<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
use App\DiscountType;  
use App\Bookingfact;  
use App\Reservation;  
use App\Lib\BookingCalculate;  
use SimpleSoftwareIO\QrCode\Facades\QrCode;  
use Illuminate\Mail\Message;  
use App\Lib\Calculator;  
class BookingController extends Controller  
{  
 public $firstForm, $discountTypes, $prices = [], $reservations,$bookingfact;  
  
 function \_\_construct (Request $request)  
 {  
 $this->firstForm = unserialize($request['firstForm']);  
 $this->discountTypes = DiscountType::orderBy('id', 'asc')->get();  
 }  
  
 public function confirm(Request $request)  
 {  
 $this->reservations = $request->except(['\_token', 'OK', 'firstForm']);  
 if ($this->calculatePrices () )  
 {  
*// dd('ss');* $this->preparationArray();  
 return view('View\_booking', $this->returnArray());  
 }  
*// dd('ss');  
// dd($this->reservations);* return view ('View\_reservation',$this->returnArray());  
 }  
 public function save(Request $request)  
 {  
 $this->reservations = unserialize( $request->input('reservations') );  
 $bookingfact = $request->except(['\_token', 'OK', 'reservations', 'firstForm']);  
 if (!$this->calculatePrices () )  
 {  
  
 return view ('View\_reservation',$this->returnArray());  
 }  
 $newBookingfact = new Bookingfact;  
 if (! $newBookingfact->validateAndSave($bookingfact) )  
 {  
 return view ('View\_booking',['reservations'=>$this->reservations, 'discountTypes'=>$this->discountTypes,   
 'validationError'=>$newBookingfact->errorMessages->keys(), 'firstForm'=>$this->firstForm,   
 'bookingfact'=>$bookingfact, 'prices'=>$this->prices ]);  
 }  
 foreach ($this->reservations as $reservation) Reservation::andSave($reservation);  
  
 return redirect()->route('booking.show',['array'=>$this->returnArray()]);  
 }  
 public function show(Request $request)  
 {  
 return view ('order',['array'=>$request->all(),'discountTypes'=>$this->discountTypes]);  
 }  
  
 public function calculatePrices ()  
 {  
 $hasErrors = false;  
 foreach ($this->reservations as &$reservation) {  
 $newReservation = new Reservation;  
 $reservation['validationError'] = [];  
  
 if (! $newReservation->validate($reservation) ){  
 $reservation['validationError'] = $newReservation->errorMessages->keys();  
 $hasErrors = true;  
  
 }  
 else {  
 $reservation['prices']=Calculator::calculate ($reservation);  
 array\_push($this->prices, Calculator::calculate ($reservation) );  
 }  
 }  
  
 if ($hasErrors) return false;  
 else return true;  
 }  
   
 private function preparationArray()  
 {  
 $i=0;  
 foreach($this->reservations as &$reservation) {  
 $reservation['price'] = $this->prices[$i++];  
 $reservation['number']=$i;  
 $reservation['guid']=(string)$this->getGuid($reservation);  
 $reservation['orderPrice']=array\_sum ($this->prices);  
 }  
 }  
 private function returnArray()  
 {  
 $data = [  
 'reservations' => $this->reservations,  
 'discountTypes' => $this->discountTypes,  
 'firstForm'=>$this->firstForm,  
 'prices'=>$this->prices,  
 ];  
 return $data;  
 }  
  
 private function getGuid($reservation)  
 {  
 $dataString='';  
 $dataString.=url()->full()  
 .$reservation['name']  
 .$reservation['discount']  
 .$reservation['fromdate']  
 .$reservation['fromtime']  
 .$reservation['todate']  
 .$reservation['totime']  
 .$reservation['price'];  
 $hash = strtoupper(md5($this->getRandomString($dataString)));  
 $hyphen = chr(45);  
 $openBracket=chr(123);  
 $closeBracket=chr(125);  
 $uuid = $openBracket  
 .substr($hash, 0, 8).$hyphen  
 .substr($hash, 8, 4).$hyphen  
 .substr($hash,12, 4).$hyphen  
 .substr($hash,16, 4).$hyphen  
 .substr($hash,20,12)  
 .$closeBracket;  
 return $uuid;  
 }  
  
 private function getRandomString($dataString)  
 {  
 $randomString = '';  
 for ($i = 0; $i < strlen($dataString)- 1; $i++) {  
 $randomString .= $dataString[rand(0, strlen($dataString)- 1)];  
 }  
 return $randomString;  
 }  
}