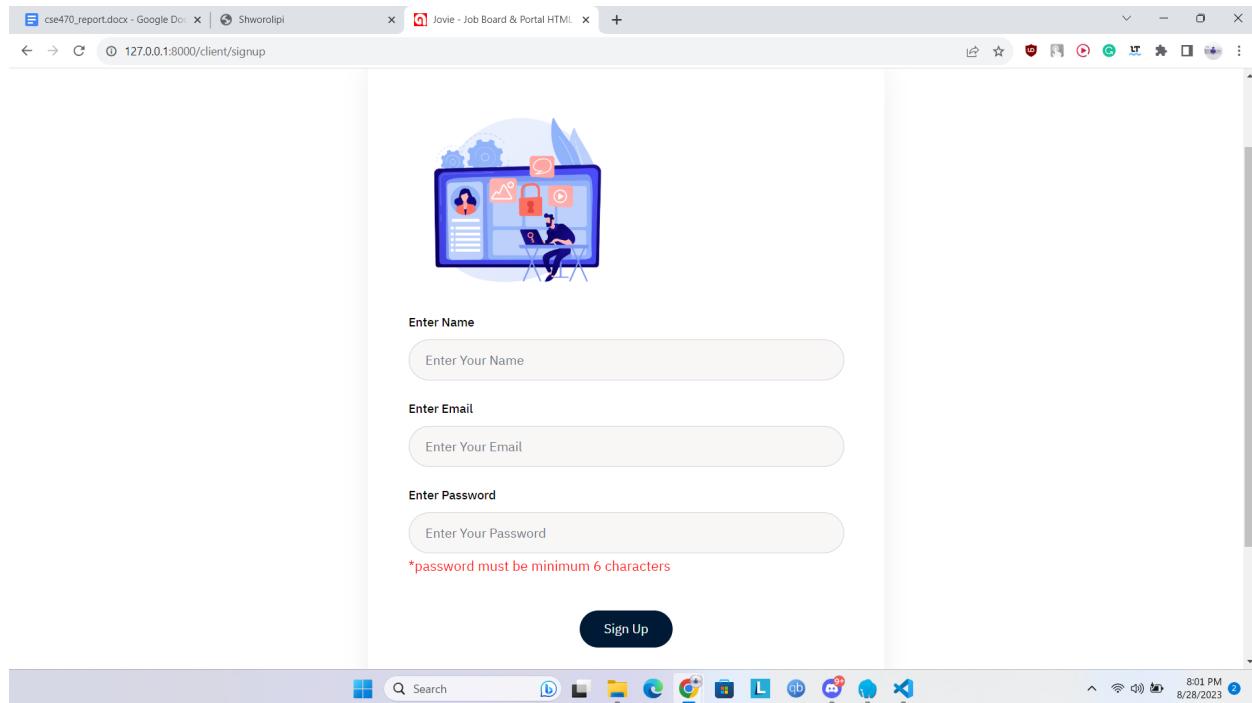
Enter Email
ayan.roy@gmail.com

Enter Password
.....

Sign In

Don't have an account? [signup »](#)

- User can Fillup the form to register



1.3 Job List

- User can View all the Jobs from this page

Job List

Home > Job List

Jobs You May Be Interested In

Placeholder text: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quis ipsum suspendisse ultrices gravida

1.4 Browse Job

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

The screenshot shows a web browser window with two tabs: 'cse470_report.docx - Google Docs' and 'Shworalipi - Shrom'. The main content area displays a job listing for 'test purpose'. It includes a placeholder profile picture, a job title 'graphics design', a role 'software developer', a salary range '₹14000 ₹15000', and a duration 'Duration: 2 month'. Below this, there's a section for 'Qualification' with the placeholder text 'test purpose'. A 'Job Experience : test purpose' section follows, also containing the placeholder text. A prominent red 'Apply Now' button is located at the bottom left of the listing.

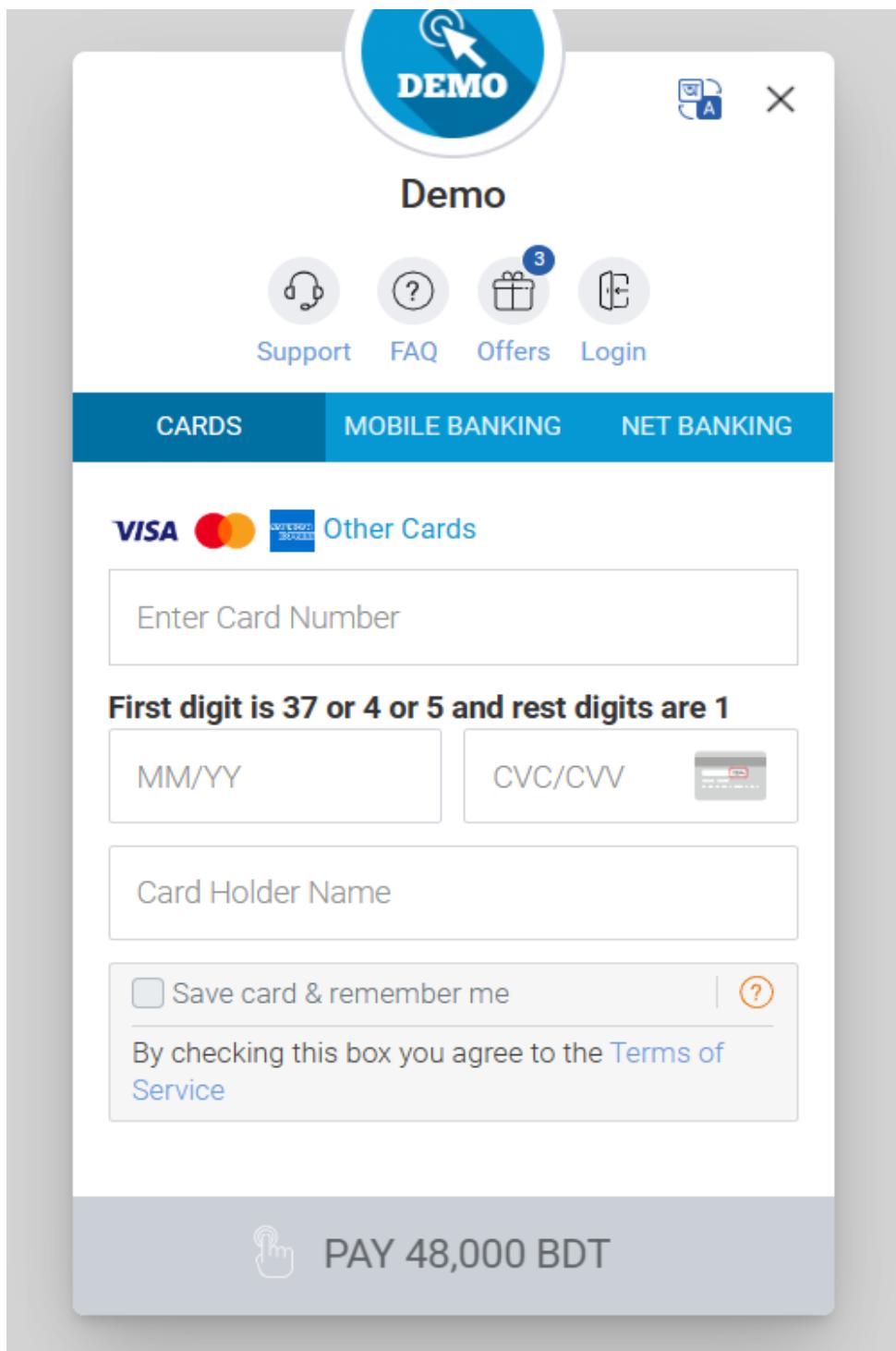
1.5 Cart

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

The screenshot shows a web browser window with two tabs: 'cse470_report.docx - Google Docs' and 'Shworalipi - Shrom'. The main content area displays a cart page. At the top, it shows 'CSE' and 'GRAPHICS DESIGN' next to user icons and a 'Logout' button. Below this, a worker named 'Ayan' is listed. The worker's profile picture is a placeholder, and their name 'Ayan' is displayed below it. They are identified as a 'software developer'. Their salary range is '₹14000 ₹15000', and their duration is 'Duration: 2 month'. To the right of the worker's information is a 'Remove' button. Further down, there's a 'Description' section with the placeholder text 'test purpose' and a 'Work Experience' section with a single entry 'test purpose'. At the bottom of the page, there are two buttons: 'Total Amount: ₹ 14000' and a red 'Hire Worker' button.

1.6 Payment System

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



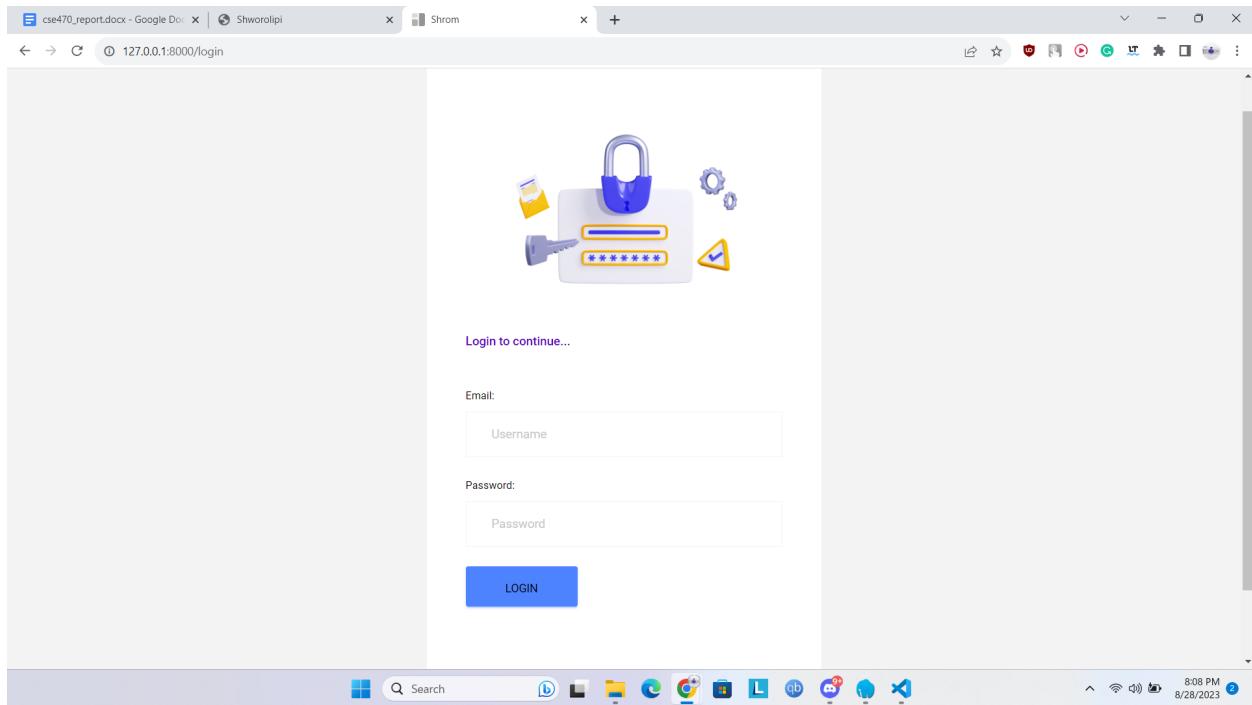
- There will be options for various mobile bankings



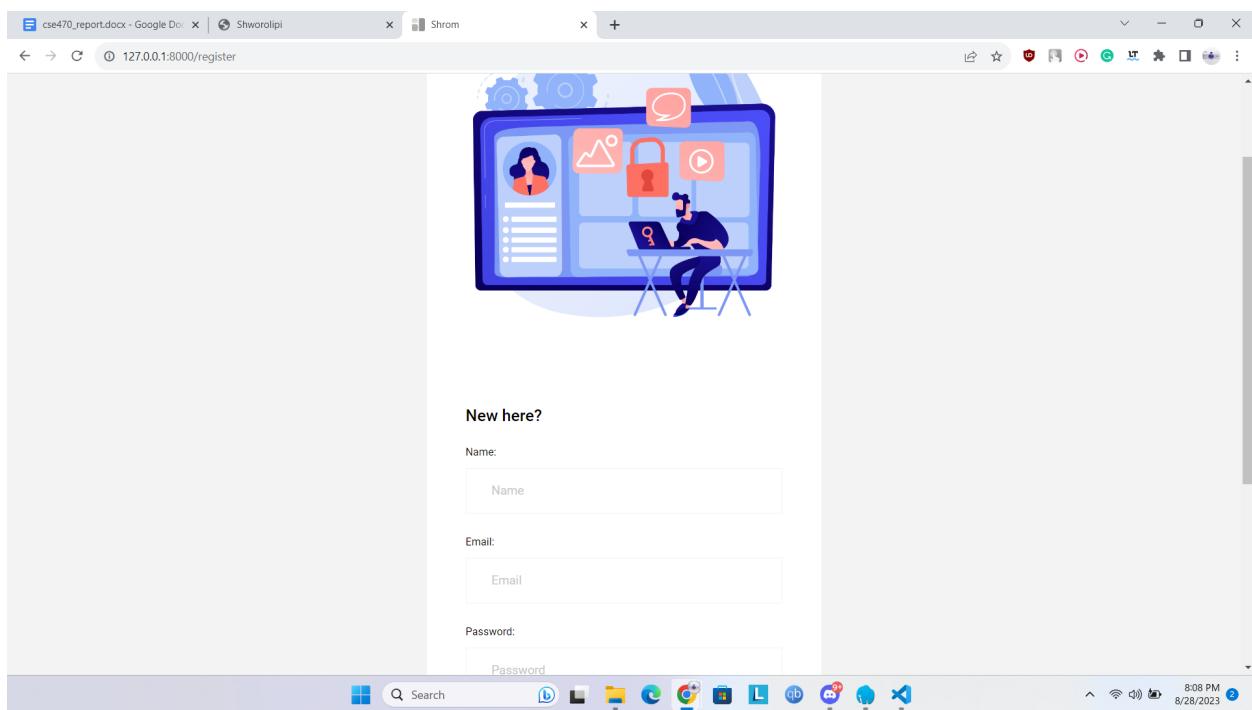
2. Worker Side

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

The screenshot shows a web application interface titled "ALL CATEGORY". On the left, there is a sidebar with navigation links: "Dashboard", "Category" (which is currently selected), and "Job Post". The main content area displays a table with two rows. The first row contains the ID "1" and the category name "cse". The second row contains the ID "2" and the category name "graphics design". Each row has two action buttons: a blue "Edit" button and a red "Delete" button.

ID	Category name	Action
1	cse	
2	graphics design	

- Worker can add categories

The screenshot shows a web application interface titled "CREATE CATEGORY". On the left, there is a sidebar with navigation links: "Dashboard", "Category" (which is currently selected), and "Job Post". The main content area displays a form with a single input field labeled "Name" and a blue "Submit" button below it. A placeholder text "Please fill out the form below." is visible above the input field.

2.3 JobPost

- Worker can see their own Jobposts and edit them

ALL POST

ID	Name	Post title	Description	Category	Designation	Profile Image	Education	Duration	Price	Discount Price	Status	Action

WhatsApp Us

- Worker can create Jobposts

CREATE JOBPOST

Please fill out the form below.

Name

Phone

Education

Title :

Description

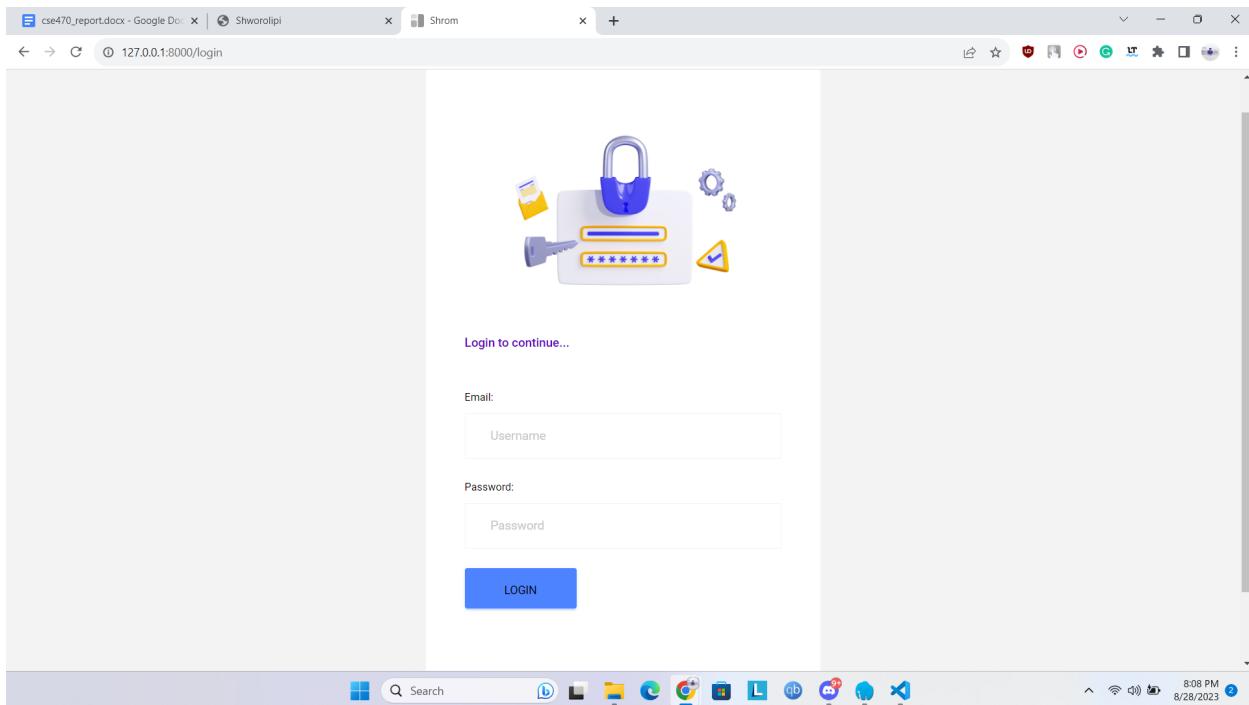
X B U I A H E M F G H I J K L M N O P Q R S T U V W X Y Z </> ?

WhatsApp Us

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

The screenshot shows a web application interface titled "ALL CATEGORY". On the left, there is a sidebar with a logo, navigation links for "Dashboard", "Category" (which is expanded to show "All Categories" and "Create Categories"), and "Job Post". The main content area displays a table with two rows. The first row contains the ID "1" and category name "cse", with action buttons for edit and delete. The second row contains the ID "2" and category name "graphics design", also with edit and delete buttons. The top of the window shows tabs for "cse470_report.docx - Google Docs" and "Shworolpi Majestic Admin", and the address bar shows "127.0.0.1:8000/all-category". The bottom of the screen shows a Windows taskbar with various pinned icons and the date/time "8/28/2023 8:09 PM". A green WhatsApp button is visible in the bottom right corner of the browser window.

- Admin can Create categories to their liking

The screenshot shows a web application interface titled "CREATE CATEGORY". On the left, there is a sidebar with a logo, navigation links for "Dashboard", "Category" (which is expanded to show "All Categories" and "Create Categories"), and "Job Post". The main content area displays a form with a text input field labeled "Name" and a blue "Submit" button. Above the form, a message says "Please fill out the form below." The top of the window shows tabs for "cse470_report.docx - Google Docs" and "Shworolpi Majestic Admin", and the address bar shows "127.0.0.1:8000/category". The bottom of the screen shows a Windows taskbar with various pinned icons and the date/time "8/28/2023 8:09 PM". A green WhatsApp button is visible in the bottom right corner of the browser window.

3.4 JobPost

- Admin can see all Jobposts And also create them

The screenshot displays two pages of the Majestic Admin application:

ALL POST Page:

- Left Sidebar:** Shows a navigation menu with 'Dashboard', 'Category', 'Job Post' (selected), and sub-options 'All Job posts' and 'Create Job posts'.
- Table Headers:** ID, Name, Post title, Description, Category, Designation, Profile Image, Education, Duration, Price, Discount Price, Status, Action.
- Right Side:** A green 'WhatsApp Us' button.

CREATE JOBPOST Page:

- Left Sidebar:** Same as the ALL POST page.
- Form Fields:**
 - Name: Input field.
 - Phone: Input field.
 - Education: Input field.
 - Title: Input field.
 - Description: Rich text editor with toolbar.
- Right Side:** A green 'WhatsApp Us' button.

3.5 All Orders

3.6 Subscribers

- Admin can view all Subscribers

The screenshot shows a web-based administrative interface. On the left, there is a sidebar with a logo at the top and a list of menu items: Dashboard, Category (with a dropdown arrow), Job Post (with a dropdown arrow), All Orders, Subscribers (which is currently selected and highlighted in blue), Pages (with a dropdown arrow), All Workers, All Users, Slider (with a dropdown arrow), and Website Setup. The main content area is titled "ALL CATEGORY". It contains a table with one row. The columns are "SI No", "Subscribers", and "Action". The "SI No" cell contains the number 1. The "Subscribers" cell contains the email address "anannya@gmail.com". The "Action" cell contains a small red square button with a white icon. At the bottom right of the main area, there is a green button with a white phone icon and the text "WhatsApp Us". The browser's address bar shows the URL "127.0.0.1:8000/subscribers". The system status bar at the bottom right shows the time as "8:19 PM" and the date as "8/28/2023".

3.7 All Workers and Users

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

ALL WORKERS

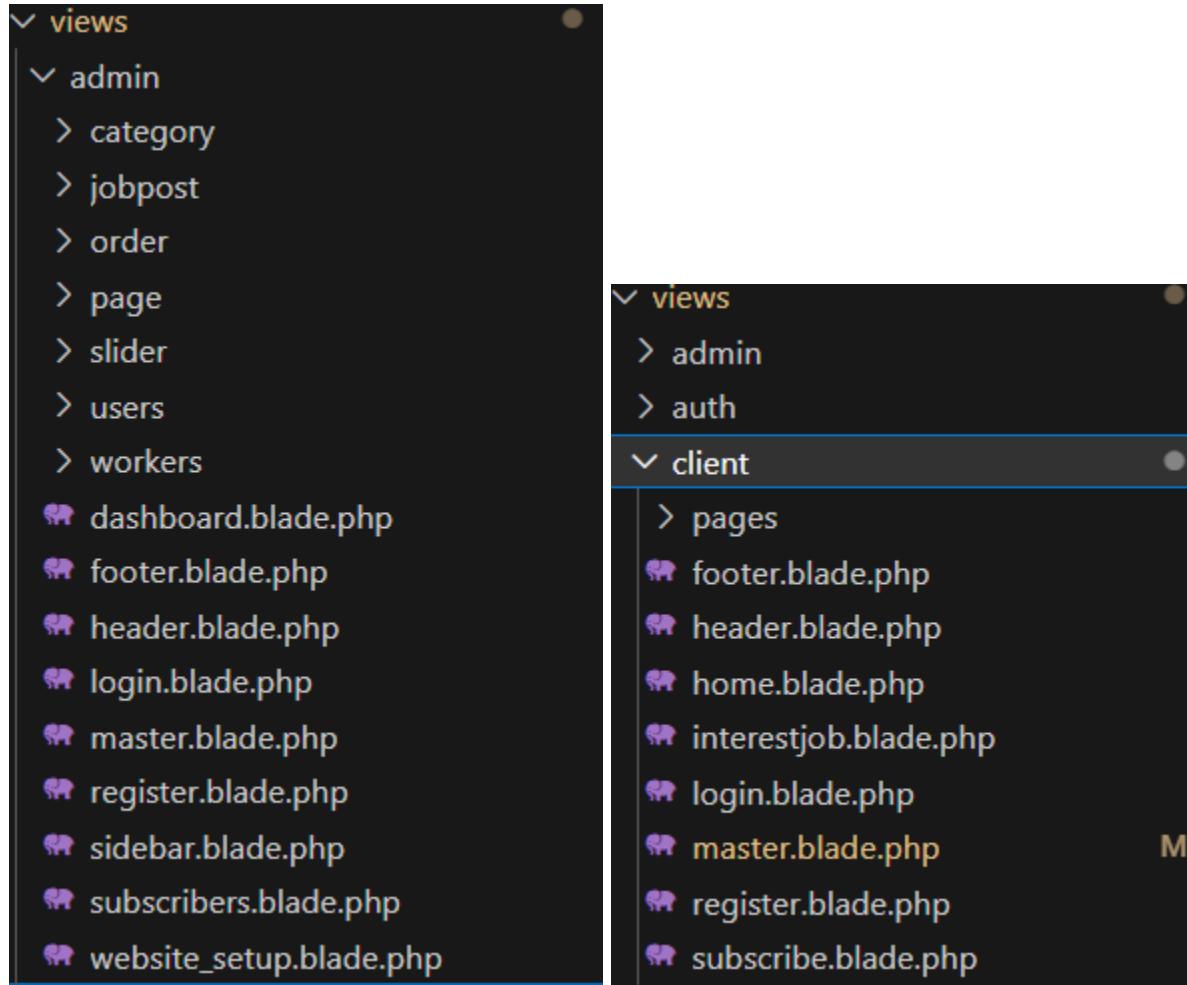
ID	Name	Email	Status	Action
2	diplo	diplo@gmail.com	Active	
5	a	aa@gmail.com	Active	
9	Ayan	ayan.roy@g.bracu.ac.bd	Active	

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	

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

resources > views > admin > master.blade.php

```

11  <link rel="stylesheet" href="{{asset('admin/vendors/base/vendor.bundle.base.css')}}">
12  <!-- endinject -->
13  <!-- plugin css for this page -->
14  <link rel="stylesheet" href="{{asset('admin/vendors/datatables.net-bs4/dataTables.bootstrap4.css')}}">
15  <!-- End plugin css for this page -->
16  <!-- inject:css -->
17  <link rel="stylesheet" href="{{asset('admin/css/style.css')}}">
18  <link rel="stylesheet" href="https://maxst.icons8.com/vue-static/landings/line-awesome/line-awesome/1.3.0/css/line-awesome.min.css" rel="stylesheet">
19  <!-- endinject -->
20
21  </head>
22  <body>
23      <div class="container-scroller">
24
25          <!-- partial:partials/_navbar.html -->
26          @include('admin.header')
27          <!-- partial -->
28          <div class="container-fluid page-body-wrapper">
29              <!-- partial:partials/_sidebar.html -->
30              @include('admin.sidebar')
31              <!-- partial -->
32              <div class="main-panel">
33                  @yield('admin.content')
34
35                  <!-- content-wrapper ends -->
36                  <!-- partial:partials/_footer.html -->
37                  {{-- @include('admin.footer') --}}
38
39                  <!-- partial -->
40
41          <!-- main-panel ends -->
42
43          <!-- page-body-wrapper ends -->
44      </div>
45
46      @include('admin.footer')
47
48  <!-- container-scroller -->
49
50  <!-- plugins:js -->

```

Ln 35, Col 38 Spaces: 2 UTF-8 CRLF PHP

master* ⌂ ⌂ 0 △ 0 8:39 PM 8/29/2023

File Edit Selection View Go Run Terminal Help master.blade.php - shrom - Visual Studio Code

resources > views > client > master.blade.php

```

21  <link rel="stylesheet" href="{{asset('client/assets/css/nice-select.css')}}">
22
23  <link rel="stylesheet" href="{{asset('client/assets/css/style.css')}}">
24
25  <link rel="stylesheet" href="{{asset('client/assets/css/dark.css')}}">
26
27  <link rel="stylesheet" href="{{asset('client/assets/css/responsive.css')}}">
28
29  <title>Shrom</title>
30  <link href="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.css" rel="stylesheet">
31  <link rel="icon" type="image/png" href="{{asset('client/assets/img/favicon.png')}}">
32
33  </head>
34  <body>
35
36  @include('client.header')
37
38  @yield('client')
39  @include('client.subscribe')
40
41
42  @include('client.footer')
43  <div class="copyright-text text-center">
44      <p>Developed By Ayan Roy</p>
45  </div>
46
47
48  <div class="top-btn">
49      <i class="bx bx-chevron-up bx-fade-up"></i>
50  </div>
51
52
53  <script data-cfasync="false" src="{{asset('..../cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js')}}"></script>
54  <script src="{{asset('assets/js/jquery.min.js')}}"></script>
55  <script src="{{asset('client/assets/js/bootstrap.bundle.min.js')}}"></script>
56
57  <script src="{{asset('client/assets/js/owl.carousel.min.js')}}"></script>

```

Ln 39, Col 29 Spaces: 4 UTF-8 CRLF PHP

master* ⌂ ⌂ 0 △ 0 8:40 PM 8/29/2023

Backend Development

1. Admin can create/remove services:

- admin can edit workers

```

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

```

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

```
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

```
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

```

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

```

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', $tran_id)
            ->update(['status' => 'Failed']);
        return redirect()->back();
        echo "Transaction is Failed";
    } else if ($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

```

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";
    }
}

```

```

//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 see several routes for how an admin can check pages dashboard, category, slider and all workers.

```

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

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