**ChatApplication in Laravel 11**

php artisan make:migration create\_admins\_table

php artisan make:migration create\_chats\_table

Users Migration

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

$table->id(); // id column

$table->string('name');

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

$table->string('password');

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

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

$table->string('phone\_number')->nullable();

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

$table->timestamps(); // created\_at and updated\_at

$table->boolean('status')->default(true); // active status

});

Admins Migration

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

$table->id(); // id column

$table->string('name');

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

$table->string('password');

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

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

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

$table->string('phone\_number')->nullable();

$table->timestamps(); // created\_at and updated\_at

$table->boolean('status')->default(true); // active status

});

Chats Migration

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

$table->id(); // id column

$table->unsignedBigInteger('sender\_id');

$table->unsignedBigInteger('receiver\_id');

$table->text('message');

$table->boolean('seen')->default(false);

$table->timestamps(); // created\_at and updated\_at

});

php artisan migrate

Database Setup Clear

composer require pusher/pusher-php-server

BROADCAST\_CONNECTION=pusher

BROADCAST\_DRIVER=pusher

PUSHER\_APP\_ID=1676199

PUSHER\_APP\_KEY=b23d71886d55f985f153

PUSHER\_APP\_SECRET=8e3c5e7d9d6ea1cda601

PUSHER\_APP\_CLUSTER=ap2

VITE\_APP\_NAME="${APP\_NAME}"

VITE\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"

VITE\_PUSHER\_HOST="${PUSHER\_HOST}"

VITE\_PUSHER\_PORT="${PUSHER\_PORT}"

VITE\_PUSHER\_SCHEME="${PUSHER\_SCHEME}"

VITE\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

Pusher Setup Clear

Routes

Route::post('/user/save', [UsersController::class, 'save'])->name('user.save');

Route::post('/user/check', [UsersController::class, 'check'])->name('user.check');

Route::get('/user/logout', [UsersController::class, 'logout'])->name('user.logout');

Route::get('/user/profile', [UsersController::class, 'profile'])->name('user.profile');

Route::get('/user/login', [UsersController::class, 'login'])->name('user.login');

Route::get('/user/profile', [UsersController::class, 'profile'])->name('user.profile');

Route::get('/user/register', [UsersController::class, 'register'])->name('user.register');

Route::get('/user/profileview', [UsersController::class, 'profile'])->name('user.profileview');

Route::get('/user/profileedit', [UsersController::class, 'edit'])->name('user.profileedit');

Route::get('/user/chats', [UsersController::class, 'chats'])->name('user.chats');

Route::put('/user/updateProfile', [UsersController::class, 'updateProfile'])->name('user.updateProfile');

Route::get('/user/dashboard', [UsersController::class, 'dashboard'])->name('user.dashboard');

Route::post('/admin/save', [AdminsController::class, 'save'])->name('admin.save');

Route::post('/admin/check', [AdminsController::class, 'check'])->name('admin.check');

Route::get('/admin/logout', [AdminsController::class, 'logout'])->name('admin.logout');

Route::get('/admin/profileview', [AdminsController::class, 'profile'])->name('admin.profileview');

Route::get('/admin/profileedit', [AdminsController::class, 'edit'])->name('admin.profileedit');

Route::get('/admin/login', [AdminsController::class, 'login'])->name('admin.login');

Route::get('/admin/profile', [AdminsController::class, 'profile'])->name('admin.profile');

Route::get('/admin/register', [AdminsController::class, 'register'])->name('admin.register');

Route::get('/admin/chats', [AdminsController::class, 'chats'])->name('admin.chats');

Route::get('/admin/dashboard', [AdminsController::class, 'dashboard'])->name('admin.dashboard');

Route::put('/admin/updateProfile', [AdminsController::class, 'updateProfile'])->name('admin.updateProfile');

Route::get('/admin/fetch-messages', [ChatsController::class, 'fetchMessages'])->name('admin.fetchMessages');

Route::post('/admin/send-message', [ChatsController::class, 'sendMessage'])->name('admin.sendMessage');

Route::get('/fetch-messages', [ChatsController::class, 'fetchMessagesFromUserToAdmin'])->name('fetch.messagesFromSellerToAdmin');

Route::post('/send-message', [ChatsController::class, 'sendMessageFromUserToAdmin'])->name('send.Messageofsellertoadmin');

Models

class Admin extends Model

{

use HasFactory;

protected $table = 'admins'; // Optional if your table name matches the default convention

// Specify which attributes are mass assignable

protected $fillable = [

'name',

'email',

'password',

'picture', // Include if you have a picture attribute

];

}

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Chat extends Model

{

use HasFactory;

public function sender()

{

return $this->belongsTo(User::class, 'sender\_id');

}

// Relationship to get receiver profile

public function receiver()

{

return $this->belongsTo(User::class, 'receiver\_id');

}

public function receiverProfilee()

{

return $this->belongsTo(Admin::class, 'receiver\_id', 'id')->select(['id', 'name', 'picture', 'bio']);

}

public function senderProfilee()

{

return $this->belongsTo(Admin::class, 'sender\_id', 'id')->select(['id', 'name', 'picture', 'bio']);

}

public function receiverSellerProfile()

{

return $this->belongsTo(User::class, 'receiver\_id', 'id')->select(['id', 'name', 'picture', 'bio']);

}

public function senderSellerProfile()

{

return $this->belongsTo(User::class, 'sender\_id', 'id')->select(['id', 'name', 'picture', 'bio']);

}

}

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

class User extends Authenticatable

{

use HasFactory, Notifiable;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array<int, string>

\*/

protected $fillable = [

'name',

'email',

'password',

];

/\*\*

\* The attributes that should be hidden for serialization.

\*

\* @var array<int, string>

\*/

protected $hidden = [

'password',

'remember\_token',

];

/\*\*

\* Get the attributes that should be cast.

\*

\* @return array<string, string>

\*/

protected function casts(): array

{

return [

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

'password' => 'hashed',

];

}

}

Controllers

UsersControllers

<?php

namespace App\Http\Controllers;

use App\Models\Admin;

use App\Models\Chat;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Storage;

class UsersController extends Controller

{

public function updateProfile(Request $request)

{

$userId = session('LoggedUserInfo');

$user = User::find($userId);

if (!$user) {

return redirect('user/login')->with('fail', 'You must be logged in to update the profile');

}

$request->validate([

'name' => 'required|string|max:255',

'phone\_number' => 'required|string|max:15',

'bio' => 'nullable|string',

'picture' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',

]);

// Update user details

$user->name = $request->name;

$user->phone\_number = $request->phone\_number;

$user->bio = $request->bio;

if ($request->hasFile('picture')) {

// Delete the old profile picture if it exists

if ($user->picture) {

Storage::disk('public')->delete($user->picture);

}

// Store the new picture

$file = $request->file('picture');

$filename = time() . '.' . $file->getClientOriginalExtension();

$path = $file->storeAs('profile\_pictures', $filename, 'public');

// Save the path to the user's picture

$user->picture = $path;

}

// Save the user profile updates

$user->save();

return redirect()->back()->with('success', 'Profile updated successfully.');

}

public function post() {

$users = User::all();

return view('user.post', [

'users' => $users

]);

}

public function register() {

return view("user.register");

}

public function login() {

return view("user.login");

}

public function chats()

{

$userId = session('LoggedUserInfo');

$LoggedUserInfo = User::find($userId);

if (!$LoggedUserInfo) {

return redirect('user/login')->with('fail', 'You must be logged in to access the dashboard');

}

// Retrieve all admins

$admins = Admin::all();

return view('user.chats', [

'LoggedUserInfo' => $LoggedUserInfo,

'admins' => $admins // Pass only admins to the view

]);

}

public function edit()

{

$userId = session('LoggedUserInfo');

$LoggedUserInfo = User::find($userId);

if (!$LoggedUserInfo) {

return redirect('user/login')->with('fail', 'You must be logged in to access the dashboard');

}

return view('user.profileedit', [

'LoggedUserInfo' => $LoggedUserInfo,

]);

}

public function profile()

{

$userId = session('LoggedUserInfo');

$LoggedUserInfo = User::find($userId);

if (!$LoggedUserInfo) {

return redirect('user/login')->with('fail', 'You must be logged in to access the dashboard');

}

return view('user.profileview', [

'LoggedUserInfo' => $LoggedUserInfo,

]);

}

public function dashboard()

{

$userId = session('LoggedUserInfo');

// Check if the session has the correct user ID

if (!$userId) {

return redirect('user/login')->with('fail', 'You must be logged in to access the dashboard');

}

$LoggedUserInfo = User::find($userId);

// Fetch the count of messages for the user

// Fetch the messages, ensuring they are ordered by the newest first

return view('user.dashboard', [

'LoggedUserInfo' => $LoggedUserInfo,

]);

}

public function check(Request $request)

{

$request->validate([

'email' => 'required|email',

'password' => 'required|min:5|max:12'

]);

$userInfo = User::where('email', $request->email)->first();

if (!$userInfo) {

return back()->withInput()->withErrors(['email' => 'Email not found']);

}

if ($userInfo->status === 'inactive') {

return back()->withInput()->withErrors(['status' => 'Your account is inactive']);

}

if (!Hash::check($request->password, $userInfo->password)) {

return back()->withInput()->withErrors(['password' => 'Incorrect password']);

}

session([

'LoggedUserInfo' => $userInfo->id,

'LoggedUserName' => $userInfo->name,

]);

return redirect()->route('user.dashboard');

}

public function logout()

{

if (session()->has('LoggedUserInfo')) {

session()->forget('LoggedUserInfo');

}

session()->flush();

return redirect()->route('user.dashboard');

}

public function save(Request $request)

{

$request->validate([

'name' => 'required|string|max:255',

'email' => 'required|string|email|max:255|unique:users',

'password' => 'required|string|min:8|regex:/^\S\*$/',

], [

'email.unique' => 'This email is already registered.',

'password.min' => 'Password must be at least 8 characters long.',

]);

$user = User::create([

'name' => $request->name,

'email' => $request->email,

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

]);

return redirect()->route('user.login')->with('success', 'User created successfully!');

}

}

AdminsController

<?php

namespace App\Http\Controllers;

use App\Models\Admin;

use App\Models\Chat;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Session;

use Illuminate\Support\Facades\Storage;

class AdminsController extends Controller

{

public function register() {

return view("admin.register");

}

public function login() {

return view("admin.login");

}public function chats()

{

$LoggedAdminInfo = Admin::find(session('LoggedAdminInfo'));

if (!$LoggedAdminInfo) {

return redirect()->route('admin.login')->with('fail', 'You must be logged in to access the dashboard');

}

// Fetch chats where the admin is either the sender or the receiver

$chats = Chat::with(['senderProfilee', 'receiverProfilee', 'senderSellerProfile', 'receiverSellerProfile'])

->where('sender\_id', $LoggedAdminInfo->id)

->orWhere('receiver\_id', $LoggedAdminInfo->id)

->get();

// Combine both results and remove duplicates

$allChats = $chats->map(function($chat) use ($LoggedAdminInfo) {

if ($chat->sender\_id == $LoggedAdminInfo->id) {

if ($chat->receiverProfilee) {

$chat->user\_id = $chat->receiver\_id;

$chat->profile = $chat->receiverProfilee;

} else {

$chat->user\_id = $chat->receiver\_id;

$chat->profile = $chat->receiverSellerProfile;

}

} else {

if ($chat->senderProfilee) {

$chat->user\_id = $chat->sender\_id;

$chat->profile = $chat->senderProfilee;

} else {

$chat->user\_id = $chat->sender\_id;

$chat->profile = $chat->senderSellerProfile;

}

}

return $chat;

})->unique('user\_id')->values();

// Pass the logged-in admin's information and chats to the view

return view('admin.chats', [

'LoggedAdminInfo' => $LoggedAdminInfo,

'chats' => $allChats

]);

}

public function dashboard()

{

$adminId = session('LoggedAdminInfo');

// Check if the session has the correct admin ID

if (!$adminId) {

return redirect('admin/login')->with('fail', 'You must be logged in to access the dashboard');

}

$LoggedAdminInfo = Admin::find($adminId);

if (!$LoggedAdminInfo) {

return redirect('admin/login')->with('fail', 'Admin not found');

}

return view('admin.dashboard', [

'LoggedAdminInfo' => $LoggedAdminInfo

]);

}

public function check(Request $request)

{

$request->validate([

'email' => 'required|email',

'password' => 'required|min:5|max:12'

]);

// Find the admin by email

$adminInfo = Admin::where('email', $request->email)->first();

// Check if the admin exists

if (!$adminInfo) {

return back()->withInput()->withErrors(['email' => 'Email not found']);

}

// Check if the admin's account is inactive

if ($adminInfo->status === 'inactive') {

return back()->withInput()->withErrors(['status' => 'Your account is inactive']);

}

// Check if the password is correct

if (!Hash::check($request->password, $adminInfo->password)) {

return back()->withInput()->withErrors(['password' => 'Incorrect password']);

}

// Set session variables

session([

'LoggedAdminInfo' => $adminInfo->id,

'LoggedAdminName' => $adminInfo->name,

]);

// Redirect to the admin dashboard

return redirect()->route('admin.dashboard');

}

public function logout()

{

if (Session::has('LoggedAdminInfo')) {

Session::forget('LoggedAdminInfo');

}

Session::flush();

return redirect()->route('admin.login');

}

public function updateProfile(Request $request)

{

$adminId = session('LoggedAdminInfo');

$admin = Admin::find($adminId);

if (!$admin) {

return redirect('admin/login')->with('fail', 'You must be logged in to update the profile');

}

$request->validate([

'name' => 'required|string|max:255',

'phone\_number' => 'required|string|max:15',

'bio' => 'nullable|string',

'picture' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',

]);

// Update admin details

$admin->name = $request->name;

$admin->phone\_number = $request->phone\_number;

$admin->bio = $request->bio;

if ($request->hasFile('picture')) {

// Delete the old profile picture if it exists

if ($admin->picture) {

Storage::disk('public')->delete($admin->picture);

}

// Store the new picture

$file = $request->file('picture');

$filename = time() . '.' . $file->getClientOriginalExtension();

$path = $file->storeAs('profile\_pictures', $filename, 'public');

// Save the path to the admin's picture

$admin->picture = $path;

}

// Save the admin profile updates

$admin->save();

return redirect()->back()->with('success', 'Profile updated successfully.');

}

public function edit()

{

$adminId = session('LoggedAdminInfo');

$LoggedAdminInfo = Admin::find($adminId);

if (!$LoggedAdminInfo) {

return redirect('admin/login')->with('fail', 'You must be logged in to access the dashboard');

}

return view('admin.profileedit', [

'LoggedAdminInfo' => $LoggedAdminInfo,

]);

}

public function profile()

{

$adminId = session('LoggedAdminInfo');

$LoggedAdminInfo = Admin::find($adminId);

if (!$LoggedAdminInfo) {

return redirect('admin/login')->with('fail', 'You must be logged in to access the dashboard');

}

return view('admin.profileview', [

'LoggedAdminInfo' => $LoggedAdminInfo,

]);

}

public function save(Request $request)

{

$request->validate([

'name' => 'required|string|max:255',

'email' => 'required|string|email|max:255|unique:admins',

'password' => 'required|string|min:8|regex:/^\S\*$/',

], [

'email.unique' => 'This email is already registered.',

'password.min' => 'Password must be at least 8 characters long.',

'picture.max' => 'Profile picture size must be less than 2MB.',

]);

$adminData = [

'name' => $request->name,

'email' => $request->email,

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

];

Admin::create($adminData);

return redirect()->route('admin.login')->with('success', 'Admin created successfully!');

}

}

ChatsController

<?php

namespace App\Http\Controllers;

use App\Events\SendAdminMessage;

use App\Events\SendSellerMessage;

use App\Events\SendUserMessage;

use App\Models\Chat;

use Illuminate\Http\Request;

use App\Models\Admin;

class ChatsController extends Controller

{

public function fetchMessagesFromUserToAdmin(Request $request)

{

$receiverId = $request->input('receiver\_id');

$sellerId = session('LoggedUserInfo');

$messages = Chat::where(function($query) use ($sellerId, $receiverId) {

$query->where('sender\_id', $sellerId)

->where('receiver\_id', $receiverId);

})->orWhere(function($query) use ($sellerId, $receiverId) {

$query->where('sender\_id', $receiverId)

->where('receiver\_id', $sellerId);

})->orderBy('created\_at', 'asc')->get();

return response()->json(['messages' => $messages]);

}

public function sendMessageFromUserToAdmin(Request $request)

{

// Validate the incoming request data

$request->validate([

'message' => 'required|string',

'receiver\_id' => 'required|exists:admins,id',

]);

// Create a new chat message

$chat = new Chat();

$chat->sender\_id = session('LoggedUserInfo');

$chat->receiver\_id = $request->input('receiver\_id');

$chat->message = $request->input('message');

$chat->seen = 0; // Default to not seen

$chat->save();

// Broadcast the message using the SendUserMessage event

event(new SendUserMessage($chat));

// Return a JSON response indicating success

return response()->json(['success' => true, 'message' => 'Message sent successfully']);

}

public function sendMessage(Request $request)

{

$request->validate([

'message' => 'required|string',

'receiver\_id' => 'required|integer|exists:users,id', // Ensure the receiver\_id is a valid user id

]);

$LoggedAdminInfo = Admin::find(session('LoggedAdminInfo'));

if (!$LoggedAdminInfo) {

return response()->json([

'success' => false,

'message' => 'You must be logged in to send a message',

]);

}

$message = new Chat();

$message->sender\_id = $LoggedAdminInfo->id;

$message->receiver\_id = $request->receiver\_id;

$message->message = $request->message;

$message->save();

broadcast(new SendAdminMessage($message))->toOthers();

return response()->json([

'success' => true,

'message' => 'Message sent successfully',

]);

}

public function fetchMessages(Request $request)

{

$receiverId = $request->input('receiver\_id');

// Fetch the logged-in admin using the session

$adminId = session('LoggedAdminInfo');

$LoggedAdminInfo = Admin::find($adminId);

if (!$LoggedAdminInfo) {

return response()->json([

'error' => 'Admin not found. You must be logged in to access messages.'

], 404);

}

// Fetch messages between the admin and the specified seller

$messages = Chat::where(function ($query) use ($adminId, $receiverId) {

$query->where('sender\_id', $adminId)

->where('receiver\_id', $receiverId);

})->orWhere(function ($query) use ($adminId, $receiverId) {

$query->where('sender\_id', $receiverId)

->where('receiver\_id', $adminId);

})->orderBy('created\_at', 'asc')->get();

return response()->json([

'messages' => $messages

]);

}

}

Events

SendAdminMessage

<?php

namespace App\Events;

use App\Models\Admin;

use App\Models\Chat;

use Illuminate\Broadcasting\Channel;

use Illuminate\Broadcasting\InteractsWithSockets;

use Illuminate\Broadcasting\PresenceChannel;

use Illuminate\Broadcasting\PrivateChannel;

use Illuminate\Contracts\Broadcasting\ShouldBroadcast;

use Illuminate\Foundation\Events\Dispatchable;

use Illuminate\Queue\SerializesModels;

use Illuminate\Contracts\Broadcasting\ShouldBroadcastNow;

class SendAdminMessage implements ShouldBroadcastNow

{

use Dispatchable, InteractsWithSockets, SerializesModels;

public $message;

/\*\*

\* Create a new event instance.

\*

\* @return void

\*/

public function \_\_construct(Chat $message)

{

$this->message = $message;

}

/\*\*

\* Get the channels the event should broadcast on.

\*

\* @return \Illuminate\Broadcasting\Channel

\*/

public function broadcastOn()

{

return new Channel('admin-messages' );

}

/\*\*

\* Prepare the data to be broadcast.

\*

\* @return array

\*/

public function broadcastWith()

{

$admin = Admin::find($this->message->sender\_id);

if ($admin) {

return [

'message' => $this->message->message,

'receiver\_id' => $this->message->receiver\_id,

'sender\_id' => $this->message->sender\_id,

'admin' => [

'id' => $admin->id,

'name' => $admin->name,

'image' => asset('storage/' . $admin->picture),

],

'created\_at' => $this->message->created\_at,

];

} else {

return [

'message' => $this->message->message,

'receiver\_id' => $this->message->receiver\_id,

'sender\_id' => $this->message->sender\_id,

'admin' => [

'id' => null,

'name' => 'Unknown Admin',

'image' => asset('storage/default-avatar.png'),

],

'created\_at' => $this->message->created\_at,

];

}

}

/\*\*

\* The name of the event to broadcast.

\*

\* @return string

\*/

public function broadcastAs()

{

return 'admin-message';

}

}

SendUserMessage

<?php

namespace App\Events;

use App\Models\Chat;

use App\Models\User;

use Illuminate\Broadcasting\Channel;

use Illuminate\Broadcasting\InteractsWithSockets;

use Illuminate\Broadcasting\PresenceChannel;

use Illuminate\Broadcasting\PrivateChannel;

use Illuminate\Contracts\Broadcasting\ShouldBroadcast;

use Illuminate\Foundation\Events\Dispatchable;

use Illuminate\Queue\SerializesModels;

use Illuminate\Contracts\Broadcasting\ShouldBroadcastNow;

class SendUserMessage implements ShouldBroadcastNow

{

use Dispatchable, InteractsWithSockets, SerializesModels;

public $message;

/\*\*

\* Create a new event instance.

\*

\* @param Chat $message

\* @return void

\*/

public function \_\_construct(Chat $message)

{

$this->message = $message;

}

/\*\*

\* Get the channels the event should broadcast on.

\*

\* @return \Illuminate\Broadcasting\Channel

\*/

public function broadcastOn()

{

return new Channel('admin-messages'); // Channel to broadcast on

}

/\*\*

\* Prepare the data to be broadcast.

\*

\* @return array

\*/

public function broadcastWith()

{

// Assuming sender\_id is a user ID

$user = User::find($this->message->sender\_id); // Replace User with the actual model name if different

if ($user) {

return [

'message' => $this->message->message,

'receiver\_id' => $this->message->receiver\_id,

'sender\_id' => $this->message->sender\_id,

'user' => [

'id' => $user->id,

'name' => $user->name,

'image' => asset('storage/' . $user->picture), // Assuming 'picture' is the image field

],

'created\_at' => $this->message->created\_at,

];

} else {

return [

'message' => $this->message->message,

'receiver\_id' => $this->message->receiver\_id,

'sender\_id' => $this->message->sender\_id,

'user' => [

'id' => null,

'name' => 'Unknown User',

'image' => asset('storage/default-avatar.png'), // Default image if user is not found

],

'created\_at' => $this->message->created\_at,

];

}

}

/\*\*

\* The name of the event to broadcast.

\*

\* @return string

\*/

public function broadcastAs()

{

return 'user-message'; // Event name to broadcast as

}

}

Laravel11ChatApp.txt

Displaying Laravel11ChatApp.txt.

userController

<?php

namespace App\Http\Controllers;

use App\Models\Chat;

use App\Models\User;

use App\Models\Admin;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Storage;

class UserController extends Controller

{

public function updateProfile(Request $request)

{

$userId = session('LoggedUserInfo');

// $user = User::find($userId);

// echo "userid :".$userId;

// echo "<br><br>".$user;

// return $request;

// if (!$user) {

// return redirect()->route('user.login')->with('fail', 'You must be logged in to update the profile.');

// }

// $request->validate([

// 'name' => 'required|string|max:20',

// 'phone\_number' => 'required|string|max:10',

// 'bio' => 'nullable|string|max:100',

// 'picture' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:100',

// ]);

//upload image here

$img = $request->picture;

$ext = $img->getClientOriginalExtension();

$imgName = time().'.'.$ext;

$img->move(public\_path('uploads/temp/'),$imgName);

// // //store the data

$update= DB::table('users')->where('id',$userId)->update([

'name' => $request->name,

'phone\_number' => $request->phone\_number,

'bio' => $request->bio,

'picture'=>$imgName

]);

// if($update){

// return redirect()->route('user.profileview');

// }else{

// return "does note update";

// }

// move image in temp directory

// Update user details

// $user->name = $request->name;

// $user->phone\_number = $request->phone\_number;

// $user->bio = $request->bio;

// if ($request->hasFile('picture')) {

// // Delete the old profile picture if it exists

// if ($user->picture) {

// Storage::disk('public')->delete($user->picture);

// }

// // Store the new picture

// $file = $request->file('picture');

// $filename = time() . '.' . $file->getClientOriginalExtension();

// $path = $file->storeAs('profile\_pictures', $filename, 'public');

// // Save the path to the user's picture

// $user->picture = $path;

// }

// $user->save();

return redirect()->back()->with('success', 'Profile updated successfully.');

}

public function post()

{

$users = User::all();

return view('user.post', compact('users'));

}

public function register()

{

return view('user.register');

}

public function login()

{

return view('user.login');

}

public function chats()

{

$userId = session('LoggedUserInfo');

$LoggedUserInfo = User::find($userId);

if (!$LoggedUserInfo) {

return redirect()->route('user.login')->with('fail', 'You must be logged in to access the dashboard.');

}

$admins = Admin::all();

return view('user.chats', compact('LoggedUserInfo', 'admins'));

}

public function edit()

{

$userId = session('LoggedUserInfo');

$LoggedUserInfo = User::find($userId);

if (!$LoggedUserInfo) {

return redirect()->route('user.login')->with('fail', 'You must be logged in to access the profile.');

}

return view('user.profileedit', compact('LoggedUserInfo'));

}

public function profile()

{

$userId = session('LoggedUserInfo');

$LoggedUserInfo = User::find($userId);

if (!$LoggedUserInfo) {

return redirect()->route('user.login')->with('fail', 'You must be logged in to access the profile.');

}

return view('user.profileview', compact('LoggedUserInfo'));

}

public function dashboard()

{

$userId = session('LoggedUserInfo');

if (!$userId) {

return redirect()->route('user.login')->with('fail', 'You must be logged in to access the dashboard.');

}

$LoggedUserInfo = User::find($userId);

return view('user.dashboard', compact('LoggedUserInfo'));

}

public function check(Request $request)

{

$request->validate([

'email' => 'required|email',

'password' => 'required|min:5|max:12',

]);

$userInfo = User::where('email', $request->email)->first();

if (!$userInfo) {

return back()->withInput()->withErrors(['email' => 'Email not found.']);

}

if ($userInfo->status === 'inactive') {

return back()->withInput()->withErrors(['status' => 'Your account is inactive.']);

}

if (!Hash::check($request->password, $userInfo->password)) {

return back()->withInput()->withErrors(['password' => 'Incorrect password.']);

}

session([

'LoggedUserInfo' => $userInfo->id,

'LoggedUserName' => $userInfo->name,

]);

return redirect()->route('user.dashboard');

}

public function logout()

{

session()->flush();

return redirect()->route('user.login');

}

public function save(Request $request)

{

$request->validate([

'name' => 'required|string|max:255',

'email' => 'required|string|email|max:255|unique:users',

'password' => 'required|string|min:8|regex:/^\S\*$/',

], [

'email.unique' => 'This email is already registered.',

'password.min' => 'Password must be at least 8 characters long.',

]);

User::create([

'name' => $request->name,

'email' => $request->email,

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

]);

return redirect()->route('user.login')->with('success', 'User created successfully!');

}

}