**ERSM (CRM)**

**Summary**

In this CRM :-

* We will create users as admins & agents.
* We will manage users and the data will be stored in users table.
* Based on the type of user (Admin or Agent) user can access different pages.

**Requirement**

* PHP( Version 8.1 )
* Laravel
* My Sql
* Wamp Server
* VS Code Editor
* Composer

**API DETAILS**

$url = "http://easygovoice.com/csc\_url\_api?api\_key=53f714f360567740&pass=E51Frwd7By&empid=" . $email . "&AgentNo=" . $pri\_no . "&CustomerNo=" . $mobile;

**CRM(FLOW)**

**Login:-**

* Enter your User Id/Email and Password to get a successful login .
* You will get redirected to the Dashboard.
* Based on the type of user (Admin or Agent) user can access different pages.

**Admin can access pages :-**

* Dashboard -

Routes:-

use App\Http\Controllers\UserController;

Route::get('/dashboard', [UserController::class, 'show'])->name('dashboard');

Controller:-

use App\Models\ClientData;

use Exception;

use Illuminate\Support\Facades\DB;

return view('dashboard');

Model:-

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class ClientData extends Model

{

use HasFactory;

    protected $table = 'client\_data';

    protected $fillable = ['vin\_no','parent\_id','customer\_name','contact\_no',

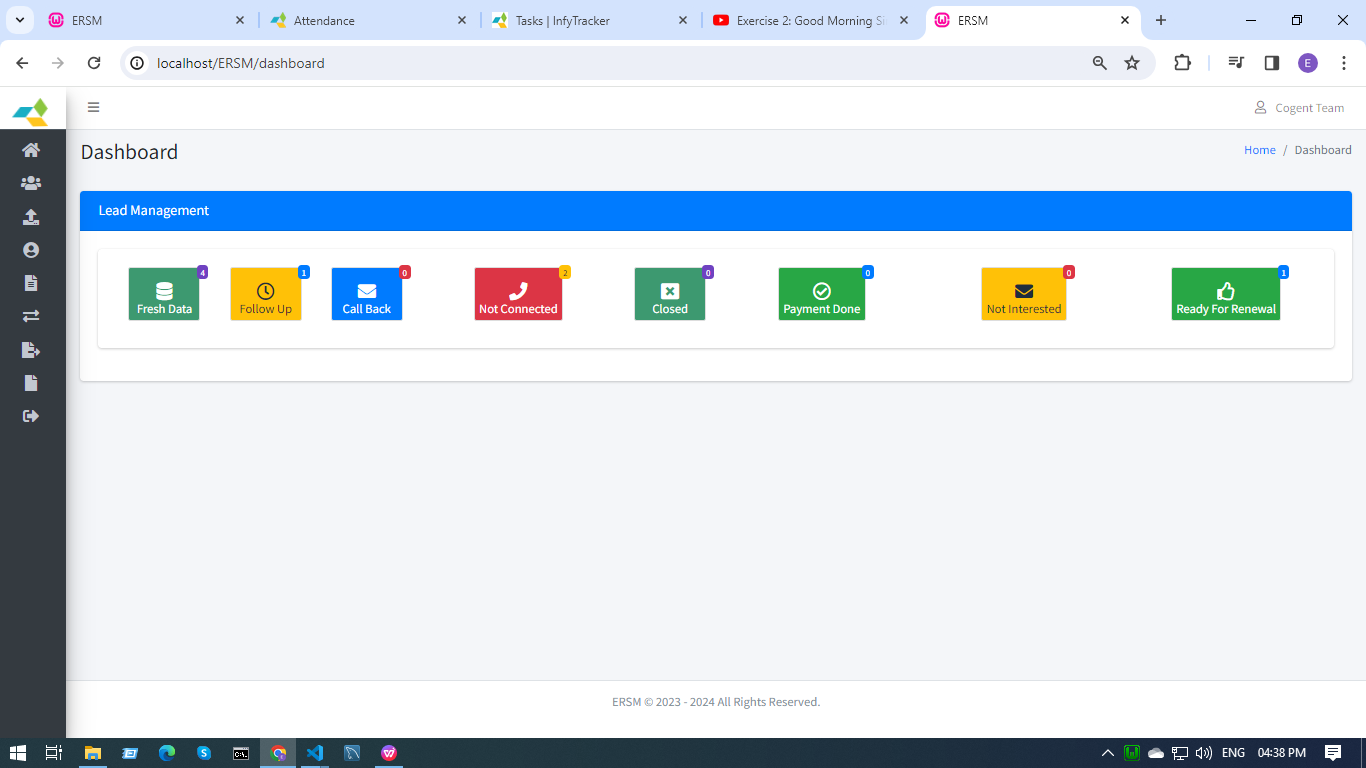
        'usage\_category','plant\_code','dealer\_name','dealer\_state','bill\_to\_party\_name',

        'device\_type','region','customer\_category','agent\_id','created\_by',];

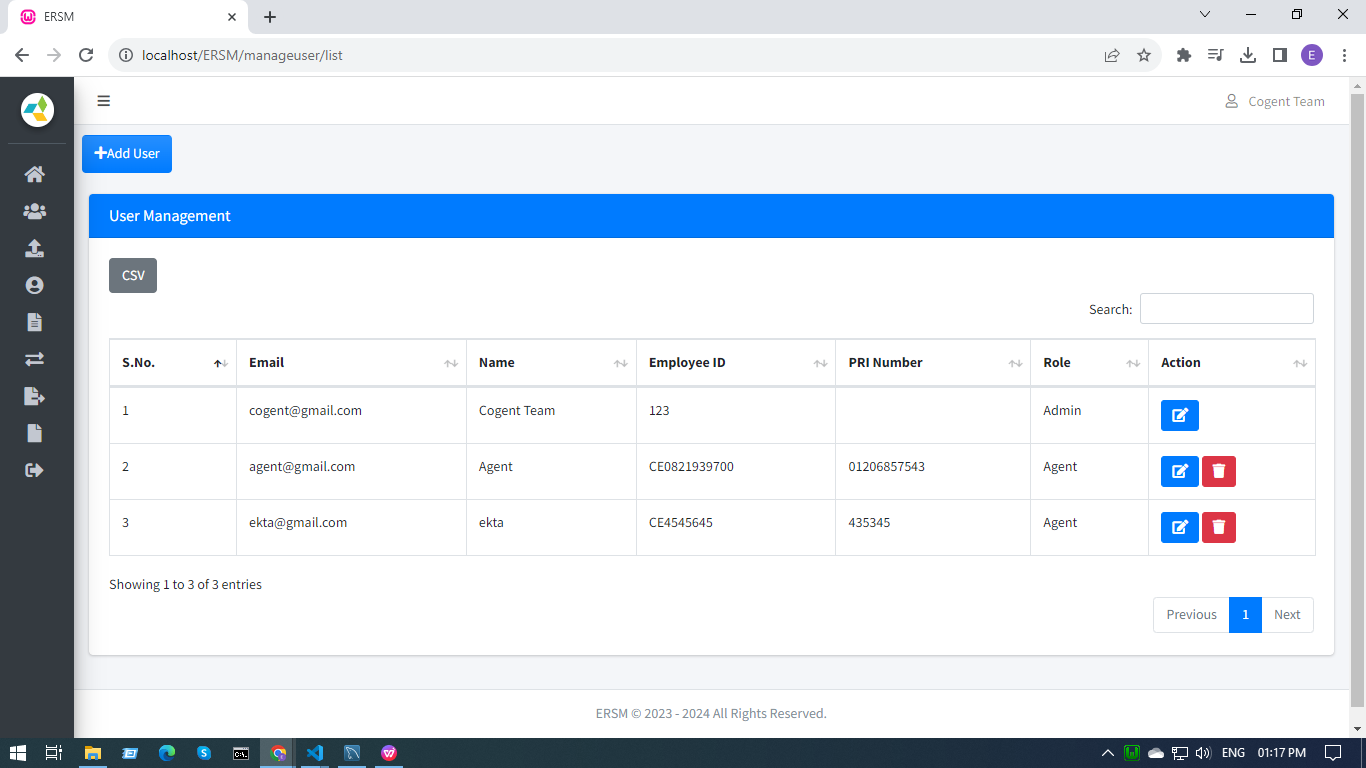
}

Features:-

* The dashboard includes the Lead management section.
* Admin can see all the leads of various categories with the no. of total lead count.
* The count of the Leads is coming through the helper file - function leadCount()
* These tabs are not clickable by the admin.



* Manage User -



Routes:-

use App\Http\Controllers\ManageUserController;

 Route::get('manageuser/list', [ManageUserController::class, 'list'])->name('manageuser.list')->middleware('admin');

    Route::get('manageuser/add', [ManageUserController::class, 'add'])->name('manageuser.add')->middleware('admin');

    Route::post('manageuser/postadd', [ManageUserController::class, 'postadd'])->name('manageuser.postadd')->middleware('admin');

    Route::get('manageuser/edit', [ManageUserController::class, 'editUser'])->name('manageuser.edit')->middleware('admin');

    Route::post('manageuser/update', [ManageUserController::class, 'update'])->name('manageuser.update')->middleware('admin');

    Route::get('manageuser/delete', [ManageUserController::class, 'delete'])->name('manageuser.delete')->middleware('admin');

    Route::get('changepassword', [ManageUserController::class, 'changepassword'])->name('changepassword');

    Route::post('updatepassword', [ManageUserController::class, 'updatepassword'])->name('updatepassword');

Controller:-

use App\Models\User;

use Exception;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use Yajra\DataTables\DataTables;

 return view('manageuser.list');

Model:-

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Fortify\TwoFactorAuthenticatable;

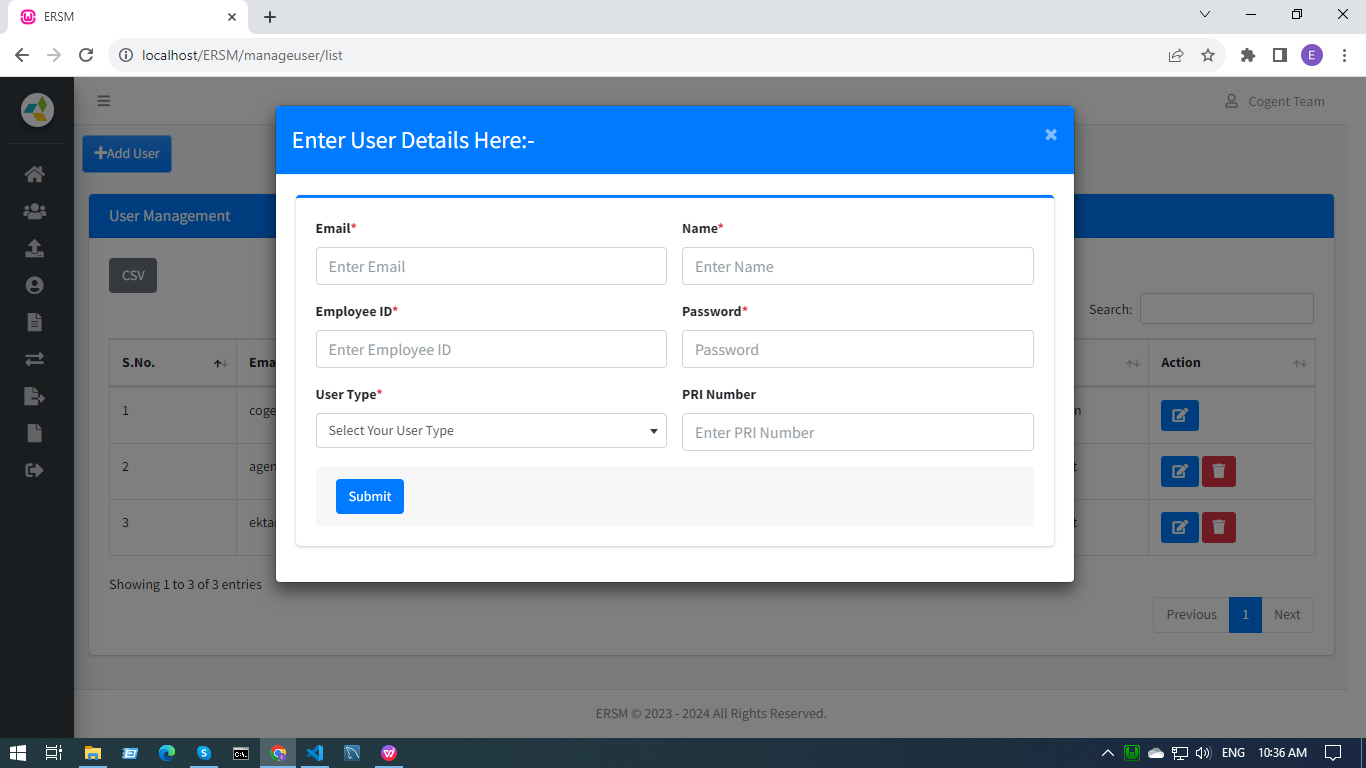
use Laravel\Jetstream\HasProfilePhoto;

use Laravel\Sanctum\HasApiTokens;

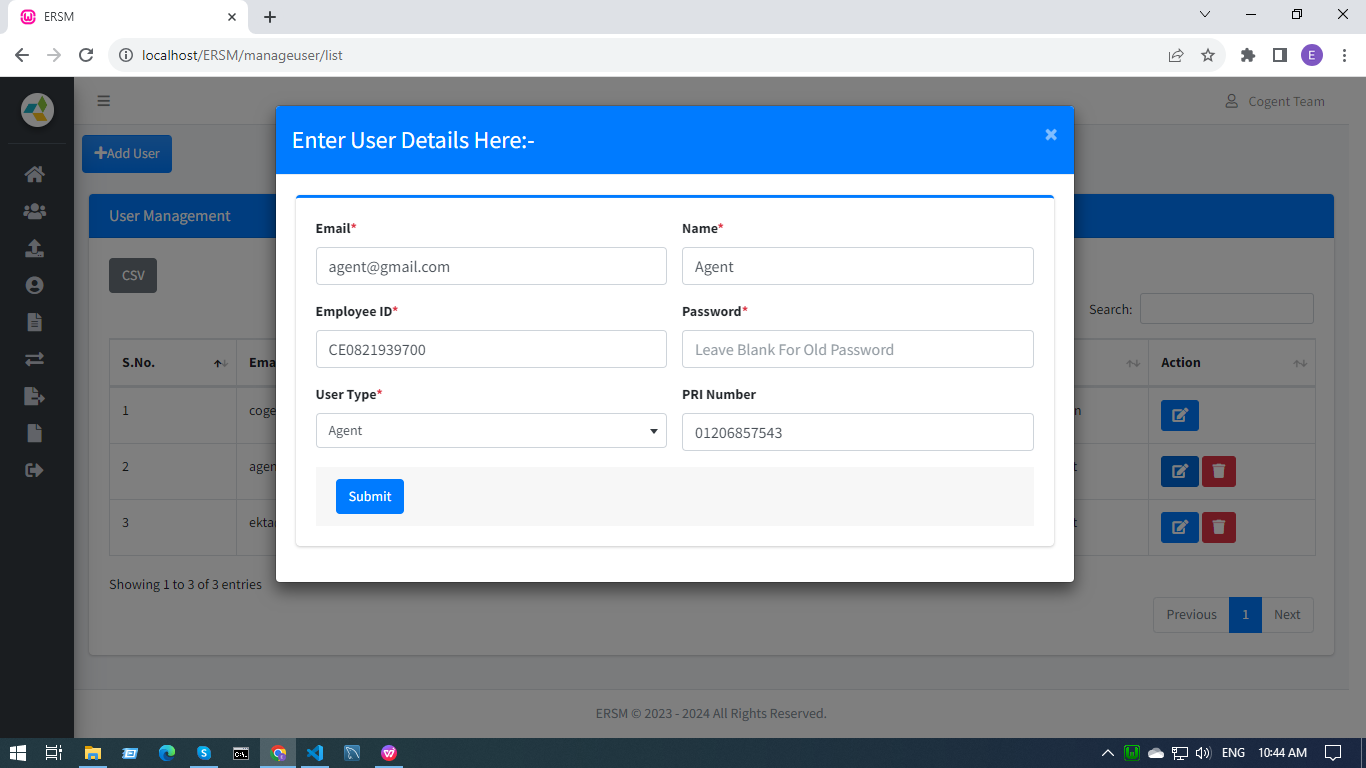
protected $fillable = ['name','email','password','usertype', ];

Features:-

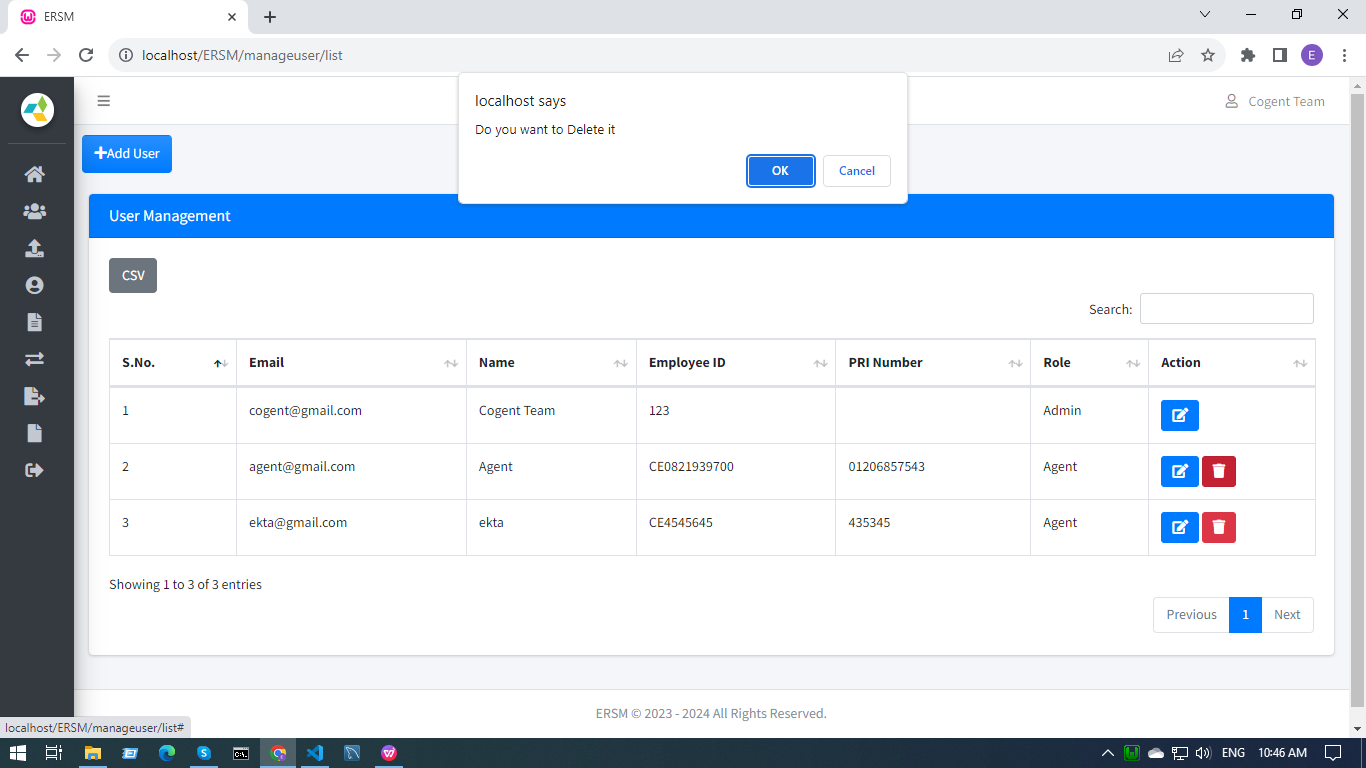
* The data of users from the database is represented in tabular form.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.
* In the Manage User page, Admin can perform 3 functions -
* Add new user:
* On clicking the Add User button, Add User Model opens , where Email, Name, Employee ID, Password, Usertype sections are mandatory and PRI Number is not mandatory.
* Click on the submit button, then the user details will be added into the database in Users table.



* Edit user :
* On clicking the Edit User button in the Action Column of the table, Edit User Model opens , where Email, Name, Employee ID, Usertype sections are mandatory and Password, PRI Number are not mandatory.
* Click on the submit button, then the user details will be updated into the database in Users table.



* Delete User :
* On clicking the Delete User button in the Action Column of the table, Delete User Model opens , where it asks for confirmation if you want to delete the user or not.
* Click on the yes button, then the user details will be deleted from the database in Users table.



* Upload Data -

Routes:-

use App\Http\Controllers\DataController;

Route::get('uploaddata/excel', [DataController::class, 'index'])->name('uploaddata.excel')->middleware('admin');

Route::post('uploaddata/import', [DataController::class, 'import'])->name('uploaddata.import')->middleware('admin');

Route::get('table-export', [DataController::class, 'export'])->name('table.export')->middleware('admin');

Controller:-

use Illuminate\Http\Request;

use App\Imports\ImportData;

use Maatwebsite\Excel\Facades\Excel;

use Exception;

use Illuminate\Support\Facades\Schema;

use Illuminate\Support\Facades\DB;

use App\Models\ClientData;

use App\Http\Controllers\Controller;

return view('uploaddata.excel');

Models:-

* ImportData :

use App\Models\ClientData;

use Illuminate\Support\Facades\Auth;

use Maatwebsite\Excel\Concerns\ToModel;

use Maatwebsite\Excel\Concerns\WithHeadingRow;

class ImportData implements ToModel, WithHeadingRow

{

    public function model(array $row)

    {

        return new ClientData([

            'vin\_no'             => $row['vin\_no'],

            'parent\_id'          => $row['parent\_id'],

            'customer\_name'      => $row['customer\_name'],

            'contact\_no'         => $row['contact\_no'],

            'usage\_category'     => $row['usage\_category'],

            'plant\_code'         => $row['plant\_code'],

            'dealer\_name'        => $row['dealer\_name'],

            'dealer\_state'       => $row['dealer\_state'],

            'bill\_to\_party\_name' => $row['bill\_to\_party\_name'],

            'device\_type'        => $row['device\_type'],

            'region'             => $row['region'],

            'customer\_category'  => $row['customer\_category'],

            'agent\_id'           => $row['agent\_id'],

            'created\_by'         => Auth::user()->name

        ]);

        return back()->with('success', 'Data Uploaded successfully');

    }

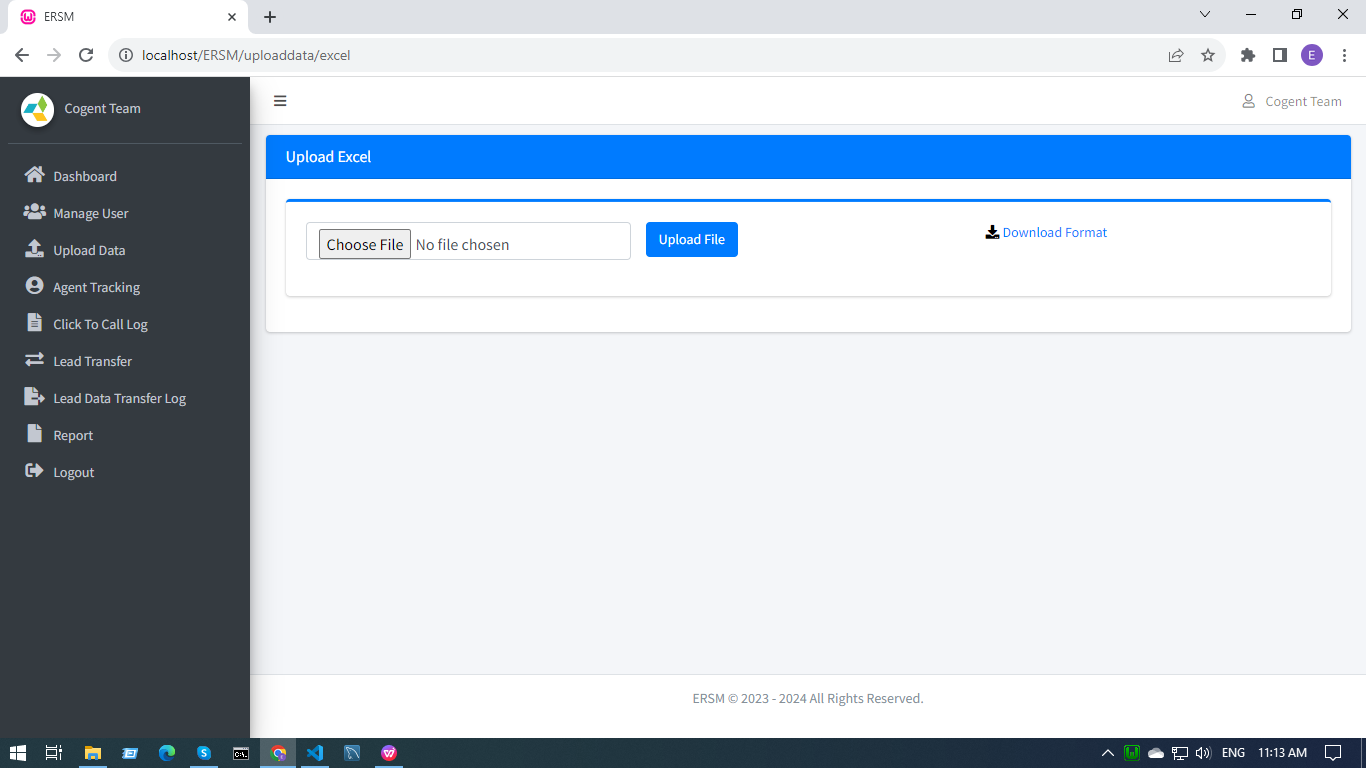
}

* Clientdata :

protected $table = 'client\_data';

Features:-

* Admins can download the format in which the data must be uploaded by clicking on the download format button.
* To upload data click on choose file button, then select a csv file then click on upload file button if the data is in correct format it will be uploaded into the database in client data table.
* The no. of records uploaded will be displayed on the screen.



* Agent Tracking -

Routes:-

use App\Http\Controllers\AgentTrackingController;

Route::get('agenttracking/list', [AgentTrackingController::class, 'list'])->name('agenttracking.list')->middleware('admin');

Controller:-

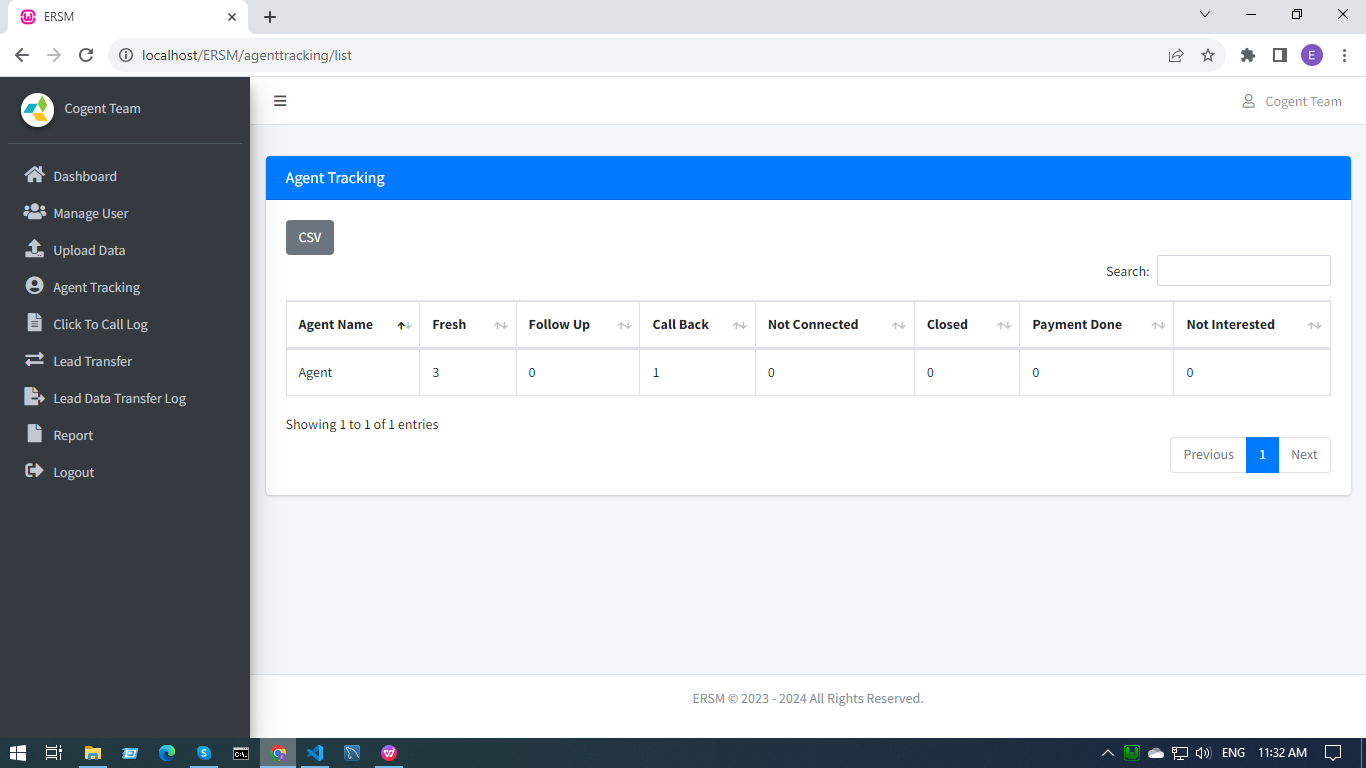
use Exception;

use Illuminate\Support\Facades\DB;

use Illuminate\Http\Request;

return view('agent.table', compact('flagCounts'));

Features:-



* Agent Tracking displays the no. And the type of leads each agent has currently.
* The data from the database is represented in tabular form.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.
* Click To Call Log -

Routes:-

use App\Http\Controllers\ReportController;

 Route::get('calllog',[ReportController::class,'calllog'])->name('calllog');

Controller:-

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Auth;

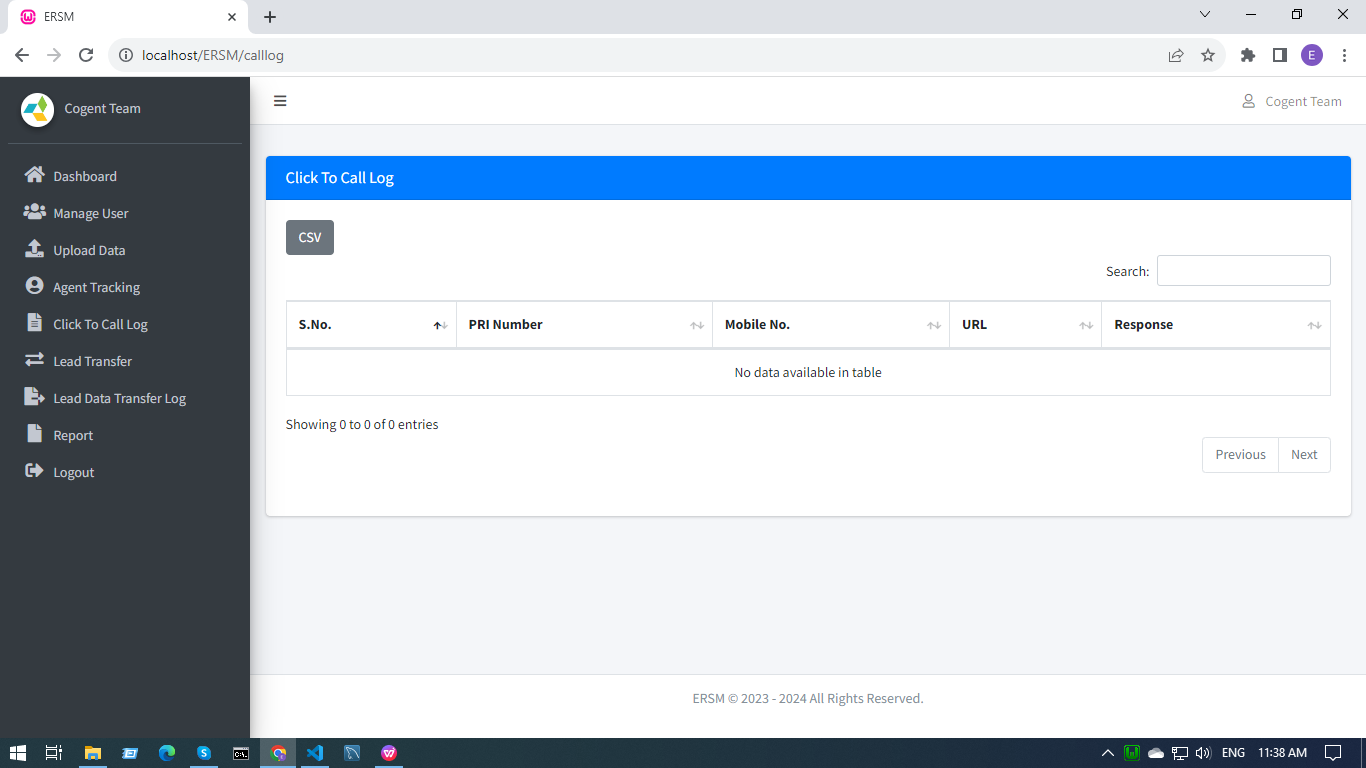
use Exception;

use DateTime;

 return view('report.calllog',compact('data'));

Features:-

* Click To Call Log displays the no. And the type of calls dialed by the agent.
* Whether the call connected or not.
* It contains the data of the last 3 days.
* The data from the database is represented in tabular form.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.



* Lead Transfer -

Routes:-

use App\Http\Controllers\LeadController;

Route::get('leadtransfer',[LeadController::class,'leadtransfer'])->name('leadtransfer');

Route::post('leadupdate',[LeadController::class,'leadupdate'])->name('leadupdate');

Route::post('/getNewAgents',[LeadController::class,'getNewAgents'])->name('getNewAgents');

Controller:-

use Illuminate\Http\Request;

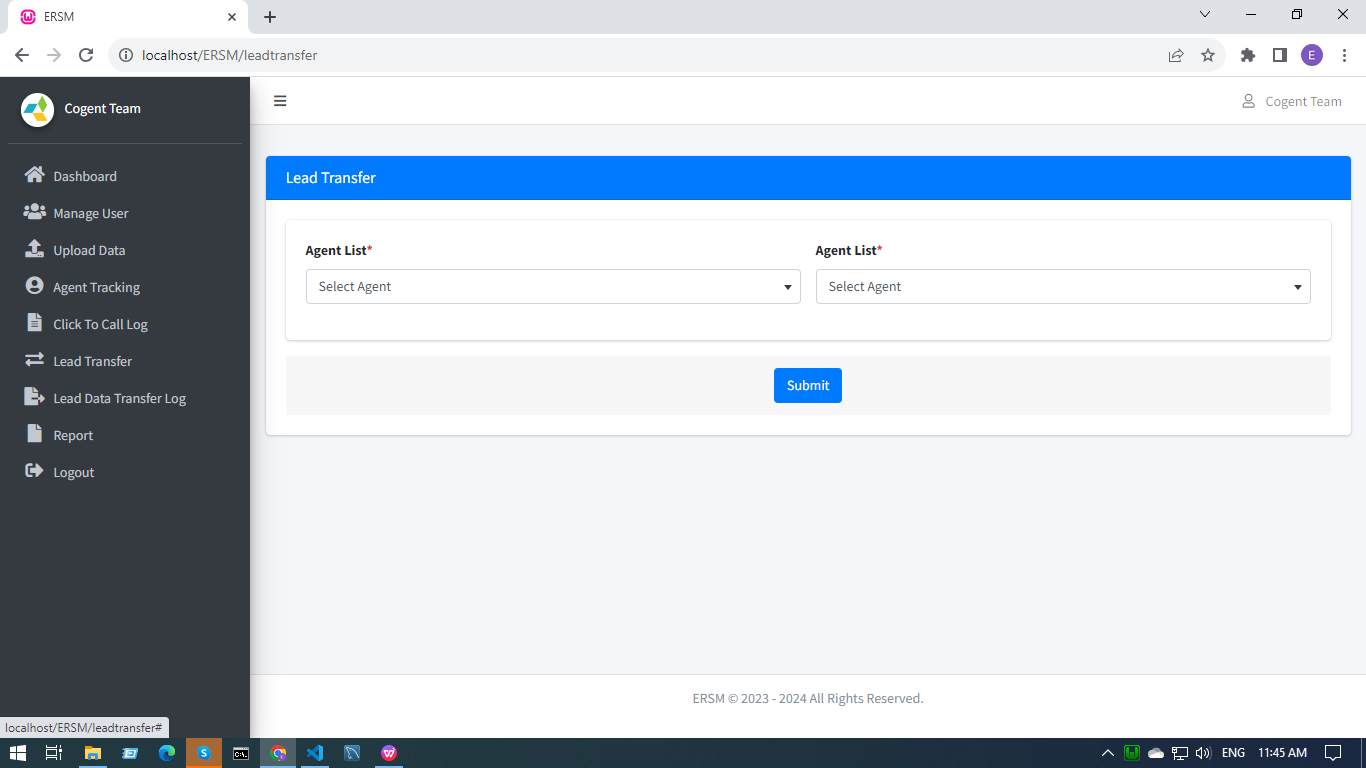
use Exception;

use Illuminate\Support\Facades\DB;

return view('lead.leadtransfer',compact('agents'));

Features:-

* On this page, leads from 1 agent can be transferred to another agent.
* Agent list drop-down consists of the names of agents.
* The value selected in the 1st drop-down will not be visible in the 2nd drop-down.
* Click on submit and the leads will be transferred.
* The updated data will be stored in the database in client data table.



* Lead Data Transfer Log -

Routes:-

use App\Http\Controllers\LeadController;

   Route::get('leadlog',[LeadController::class,'leadlog'])->name('leadlog');

Controller:-

use Illuminate\Http\Request;

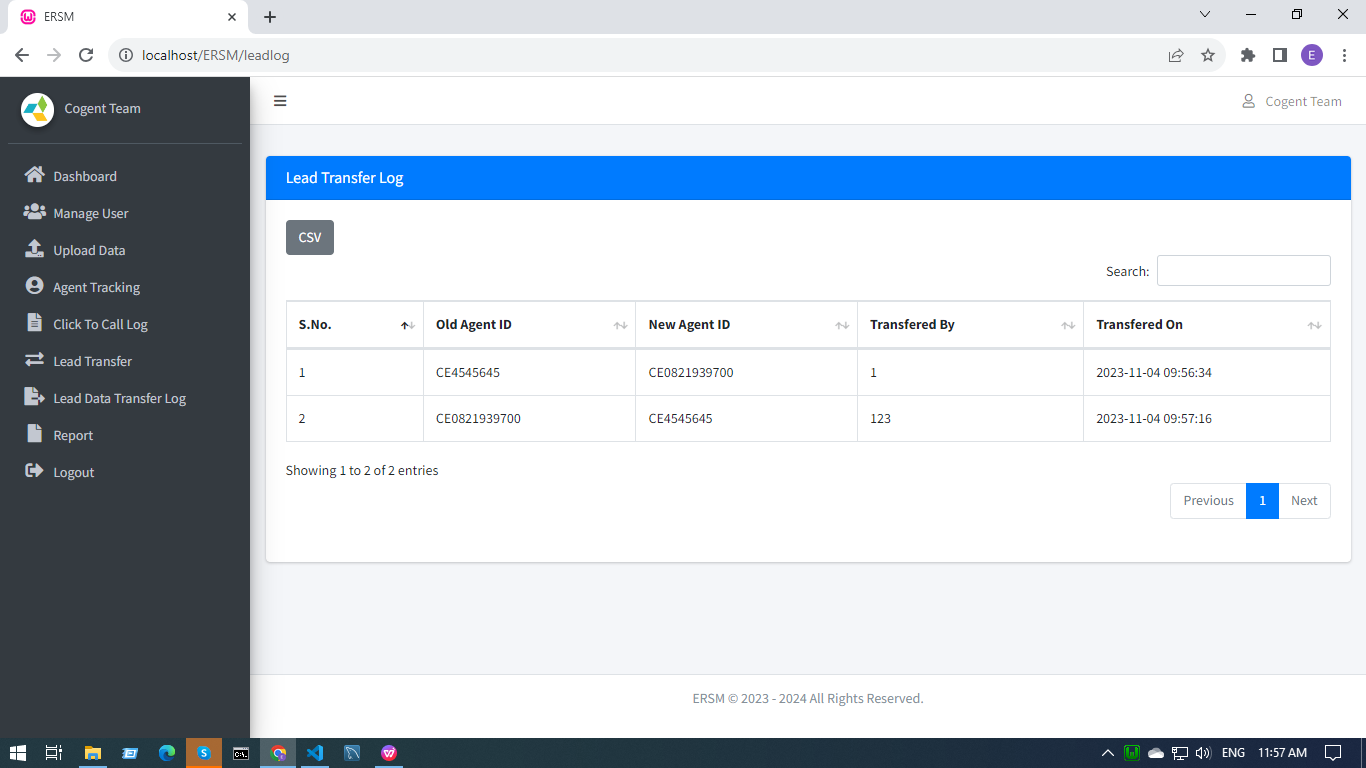
use Exception;

use Illuminate\Support\Facades\DB;

 return view('lead.leadlog',compact('data'));

Features:-

* This page consists of the details of lead transfer from 1 agent to another on lead transfer page.
* It contains the data of the last 3 months.
* The data from the database is represented in tabular form.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.



* Report -

Routes:-

use App\Http\Controllers\ReportController;

Route::get('report',[ReportController::class,'report'])->name('report');

Route::post('daterange',[ReportController::class,'daterange'])->name('daterange');

Controller:-

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Auth;

use Exception;

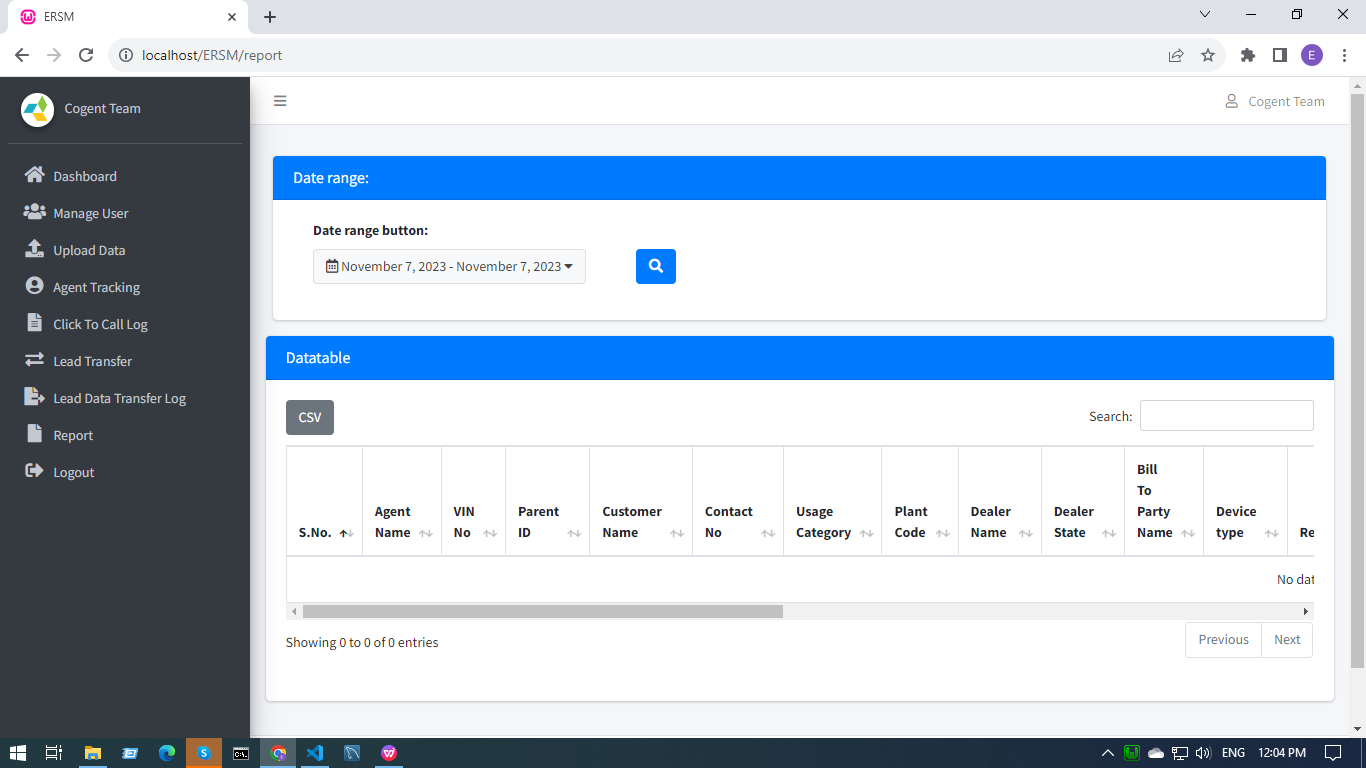
use DateTime;

 return view('report.report');

return view('report.report',compact('data'));

Features:-

* This page consists of all the details of leads from both the client data table and the agent inputs table from the database.
* Join is used in the query to fetch data.
* It contains the date range picker on which we can fetch the data from a desired duration of time.
* The data from the database is represented in tabular form from selected dates.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.



**Agents can access pages :-**

* Dashboard -

Routes:-

use App\Http\Controllers\UserController;

Route::get('/dashboard', [UserController::class, 'show'])->name('dashboard');

Controller:-

use App\Models\ClientData;

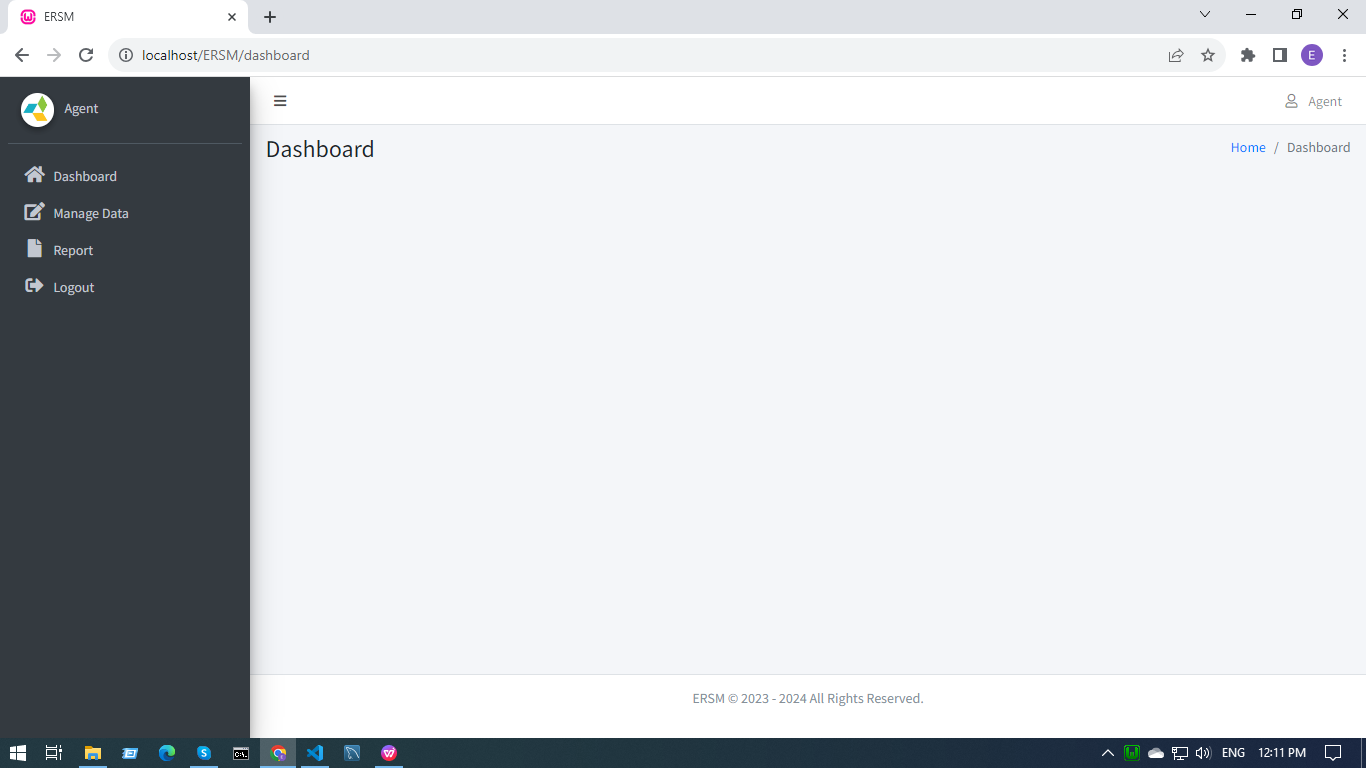
use Exception;

use Illuminate\Support\Facades\DB;

return view('dashboard');

Features:-

* The dashboard currently is blank for the agents.



* Manage Data -

Routes:-

use App\Http\Controllers\ManageDataController;

Route::get('managedata/list', [ManageDataController::class, 'freshlist'])->name('managedata.list');

Route::get('managedata/edit', [ManageDataController::class, 'freshedit'])->name('managedata.edit');

Route::post('managedata/update', [ManageDataController::class, 'update'])->name('managedata.update');

Route::get('gettabledata', [ManageDataController::class, 'gettabledata'])->name('gettabledata');

Route::get('call', [ManageDataController::class, 'call'])->name('call');

Route::post('/disposition', [ManageDataController::class, 'disposition'])->name('disposition');

Controller:-

use App\Models\ClientData;

use App\Models\Disposition;

use Illuminate\Support\Facades\DB;

use App\Models\AgentInput;

use App\Models\ClickToCall;

use Illuminate\Http\Request;

use Illuminate\Support\Carbon;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Http;

use Exception;

return view('managedata.fresh', compact('freshcount', 'followupcount', 'callbackcount', 'notconnectedcount', 'closedcount', 'paymentdone', 'notinterested'));

Models:-

* Client Data :

use Illuminate\Database\Eloquent\Model;

class ClientData extends Model

{

    protected $table = 'client\_data';

}

* Dispositions :

class Disposition extends Model

{

    protected $table = 'dispositions';

}

* AgentInput :

class AgentInput extends Model

{

    use HasFactory;

    protected $table = 'agentinputs';

    protected $fillable = ['dataID', 'source\_of\_calling', 'calling\_status',

        'connect\_status', 'year','price','remarks', 'created\_by',];

}

* ClickToCall :

class ClickToCall extends Model

{

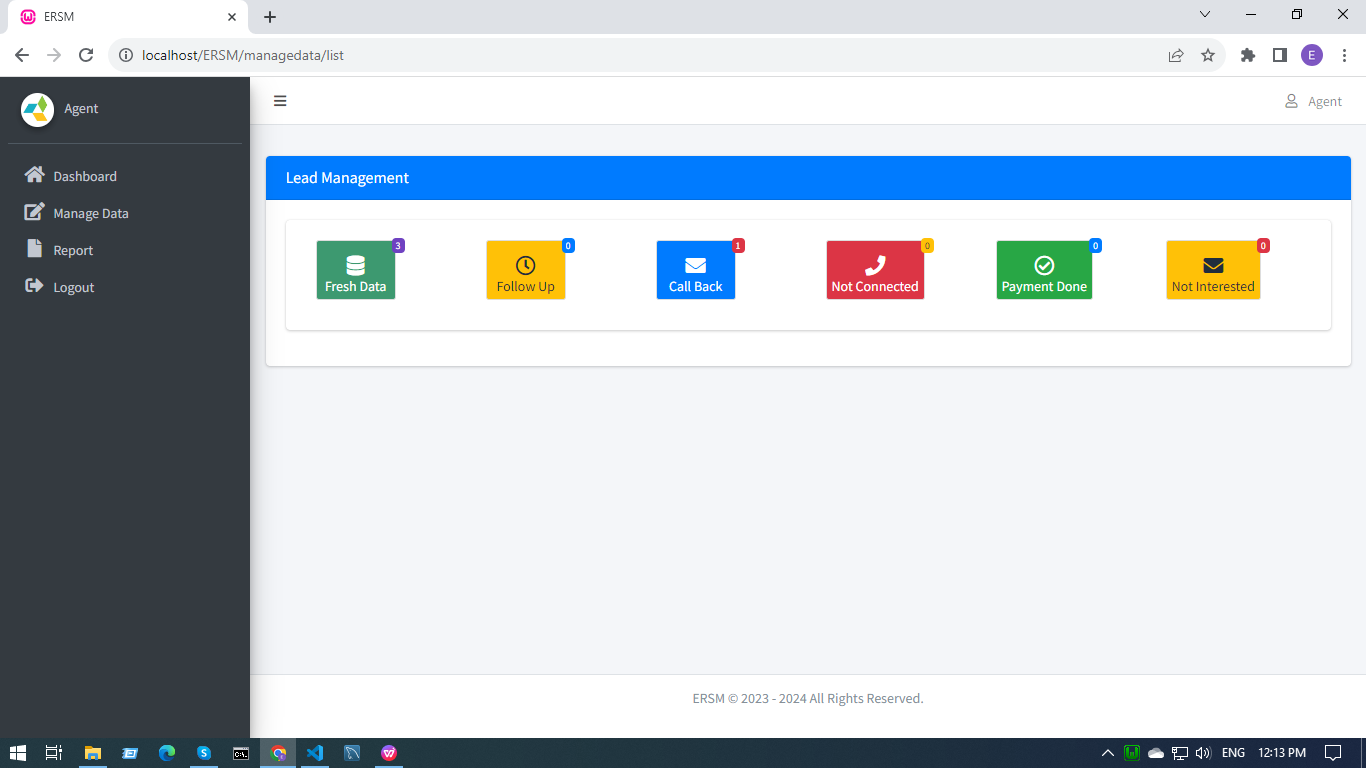
    use HasFactory;

    protected $table = 'click\_to\_calls';

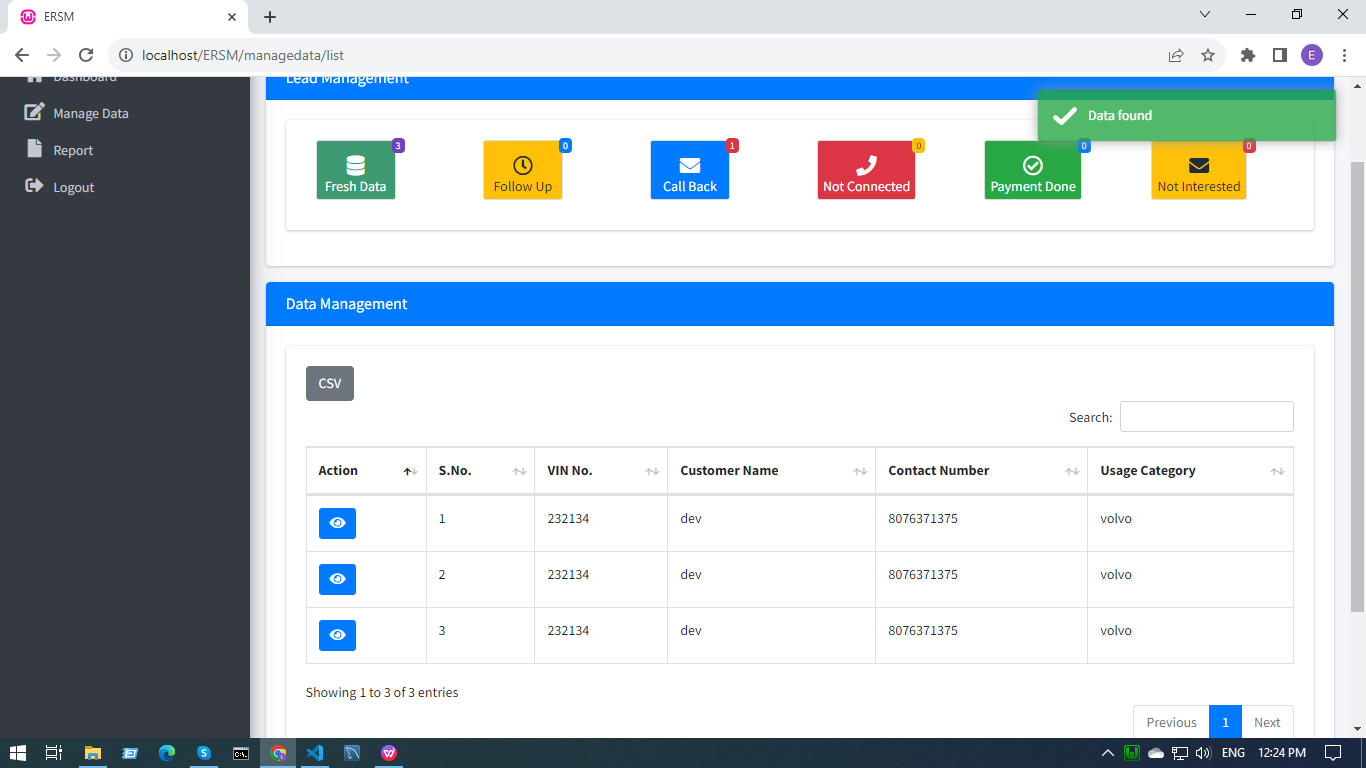
}

Features :-

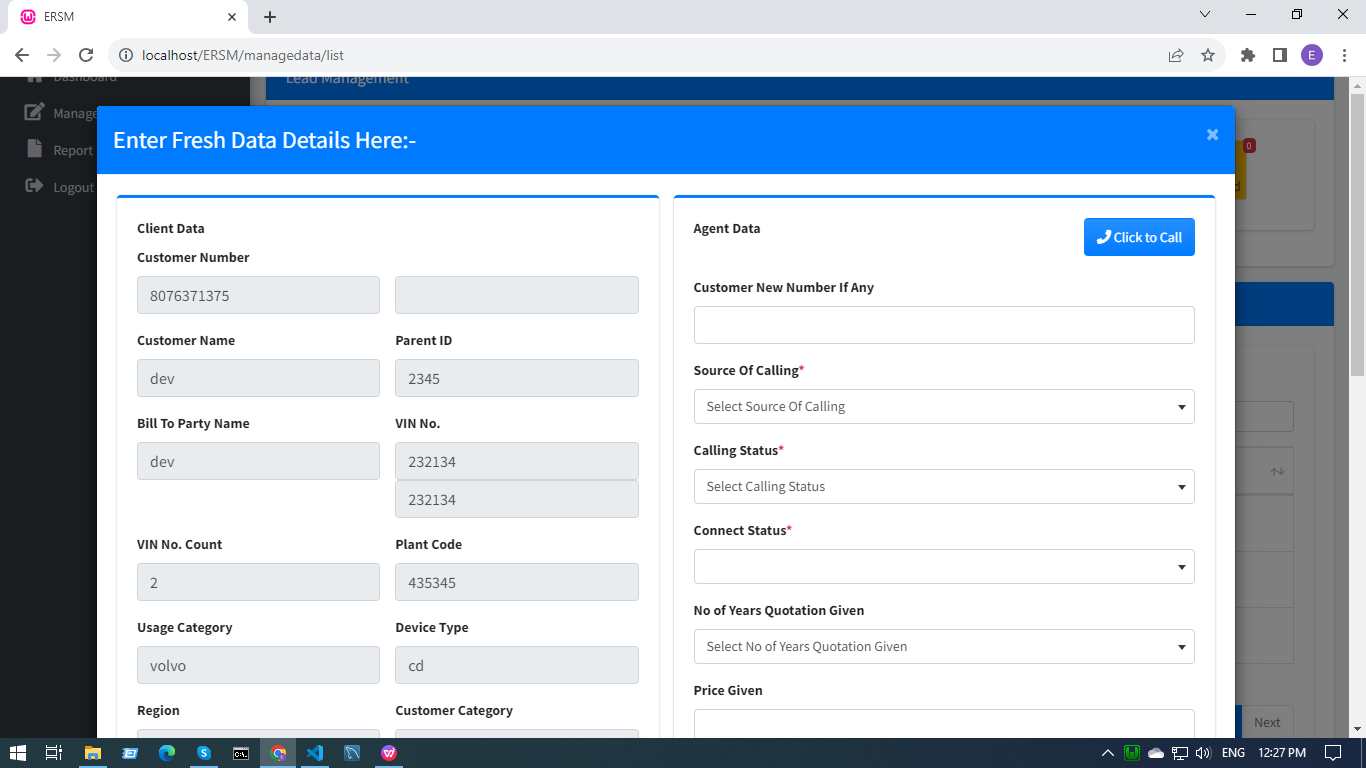
* This page contains the no. And the category of the leads as per agent.
* The tabs here are clickable and on click fetch the details of that lead type from the database.



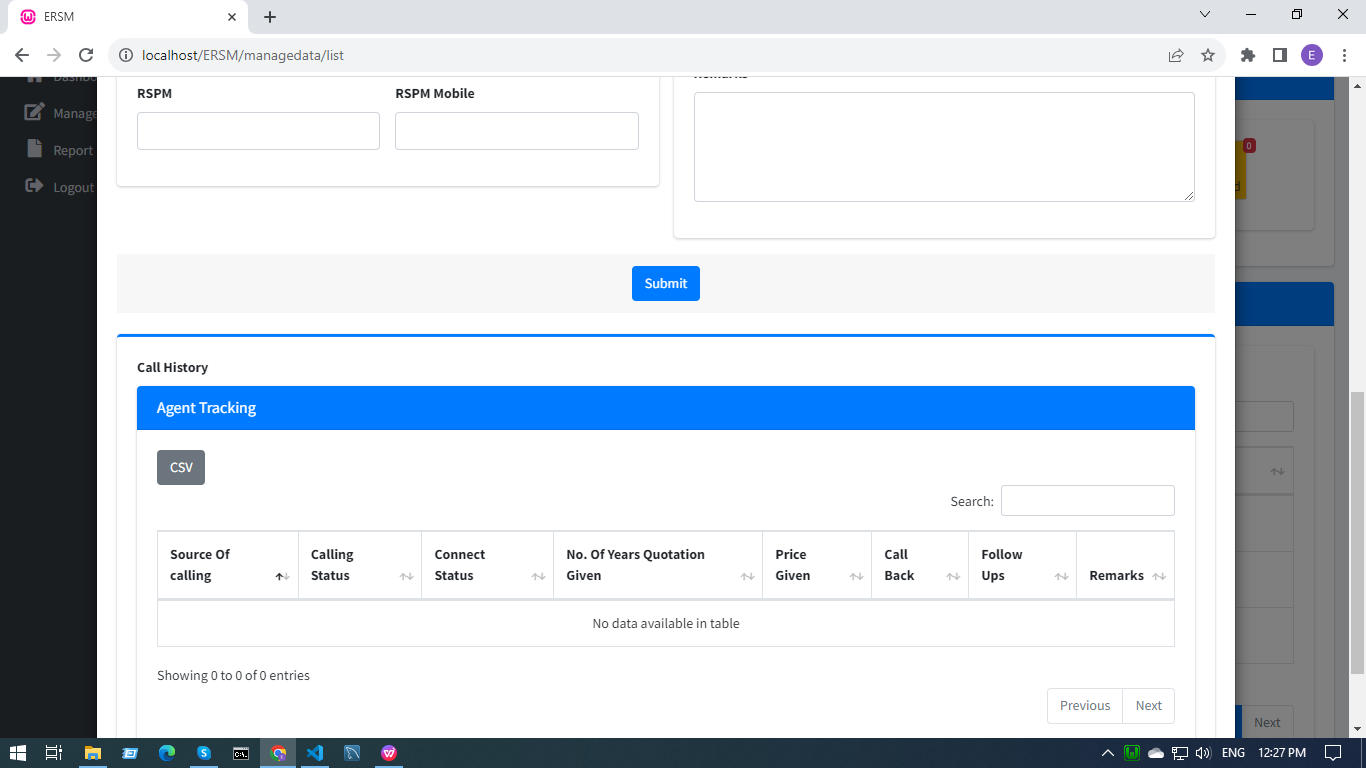
* By clicking on fresh data tab, the data management table is appended on the page using Ajax.
* The data from the database is represented in tabular form from selected dates.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.
* By clicking on the view button, model for that particular entry opens on the screen.



* Agents can click on the click to call button to make a call and the entry will be stored on the database and will be shown in click to call log.



* On the basis of the value selected in calling status, the values in connect status changes.
* On the basis of the option selected in the connect status drop-down, call back div, follow up div, quotation div, price div appear and disappear.
* Source of callsssss, calling status, connect status and remarks sections are mandatory.
* Click on the submit button and the data will be stored in their respective tables in a database.
* On the basis of agents' input, the value are also stored in a history table displayed at the end of the model.
* Admin can download the record of data in form on csv file by clicking on csv button.
* Admins can search for a particular user too using the search bar.



* Report -

Routes:-

use App\Http\Controllers\ReportController;

Route::get('report', [ReportController::class, 'report'])->name('report');

Route::post('daterange', [ReportController::class, 'daterange'])->name('daterange');

Controller:-

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Auth;

use Exception;

use DateTime;

 return view('report.report');

return view('report.report', compact('data'));

Features:-

* This page consists of all the details of leads from both the client data table and the agent inputs table from the database for the particular agent only.
* Join is used in the query to fetch data.
* It contains the date range picker on which we can fetch the data from a desired duration of time.
* The data from the database is represented in tabular form from selected dates.
* Agents can download the record of data in form on csv file by clicking on csv button.
* Agents can search for a particular entry too using the search bar.

