T.C.

ALTINBAS UNIVERSITY

(Electrical and Computer Engineering)

DESIGN AND IMPLEMENTATION OF WEB BASED FOR INTERMEDIATE ONLINE SHOP BY USING MVC LARAVEL FRAMEWORK

Mohammed Thakir MAHMOOD

Master Thesis

Prof. Dr. Osman Nuri UCAN

Istanbul (2019)

DESIGN AND IMPLEMENTATION OF WEB BASED FOR INTERMEDIATE ONLINE SHOP BY USING MVC LARAVEL FRAMEWORK

by

Mohammed Thakir MAHMOOD

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Osman Nuri UCAN

Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Prof. Dr. Osman Nuri UCAN	School of Engineering and Natural Science, Altinbas university	_____
Assoc. Prof. Dr. Oguz BAYAT	School of Engineering and Natural Science, Altinbas university	_____
Asst. Prof. Dr. Adil Deniz DURU	Physical Education and Sport Marmara University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Çağatay AYDIN

Head of Department

Assoc. Prof. Dr. Oguz BAYAT

Director

Approval Date of Graduate School of Science and Engineering: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mohammed Thakir MAHMOOD

DEDICATION

I would like to dedicate this work to my lovely family and specially my mother, for their invaluable efforts when I felt hopeless and weak in solving problems.

ACKNOWLEDGEMENTS

I wish to express my acknowledgments to my supervisor, Asst. Prof. Dr. Osman Nuri UCAN who was abundantly helpful and offered invaluable support with his sincerity and belief in me.

ABSTRACT

DESIGN AND IMPLEMENTATION OF WEB BASED FOR INTERMEDIATE ONLINE SHOP WITH LARAVEL FRAMEWORK

Mohammed Thakir MAHMOOD

M.S., Electrical and computer engineering, Altınbaş University,

Supervisor: Dr. Osman Nuri UCAN

Co-Supervisor:

Date: 4/2019

Pages: 54

Working with traditional methods to develop a web applications causing large limitations and a lot of time consuming and with number of unexpected errors . for this reason a new technology like MVC pattern frameworks found by some companies to deal with such issues . in this research we presented a design and implementation for web based application for e-commercial shop and third-part to buy products from online shops . we have used Laravel framework to do so . as result of this research we can find out the development was standardized and non-business logic relationships automatically processed, there was much scalability so this gives us more efficiency through the implementations.

Keywords: MVC pattern, PHP frameworks , Laravel framework.

TABLE OF CONTENTS

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

MVC : Model-View-Control

REST : Representational State Transfer

LIST OF TABLES

1. INTRODUCTION

With the broad use of Web innovation, many organizations have earnest requirements to construct their own Web business frameworks rapidly and proficiently [1]. however , an high quality application relies upon the support of very much well-designed system structure. . Step by step instructions to accurately apply the center innovation to plan and fabricate a steady, adaptable and reusable Web application framework structure is the test we face [2].Amazon , ebuy , and many other online shops are so popular nowadays for online shopping and it's have been using widely by customers to buy products that don't exist in their countries or for saving their time in shopping or to buy a specific brands in that shops , so for many reasons the online shopping has been important in our life but for some reasons some online shops don't sell their products for some countries specially in the middle east for many reasons , so this is a problem need to be fixed . we will build a website to deal with this problem using a business model to buy these products for users and ship them to their homes . we will use php language for programming but by using the MVC framework for some reasons that will justify it later in this chapter.

.

1.1 PROBLEM STATEMENT

As we mentioned before , we are trying to design and implement a website as a third part to buy products for users that can't buy them directly . Our idea is the next , first there will be a registration part for users then they will login in their accounts . in the user profile system there will be a part to submit an order for products . in the order, there will be all information about product and the quantity of the product and the color and size (optional) . after submit the order the forma of the order will received in the administration side to review it and reply to the customer or user with the acceptance or refuse on order . if the admin accept the order there will be a receipt for the order including the price information with the website's commission on the products (mostly 10% on products) .The receipt will be shown to user in his profile , if the user satisfy he will pay and send a copy of the receipt for payment as an answer to buy his order .the administration will check the payment and then buy the products for user .

To build this website we are going to use the HTML and CSS with JavaScript help for the interface . for programming and data storing we are going to use php MVC framework .

1.2 SIGNIFICANCE OF THE STUDY

In our study we are trying to create a website based on MVC Pattern framework . we will show in our study how we may use MVC framework works and the advantages and disadvantages of the framework, according to HE REN YU[6] he said that "Laravel make the development process is standardized ,processing some non-business logic relationship automatically. In his paper designs and implements a simple Laravel model, which achieved automated processing for part of the design. The experimental and simulation proved, web design based on Laravel framework, has scalability and robust scalability, so as to improve the developing efficiency." and the difficulties that may any web developer may face . on the other hand the idea of website to serve customers who have a problem to buy some products from global online shops .

1.3 MVC PATTERN FRAMEWORK

The Model-View-Controller pattern nowadays is widely been used by companies and developers . first time that used in Smalltalk and after that adopted and published by Java. The MVC pattern working in the following way : separates the application into three modules: Model, View and Controller. The Model is to control and to manage the data .it save and receive the entities that used by an application, usually from a database, and contains the logic implemented by the application. The View is responsible to view the data that supplied by the model in a specific format. The Controller process the Model and View layers to work together. The Controller receives a request from the client, invokes the Model to perform the requested operations and sends the data to the View. The View formats the data to be presented to the user, in a web application as a HTML output. The MVC pattern is shown in Fig. 1

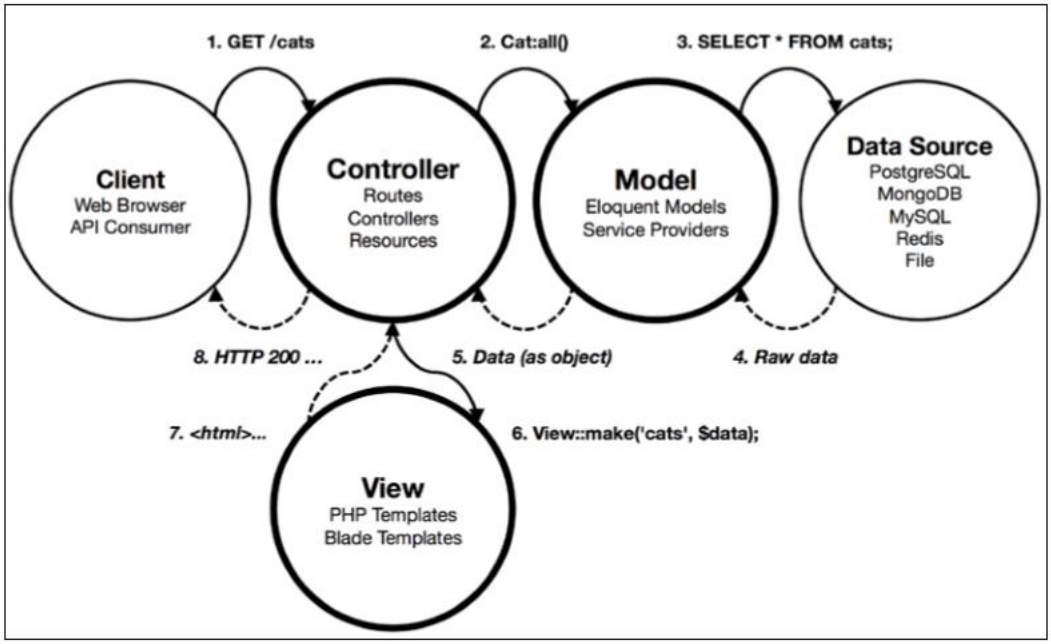

Figure 1.1: MVC pattern

1.4 PHP FRAMEWORKS

PHP is a server-side scripting language which is used to develop web applications. Database controllers. Frameworks are built to make it even faster to develop a website. The biggest PHP Frameworks with MVC thinking are Code Igniter, Laravel and CakePHP.

Laravel: for the first time has released in June 9, 2011. According to Laravels creator Taylor Otwell, the reason for creation the framework was the lack of some essential functionality, like user authentication in the CodeIgniter framework [3]. Laravel is an Open Source framework It has an exceptionally rich arrangement of highlights which will support the speed of Web Development. In the event that you acquainted with Core PHP and Advanced PHP, Laravel will make your errand less demanding. Laravel is based on the shoulders of goliaths. It utilizes parts from other reliable systems like Symfony. On the off chance that populating is anything to pass

by, Laravel MVC has the Lion‟s share in the PHP MVC structures piece of the overall industry. As per an overview done by SitePoint, Laravel MVC structure is the most utilized system in both individual tasks and at work. Since it figures out how to do every single basic undertaking extending from web serving and database the executives directly to HTML age, Laravel is known as a full stack structure. This vertically incorporated web advancement condition is intended to offer an improved and smooth work process for the designer.

Features of LARAVEL

- It Supports Rapid Application Developing (RAD):
- It support Blade template: this is a built in template engine Built in features for common web application tasks such as authentication, routing, database managing, sending emails Class auto loading.
- It support RESTful controllers that allows you to take advantage of HTTP verbs such as GET, POST, PUT, and DELETE etc.
- It support Unit Testing which built in unit testing features that can be run from the artisan command line.

1.5 THESIS OBJECTIVES

Our research focuses on building a website based on Laravel framework .we want to show to the web developers the Significance of using MVC frameworks and how to use programming languages to apply businesses models and e-commercial ideas . We believe that Our website will be used by many users who have difficulties to buy some products from global online shops and on other hand get revenue from our knowledge of using programming languages .

2. SYSTEM DESIGN AND ANALYSIS

2.1 INTRODUCTION

In this chapter , online shop web system design and planning will be presented through Unified Modeling Language UML. The design part for the application is so important for the quality management and to understand how the business are working according to the requirements .Usually , the requirements are been collected by analysts and then converted into diagrams .these diagrams to show the developer how the system are working and how the data are been flowed through the system . Four UML Diagram we will use in this chapter , activity diagram, ER diagram, SQL schema Other complementary diagram is user interface.

2.2 SYSTEM ARCHITECTURE

2.2.1 Customer Registration System

The most important part in the in User side is registration into the system to create customer profile . can register to the system by this part and manage his profile . The user will be asked about his personal information like full name and phone or email and full address. After that if the information is true and there is no lack in them , the code verification will be sent via email address or phone number , the customer will be asked to enter the verification code to confirm his information for security . In figure 2 the Activity Diagram for customer registration .

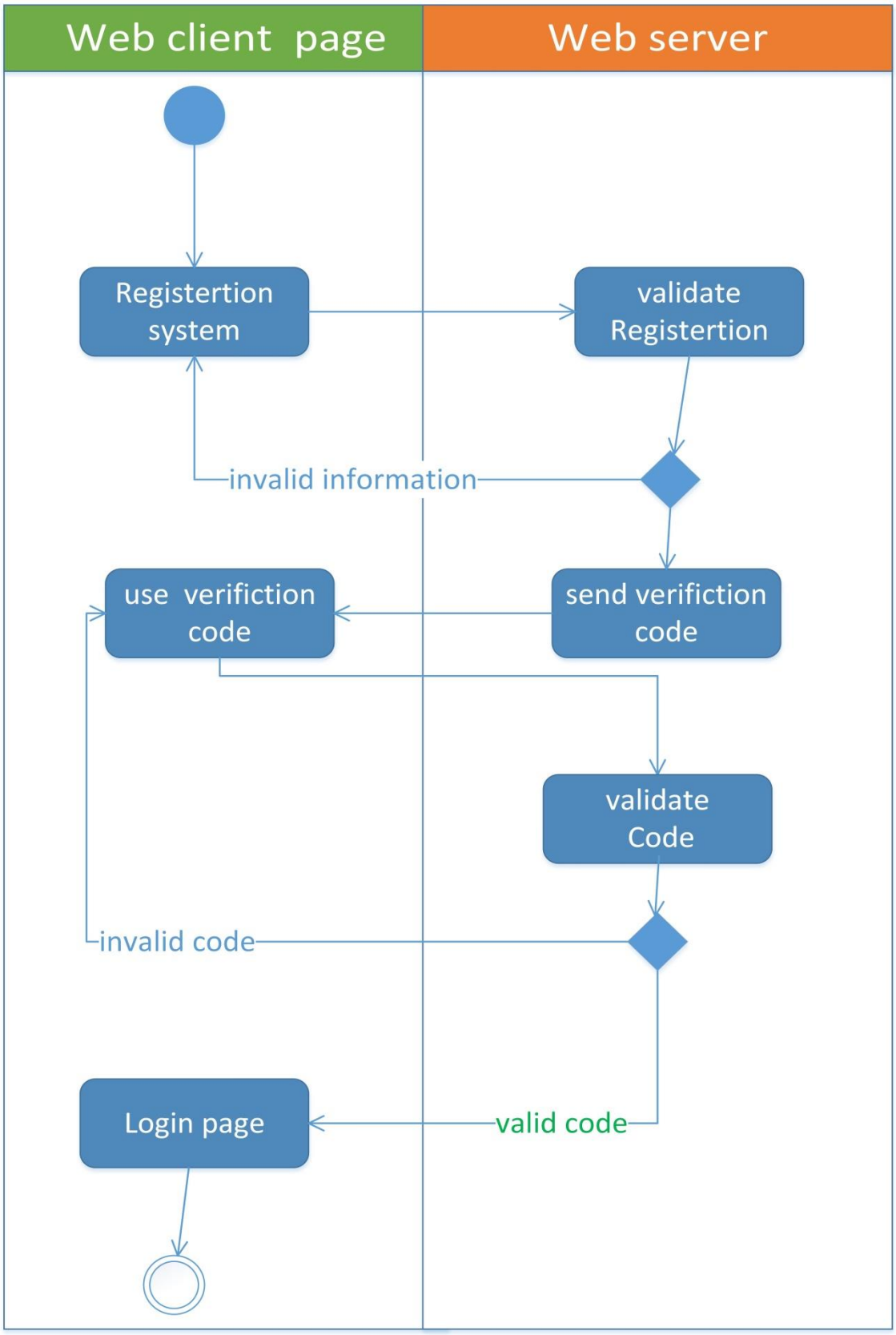

Figure 2.1: Registration Activity Diagram

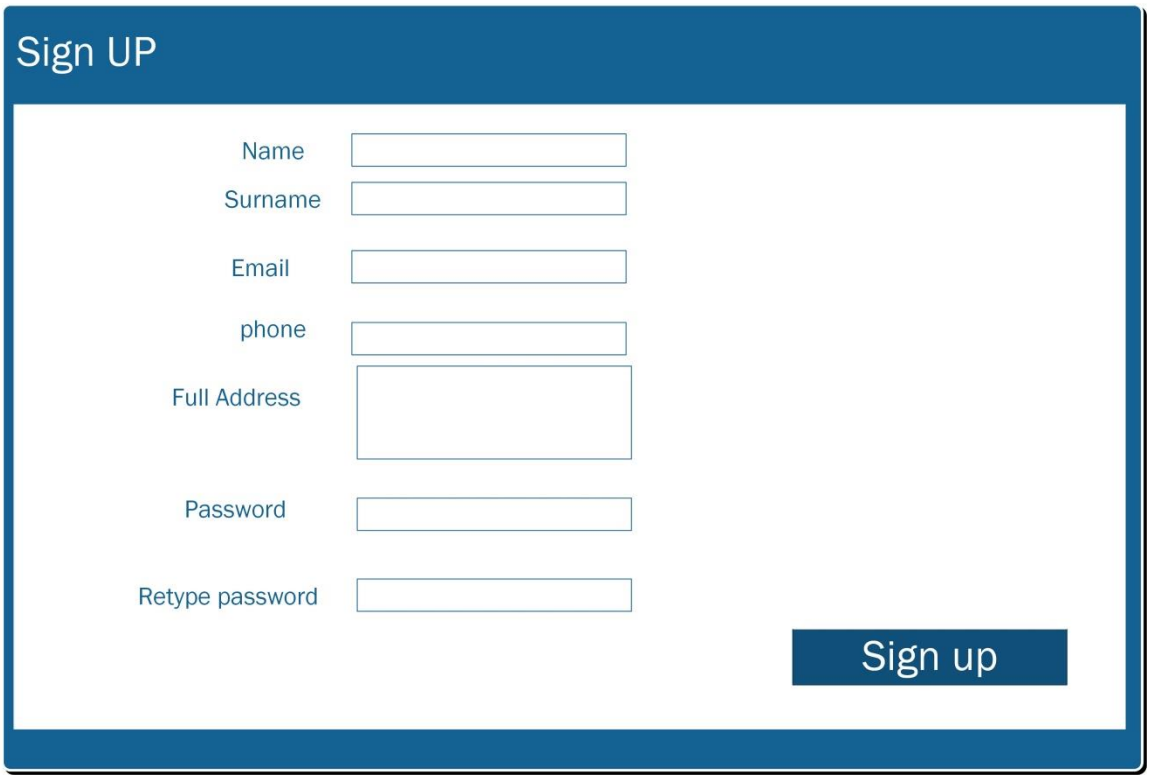

Figure 2.2: User Interface Diagram

2.2.2 Customer Login System

In this part Customer can access his profile by the email and password that registered before .

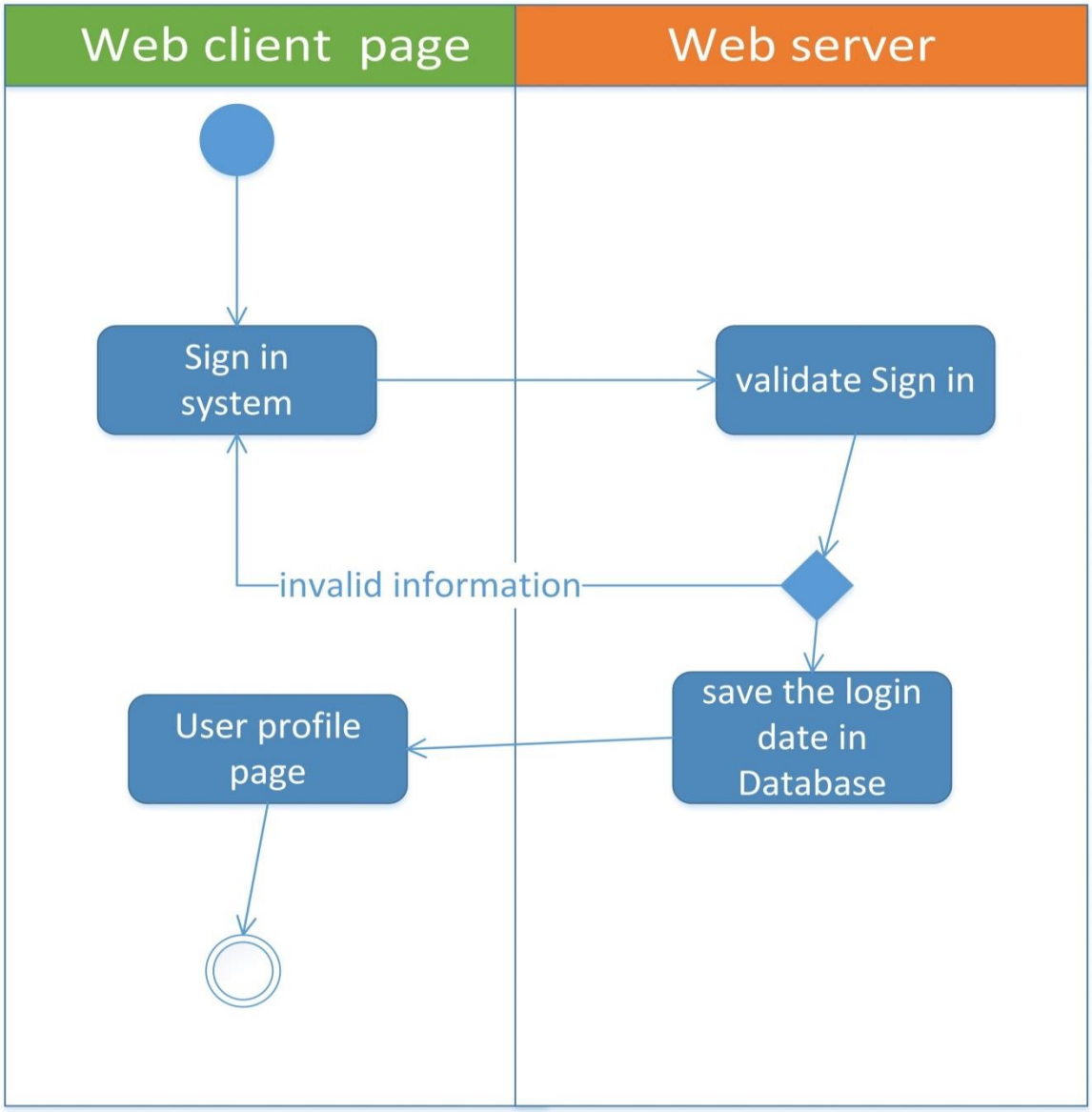

Figure 2.3: Sign In Activity Diagram

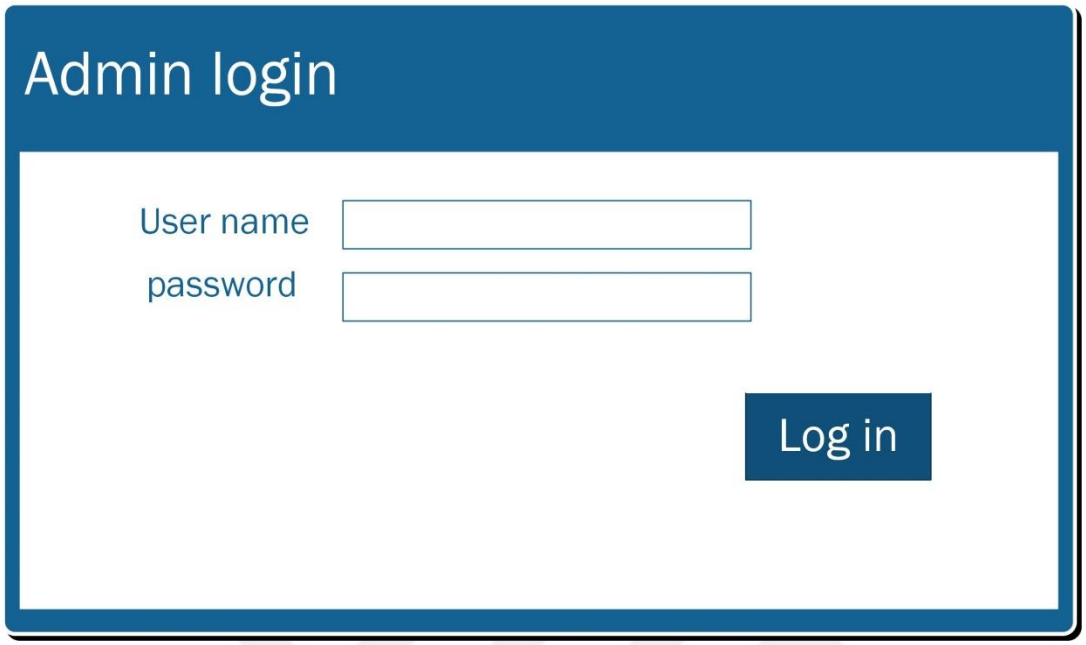

Figure 2.4: Sign In Interface Diagram

2.2.3 Customer Profile System

User have his own profile to manage his service and buy products by add an order for products by the name of the product , link , price , quantity…etc.

After user submit his order he will wait the administration reply on his order with invoice of total payment . The invoice contain the price of products and charging fees and commission of the website . user can pay the money in two way , first by deposit the money in a bank and send a copy of invoice to admin to approve his order or by electronic payment with PayPal payment .in this user profile , the customer can change his information and manage his orders and he can contact with administration , he can know the price of charge for each kilogram and he can give his opinion in the website's service.

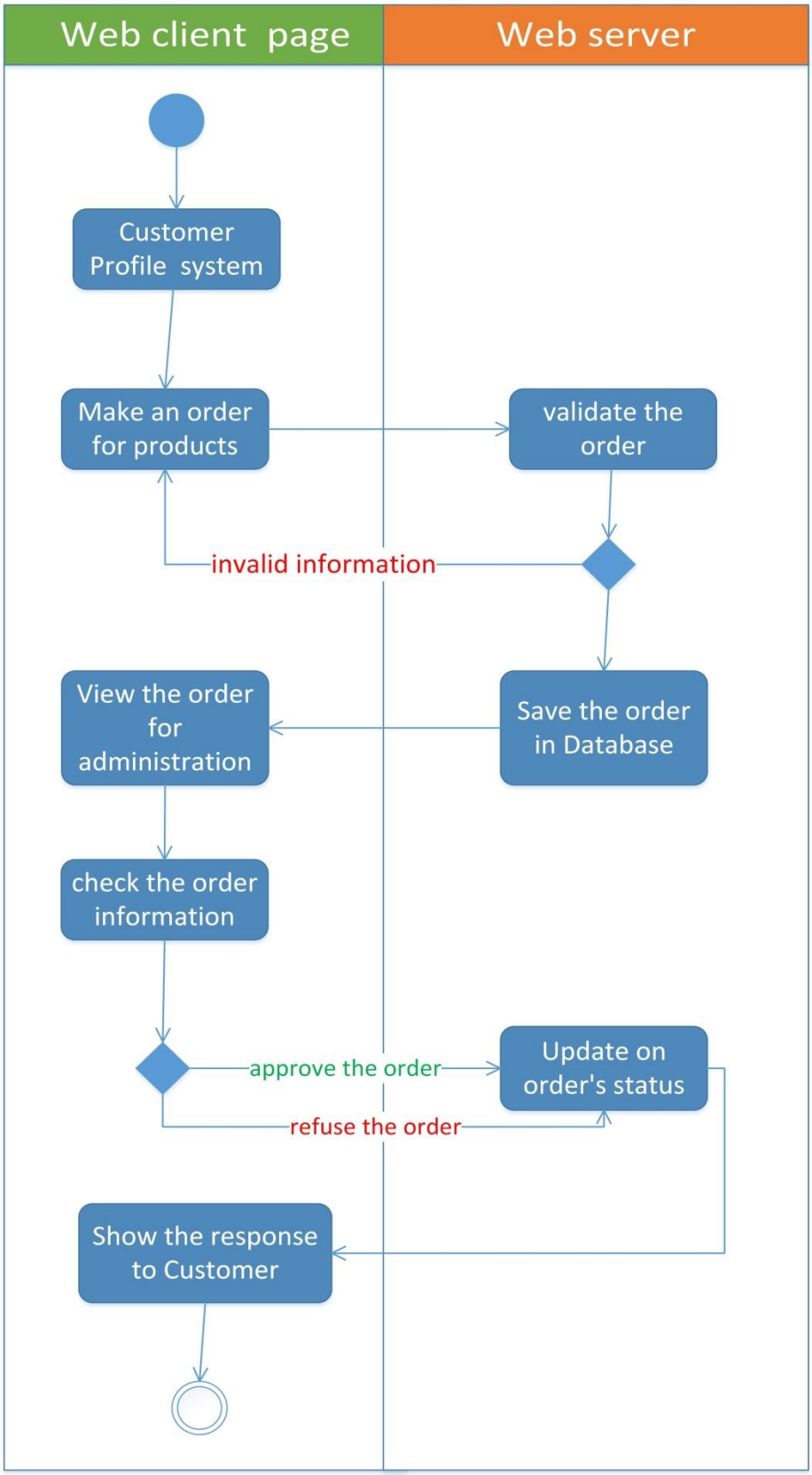

Figure 2.5: Activity Diagram For Making An Order

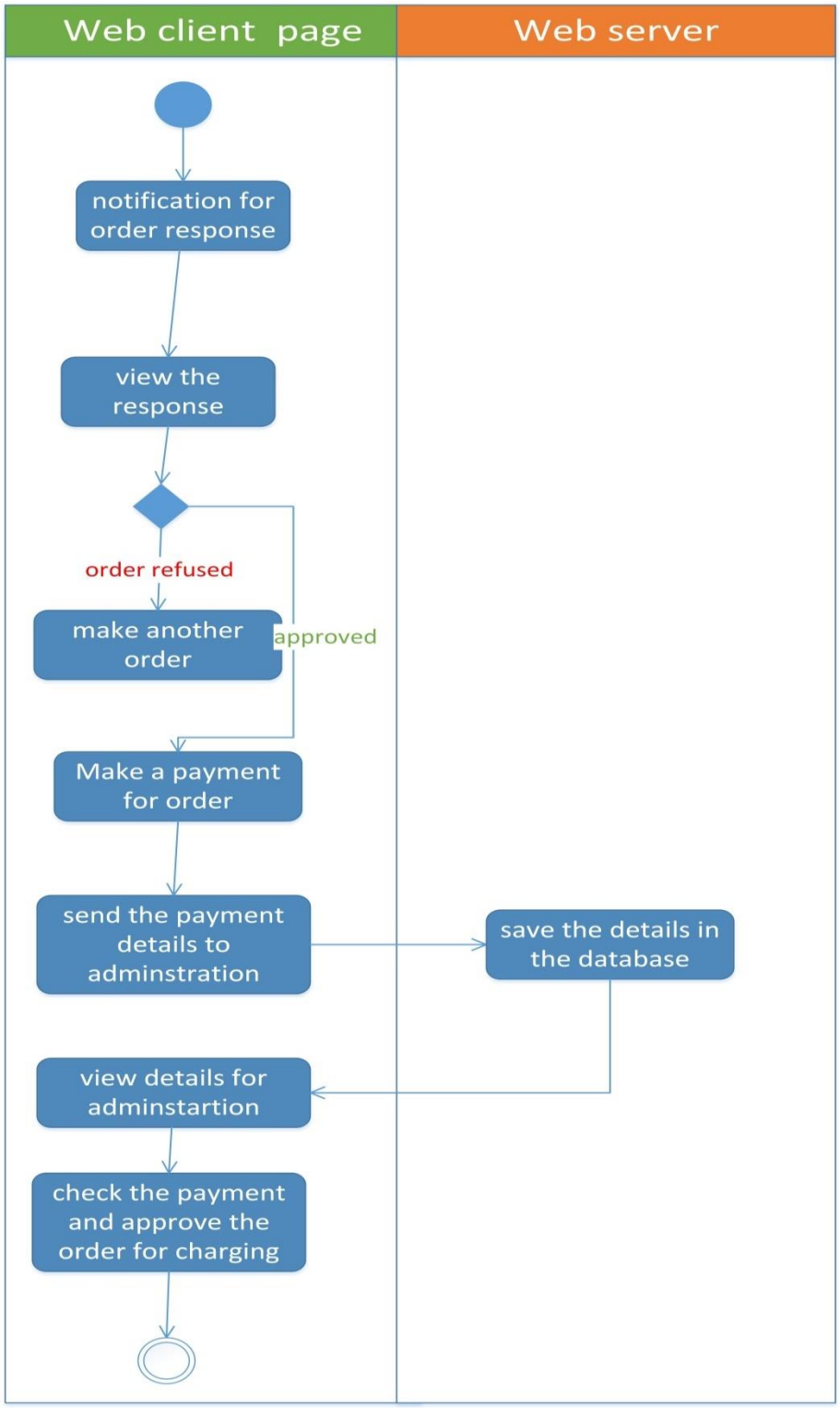

Figure 2.6: Activity Diagram For Order Payment

2.2.4 Admin Login System

Admin can login to system by this part and manage his authority in the website . every supervisor has a limited authority in the system ,the Administrator who determine these authorities during the addition of a new supervisor in the system . Before the addition of supervisors the Administrator must add a supervisors groups and each group has an authority and limitations in the system . When the supervisor make a login and after validation of his accessibility in the database has been succeed , we check his group in the system and manage his authority , so if he enter a department in the admin side through his group we validate if this page is authorized within the group of supervisor in figure 8 we will explain it through the activity diagram .

2.2.5 Supervisor Profile Page

In this part the admin can see his own profile to update his information like name and email , password …etc.

2.2.6 Add New Supervisor

Administrator can add supervisors and supervisors who have the authority to use this section. Administrator will provide the information of the supervisor like the name , username, phone , email and some notices and choose his authority inside the administration by choose the appropriate group .In figure 11 the activity diagram for this procedure and figure 10 for interface diagram.

2.2.7 View All Supervisors

In this section the Administrator can see all the existing supervisors and he can update on their information or delete them .figure 12 the interface diagram for this section .

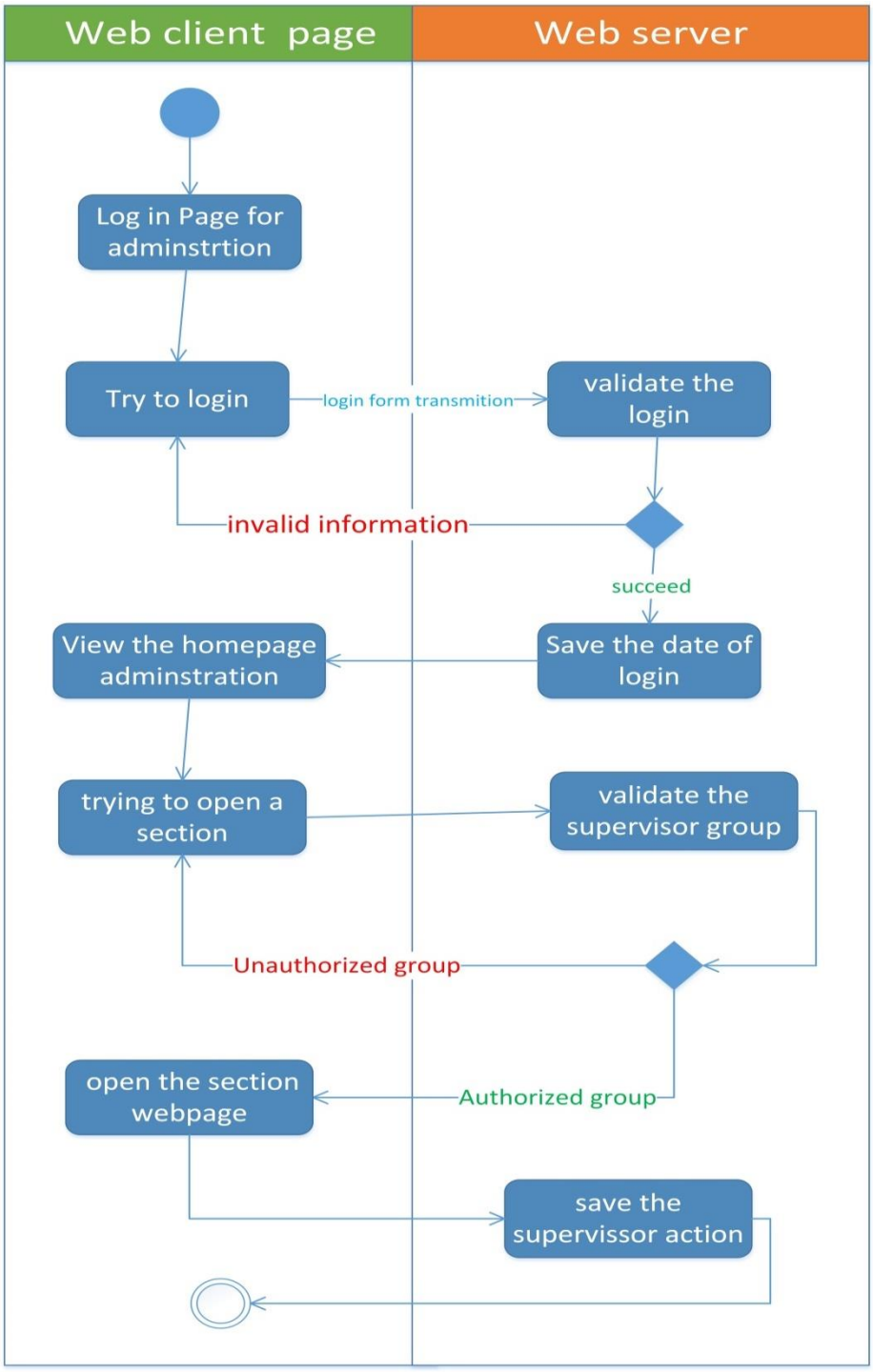

Figure 2.7: Admin Login And Authority System

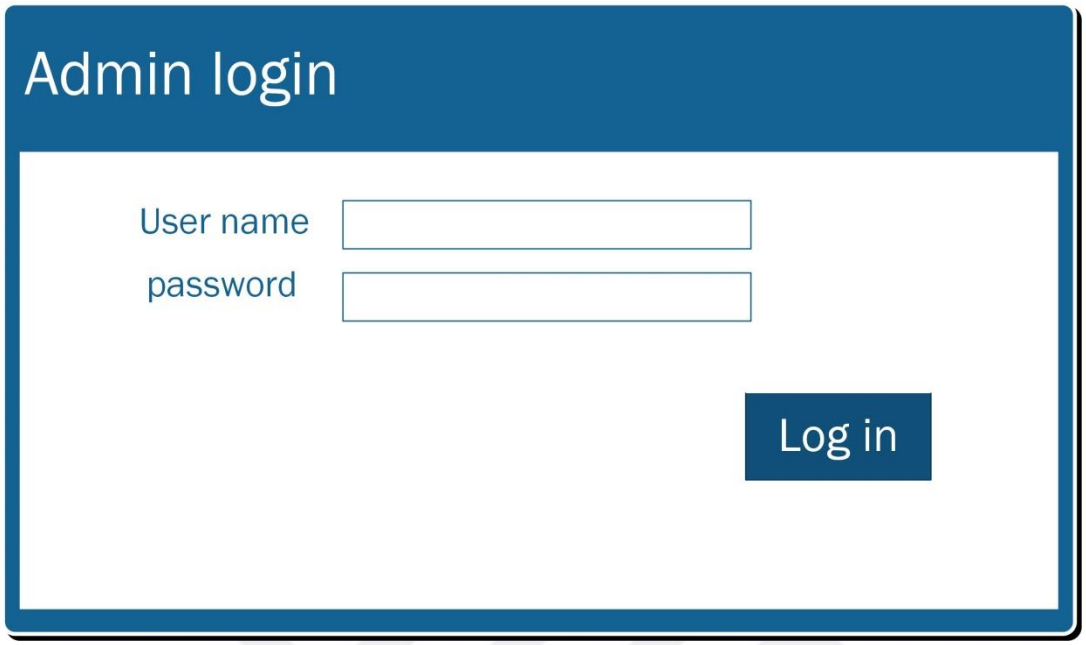

Figure 2.8: The Interface Diagram For Login

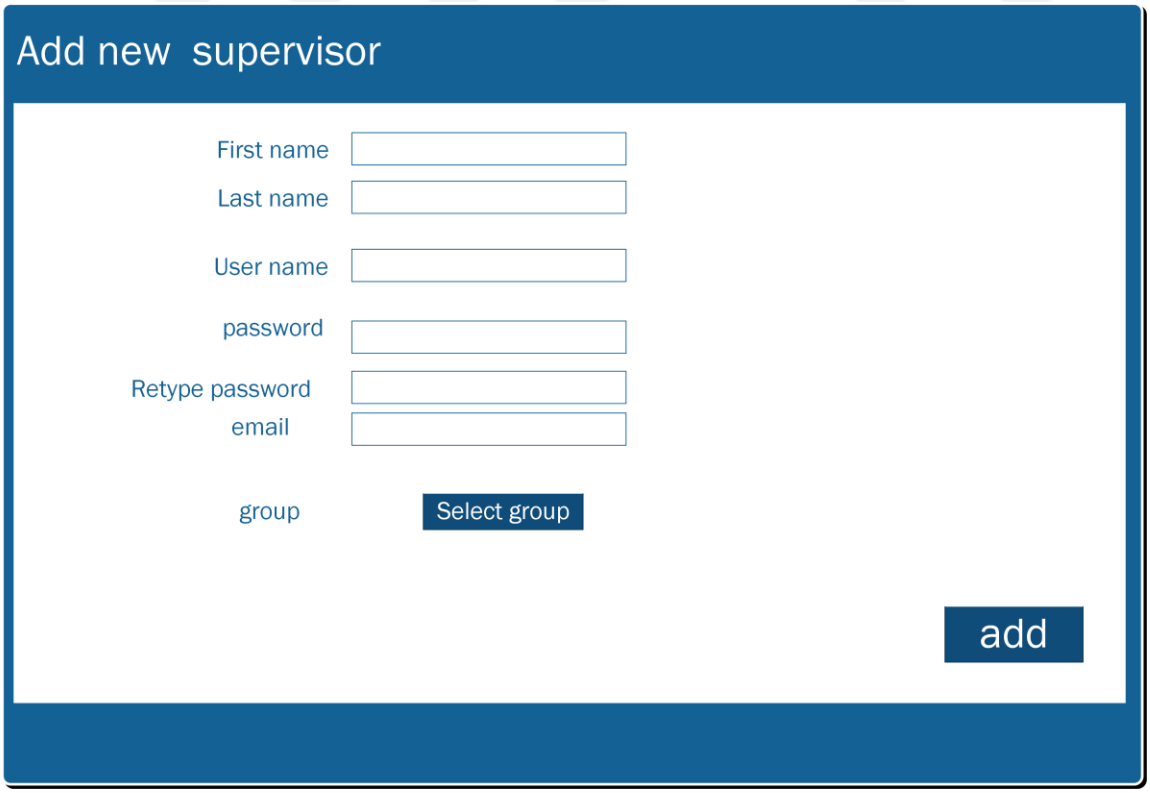

Figure 2.9: The Interface Diagram For Add A New Supervisor

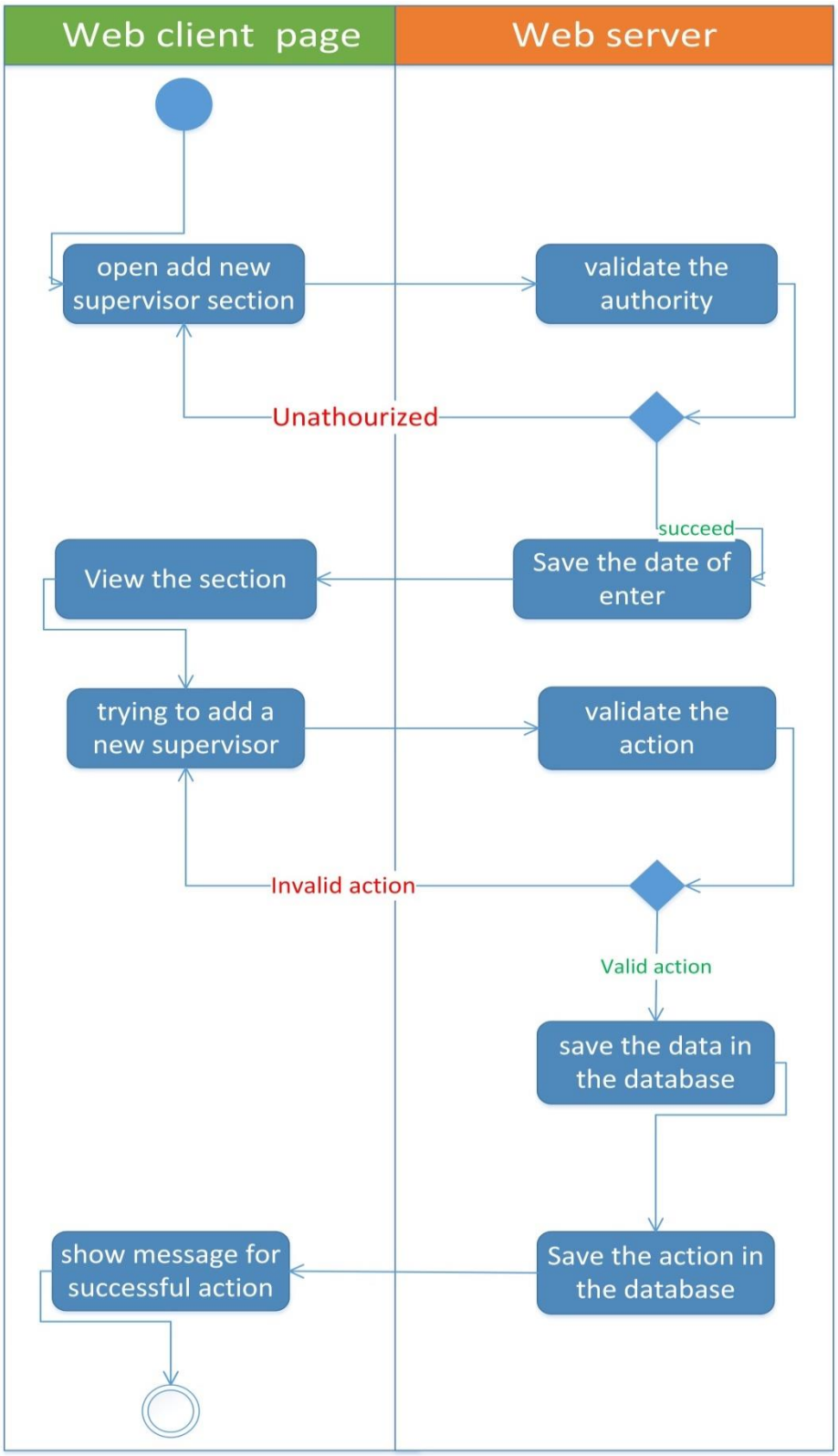

Figure 2.10: The Activity Diagram For Add A New Supervisor Procedure

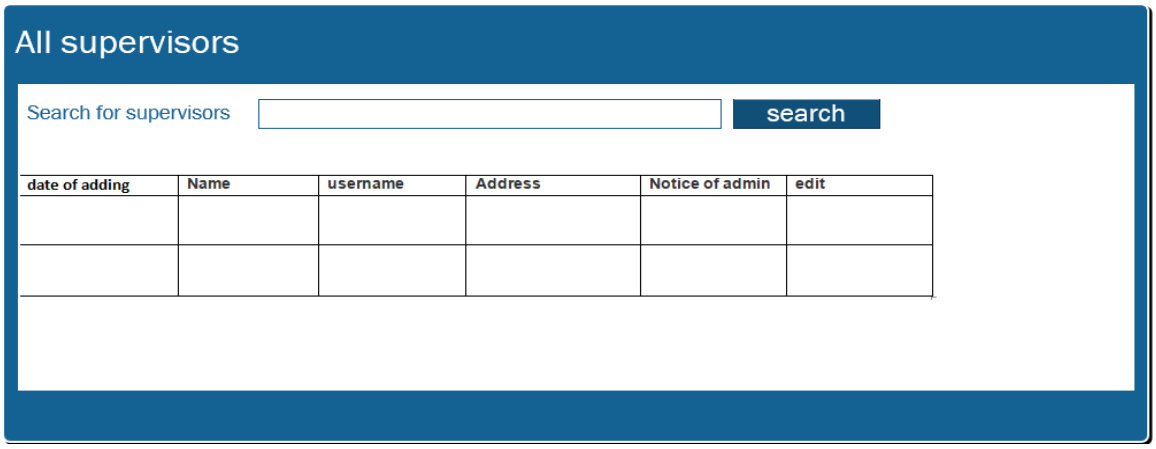

Figure 2.11: View All Supervisors Interface Diagram

2.2.8 Add New Supervisor Group

In this part admin cad add group to determines the authorities of the supervisors for security issues and to organize them . admin can add the group by provide the name of group and choose the sections that group can access them .When we select the section for a group , we save the name of the group in a table with group's ID , and the sections on another table with many to many relationship , that's mean the name of the section and the group's ID in the same data row .fig 14 the activity diagram for this section .

2.2.9 View All Groups

In this part admin cad see all groups and he can edit them or delete them.

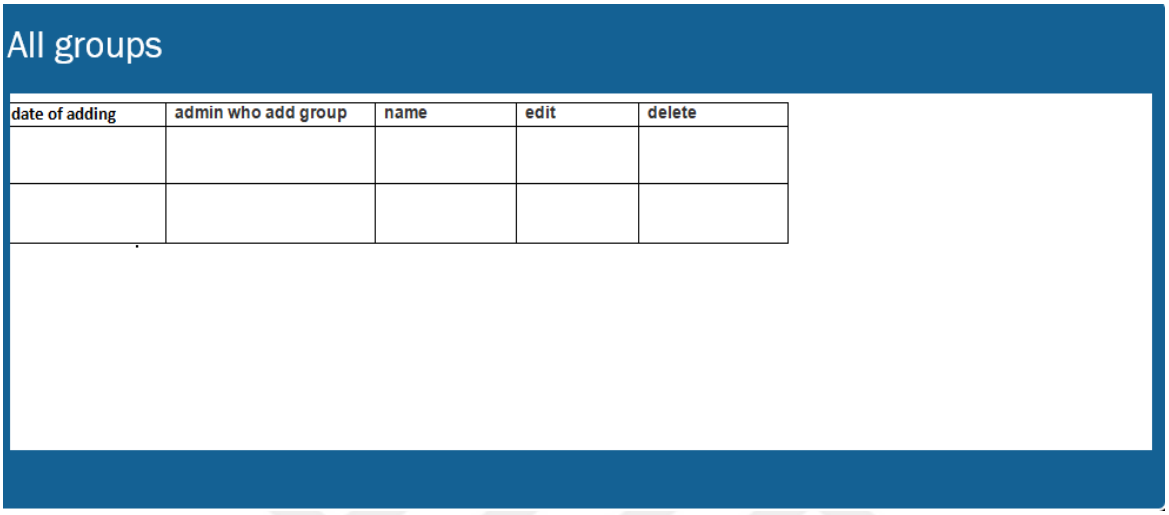

Figure 2.12: Interface Diagram For View All Groups Section

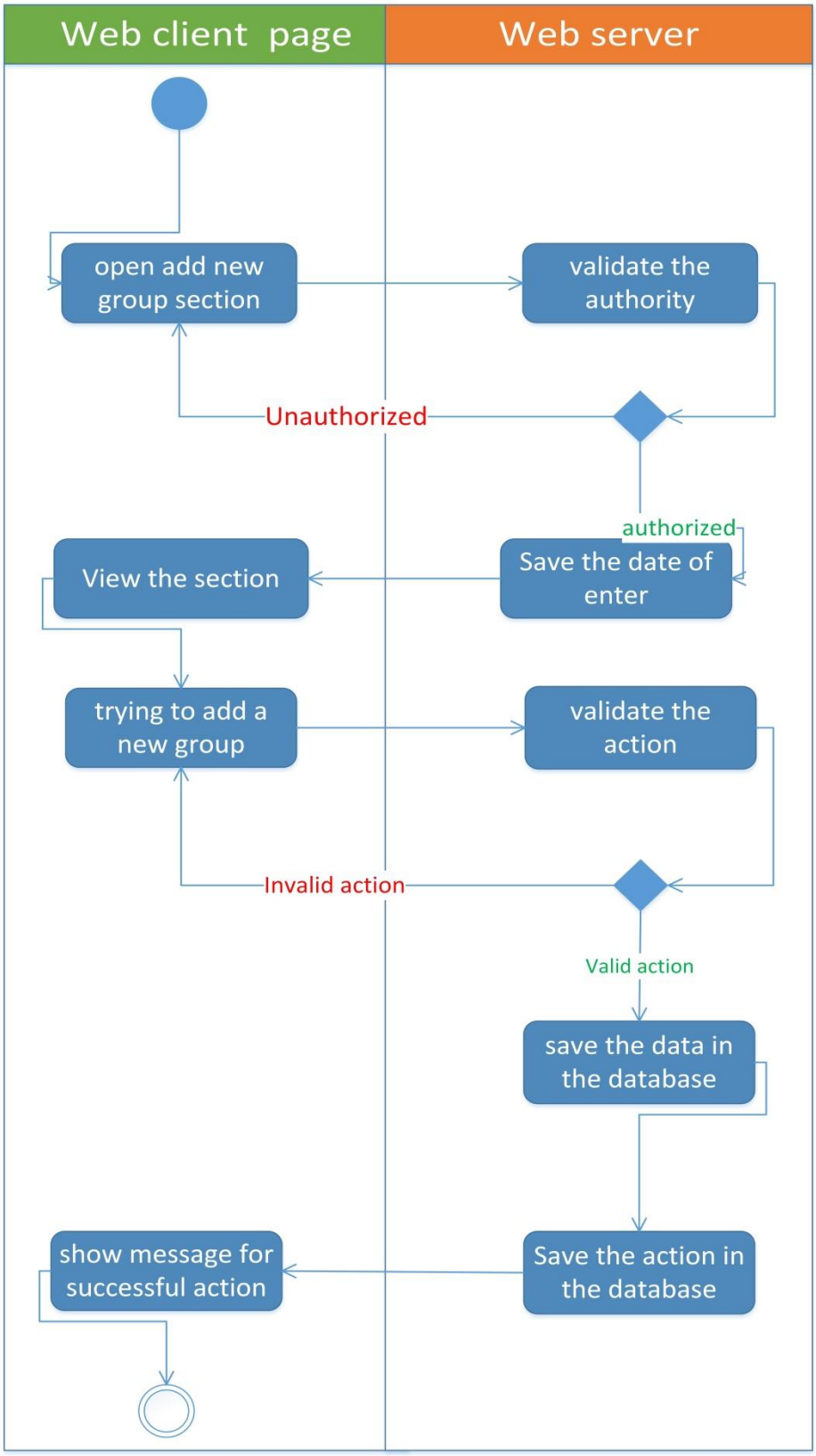

Figure 2.13: Activity Diagram For Add A New Supervisor Group

2.2.10 Add Slid Show Section

In this section admin can add and manage the slides show , he can add new slide show by provide the name of it , a picture and if there any link to access for that picture because slid show for advertisements sometimes .figure 15 the interface diagram for this section .

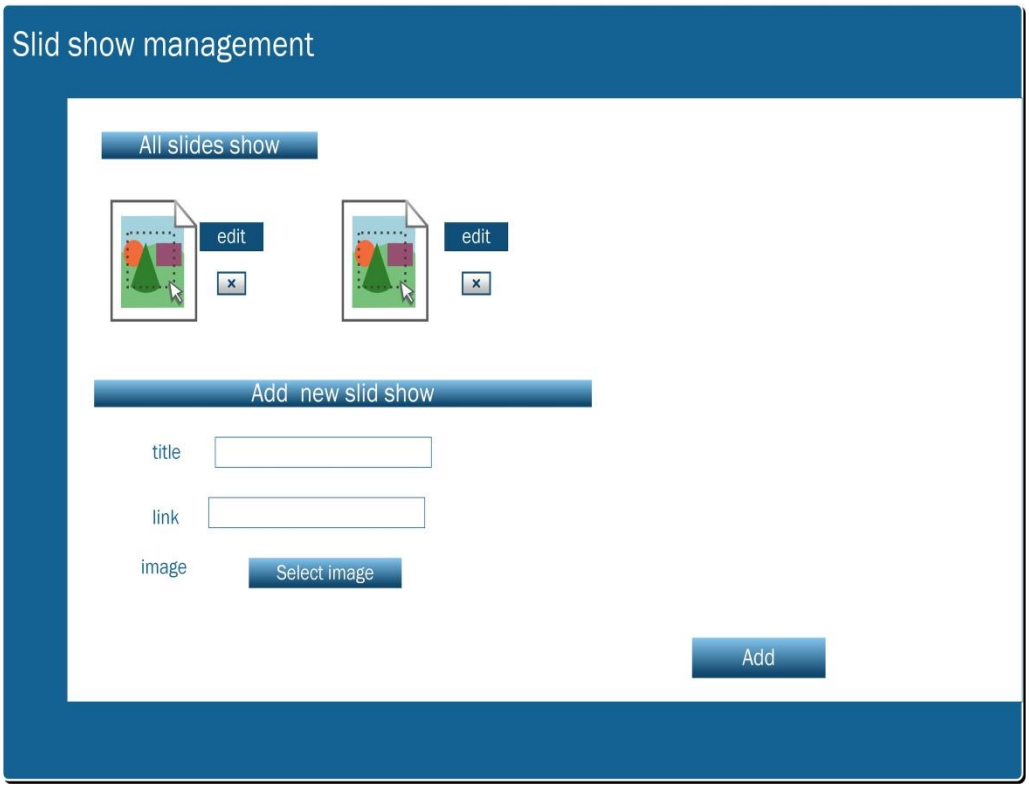

Figure 2.14: The Interface Diagram For Slides Show .

2.2.11 Add New Customer Section

Admin can add customers by providing the information that we mentioned before in registration page for user . figure 16 the interface diagram for this section . figure 16 the interface diagram for this section .

2.2.12 View Customer Section

Admin can add customers by providing the information that we mentioned before in registration page for user . figure 16 the interface diagram for this section . figure 16 the interface diagram for this section .

Admin can see all users and their information and he can edit or delete them. figure 17 the interface diagram for this section .

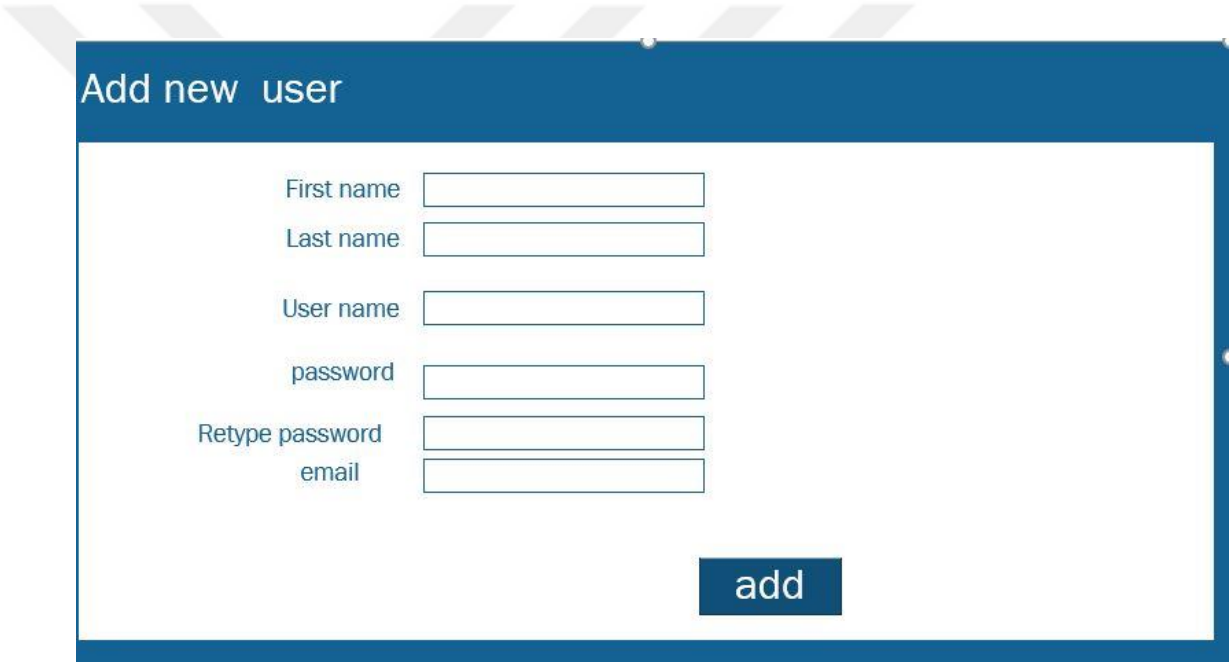

Figure 2.15: New Customers Section

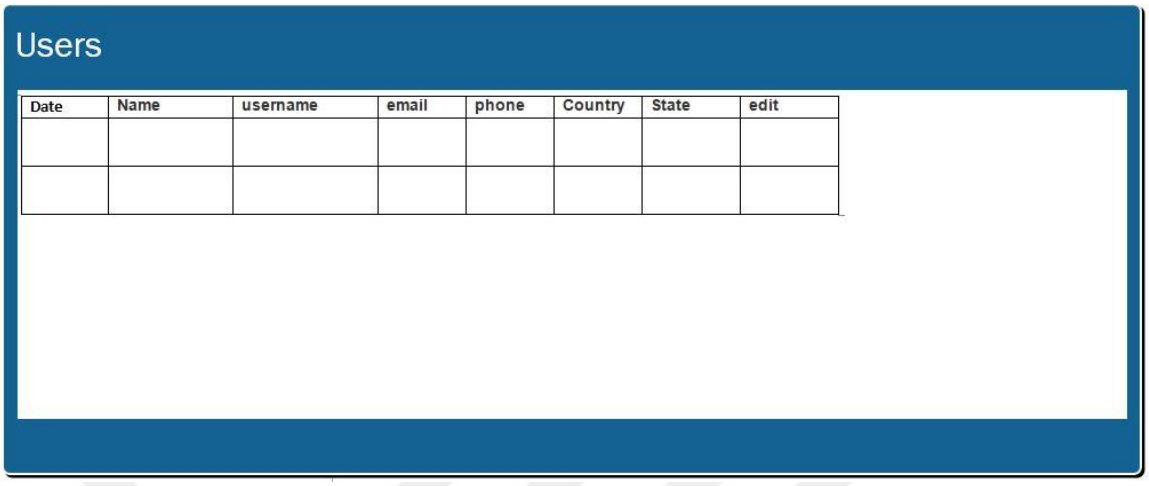

Figure 2.16: All Customers Section

2.2.13 Add News Section

Admin can add news by provide the title and the picture and content of the news by editor which can add videos links or manipulate with font size and colors .

2.2.14 New Orders Section For Products Waiting To Approve

In this part admin can see the new orders for products by users and he can see the order with the details of the products like price , link , color , size . Admin can reply to user if the can buy this product and provide him with price and other details of payment through an invoice to pay it . figure 18 the interface diagram for this section .

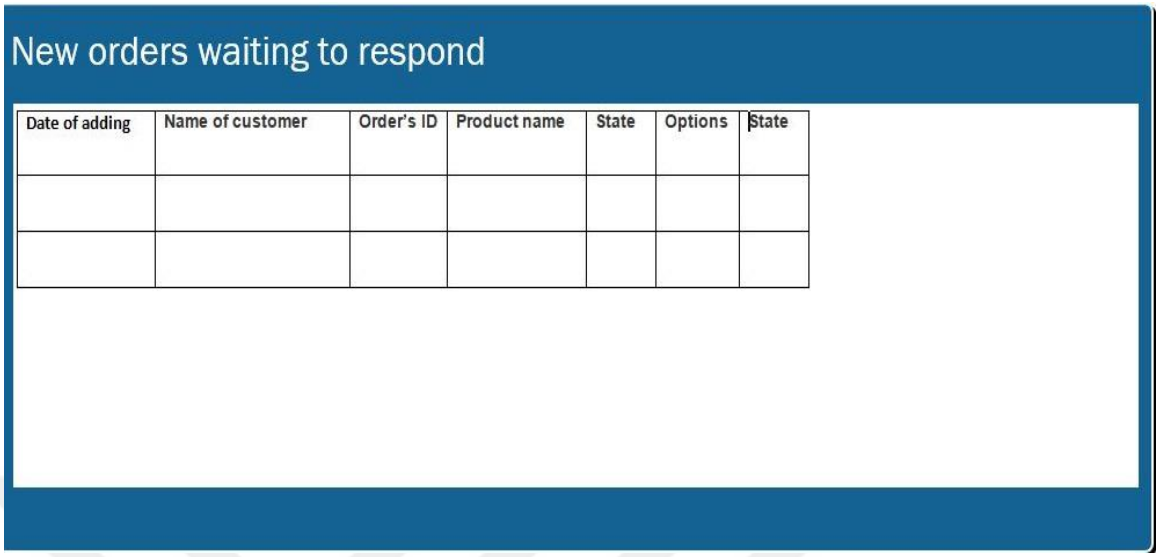

Figure 2.17: Orders Section For Products Waiting To Approve

2.2.15 Orders Waiting To Confirm The Payment

In this section admin can see orders that wait for customers payment in order to buy products for them . in this part admin will see the full details of order , the payment details and the invoice of the order and if customer have paid the money he will approve the order buying and change the state of order to charged products . figure 19 the interface diagram for this section .

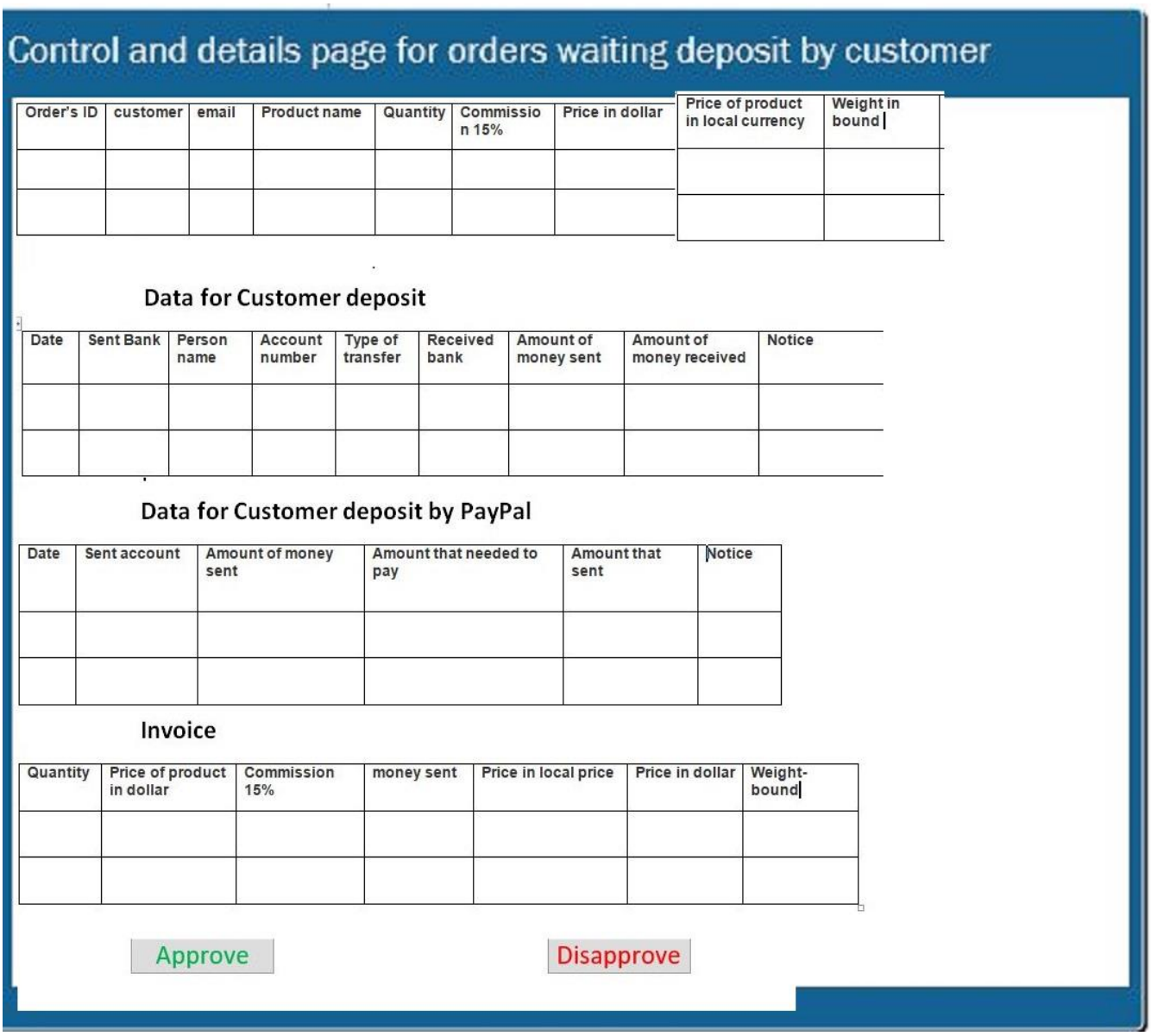

Figure 2.18: Orders Waiting To Confirm The Payment

2.2.16 Orders Charge Section

This part admin will see the orders that under charging with code of track shipment

2.2.17 Web Configuration

In this part admin will set the website information and control like the title , the meta data for search engines browsers , the state of website if off or on and the message of off state ,the banner ,and all other information. Figure 20 the interface diagram for this section .

Figure 2.19: Interface Diagram For Web Configuration

2.3 ER DIAGRAM

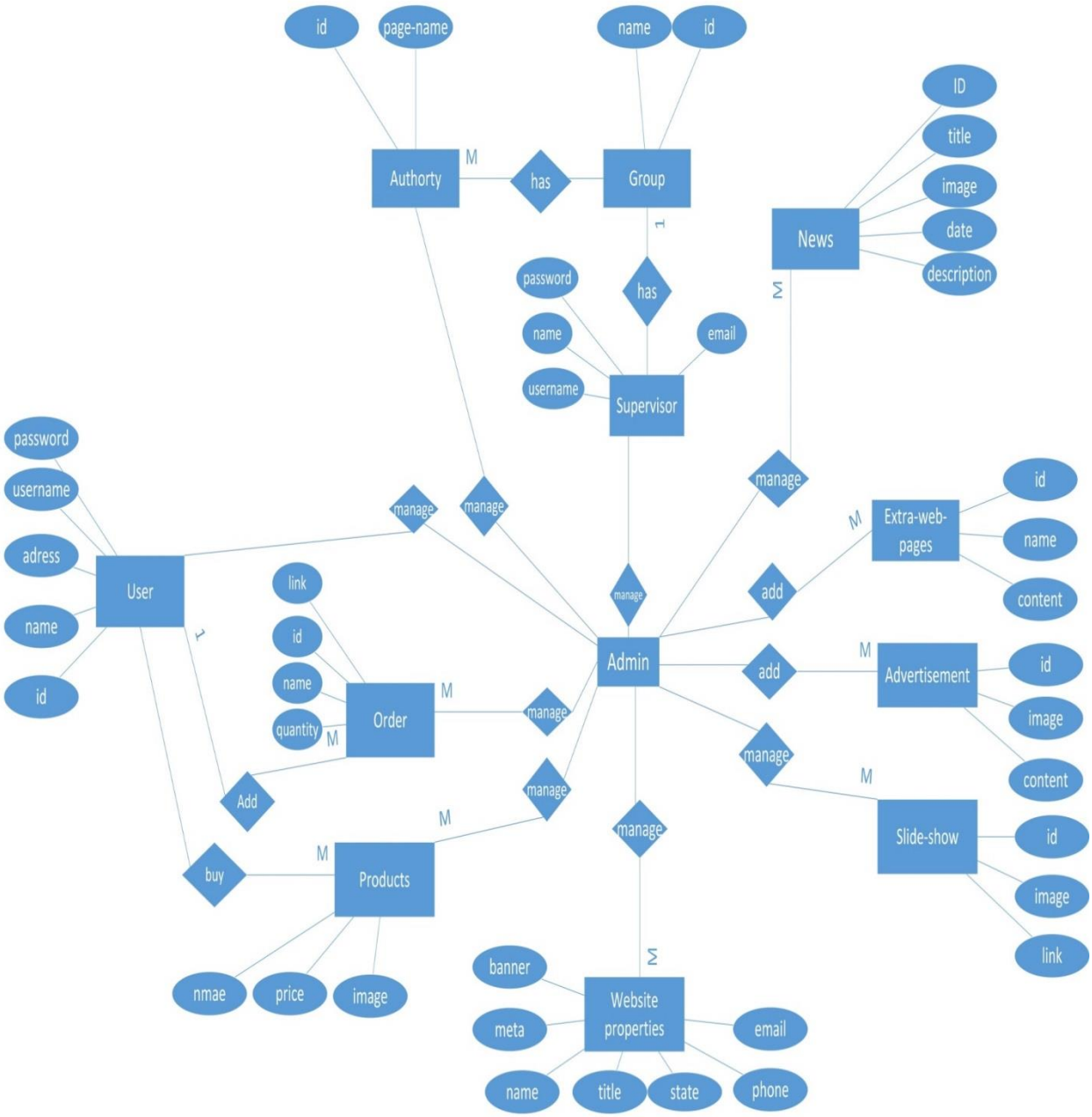

Figure 2.20: ER Diagram

2.4 SQL SECHEMA

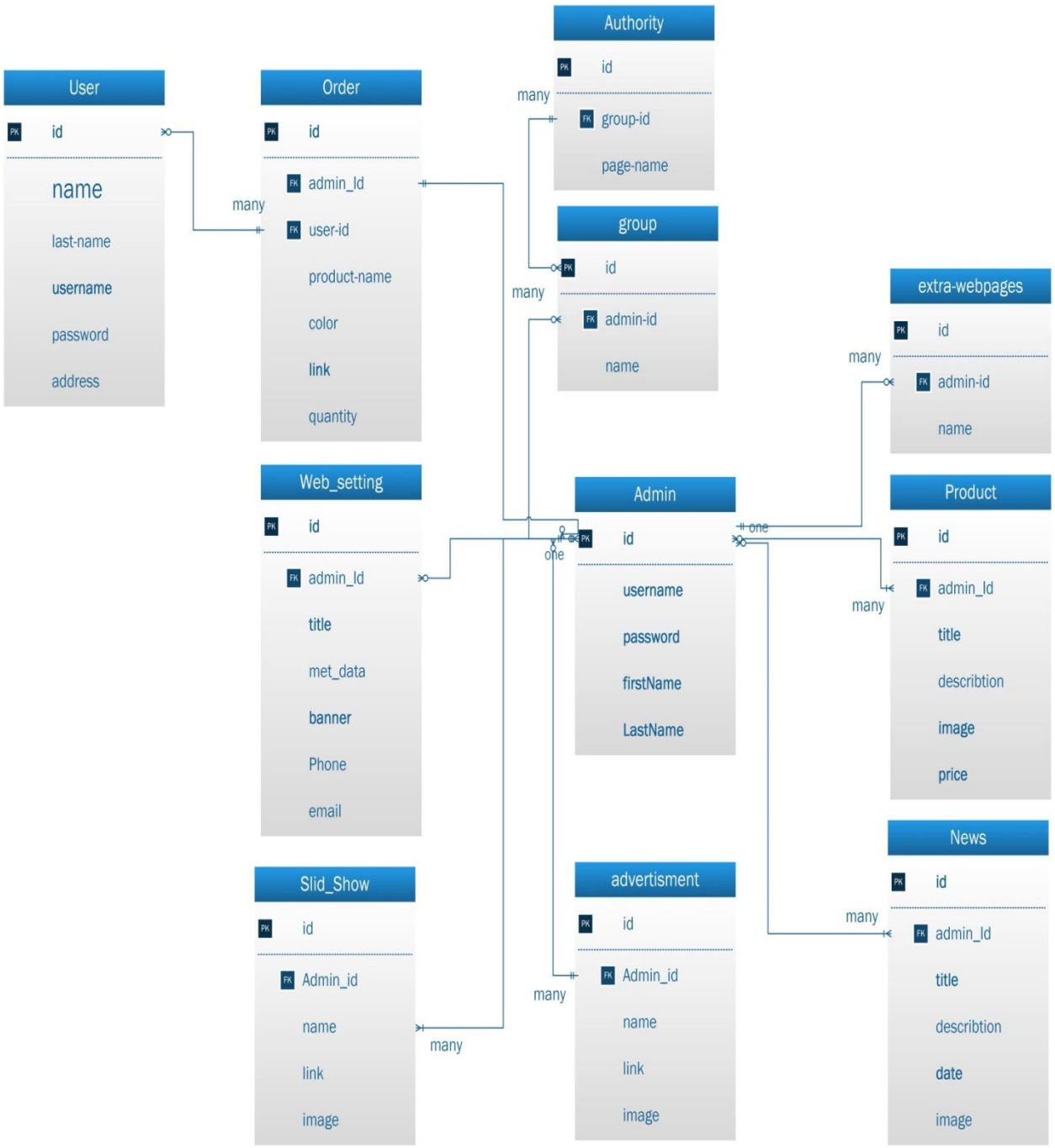

Figure 2.21: SQL Sechema

3. SYSTEM IMPLEMENTATION

3.1 INTRODUCTION

This is the longest phase of software development cycle where the process of writing a source code begins. The objective is to transform the gathered information from analysis and design into code.

3.2 MAPPING THE APPLICATION

We will build a customer database and these customers have an orders for products to buy these products for them and ship these products to their address . Visitors will be able to register in the website and put the basic information such as the name, date of birth, phone number ,email …etc. This application will implement the default Create-Retrieve- Update-Delete (CRUD) operations. In this chapter we will take the customers and their orders to show that how we can implement the theory, other parts are include in the CD .

3.3 RELATIONSHIPS ,ENTITIES AND ATTRIBUTES

At the beginning , we must define the entities of the application. From the requirements of our application , we can find the following entities and attributes:

• customers: They have a username , a name, a date of birth, mobile phone number , password , email .

• Orders: They have an identifier and a name of product , link of the product , quantity , color and size if there is any .

We need this information because according to it we will build and design the database that will store the relationships, attributes, entities.

3.4 THE MAP OF OUR APPLICATION

We need to think about the application URLs and we need to express the URLS clearly . for usability level the clear URL will be easy to navigate , and for users to easily remember the URL to come back again . for search engines will rank the website highly if they contain relevant keywords .

To fellow the requirements we will need to create routes in our application . A route is a URL and HTTP method to which the application will respond.

Method	Route	Description
GET	/	Index page
GET	/Registration	Registration page for customers
POST	/registration	To create an customer in the database
GET	/New/Order	View the page to request a new order
POST	/newOrder	To create an order in the database
PUT	/edit_order	To update on order

As I mentioned before we take the customer and orders as an example for the implementation.

3.5 WRITING THE ROUTE

To be directed to our application by the web browser we need to make a logic for some form of routing if take a look to this URL :

www.domain/laravelproject/index .com

we are using http protocols to get accessing to our laravel application that been hosted on domainName.com , the main webpage with is laravel Project/index is portion of the URL that we will use to route web request to the appropriate logic .

Routes are defined in the app/Http/routes.php file, so let's go ahead and create a route that will listen for the request we mentioned above.

```
Route::get('/', [
        'uses' => 'index@index',
        'as' => 'index'
    ]
  );
```

In this Route code we are telling the request if the URL starting with / , the request method will be get and go to control files and find the control with name app/http/controllers/index.php and use the function that name is index .the keyword 'as' mean the name of route .so we go to link www.mydomain.come we will get the index page . if we see the request to register a new user

```
Route::post('index/registerUser', [
        'uses' => 'index@register',
        'as' => 'registerUser'
] );
```

The request method is POST and the controller is index and the function is register and the name of route is register User.

3.6 CONTROL FILES

Controller files a collections of class contain number of methods also is been called as actions . usually there is a route map to a controller action and this is can greatly improve the reusability and testability of our code . When we have written the route code for index page it was like this

```
Route::get('/', [
```

```
        'uses' => 'index@index',
        'as' => 'index'
  ] );
```

We said that the route will use app/http/controllers/index.php if we look at:

app/http/controllers/index.php it will be like this :

```
public function index(){
return view('index');
}
```

This function return a view which we will explain it in next point .

3.7 RETURING VIEWS

The view objects is the most frequent objects that we will return from routes . the routes send data to views and put them in a template , and this is help us to separate the business and presentation logic in our application . the view objects in path resources/views . we have explained the main parts of application in general , now in next points we will go deeper with models and how can connects the models with each other's .

3.8 PREPARING THE DATABASE

Before we will go further in our routes and controls , we will create the models of our application , and we will prepare the basic of the database schema . first of all is to configure the database setting by open the configuration file which is with extension .env in the main folder . then we write the database name and password and the domain URL or server URL .

3.9 CREATING ELOQUENT MODEL

At the beginning of this chapter, we identified two main entities: Customers and Orders . Laravel ships with Eloquent, a ground-breaking ORM that gives you a chance to characterize these substances, map them to their comparing database tables, and cooperate with them utilizing PHP techniques, instead of crude SQL. By tradition, they are written in the solitary structure; a model named Customers will guide to the Customers table in the database.

You can likewise physically characterize the name of the database table utilizing the suitably named $table property, in the event that your table name doesn't pursue the tradition expected by Laravel:protected $table = 'custom_table_name'; The Customer model, saved at app/Customers.php, will have a hasMany relationship with the Orders model, which is defined in the following code snippet:

```
<?php namespace Eshtry;

use Illuminate\Database\Eloquent\Model;

class Customers extends Model {

  protected $fillable = ['name','date_of_birth','Customer_id'];

 public function customer() { return  $this->hasMany('App\orders');   } }
```

The $fillable array characterizes the rundown of fields that Laravel can fill by mass task, which is an advantageous method to allot credits to a model. By tradition, the segment that Laravel will use to observe the related model must be called Customer_id in the database. The Order display, application/Order.php, is characterized with the reverse hasMany relationship as pursues:

```
namespace App;

use Illuminate\Database\Eloquent\Model;

class orders extends Model

{
```

```
    public function users(){
        $this->belongsTo('App\user');
    }
}
```

Now after creating a model we should start with database .

3.10 BUILDING THE DATABASE SECHEMA

Now we must create the database schema., you don't have to write any SQL code and you will also be able to keep track of any schema changes in a version control system. To create your first migration, you need to use terminal window and enter the following command:

```
$ php artisan make:migration create_customers_table --create=customers
```

This code will create a new migration at database/migrations/. If you open the newly created file, you will discover some code that Laravel has produced for you. Movements dependably have an up() and down() strategy that characterizes the diagram changes while relocating up or down. Moving up is adjusting the database diagram (that is, including a table sometime in the not too distant future), while, moving down is the way toward fixing that pattern change. By tradition, the table and field names are written in snake_case. Additionally, the table names are written in plural structure.

Our customer plan migration will look like this:

```
6   class CreateUsersTable extends Migration
7   {
8       /**
9        * Run the migrations.
10       *
11       * @return void
12       */
13      public function up()
14      {
15          Schema::create('users', function (Blueprint $table) {
16              $table->increments('id');
17              $table->string('name');
18              $table->string('email');
19              $table->string('city');
20              $table->string('contry');
21              $table->string('mobile');
22              $table->string('password');
23              $table->string('address');
24              $table->string('state');
25              $table->string('forget_key');
26              $table->string('remember_token');
27              $table->timestamp('last_login');
28              $table->integer('group_id');
29              $table->timestamps();
30          });
31      }
32
33      /**
34       * Reverse the migrations.
35       *
36       * @return void
37       */
38      public function down()
39      {
40          Schema::drop('users');
41      }
42  }
43
```

3.11 CREATE A MASTER VIEW

Blade give us a usability to build hierarchical layouts by allowing the templates to be nested and extended . The next code screen shot is the master template that we will use for our application. We will save it as resources/views/layouts/main.blade.php.

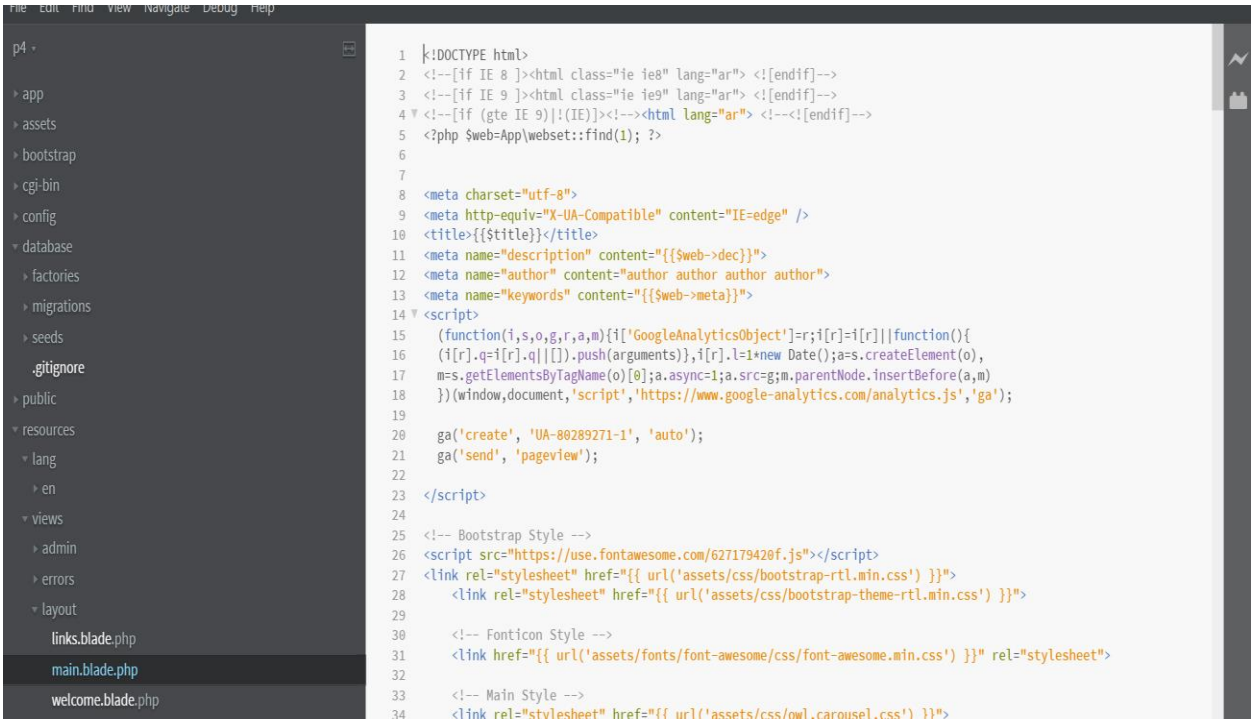

Figure 3.1: Create A Master View

we use the master view to put all necessary files and links that we need in all pages. As we see the CSS and JavaScript files are linked and used in the master view , now we should extend these files in pages that we need by invoke the master view using the following code:

@extends('layouts.main')

This code will extend the main file and view it in the page that we invoke and this will make codes more usable and re-use these files in different pages .

3.12 BUILD THE REGISTRATION SYSTEM UISNG MVC PATTERN

Now after we have shown how the MVC works in general terms , we are going to show how registration system for customers works with larvae MVC pattern in our application to extend the explanation and to go deeper with functions and models that may developers need to build any system

3.12.1 Creating Route To View Registration's Page

Firstly , We have to create a rout to view registration page for customers in order to use the system , the route should be with GET method , the following code for route :

```
Route::get('/registerv', [
        'uses' => 'index@registerV',
        'as' => 'registerV'
    ]

);
```

As we see we are going to use control files that its name index and invoke the function register.

3.12.2 Creating Control To View Registration's Page

Now we need to create a control to view the registration page , the following code to view the blade files for viewing

```
public function registerV(){

    return view('registerNewUser');


}
```

This is a public function that return a view files that named register New User.

3.12.3 Creating View for Registration's Page

Now we are going to use HTML codes and CSS and we must extend the master view for the necessary files that we need , the following code is for view :

```
resources/views/registerNewUser.blade.php (p4) - Brackets
le Edit Find View Navigate Debug Help
Working Files
main.blade.php
index.php
p4
edit_user.blade.php
edite_order.blade.php
index.blade.php
login_Users.blade.php
mainPages.blade.php
newPassword.blade.php
news.blade.php
newsPages.blade.php
orders_added.blade.php
orders_bill.blade.php
orders_box.blade.php
orders_end.blade.php
passwordReset.blade.php
payment_bank.blade.php
payment_methods.blade.php
payment_paypal.blade.php
print.blade.php
quesions.blade.php
registerNewUser.blade.php
shop.blade.php
Single.blade.PHP

1  @extends('layout.main')
2
3
4
5    @section('content')
6  <?php $web=App\webset::find(1); ?>
7        <?php View::share('title',$web->name.' Registeration  ' ); ?>
8        <?php $con=App\country::all(); ?>
9
10
11       <script type="text/javascript">
12           //<![CDATA[
13           // array of possible countries in the same order as they appear in the country selection list
14           var countryLists = new Array(4)
15           countryLists["empty"] = ["الفرع"];
16           @foreach($con as $cons)
17           <?php $city=\App\city::where('country_id',$cons->id)->get(); ?>
18                   countryLists["{{$cons->name}}"] = [ @foreach($city as $citys) "{{$citys->name}}" ,
                     @endforeach  ];
19
20           @endforeach
21           /* CountryChange() is called from the onchange event of a select element.
22            * param selectObj - the select object which fired the on change event.
23            */
24           function countryChange(selectObj) {
25               // get the index of the selected option
26               var idx = selectObj.selectedIndex;
27               // get the value of the selected option
28               var which = selectObj.options[idx].value;
29               // use the selected option value to retrieve the list of items from the countryLists array
30               cList = countryLists[which];
31               // get the country select element via its known id
32               var cSelect = document.getElementById("country");
33               // remove the current options from the country select
34               var len=cSelect.options.length;
35               while (cSelect.options.length > 0) {
36                   cSelect.remove(0);
37               }
38               var newOption;
39               // create new options
Line 92, Column 1 -- 269 Lines    INS    UTF-8    PHP    Spaces: 4
```

Figure 3.2: HTML Codes And CSS For Registration's Page

As we see that we have extend layout.main and we used section to delimit the blocks of content that are going to be injected into the master template.

Now we need the HTML form to send data to be saved in database . so again we will use route to send these data to control and then to be saved in the database .

The following HTML to create the form for registration .

```
                    <div class="title">Register new user  </div>

                    <form class="form-horizontal" action="{{route('registerUser')}}" method="post"
                    onsubmit="return checkForm(this)">
                        <div class="form-group">
                            <label for="inputEmail3" class="col-sm-2 control-label">user name  </label>
                            <div class="col-sm-4">
                                <input name="name" class="form-control" id="inputEmail3" placeholder=""
                                type="text">
                            </div><!-- end col-sm-4 -->
                        </div><!-- end form-group -->
                        <div class="form-group">
                            <label for="inputEmail3" class="col-sm-2 control-label"> Email </label>
                            <div class="col-sm-4">
                                <input name="email" class="form-control" id="inputEmail3" placeholder="email"
                                type="email">
                            </div><!-- end col-sm-4 -->
                        </div><!-- end form-group -->
                        <div class="form-group">
                            <label for="inputEmail3" class="col-sm-2 control-label"> mobile </label>
                            <div class="col-sm-4">
                                <input name="mobile" class="form-control" id="inputEmail3" placeholder="mobile"
                                type="text">
                            </div><!-- end col-sm-4 -->
                        </div><!-- end form-group -->
<div class="form-group">
                            <label for="inputEmail3" class="col-sm-2 control-label">  currency </label>
                                 <div class="col-sm-4">

<select class="" name="currence" id="continent">
                      <option value="empty">choose   </option>
```

As we see on second line we tell the HTML code to use the route name registerUser with POST method .

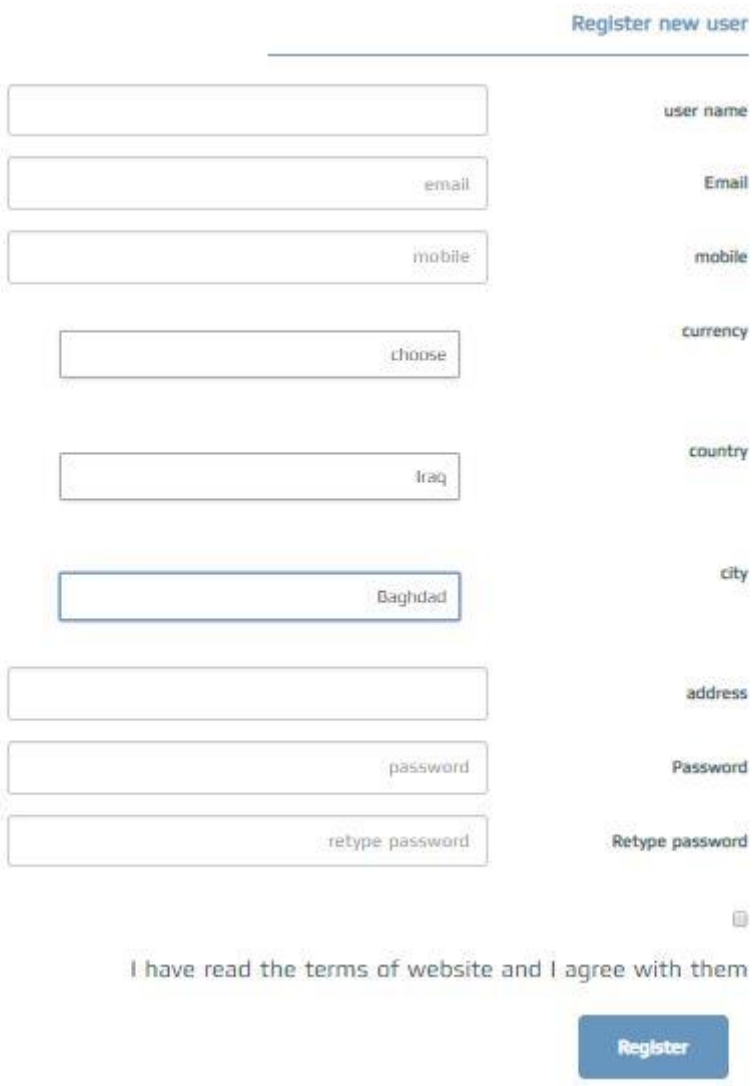

Figure 3.3: Register new user

3.12.4 Creating Route To Save Registration's Data

From the registration page the data flow start to be sent the route with POST method , the following code for route :

```
Route::post('index/registerUser', [
        'uses' => 'index@register',
        'as' => 'registerUser'
    ]

);
```

Now as we told before that we will use control files and function register will save data to database .

3.12.5 Creating Control To Save Registration's data

Now we should create control to save registration data on database .

The following code for register function :

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

    $validator = Validator::make($request->all(), [
        'name' => 'required|unique:users',
        'email' => 'required|unique:users',
        'password' => 'required|confirmed',
        'contry' => 'required',
        'city' => 'required',
        'agree_check' => 'numeric',
        'mobile' => 'required|unique:users', ]);
    if ($validator->fails()) {
        return redirect()->back()
            ->withErrors($validator)
            ->withInput();
    }
     if(!isset($request['MyCheckbox'])){
         Session::flash('sucess','you have to agree to terms');
         return redirect()->back();
     }

    $user= new user();
     $email=$request['email'];
     $wagtail=webset::find(1);

      $user->group_id=$wagtail->newUsersState;
       $user->name=$request['name'];
       $user->currince_id= $request['currence'];
       $user->password=bcrypt($request['password']);
    $user->email=$request['email'];
    $user->mobile=$request['mobile'];
    $user->contry=$request['contry'];
    $user->city=$request['city'];
    $user->address=$request['adress'];

          if($user->save()){
              Session::flash('sucess','we registered you please log in ');
              return redirect()->route('loginuser');
          } } }
```

Now we should explain these parts of code . firstly the Request class , to obtain an instance of the current HTTP request via dependency injection, you should type-hint the Illuminate\Http\Request class on your controller method. The incoming request instance will automatically be injected by the service container. So the function is expected to receive an http request from route and the class Request is handling this request . the $request is an object from that class , actually it’s an array of objects .

The next piece of code is to validate the data as we see in the second line we call an static function (make) from class Validator , Laravel provides different approaches to validate your application's incoming data. The first argument passed to the make method is the data under validation. The second argument is the validation rules that should be applied to the data.

Next step is to create an object from model User which is the customer . $user is an object from class model which is the database model , as we see $user→name=$request['name'] , we assign the name from http request to object name from database after that we just invoke the function save() then flash the message that we have successful registered the customer in the system now he can login .

3.12.6 How To Make a Login

Now, after we have registered the customer in the system , we need to make a login to the profile of customer .first as we mentioned before , we need to create a route , a view and control .the route code to view the login page is here :

```
Route::get('/loginuser', [
        'uses' => 'index@loginv',
        'as' => 'loginuser'
    ]
);
```

```
            <div class="title"><h3>Log in<h3></div>
            <form class="form-horizontal" action="{{route('loginUserIndex')}}" method="post">
                <div class="form-group">
                    <div class="col-sm-4">
                        <input name="email" class="form-control" id="inputEmail3" placeholder="Email "
                        type="text">
                    </div>
                    <label for="inputEmail3" class="col-sm-2 control-label"><h3>Email </h3></label>

                </div>
                <div class="form-group">
                    <div class="col-sm-4">
                        <input name="_token" type="hidden" value="{{ Session::token() }}">
                        <input name="password" class="form-control" id="inputPassword3" placeholder="password
                        " type="password">
                    </div>
                    <label for="inputPassword3" class="col-sm-2 control-label"> <h3>Password</h3></label>

                </div>

                <div class="form-group">
                    <div class="col-sm-2">
                        <button type="submit" name="submit"><h3>Login </h3> </button>
                    </div>

                </div>









    </form>
```

And the view page is here :

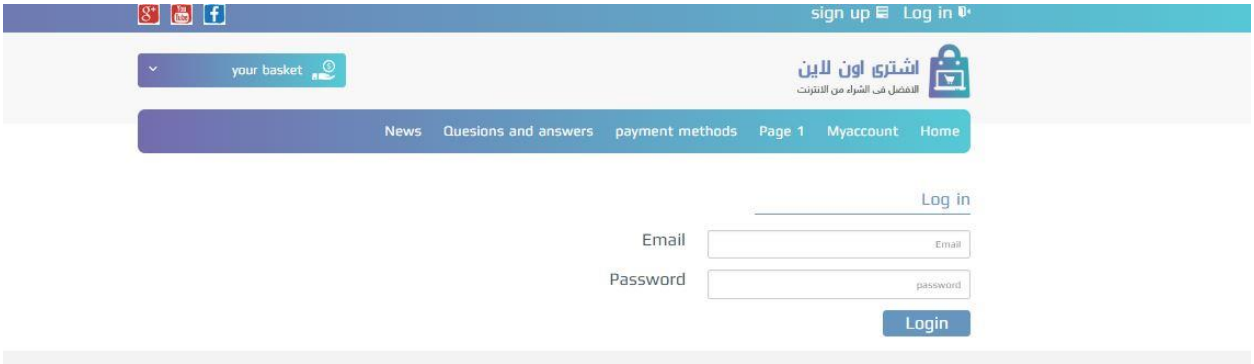

Figure 3.4: View Page

And the route to make a login and link the view with control is here and it's a post method http :

```
Route::post('/loginUserIndex', [
        'uses' => 'index@loginUserIndex',
        'as' => 'loginUserIndex'
    ]


);
```

And the control function that make login is here

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


            if(Auth::guard('user')->attempt(['email'=>$request['email'],'password'=>$request['password']])){

                $user= Auth::guard('user')->user();
                $users=user::find($user->id);
                $users->last_login=carbon::now();
                $users->save();

               return redirect()->route('client_place') ;


                            } else {

                 return redirect()->route('loginuser')->with(['fail'=>" غير صحيحه    البريد الالكتروني او كلمة المرور"]);


  }
 }
}
```

In the next line if authentication was successful the application will direct us to the client place or customer area , otherwise the customer will direct back to the login page with error message .

3.12.7 How a Customer Make an Order

Now after customer has made the login to the system , he will make an order to buy some products . the some procedure every time we should fellow to make any process , first the route to access the view of the page that we can make an order through it , the next code show the route to view the page for orders .

```
Route::get('/index/client_place/{state?}/{id?}/{other?}', [
        'uses' => 'index@client_place',
        'as' => 'client_place'
    ]
);
```

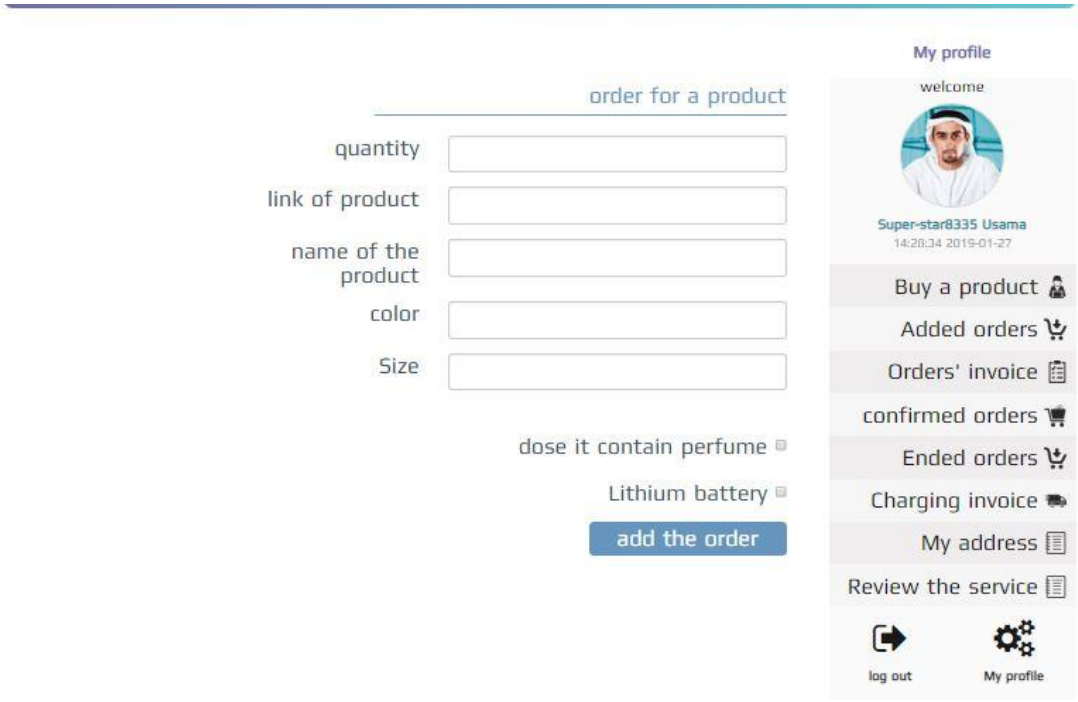

Figure 3.5: Customer Make An Order

now the route that used to add order to the database ,

```
Route::post('index/add_to_cart', [
        'uses' => 'index@add_to_cart',
        'as' => 'add_to_cart'
    ]

);
```

Now we will see the control function (add_to_cart) that save the order data to database :

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

        $validator = Validator::make($request->all(), [
            'name' => 'required',
            'linke' => 'required',
//            'color' => 'required',
            'quantity' => 'required',
//            'size' => 'required',


        ]);

        if ($validator->fails()) {
            return redirect()->back()
                ->withErrors($validator)
                ->withInput();
        }
        $user= new orders();
        $user->name=$request['name'];
        $user->size=$request['size'];
        $user->link=$request['linke'];
        $user->color=$request['color'];
        $user->quantity=$request['quantity'];
        $user->state_oreder='cart';
        if(isset($request['checkbox1'])){
            $user->opition1=$request['checkbox1'];
        }
        if(isset($request['checkbox2'])){

            $user->opition2=$request['checkbox2'];
        }
        Auth::guard('user')->user()->orders()->save($user);
        $this->logs($request['name'].' he added product to basket: ');
                return redirect()->route('client_place',['state'=>'cart']);


        }

```

We have explained the role of Request and validator classes , we will focus on the relationship between customers and the orders . if we look closely at the line ,

Auth::guard('user')->user()->orders()->save($user);

We used the Auth::guard(‘user’)->user to refer to the user that login and make an order , and the order() function is the relationship between the customer and the order this function is located in the model file user and this function is look like :

```
public function orders(){

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

  }
```

hasMany relationship , that’s mean the user can make many orders and if you wonder what is the benefit of this function , it’s for database relationships , now after the customer submit and order the customer id will be saved automatically in the table orders .

3.12.8 Display The Orders In The Admin Side

Now we will display these orders by admin , as usually we must define the routes and controls ,the following code for display the view of the orders ,

```
Route::get('/admin/ordersView/{state?}/{id?}', [
        'uses'=>'admin@ordersView',
        'as'=>'ordersView'
    ]

);
```

The following the control codes to display the order ,

```
public function ordersView($state='', $id=''){

  |

        $id=orders::all();

        return view('admin.ordersProductsView',['all'=>$id]);

    }
```

Now we put into the variable $id all orders that exist in the database by using the model orders and call the static function all(). Then we return the view ordersProductView in the admin folder and we just pass or send the variable $id to the view under ‘all’ name now let us look at view code :

```
                                        <th> Date</th>
                                        <th>order number</th>
                                        <th>customer name</th>
                                        <th>product name</th>
                                        <th>quantity </th>
                                        <th>order state</th>
                                        <th>options </th>
                                    </tr>
                                </thead>
                                <tbody>
                                 <?php use App\catgory; ?>
                                  <?php use App\products; ?>
                                   @foreach($all->sortByDesc('id') as $alls)
                                    <?php if($alls->state_oreder=='cart'){

}else{ ?>

                                    <tr>
                                        <td class="text-center"><label class="csscheckbox csscheckbox-
                                        primary"><input type="checkbox"><span></span></label></td>
                                        <?php $find=\App\user::find($alls->user_id); ?>
                                        <td>{{$alls->created_at}}</td>
                                        <td>@if($find!=null) <?php echo $alls->id; ?> @endif</td>
                                        <td>@if($find!=null) <?php echo $find->name; ?> @endif</td>
                                        <td>{{$alls->name}}</td>
                                         <td>{{$alls->quantity}}</td>
                                      <td>{{$alls->state}} </td>
                                        <td><a href="{{route('ordersView',['state'=>'update','id'=>$alls-
                                        >id])}}" data-toggle="tooltip" title="" class="btn btn-effect-
                                        ripple btn-sm btn-success" style="overflow: hidden; position:
                                        relative;" data-original-title=" see order and edit"><i class="hi
                                        hi-folder-open"></i></a></td>

                                    </tr>
                                    <?php } ?>
                                    @endforeach

                                </tbody>
                            </table>
```

If we look at the following code :

@foreach($all->sortByDesc('id') as $alls)

The $all variable that we have sent it from control ,this is an array of objects , now we extract the element of this array using foreach loop and functions that arrange these elements and we refer to each object that present the database rows as we see the information from database and we used another functions to get data from another models like users model as we see the code :

<?php $find=\App\user::find($alls->user_id); ?>

We are find function that will find the user by his id that saved in the orders.

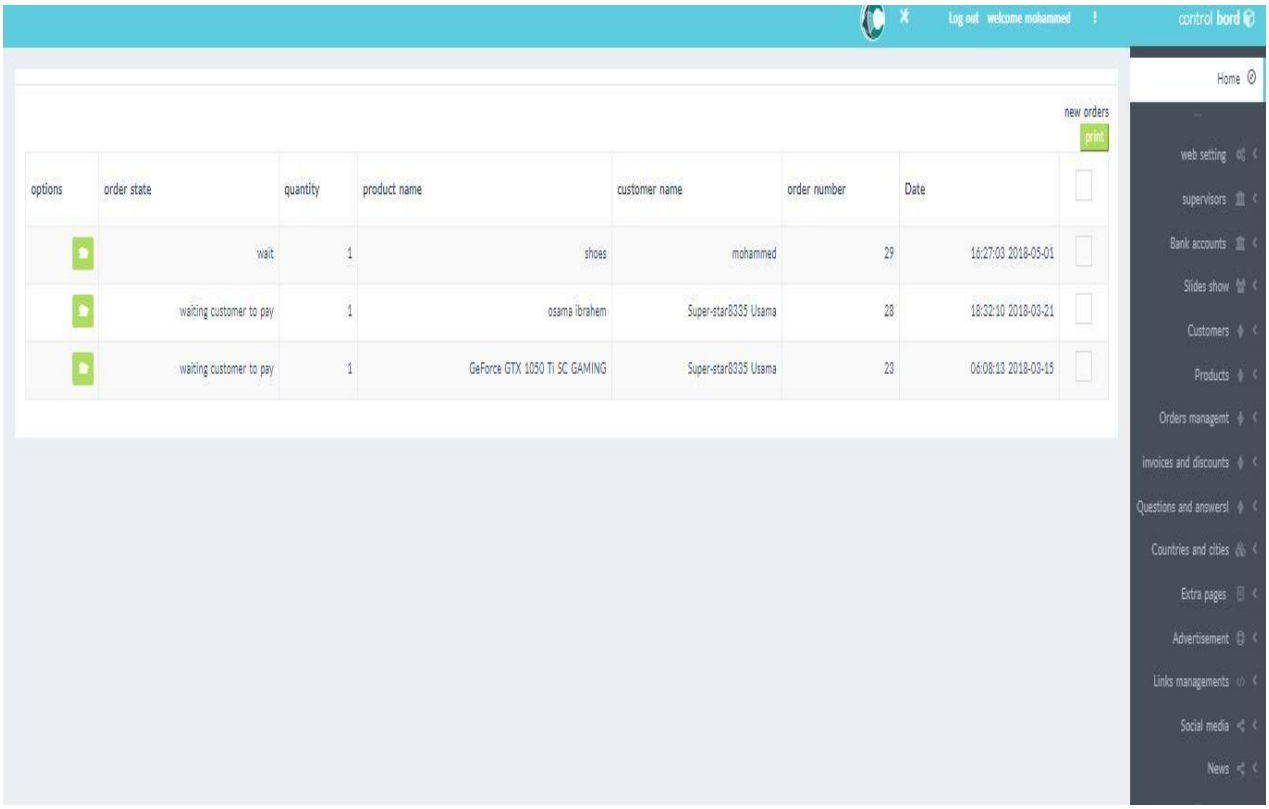

Figure 3.6: Function To Find User By ID

3.13 SUMMARY

In this chapter we have convert the design part to real application by using MVC pattern . through the implementation of our application we could find out that the MVC technology has been made the development rapid and more usable with OOP structure .

3.14 CONTRIBUTIONS

3.14.1 Security

in our application we have almost five security futures that we applied as following :

1. In the user registration we make sure the identity of the user through his phone number or email . in this future the server send a message or an email to the user contain a code verification . the user will be asked to inter the code after registration otherwise the registration will not be completed . If the user will forget the password of his account , he will follow the same procedure in above .
2. In the administration part we have groups for supervisors to limit their accessibility inside the web site .
3. We track every movement for supervisors inside the website by using a functions that register all their movements just in case a problem may happened with some customers .
4. All passwords are encrypted using OpenSSL and the AES-256-CBC cipher
5. PDO parameter binding to protect our application against SQL injection attacks

3.14.2 Shopping Using GPS

Collecting GPS data is becoming quite pervasive. Using the knowledge of where a customer goes, which path she travels, and how much time she spends at various locations can improve the quality of customer interactions and types of marketing offers, and increase the likelihood that she'll redeem an offer.

GPS data is a time-series of an individual's position information in terms of latitude and longitude. This data provides a wealth of hidden predictive information about your customers' activity that could be used to improve marketing decisions.

In our application we can fellow the customers that prefer to buy products that may be available in our shop . we can offer to customers some products that may interest them . we send notification to their emails and phones that exist in our database .

4. CONCLUSION

The thesis objectives were to analyze, design and implement a website to buy products for customers from different global shops using MVC pattern technology.

Within the Theory chapter, a brief description of used technologies such as Entity Framework, Laravel php , and so forth were introduced. The Analysis and Design chapter clear up the website basic features together with initial description of mentioned subcomponents. Additionally, this chapter presents gathered requirements based on not only customer's expectations but also the needs for an application and management requirements , also this chapter reflects these requirements into cohesive blocks consisting of data model, server and client side application architecture, subcomponent design and finally the application design. The last thesis objective was fulfilled via Implementation chapter which essentially follows the same structure as previous chapter. Through the implementation of the system we can figure out the following points :

- The requirement gathering and design make the development process rapid and help the developers to avoid the flaws and understand the business easily .
- Using the OOP pattern make the codes more efficient and easier to reuse the functions and that make the development faster .
- Using the MVC to develop an applications is more securable because of the built classes such as authentication, routing, database managing, sending emails Class auto loading.

By the time the thesis is written, the first version of the website was successfully deployed. According to customer feedback, this solution met with success in all aspects. Thus, it can be presumed that implemented solution suits all defined requirements.

On the basis of previous information it can be claimed that all thesis objectives are considered as successfully accomplished.

4.1 FUTURE WORK

- This project can be a perfect if we make it for mobile platforms such as android and ISO it can be easily apply this idea as a mobile application.
- The login system can be by face detection algorithms but we don't know what kind of challenges will we have because of the web environments and the obstacles of servers to handle such issues.
- The payment methods will be on visa and credit cards .
- Add more futures to fellow the shipments of customers by the ID of shipment .
- Add more API for global shops such as amazon.
- Add more futures related to products that do not available in customer's countries , for example the customer will see a suggestions for a new products that available in our websites but do not available in others.

REFERENCES

[1] Yuan Jing, Cao Yaoqin, Wangb Wenhai, Li Jidong. Net Based Office Automation System Based on PHP Technology [J]. Microcomputer development. 2003.8:61-63.

[2] Su Chengjun, He Pilian. Applied Research on Developing Web Database with PHP [J]. Computer engineering, 2000.9:184-185

[3] Alfat, Lathifah, Aris Triwiyatno, and R. Rizal Isnanto. "Sentinel web: Implementation of Laravel framework in web based temperature and humidity monitoring system." 2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). IEEE, 2015.

[4] Anif, Muhammad, Arya Dentha, and H. W. S. Sindung. "Designing internship monitoring system web based with Laravel framework." 2017 IEEE International Conference on Communication, Networks and Satellite (Comnetsat). IEEE, 2017.

[5] He, Ren Yu. "Design and implementation of web based on Laravel framework." 2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014). Atlantis Press, 2015.

[6] Rees, Dayle. "Laravel: Code Bright." (2012).

[7] Das, Ripunjit, and Lakshmi Prasad Saikia. "Comparison of Procedural PHP with Codeigniter and Laravel Framework." International Journal of Current Trends in Engineering & Research 2.6 (2016): 42-8.

[8] Bean, Martin. Laravel 5 essentials. Packt Publishing Ltd, 2015.

[9] Verma, Archit. "MVC Architecture: A Comparative Study Between Ruby on Rails and Laravel." Indian Journal of Computer Science and Engineering (IJCSE) 5.5 (2014): 196-198.

[10] Gupta P, Govil MC (2010) MVC Design pattern for the multi framework distributed applications using XML, spring and struts framework. International Journal on Computer Science and Engineering.

[11] Shu-qiang H, Huan-ming Z (2008) Research on improved MVC design pattern based on struts and XSL. 2008 International Symposium on Information Science and Engineering.

[12] Selfa DM, Carrillo M, Boone MDR (2006) A database and web application based on MVC architecture. 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06).

[13] Ahmed M, Uddin MM, Azad MS, Haseeb S (2010) MySQL performance analysis on a limited resource server. SpringSim 10 Proceedings of the 2010 Spring Simulation Multiconference.

[14] Anley C (2002) Advanced SQL injection in SQL server applications. Next Generation Security Software Ltd, UK.

[15] Kroenke DM, Auer DJ. Database Concepts 7th Edition Online Appendix E SQL Views, SQL/PSM and Importing Data.

[16] Benson C, Prove MM, Mzourek J (2004) Professional usability in open source projects. CHI '04 Extended Abstracts on Human Factors in Computing Systems.

[17] Moulton B, Chaczko Z, Karatovic M (2009) International journal of digital content technology & amp; its Applications GYEONGJU. The International Association for Information, Culture, Human and Industry Technology.

[18] Doyle B, Lopes CV (2008) Survey of technologies for web application development. University of California, USA.

[19] Stratmann E, Ousterhout J, Madan S (2011) Integrating long polling with an MVC framework. Stanford University.