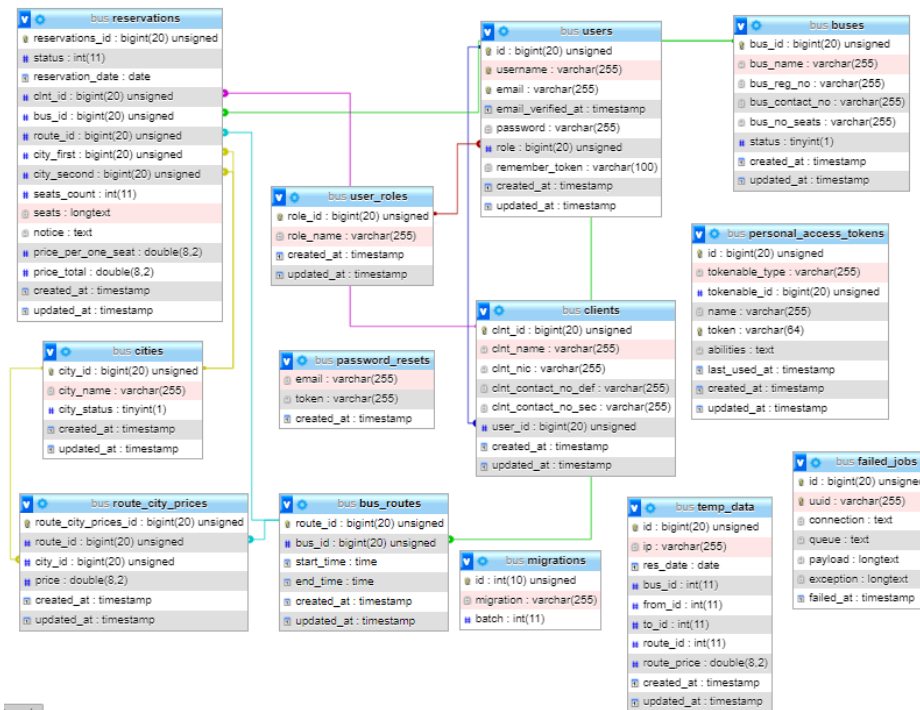


Criterion C: Development

Relationship and Tables



Commented [B1]: Relationship between tables are displayed representing Internal Structure.

Connection to database

```
APP_NAME="Jaiswal Bus Services"
APP_ENV=local
APP_KEY=base64:jbBpNgWDxkmEo9yBfI7dOqzE+GePfABpPCpHEpMC4Vc=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=bus
DB_USERNAME=root
DB_PASSWORD=
```

Database Connection
Database Name
Username
Password

Commented [B2]: Setup of Global variables and database connectivity.

Creating the configuration of connecting the database. It is in the .env file. DB_Connection is one that connects to MySQL. DB_HOST is for the type of server such as Localhost or Live server. In the local host, it will give the password as empty but on the live server, we have to give the password.

Setting SMTP server

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=jaiswal.bus.servises@gmail.com
MAIL_PASSWORD=9lvygpgjlr
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=jaiswal.bus.servises@gmail.com
MAIL_FROM_NAME="${APP_NAME}"
```

Configure smtp, so the application can send mails. It is setted in .env file which contains all the environment setups for an application.

Technique: Migrations and seeders

Explanation: The application contains migration and seeders. Migrations are created in the application so that it can easily be shared. Migration contains all the tables and relationships for the database of the application. The database can easily be migrated using php artisan migrate command. It contains a seeder for the admin which means it will create the data in the database, so that the admin can login using the id and password and access all the administrator functionalities. The seeders can be integrated with command php artisan db:seed.

Commented [B3]: Use of Database migration and seeders

```
public function up()
{
    Schema::create('buses', function (Blueprint $table) {
        $table->id('bus_id');
        $table->string('bus_name');
        $table->string('bus_reg_no');
        $table->string('bus_contact_no')->nullable();
        $table->string('bus_no_seats');
        $table->boolean('status')->default('0');
        $table->timestamps();
    });
}
```

```

public function up()
{
    Schema::create('reservations', function (Blueprint $table) {
        $table->id('reservations_id');
        $table->integer('status')->default(3); //2- approved, 3-pending, 4-reject
        $table->date('reservation_date');
        $table->unsignedBigInteger('clnt_id');
        $table->foreign('clnt_id')->references('clnt_id')->on('clients');
        $table->unsignedBigInteger('bus_id');
        $table->foreign('bus_id')->references('bus_id')->on('buses');
        $table->unsignedBigInteger('route_id');
        $table->foreign('route_id')->references('route_id')->on('bus_routes');
        $table->unsignedBigInteger('city_first');
        $table->foreign('city_first')->references('city_id')->on('cities');
        $table->unsignedBigInteger('city_second');
        $table->foreign('city_second')->references('city_id')->on('cities');
        $table->integer('seats_count');
        $table->json('seats');
        $table->text('notice')->nullable();
        $table->float('price_per_one_seat');
        $table->float('price_total');
        $table->timestamps();
    });
}

```

Reservation table
migration

```

public function run()
{
    User::create([
        'username' => 'Admin',
        'email' => 'admi@admin.com',
        'password' => Hash::make('password'),
        'role' => '1'
    ]);
}

```

Seeder for admin

Commented [B4]: Use of array data structure.

Technique: Laravel eloquent model

Explanation: In the applications queries are used throughout and this queries was built using an eloquent model which helps perform actions such as inserts, updates, deletes. The database has models which help to interact with tables. It automatically handles the models; the database queries can be performed without writing complex queries.

Commented [B5]: Use of queries to manage CRUD operations.

```

*/
protected function create(array $data)
{
    $user= User::create([
        'username' => $data['username'], //Username
        'email' => $data['email'], //Email
        'password' => Hash::make($data['password']), //hashed password
        'role' => '2', // role 2 defines that the user is cutomer
    ]);

    $client_info = Client::create([
        'clnt_name' => $data['username'], //client name
        'clnt_nic' => $data['nic'], //nic card number
        'clnt_contact_no_def' => $data['contact_no'], //contact number
        'user_id' => $user->id,
    ]);

    return $user;
}
}

```

Insert query for Customer registration

```

try{ // Inserting data for bus
    Bus::create([
        'bus_name' => request('name'),
        'bus_reg_no' => request('reg_no'),
        'bus_contact_no' => request('contact_no'),
        'bus_no_seats' => request('no_of_seats'),
        'status' => 1,
    ]);
}

```

Insert query for Bus Registration

```

try{
    City::create([
        'city_name' => $city_name,
    ]);
}

```

Insert query for City Registration

```

try{
    City::find($city_id)->update([
        'city_name' => $city_name,
    ]);
}

```

Update query to Update the City

```

try{// this adding query
    $route = BusRoute::create([
        'bus_id' => $bus_id,
        'start_time' => $start_time,
        'end_time' => $end_time,
    ]);
}

```

Insert query for new route

```

$route_data = RouteCityPrice::create([
    'route_id' => $route_id,
    'city_id' => $city_id,
    'price' => $price,
]);

```

Insert query for adding price for fare chart

```
$city_details = City::where('city_status', '1')->get();
```

Select query for showing the list of cities that are active

```
City::find($city_id)->update(['city_status' => '0',  
City::find($city_id)->update(['city_name' => $city_name,
```

Update query for deactivating the city Update query for updating details of city

```
try{  
    Bus::find($bus_id)->update(['status' => $status  
]);
```

Update query for updating the status of the bus (Status is for active and deactivating bus works as boolean)

```
$route_details = RouteCityPrice::where('route_id',  
request('route_id'))->orderByDesc('route_city_prices_id')->get();
```

Select query for showing list for specific route

```
$bus_details = Bus::where('status', '1')->get();
```

Select query for showing list of busses which are active

```
$route_details = BusRoute::all();  
$city_details = City::all();
```

Select query for storing values in variables

```
try{  
    Bus::find($bus_id)->update(['bus_name' => request('name'),  
    'bus_reg_no' => request('reg_no'),  
    'bus_contact_no' => request('contact_no'),  
    'bus_no_seats' => request('no_of_seats'),  
]);
```

Update query for updating the details of specific bus

```
try{
    Reservation::create([
        'reservation_date' => $res_date,
        'clnt_id' => $client_data->clnt_id,
        'bus_id' => $route_info->bus_id,
        'route_id' => $route_id,
        'city_first' => $from,
        'city_second' => $to,
        'seats_count' => $no_of_seats,
        'seats' => $seat,
        'notice' => $notice,
        'price_per_one_seat' => $route_price,
        'price_total' => $total_price,
    ]);
}
```

Reservation insert query

```
try{
    $id = request('route_price_city_id');
    RouteCityPrice::find($id)->delete();
}
```

Delete query for deleting city from

```
$res_details = Reservation::where('status', '3')->get();
```

Select query for showing the list of reservation of status 3

(status 3 refers to pending reservation)

```
$res_details = Reservation::where('status', '2')->get();
```

Select query for showing the list of reservation of status 2

(status 2 refers to approved reservation)

```
$res_details = Reservation::where('status', '4')->get();
```

Select query for showing the list of reservation of status 4

(status 4 refers to rejected reservation)

Update query for button action r

Update query for button action approve

```
try{
    Reservation::find($res_id)->update([
        'status' => '4',
    ]);
}
```

```
try{
    $res->update([
        'status' => '2',
    ]);
}
```

Update query for button action reserve

```
try{
    Reservation::find($res_id)->update([
        'status' => '3',
    ]);
}
```

```
$data = DB::select(DB::raw("SELECT (`reservation_date`)
AS Date,(`price_total`) AS Sales
FROM `reservations` WHERE status=2"));
```

Select query for selecting the data for approved reservation which is used to create graph

Technique: Laravel Validation and HTML 5 validation

Explanation: In this web-application the various validations are used to get appropriate data from the input and let the application function properly. The laravel validation and HTML 5 validations are used. The rules have been set and the validations are performed. If the validation rules pass the app works normally and if it fails an error response is thrown to the user. I have customized the errors using css. Validations such as required, unique, max, min, string, email and inte are used in the application.

Commented [B6]: Use of Input validation techniques. Both client and server side validations.

```
protected function validator(array $data) // These are used for the validation of the registration
{
    return Validator::make($data, [
        'username' => ['required', 'string', 'max:255', 'unique:users'], // If the user already exists with the username, you
        have to try another
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'], // If the user already exists with the email
        address, you have to try another
        'password' => ['required', 'string', 'min:8', 'confirmed'], // The minimum password is allowed at least 8 characters
    ]);
}
```

Jaiswal Bus Services

Customer

The username has already been taken.

cutomer@cutomer.com

The email has already been taken.

Contact No.

NIC

Password

The password must be at least 8 characters.

Password Confirm

Sign Up

Validations in registration of user

```
public function bus_reg(Request $request) // Registration for the Bus
{
    if(Auth::user()->role == 1){
        $request->validate([ // These are for the Validation of the bus registration form
            'name' => 'required',
            'reg_no' => 'required|unique:buses,bus_reg_no',
            'contact_no' => 'required',
            'no_of_seats' => 'required|integer',
        ]);
    }
}
```

Validations in bus registration

```
public function bus_reg(Request $request) // Registration for the Bus
{
    if(Auth::user()->role == 1){
        $request->validate([ // These are for the Validation of the bus registration form
            'name' => 'required',
            'reg_no' => 'required|unique:buses,bus_reg_no',
            'contact_no' => 'required',
            'no_of_seats' => 'required|integer',
        ]);
    }
}
```

Validations in registration of
BUS

ADD NEW BUS

Bus Name : <input type="text"/>	Registration Number : <input type="text"/>
The name field is required.	The reg no field is required.
Contact Number : <input type="text"/>	Number of Seats : <input type="text"/>
The contact no field is required.	The no of seats field is required.

```
public function search_route(Request $request)
{
    $request->validate([
        'city_1' => 'required',
        'city_2' => 'required',
        'res_date' => 'required'
    ]);
}
```

Validations in BUS Search

Select Start Destination <input type="text"/>	Select End Destination <input type="text"/>	dd-mm-yyyy <input type="text"/>	<input type="button" value="Find Buses"/>
The city 1 field is required.	The city 2 field is required.	The res date field is required.	



```

<div class="card">
  <div class="card-header">
    <h2>ADD CITY</h2>
  </div>
  <div class="card-body">
    <form action="{{ route('city_add') }}" method="POST">
      @csrf
      <div class="form-row">
        <div class="form-group col-md-8">
          <input type="text" name="city_name" id="" placeholder="City Name" class="form-control" required>
        </div>
        <div class="form-group col-md-4">
          <button class="btn btn-primary btn-block" type="submit">Add City</button>
        </div>
      </div>
    </form>
  </div>
</div>
</div>

```

Validations in registration of new city

ADD CITY

 Please fill in this field.

```

<div class="form-row">
  <div class="form-group col-md-6">
    <label for="">City</label>
    <select name="city_id" id="city_id" class="form-control" data-toggle="select" required>
      <option selected disabled>Select City</option>
      @foreach ($city_details as $city)
        <option value="{{ $city->city_id }}">{{ $city->city_name }}</option>
      @endforeach
    </select>
  </div>
  <div class="form-group col-md-6">
    <label for="">Price</label>
    <input type="number" name="price" id="price" value="" class="form-control" required>
  </div>
</div>
<div class="form-row">
  <div class="form-group col-md-4">
    <button type="button" class="btn btn-primary btn-block btn-submit">Add to Route</button>
  </div>
</div>
</div>

```

Validations in daily route

ADD DAILY ROUTINE

Select Bus

Select Bus

Start Time :

End Time :

Add Route

Please fill in this field.

ADD DAILY ROUTINE

Select Bus

Select Bus

Start Time :

End Time :

Add Route

Please fill in this field.

Technique: **Laravel email verification**

Explanation: In verification controller which states that after the user has been signed up, it will show the email verification to confirm the registration. RouteServiceProvider are defined in the route files. It generates a unique link and it is sent to the email taken from the user input. When the unique link is clicked and it matches then the user is logged in to the account and now has all the access which a user should have. When the link is clicked it stores the time and date when the link is clicked into the database and it is stored in users table at email verified at. It is to ensure that the email entered is genuine and of the user who intended to be the owner of the mail address. The email is sent from the SMP mail server configuration in the application.

Commented [B7]: Use of email verification in the form of separate function modules.

```
use VerifiesEmails;

/**
 * Where to redirect users after verification.
 *
 * @var string
 */
protected $redirectTo = RouteServiceProvider::HOME;

/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
{
    $this->middleware('auth');
    $this->middleware('signed')->only('verify');
    $this->middleware('throttle:6,1')->only('verify', 'resend');
}
```

Verify Your Email Address

Before proceeding, please check your email for a verification link. If you did not receive the email,
[click here to request another](#).

Verify Email Address Inbox X



Jaiswal Bus Services <jaiswal.bus.services@gmail.com>
to me

Jaiswal Bus Services

Hello!

Please click the button below to verify your email address.

Verify Email Address

If you did not create an account, no further action is required.

Regards,
Jaiswal Bus Services

If you're having trouble clicking the "Verify Email Address" button, copy and paste the URL below into your web browser: <http://127.0.0.1:8000/email/verify/6/56ea9db77df55050c8999b201002f623f20bb78c?expires=1642942873&signature=044ec5643fa3d54e38c2db204359f780bd0556145e95adeb6c3a938851250ca7>

© 2022 Jaiswal Bus Services. All rights reserved.

Technique: Laravel mail

Explanation: Laravel mail library is used to send the mail to the customer when the reservation is approved. The mail works as a notification for customers. The mail is sent using smtp which setup in the .env file. When reservation is approved, the application fetches data from the database and stores it in variables. Then the data is passed through the blade and the mail is sent to the customer who reserved the seats in the bus.

```

try{
    $res->update([
        'status' => '2',
    ]);
}catch(Throwable $e){
    return back()->with(['error' => 'Approval faild', 'error_type' => 'warning']);
}

$bus_name = Bus::find($res->bus_id)->bus_name;
$route_time = BusRoute::find($res->route_id)->start_time;
$client_user_id = Client::find($res->clnt_id)->user_id;
$client_email = User::find($client_user_id)->email;
$data = [
    'bus_name' => $bus_name,
    'total_price' => $res->price_total,
    'date_time' => $res->reservation_date.' '.$route_time,
    'no_of_seats' => $res->seats_count,
    'seats_no' => $res->seats,
];
Mail::to($client_email)->send(new ReservednMail($data)); // mail funtion

return back()->with(['success' => 'Reservation approval successful']);

```

When the admin will approve the reservation
it will mail the reservation details to the
customer's email

```
@component('mail::message')
```

```
# Bus Reservation.
```

Contents of html mail

```

You have successful reserved bus, The relevent details attach in below.<br>
Bus Name: {{ $data['bus_name'] }}<br>
Time/Date: {{ $data['date_time'] }}<br>
Number of seats: {{ $data['no_of_seats'] }}<br>
Seat No(s): @foreach($data['seats_no'] as $seat){{ $seat }}, @endforeach<br>
Total Price: {{ $data['total_price'] }}

```

```
Thanks,<br>
```

```
{{ config('app.name') }}
```

```
@endcomponent
```



Technique: use of multipart/formdata and File upload

Explanation: This technique is used mainly to personalize profile avatar, allowing customer to add a custom avatar for their profile. When the customer registers in the application it has the default profile avatar but it can be changed from the form. The form uses an encoding that allows the post method to send files to the server. When the file is sent, its original name is fetched from the function, and an update query is performed to change the avatar.

```
<form enctype="multipart/form-data" action="{{ route('avatar_com') }}" method="POST">
  @csrf
  <div class="form-row">
    <div class="form-group row-md-3">
      <input type="file" name="avatar" id="avatar"
        placeholder="avatar" style="" required>
    </div>
    <div class="form-group row-md-2">
      <button class="btn btn-primary btn-block " type="submit">Update Avatar</button>
    </div>
  </div>
</form>
```

```

public function avatar_com (Request $request){
    if ($request->hasFile('avatar')){
        $avatar = $request->file('avatar');
        $avatarexten = $avatar->getClientOriginalExtension();
        $avatarfir_name = $avatar->getClientOriginalName();
        $avatarname = $avatarfir_name;
        $userid = Auth::user()->id;
        $query = User::find($userid);
        $query->avatar = $avatarname;
        $query->save();
        return redirect('/update_ava');
    }
    return view('user.avatar.home', array('user' => Auth::user()->role == 2));
}

```

Commented [B8]: Use of control structures and use of built in methods from the Laravel library.

Default avatar

Customer's avatar



Choose file No file chosen

Update Avatar

Custom avatar

Customer's avatar



Choose file No file chosen

Update Avatar

Technique: nested if

Explanation: nested if has been used multiple times in the application. The nested if allowed the application to check multiple criteria in the application at once. In the code the nested if is implemented for the delete city.

```
public function delete_city(Request $request)
{
    if(Auth::user()->role == 1){

        $city_id = request('city_id');
        if($city_id != null){
            try{
                City::find($city_id)->update([
                    'city_status' => '0',
                ]);

            }catch(Throwable $e){
                dd($e);
                return back()->with(['error' => 'Request faild', 'error_type' => 'error']);
            }
            return back()->with(['success' => 'City deleted']);
        }else{
            return back()->with(['error' => 'Request faild', 'error_type' => 'error']);
        }
    }else{
        Auth::logout();
        return redirect('/home');
    }
}
```

Technique: CSRF Protection and TryCatch error handling

Explanation: Using CSRF Protection helps to avoid cross-site forgery attacks on web applications. It is a build-in plugin in laravel that generates tokens for each active session, the token verifies that the requests have been sent by an authenticated user. The trycatch is used for the error handling in the application. In the code catch block catches and handles try block exceptions.

Commented [B9]: Use of exception handling using Try catch block and CSRF variable used to protect from cross site request forgery.

```
try{
    City::find($city_id)->update([
        'city_status' => '0',
    ]);

}catch(Throwable $e){
    dd($e);
    return back()->with(['error' => 'Request faild', 'error_type' => 'error']);
}
```

```
<form action="{{ route('search_route') }}" method="POST">
    @csrf
```

Technique: loop and jquery

Explanation: the loop is used in the form to create multiple buttons in the seat selector with different names. Using the jquery the chosen seats names are stored into an array. This array is stored in database so the admin can see the requested seats.

Commented [B10]: Use of loops, control structure and use of imported functionality from 3rd party modules and/or libraries representing algorithmic thinking.

```
@for ($i = 22; $i <= 40; $i=$i+2)
@{
    if(count($reservation_info) > 0)
    @php $k=0;@endphp
    @foreach ($reservation_info as $res)
        @foreach ($res->seats as $r)
            @if($r == $i)
                @if($res->clnt_id == $my_client_id)
                    <div class="form-group col-md-1">
                        <button type="button" class="btn btn-primary btn-block checkitem" value="{{ $i }}" disabled>A{{ $i }}</button>
                    </div>
                @continue
            @else
                <div class="form-group col-md-1">
                    <button type="button" class="btn btn-warning btn-block checkitem" value="{{ $i }}" disabled>A{{ $i }}</button>
                </div>
                @continue
            @endif
        @endforeach
    @else
        @foreach ($reservation_info as $res)
            @foreach ($res->seats as $r)
                @if($r == $i)
                    @php $k =1; @endphp
                    @break
                @endif
            @endforeach
        @endforeach
    @endif
    @if($k == 0)
        <div class="form-group col-md-1">
            <button type="button" class="btn btn-success btn-block checkitem" value="{{ $i }}">A{{ $i }}</button>
        </div>
        @break
    @endif
    @endforeach
    @else
        <div class="form-group col-md-1">
            <button type="button" class="btn btn-success btn-block checkitem" value="{{ $i }}">A{{ $i }}</button>
        </div>
    @endif
@endfor
<div class="form-group col-md-1">
</div>
```



```

<script>
$(document).ready(function(){
    var val_array=[];
    $(".checkitem").on('click', function(){
        if($(this).on('click')){
            if($(this).hasClass('btn-checked')){
                $(this).removeClass('btn-checked');
                $(this).removeClass('btn-danger');
                $(this).addClass('btn-success');
                let thisid_val = $(this).val();

                val_array = jQuery.grep(val_array, function(value) {
                    return value != thisid_val;
                });
            }else{
                let thisid_val = $(this).val();
                val_array.push(thisid_val);
                $(this).removeClass('btn-success');
                $(this).addClass('btn-danger');
                $(this).addClass('btn-checked');
            }
        }
        // console.log(val_array);
    });

    $('#reserve_seat').click(function(){
        $('#seats').val(val_array);
        $('#reservation_form').submit();
    });
});
</script>

```

Script for storing the
chosen seats into the array

Commented [B11]: Use of JavaScript function with control structures.



REQUESTED SEATS
7, 8,

The array output, giving details which seats are requested by the user

Technique: Foreach loop and cloud tables

Explanation: Foreach loop has been used to return the data for the table. All the data has been returned to the HTML table body by which the data is visible to the admin of all the approved reservations. The cloud table is used to create the table which enables the search option, download options, print option, navigation options and sorting options. The same technique is used to generate all tables in application.

Commented [B12]: Creativity through form design

Commented [B13]: Good use of screenshots to demonstrate functionality.

```

@foreach ($res_details as $res)
    @php
        $client_details = App\Models\Client::find($res->clnt_id);
        $from_city = App\Models\City::find($res->city_first);
        $to_city = App\Models\City::find($res->city_second);
        $bus_details = App\Models\Bus::find($res->bus_id);
        $bus_route_details = App\Models\BusRoute::where('route_id', $res->route_id)->first();
    @endphp
    <tr>
        <th scope="row">{{ $res->reservations_id }}</th>
        <td>{{ $client_details->clnt_name }}</td>
        <td>{{ $client_details->clnt_contact_no_def }}</td>
        <td>{{ $bus_details->bus_name }} ---> {{ $bus_route_details->start_time }} -
            {{ $bus_route_details->end_time }}</td>
        <td>{{ $res->seats_count }}</td>
        <td>{{ $from_city->city_name }} to {{ $to_city->city_name }}</td>
        <td>{{ $res->reservation_date }}</td>
        <td>{{ $res->price_total }}</td>
        <td>
            <button class="btn btn-warning btn-sm btn-reverse" data-id="{{ $res->reservations_id }}"
                type="button">Reverse</button>
        </td>
    </tr>
@endforeach

```

Foreach loop for showing list of approved reservation

APPROVED RESERVATIONS

Download options **Print option** **Search option**

Copy CSV Excel PDF Print Show 10 rows **Sorting options** Search:

#	NAME	CONTACT NUMBER	BUS NAME/ ROUTE	REQUESTED SEATS	DESTINATION	DATE
14	revisionvilla@gmail.com	85205564654	Vijayant ---> 12:20:00 - 17:20:00	4	Khategaon (MP) to Double Choki (MP)	2022-01-21
13	revisionvilla@gmail.com	85205564654	Vijayant ---> 07:00:00 - 11:45:00	1	Khudel (MP) to Harda (MP)	2022-01-14
11	Customer	85206464	Vijayant ---> 12:20:00 - 17:20:00	20	Harda (MP) to Indore (MP)	2022-02-22
10	Customer	85206464	Vijayant ---> 12:20:00 - 17:20:00	60	Harda (MP) to Indore (MP)	2022-01-12
9	Customer	85206464	Vijayant ---> 12:20:00 - 17:20:00	43	Harda (MP) to Indore (MP)	2022-02-01
8	Customer	85206464	Vijayant ---> 12:20:00 - 17:20:00	43	Harda (MP) to Indore (MP)	2022-02-01
7	Customer	85206464	Vijayant ---> 12:20:00 - 17:20:00	40	Harda (MP) to Indore (MP)	2022-01-19
6	Customer	85206464	Vijayant ---> 12:20:00 - 17:20:00	50	Harda (MP) to Indore (MP)	2022-01-18
5	Customer	85206464	Vijayant ---> 07:00:00 - 11:45:00	5	Khudel (MP) to Double Choki (MP)	2022-01-04

Showing 1 to 9 of 9 entries

Navigation options Previous 1 Next

Commented [B14]: Annotation of codes as well as frontend User Interface.

Technique: Foreach loop and google charts

Explanation: data of sales are fetched from the database where status is 2(means the reservation approved by admin). It uses a foreach loop to make data in appropriate format for google chart and passes the variable to the blade. The data is returned to the graph script and the graph is displayed

Commented [B15]: Use of third party API

```

public function chart(){
    $data = DB::select(DB::raw("SELECT (`reservation_date`
    AS Date,(`price_total`) AS Sales
    FROM `reservations` WHERE status=2"));
    $result = '';
    foreach ($data as $value) {
        $result.="['".$value->Date."','".$value->Sales."'],";
    }
    // return $result;
    return view('owner.dashboard.home',compact('result'));
}

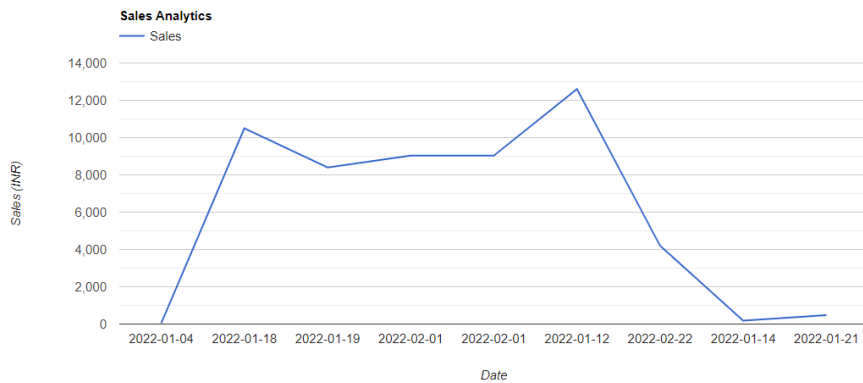
```

```

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
    google.charts.load('current', {'packages':['corechart']});
    google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
        var data = google.visualization.arrayToDataTable([
            ['Date', 'Sales'],
            <?php echo $result?>
        ]);
    }

```



Commented [B16]: Use of graphs and charts

Technique: foreach loop and linear search

Explanation: The linear search is implemented using the foreach loop for searching the routes. The algorithm compares the input given by the user and then compares it with all the data in the database to show the result. The algorithm searches the busses available in the routes and shows the results.

```
if (isset($first_route)) {  
    foreach($first_route as $first){  
        $route_details= App\Models\RouteCityPrice::where([[ 'route_id', $first->route_id],  
            [ 'city_id', $end_city]])->get();  
    }  
}
```

Commented [B17]: Use of linear search functionality.

Output of search algorithm

Bus List for Indore to Harda - 2022-02-18

#	BUS NAME	START TIME	END TIME	PRICE	AVAILABLE SEETS	OPTIONS
1	Vijayant	07:00:00	12:20:00	210	42	<input type="button" value="Contact"/> <input type="button" value="Reserve"/>

Technique: Foreach loop and arithmetic operators

Explanation: Foreach loop is used to fetch the data in the algorithm. The start city price and end city price are taken from the corresponding destination selected by the customer. Both prices are stored in a variable and then using subtraction operator to calculate the price per person for selected start destination and end destination. When the customer reserves the reservation then the form asks for a number of seats and then using the multiplication operator the total price is calculated.

```
@foreach ($route_details as $route)
    @php
        $start_city_price = App\Models\RouteCityPrice::where([[ 'city_id', $start_city],
        [ 'route_id', $route->route_id]])->first()->price;
        $end_city_price = App\Models\RouteCityPrice::where([[ 'city_id', $end_city],
        [ 'route_id', $route->route_id]])->first()->price;
        $route_price = ($end_city_price - $start_city_price);

        $bus_route_info = App\Models\BusRoute::find($route->route_id);
        $bus_details = App\Models\Bus::find($bus_route_info->bus_id);

        $booked_seat_count = App\Models\Reservation::where([[ 'route_id', $route->route_id],
        [ 'status', '!=', '4'], [ 'reservation_date', $res_date]])->sum('seats_count');
        $available_seats = ($bus_details->bus_no_seats - $booked_seat_count);
    @endphp

    $total_price = $route_price*$no_of_seats;
```

Badi

Hundiya

05-02-2022

Find Buses

#	ROUTE NAME	PRICE	ACTION	
11	Harda	210 INR	Edit	Delete
10	Hundiya	190 INR	Edit	Delete
9	Nemawar	180 INR	Edit	Delete
8	Khategaon	160 INR	Edit	Delete
7	Kanod	130 INR	Edit	Delete
6	Bijawar	110 INR	Edit	Delete
5	Badi	100 INR	Edit	Delete
4	Chapda	60 INR	Edit	Delete
3	Double Choki	40 INR	Edit	Delete
2	Khudel	30 INR	Edit	Delete
1	Indore	0 INR	Edit	Delete

Bus List for Badi to Hundiya - 2022-02-05

#	BUS NAME	START TIME	END TIME	PRICE	AVAILABLE SEETS	OPTIONS
1	Vijayant	07:00:00	12:20:00	90	42	<div>Contact</div> <div>Reserve</div>

From	To	Pice per person	Number of Seats		
Badi	Hundiya	90	3		
RESERVATOIN ID	BUS NAME/ ROUTE	REQUESTED SHEETS	DESTINATION	DATE	TOTAL PRICE
1	Vijayant	.	Hundiya	2022-02-13	270

Word count: 1036 words

Bibliography (MLA 8)

Commented [B18]: Reference material acknowledged.

"Charts | Google Developers/Interactive charts for browsers and mobile devices." *Google Developers*, developers.google.com/chart.

"Database driven applications in a snap." *CloudTables*, clouddtables.com/.

"DateDiff function." *Microsoft Support*, support.microsoft.com/en-us/office/datediff-function-e6dd7ee6-3d01-4531-905c-e24fc238f85f.

"DateDiff function." *Microsoft Support*, support.microsoft.com/en-us/office/datediff-function-e6dd7ee6-3d01-4531-905c-e24fc238f85f.

"Find the best answer to your technical question, help others answer theirs." *Stack Overflow*, stackoverflow.com/.

"A free CDN for Open Source fast, reliable, and automated serving ~ 147 billion requests / month CDN for npm CDN for GitHub CDN for WordPress." *JsDelivr*, www.jsdelivr.com/.

"Google Fonts | Google Developers/High-quality fonts to use on your web site." *Google Developers*, developers.google.com/fonts.

Otto, Mark, et al. "Build fast, responsive sites with Bootstrap/Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins." *Bootstrap*, getbootstrap.com/.

"The PHP Framework for Web Artisans/Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things." *Laravel - The PHP Framework For Web Artisans*, laravel.com/.

"Rapidly build modern websites without ever leaving your HTML. A utility-first CSS framework packed with classes like flex, pt-4, text-center and rotate-90 that can be composed to build any design, directly in your markup." *Rapidly Build Modern Websites Without Ever Leaving Your HTML*, tailwindcss.com/.

Tailblocks — Ready-to-use Tailwind CSS Blocks, tailblocks.cc/.

"Stunning free images & royalty free stock." *Attention Required!* | [Cloudflare](https://cloudflare.com/), pixabay.com/.
Unsplash. "The internet's source of freely-usable images. Powered by creators everywhere." *Beautiful Free Images & Pictures* | [Unsplash](https://unsplash.com/), unsplash.com/.