

REPORT

Hotel De Luna

Simple made perfect

Subject:

Internet Programming

Professor:

Sarvar Abdullaev

Inha University in Tashkent

Team members:

Farrukhbek Zokirov(U1810132) – Team Leader

Masrur Bekmirzaev(U1810156) – Member

Akbarjon Olimov(U1810150) – Member

Islomkhuja Akhrorov(U1810037) – Member

Abduvohid Isroilov(U1810006) – Member

Abrolov Mirsolikh(U1810277) – Member

Link to our github repo: <https://github.com/iuthub/ip-group-project-java-spring-boot>

Link to our website: <http://hotelhub.hopto.org> or

<http://hotelhub.francecentral.cloudapp.azure.com>

Superuser credentials: Admin cannot register themselves, therefore we provided you admin's credentials which stored in database.

Enter the following login and password for admin's login page (/admin):

Login: admin

Password: password

About our hotel

Our hotel is not that much different from other hotels. It has all the features that an every ordinary hotel has. It has all the facilities. When people are visiting to the website of Hotel De Luna, firstly they are headed to the Login page and they have register first. After having finished registration, they choose and book the room they want. Once the room is booked, person receives an email saying that they have successfully booked the room. What is more, customers are given an opportunity to give any feedback they want. In order to give a feedback, they don't need to register at all. They just have to fill the form and submit it. Once they submit their feedback, automatically the message to the customers' email will be delivered saying that staff of Hotel De Luna has received their feedback

Admin

```
bookings.blade.php x
1 extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5   <div class="row justify-content-center">
6     <div class="col-md-8">
7       <div class="card">
8         <div class="card-header">Bookings</div>
9
10         {{-- Taken from https://getbootstrap.com/docs/4.4/content/tables/ --}}
11         <table class="table table-bordered table-dark">
12           <thead>
13             <tr>
14               <th scope="col">#</th>
15               <th scope="col">Arrival</th>
16               <th scope="col">Book Time</th>
17               <th scope="col">Checkout</th>
18               <th scope="col">Breakfast</th>
19               <th scope="col">Number of nights</th>
20               <th scope="col">Comments</th>
21             </tr>
22           </thead>
23           <tbody>
24             @foreach($bookings as $booking)
25               <tr>
26                 <th scope="row"></th>
27                 <td>{{ $booking->arrival }}</td>
28                 <td>{{ $booking->book_time }}</td>
29                 <td>{{ $booking->checkout }}</td>
30                 <td>{{ $booking->breakfast }}</td>
31                 <td>{{ $booking->night }}</td>
32                 <td>{{ $booking->comment }}</td>
33               </tr>
34             @endforeach
35           </tbody>
36         </table>
37       </div>
38     </div>
39   </div>
40 </div>
41 @endsection
42
```

In this function, "foreach" loop is used to make blade file work. \$bookings is an array and \$booking is an one object of that array. This booking is coming from a variable in controller. The {{ }} brackets are used to access to those variables and to differentiate between html code and php code. For example, {{ \$booking-> arrival }} means to take the attribute of arrival in the \$booking object. @endforeach means the end of foreach loop.

```

1 @extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5     <div class="row justify-content-center">
6         <div class="col-md-8">
7             <div class="card">
8                 <div class="card-header">Admin Dashboard</div>
9
10                <div class="card-body">
11                    @if (session('status'))
12                        <div class="alert alert-success" role="alert">
13                            {{ session('status') }}
14                        </div>
15                    @endif
16
17                    You are logged in as <strong>ADMIN</strong>!
18                </div>
19            </div>
20        </div>
21    </div>
22 </div>
23 @endsection
24

```

In this function, @if and @endif are being used and between them there is css part. It is making sure the status of css whether it is TRUE or FALSE. It is also showing admin whether cookies are included or not

Layouts

```

22 <body>
23 <div id="app">
24 <nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
25 <div class="container">
26 <a class="navbar-brand" href="{{ url('/') }}">
27 {{ config('app.name', 'Laravel') }}
28 </a>
29 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="
30 navbarSupportedContent" aria-expanded="false" aria-label="{{ __('Toggle navigation') }}">
31 <span class="navbar-toggler-icon"></span>
32 </button>
33 <div class="collapse navbar-collapse" id="navbarSupportedContent">
34 <!-- Left Side Of Navbar -->
35 <ul class="navbar-nav mr-auto">
36
37 </ul>
38
39 <!-- Right Side Of Navbar -->
40 <ul class="navbar-nav ml-auto">
41 <!-- Authentication Links -->
42 @guest
43 <li class="nav-item">
44 <a class="nav-link" href="{{ route('login') }}">{{ __('Login') }}</a>
45 </li>
46 @if (Route::has('register'))
47 <li class="nav-item">
48 <a class="nav-link" href="{{ route('register') }}">{{ __('Register') }}</a>
49 </li>
50 @endif

```

```

51 @else
52 <li class="nav-item dropdown">
53 <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown" aria-haspopup="
54 true" aria-expanded="false" v-pre>
55 {{ Auth::user()->name }} <span class="caret"></span>
56 </a>
57 <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
58 <a class="dropdown-item" href="{{ route('logout') }}"
59 onclick="event.preventDefault();
60 document.getElementById('logout-form').submit();"
61 >{{ __('Logout') }}
62 </a>
63
64 <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;"
65 @csrf
66 </form>
67 </div>
68 </li>
69 @endguest
70 </ul>
71 </div>
72 </nav>
73
74 <main class="py-4">
75 @yield('content')
76 </main>
77 </div>
78 </body>
79 </html>
80
81

```

In this function, we have an URL {{url('/')}. This gives the link to the homeroad. {{ config('app.name', 'Laravel') }} means that Laravel is showing the text. There is also {{ __('Toggle navigation') }} which is used for localization. @guest part means user has been registered yet. @if (Route::has('register')) means if there is a route called "register" among other routes then show the button called "register" and giving route register as its link. {{ Auth::user()->name }} finds the name of the user by token and shows it. {{ route('logout') }} is link of logout. {{ __('Logout') }} shows the Logout. @csrf is used to know the token of the user

```

22 <body>
23 <div id="app">
24 <nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
25 <div class="container">
26 <a class="navbar-brand" href="{{ url('/') }}">
27 {{ config('app.name', 'Laravel') }}
28 </a>
29 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="
30 navbarSupportedContent" aria-expanded="false" aria-label="{{ __('Toggle navigation') }}">
31 <span class="navbar-toggler-icon"></span>
32 </button>
33 <div class="collapse navbar-collapse" id="navbarSupportedContent">
34 <!-- Left Side Of Navbar -->
35 <ul class="navbar-nav mr-auto">
36
37 </ul>
38
39 <!-- Right Side Of Navbar -->
40 <ul class="navbar-nav ml-auto">
41 <!-- Authentication Links -->
42 @guest('admin')
43 <li class="nav-item">
44 <a class="nav-link" href="{{ route('login') }}">{{ __('Login') }}</a>
45 </li>
46 @else
47 <li class="nav-item dropdown">
48 <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown" aria-haspopup="
49 true" aria-expanded="false" v-pre>
50 {{ Auth::user()->name }} <span class="caret"></span>
51 </a>
52 <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
53 <a class="dropdown-item" href="{{ route('logout') }}"
54 onclick="event.preventDefault();
55 document.getElementById('logout-form').submit();"
56 >{{ __('Logout') }}
57 </a>
58
59 <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;"
60 @csrf
61 </form>
62 </div>
63 </li>
64 @endguest

```

```

65 </ul>
66 </div>
67 </div>
68 </nav>
69
70 <main class="py-4">
71 @yield('content')
72 </main>
73 </div>
74 </body>
75 </html>
76

```

In this function, view of admin page is shown. Here it is saying that if admin page is guest, then show Login button, else just show the name of the user. Other functions are same as the previous ones.

Passwords

```

confirm.blade.php
1 @extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5   <div class="row justify-content-center">
6     <div class="col-md-8">
7       <div class="card">
8         <div class="card-header">{{ __('Confirm Password') }}</div>
9
10        <div class="card-body">
11          {{ __('Please confirm your password before continuing.') }}
12
13          <form method="POST" action="{{ route('password.confirm') }}">
14            @csrf
15
16            <div class="form-group row">
17              <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>
18
19              <div class="col-md-6">
20                <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password"
21                  required autocomplete="current-password">
22
23                @error('password')
24                  <span class="invalid-feedback" role="alert">
25                    <strong>{{ $message }}</strong>
26                  </span>
27                @enderror
28              </div>
29            </div>

```

```

confirm.blade.php
30    <div class="form-group row mb-0">
31      <div class="col-md-8 offset-md-4">
32        <button type="submit" class="btn btn-primary">
33          {{ __('Confirm Password') }}
34        </button>
35
36        @if (Route::has('password.request'))
37          <a class="btn btn-link" href="{{ route('password.request') }}">
38            {{ __('Forgot Your Password?') }}
39          </a>
40        @endif
41      </div>
42    </div>
43  </form>
44 </div>
45 </div>
46 </div>
47 </div>
48 </div>
49 @endsection
50

```

It is showing that it is extending from layouts.app. @section is used in the content. @error means if the password is incorrect or there is not enough symbols, then this functions shows error. @if parts checks whether there is a route called password. request and if such route exists, then show "Forgot your password?" and it is giving its link by route.

```

email.blade.php
1 @extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5   <div class="row justify-content-center">
6     <div class="col-md-8">
7       <div class="card">
8         <div class="card-header">{{ __('Reset Password') }}</div>
9
10        <div class="card-body">
11          @if (session('status'))
12            <div class="alert alert-success" role="alert">
13              {{ session('status') }}
14            </div>
15          @endif
16

```



```

16
17
18 <form method="POST" action="{{ route('password.email') }}">
19     @csrf
20
21     <div class="form-group row">
22         <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>
23
24         <div class="col-md-6">
25             <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{
26                 old('email') }}" required autocomplete="email" autofocus>
27
28             @error('email')
29                 <span class="invalid-feedback" role="alert">
30                     <strong>{{ $message }}</strong>
31                 </span>
32             @enderror
33         </div>
34
35         <div class="form-group row mb-0">
36             <div class="col-md-6 offset-md-4">
37                 <button type="submit" class="btn btn-primary">
38                     {{ __('Send Password Reset Link') }}
39                 </button>
40             </div>
41         </div>
42     </form>
43 </div>
44 </div>
45 </div>
46 </div>
47 @endsection
48

```

In this function, it is checking session. If the person has entered to his page, then his status will be shown. @error('email') checks the error whether email is entered correctly or not. If there is error, function will send password reset link

```

1 |@extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5     <div class="row justify-content-center">
6         <div class="col-md-8">
7             <div class="card">
8                 <div class="card-header">{{ __('Reset Password') }}</div>
9
10                <div class="card-body">
11                    <form method="POST" action="{{ route('password.update') }}">
12                        @csrf
13
14                        <input type="hidden" name="token" value="{{ $token }}">
15
16                        <div class="form-group row">
17                            <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>
18
19                            <div class="col-md-6">
20                                <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{
21                                    $email ?? old('email') }}" required autocomplete="email" autofocus>
22
23                                @error('email')
24                                    <span class="invalid-feedback" role="alert">
25                                        <strong>{{ $message }}</strong>
26                                    </span>
27                                @enderror
28                            </div>
29                        </div>

```

```

30 <div class="form-group row">
31 <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>
32
33 <div class="col-md-6">
34 <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password"
    required autocomplete="new-password">
35
36 @error('password')
37 <span class="invalid-feedback" role="alert">
38 <strong>{{ $message }}</strong>
39 </span>
40 @enderror
41 </div>
42 </div>
43
44 <div class="form-group row">
45 <label for="password-confirm" class="col-md-4 col-form-label text-md-right">{{ __('Confirm Password') }}</label>
46
47 <div class="col-md-6">
48 <input id="password-confirm" type="password" class="form-control" name="password_confirmation" required autocomplete="
    new-password">
49 </div>
50 </div>
51
52 <div class="form-group row mb-0">
53 <div class="col-md-6 offset-md-4">
54 <button type="submit" class="btn btn-primary">
55 {{ __('Reset Password') }}
56 </button>
57 </div>
58 </div>
59 </form>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 @endsection
66

```

In this function, it is using @token. Then variables of email "label frame" is being used and if label frame equals to null, then old email will be taken. Old email means email in any kind of browser and sends request to its server. Other functions are same as previous ones.

View

```

1 @extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5 <div class="row justify-content-center">
6 <div class="col-md-8">
7 <div class="card">
8 <div class="card-header">Dashboard</div>
9
10 <div class="card-body">
11 @if (session('status'))
12 <div class="alert alert-success" role="alert">
13 {{ session('status') }}
14 </div>
15 @endif
16
17 You are logged in as <strong>USER</strong>!
18 </div>
19 </div>
20 </div>
21 </div>
22 </div>
23 @endsection
24

```

This function finds the session and shows it.

```
welcome.blade.php x
66 <body>
67 <div class="flex-center position-ref full-height">
68     @if (Route::has('login'))
69         <div class="top-right links">
70             @auth
71                 <a href="{{ url('/home') }}">Home</a>
72             @else
73                 <a href="{{ route('login') }}">Login</a>
74             @if (Route::has('register'))
75                 <a href="{{ route('register') }}">Register</a>
76             @endif
77         </div>
78     @endif
79 </div>
80 <div class="content">
81     <div class="title m-b-md">
82         Hotel Booking Service
83     </div>
84 </div>
85 </div>
86 </body>
87 </html>
88
89
90
91
92
```

In this function, there is a route called register, and its link will be shown. If you press the link, it will go to the register route. There is also login route. If there is a login route, link will be sent to it if and only if user has not been registered yet. If user has already logged in, then he will be directed to the homepage.

View/auth

```
admin-login.blade.php x
1 @extends('layouts.app-admin')
2
3 @section('content')
4 <div class="container">
5     <div class="row justify-content-center">
6         <div class="col-md-8">
7             <div class="card">
8                 <div class="card-header">{{ __('ADMIN Login') }}</div>
9
10                <div class="card-body">
11                    <form method="POST" action="{{ route('adminLoginSubmit') }}">
12                        @csrf
13
14                        <div class="form-group row">
15                            <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>
16
17                            <div class="col-md-6">
18                                <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}" required autocomplete="email" autofocus>
19
20                                @error('email')
21                                    <span class="invalid-feedback" role="alert">
22                                        <strong>{{ $message }}</strong>
23                                    </span>
24                                @enderror
25                            </div>
26                        </div>
27
```

```
admin-login.blade.php x
27
28 <div class="form-group row">
29 <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>
30
31 <div class="col-md-6">
32 <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password"
    required autocomplete="current-password">
33
34 @error('password')
35 <span class="invalid-feedback" role="alert">
36 <strong>{{ $message }}</strong>
37 </span>
38 @enderror
39 </div>
40 </div>
41
42 <div class="form-group row mb-0">
43 <div class="col-md-8 offset-md-4">
44 <button type="submit" class="btn btn-primary">
45 {{ __('Login') }}
46 </button>
47 </div>
48 </div>
49 </form>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 @endsection
56
```

Function is being extended from layouts.app-admin. @csrf, @error, @enderror functions are used. @csrf is used to know the token of the user, @error is used to check whether email is correct or not. @enderror ends the error.

```
login.blade.php x
1 @extends('layouts.main')
2
3 @section('title', 'De Luna')
4 @section('title2', 'De Luna')
5 @section('title3', 'De Luna')
6
7 @section('paragraph', 'Sign In')
8 @section('paragraph2', 'Sign In')
9 @section('paragraph3', 'Sign In')
10
11 @section('content')
12 <div class="site-section bg-light">
13 <div class="container">
14 <div class="row justify-content-center">
15 <div class="col-md-8">
16 <div class="card">
17 <div class="card-header">{{ __('Login') }}</div>
18
19 <div class="card-body">
20 <form method="POST" action="{{ route('login') }}">
21 @csrf
22
23 <div class="form-group row">
24 <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>
25
26 <div class="col-md-6">
27 <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{
    old('email') }}" required autocomplete="email" autofocus>
28
29 @error('email')
30 <span class="invalid-feedback" role="alert">
31 <strong>{{ $message }}</strong>
32 </span>
33 @enderror
34 </div>
35 </div>
36
37 <div class="form-group row">
38 <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>
39
40 <div class="col-md-6">
41 <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password"
    required autocomplete="current-password">
42
```



```
register.blade.php x
1 |@extends('layouts.main')
2
3 |@section('title', 'De Luna')
4 |@section('title2', 'De Luna')
5 |@section('title3', 'De Luna')
6
7 |@section('paragraph', 'Sign Up')
8 |@section('paragraph2', 'Sign Up')
9 |@section('paragraph3', 'Sign Up')
10
11 |@section('content')
12 |<div class="site-section bg-light">
13 |    <div class="container">
14 |        <div class="row justify-content-center">
15 |            <div class="col-md-8">
16 |                <div class="card">
17 |                    <div class="card-header">{{ __('Register') }}</div>
18
19 |                    <div class="card-body">
20 |                        <form method="POST" action="{{ route('register') }}">
21 |                            @csrf
22
23 |                            <div class="form-group row">
24 |                                <label for="name" class="col-md-4 col-form-label text-md-right">{{ __('Full Name') }}</label>
25
26 |                                <div class="col-md-6">
27 |                                    <input id="name" type="text" class="form-control @error('name') is-invalid @enderror" name="name" value="{{
28 |                                        old('name') }}" required autocomplete="name" autofocus>
29
30 |                                    @error('name')
31 |                                        <span class="invalid-feedback" role="alert">
32 |                                            <strong>{{ $message }}</strong>
33 |                                        </span>
34 |                                    @enderror
35 |                                </div>
36 |                            </div>
```

```
register.blade.php x
36
37 |
38 |    <div class="form-group row">
39 |        <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>
40
41 |        <div class="col-md-6">
42 |            <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{
43 |                old('email') }}" required autocomplete="email">
44
45 |            @error('email')
46 |                <span class="invalid-feedback" role="alert">
47 |                    <strong>{{ $message }}</strong>
48 |                </span>
49 |            @enderror
50 |        </div>
51 |    </div>
52
53 |    <div class="form-group row">
54 |        <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>
55
56 |        <div class="col-md-6">
57 |            <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password"
58 |                required autocomplete="new-password">
59
60 |            @error('password')
61 |                <span class="invalid-feedback" role="alert">
62 |                    <strong>{{ $message }}</strong>
63 |                </span>
64 |            @enderror
65 |        </div>
66 |    </div>
```

```
register.blade.php x
64
65 |
66 |    <div class="form-group row">
67 |        <label for="password-confirm" class="col-md-4 col-form-label text-md-right">{{ __('Confirm Password') }}</label>
68
69 |        <div class="col-md-6">
70 |            <input id="password-confirm" type="password" class="form-control" name="password_confirmation" required
71 |                autocomplete="new-password">
72 |        </div>
73 |    </div>
74
75 |    <div class="form-group row mb-0">
76 |        <div class="col-md-6 offset-md-4">
77 |            <button type="submit" class="btn btn-primary">
78 |                {{ __('Register') }}
79 |            </button>
80 |        </div>
81 |    </div>
82 | </form>
83 | </div>
84 | </div>
85 | </div>
86 | </div>
87 | @endsection
88
```

All the functions are same as previous ones

```
verify.blade.php
1 @extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5   <div class="row justify-content-center">
6     <div class="col-md-8">
7       <div class="card">
8         <div class="card-header">{{ __('Verify Your Email Address') }}</div>
9
10        <div class="card-body">
11          @if (session('resent'))
12            <div class="alert alert-success" role="alert">
13              {{ __('A fresh verification link has been sent to your email address.') }}
14            </div>
15          @endif
16
17          {{ __('Before proceeding, please check your email for a verification link.') }}
18          {{ __('If you did not receive the email') }},
19          <form class="d-inline" method="POST" action="{{ route('verification.resend') }}">
20            @csrf
21            <button type="submit" class="btn btn-link p-0 m-0 align-baseline">{{ __('click here to request another') }}</button>
22          </form>
23        </div>
24      </div>
25    </div>
26  </div>
27 </div>
28 @endsection
29
```

In this function, session('resent') looks for TRUE, FALSE. If the user has been in session nearly, "A fresh verification link has been sent to your email address" will be displayed, if not then "Before proceeding, please check your email for a verification link" will be displayed.

Code of first header

```

1  <?php
2
3  namespace App;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Foundation\Auth\User as Authenticatable;
7  use Illuminate\Notifications\Notifiable;
8
9  class Admin extends Authenticatable
10 {
11     use Notifiable;
12
13     protected $guard = 'admin';
14     /**
15      * The attributes that are mass assignable.
16      *
17      * @var array
18      */
19     protected $fillable = [
20         'name', 'email', 'password',
21     ];
22
23     /**
24      * The attributes that should be hidden for arrays.
25      *
26      * @var array
27      */
28     protected $hidden = [
29         'password', 'remember_token',
30     ];
31
32     /**
33      * The attributes that should be cast to native types.
34      *
35      * @var array
36      */
37     protected $casts = [
38         'email_verified_at' => 'datetime',
39     ];
40 }
41

```

Admin.php:

\$guarded sets which fields cannot be mass assigned so when you *fill()* with properties that are guarded they won't get saved in database.

\$fillable property specifies which attributes should be mass-assignable. This can be set at the class or instance level.

\$hidden are the fields that are hidden from passing to model json data.

\$casts property on your model provides a convenient method of converting attributes to common data types. The *\$casts* property should be an array where the key is the name of the attribute being cast and the value is the type you wish to cast the column.


```

1 |?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Booking extends Model
8 {
9     protected $fillable = ['arrival', 'book_time', 'checkout', 'breakfast', 'night', 'comment'];
10
11     public function user()
12     {
13         return $this->belongsTo('App\User');
14     }
15
16     public function payment()
17     {
18         return $this->hasOne('App\Payment');
19     }
20
21     public function room()
22     {
23         return $this->hasOne('App\Room');
24     }
25
26     public function cancellation()
27     {
28         return $this->hasOne('App\Cancellation');
29     }
30
31     public function booktype()
32     {
33         return $this->hasOne('App\BookType');
34     }
35
36
37 }

```

Booking.php:

\$fillable [] – this array allows users to fill the form which is given in booking process.

function user () – this function indicates that class Booking belongs to class User or has connection to class User.

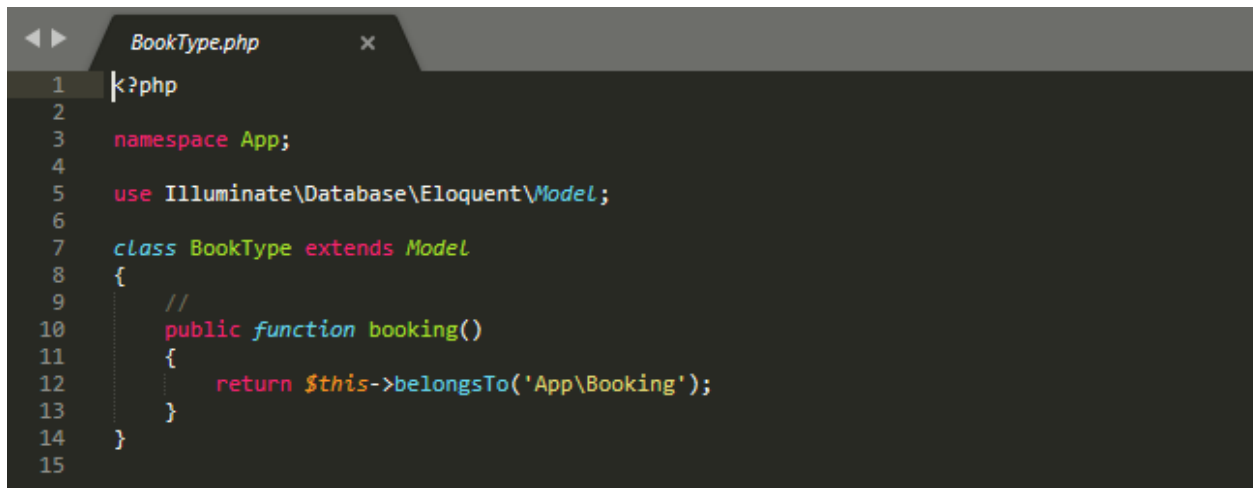
function payment () works for to connect booking table with payment table.

function room () works for to connect booking table with room table.

function cancellation () works for to connect booking table with cancellation table.

function bookType () works for to connect booking table with booktype table.

functions *payment, room, cancellation, booktype* which has one to one (*hasOne*) connection with class Booking. this *hasOne ()* connection allows us to use functions only once in the process. For example if you want to cancel the booking you just have to cancel once.



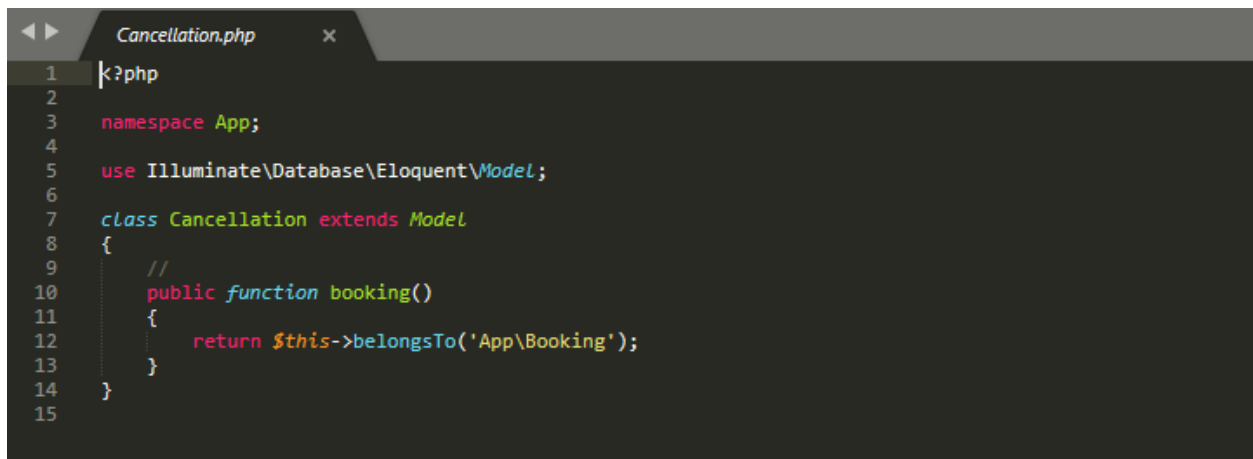
```

1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class BookType extends Model
8  {
9      //
10     public function booking()
11     {
12         return $this->belongsTo('App\Booking');
13     }
14 }
15

```

BookType.php:

function booking () – this function indicates that class BookType belongs to class Booking or has connection to class Booking



```

1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Cancellation extends Model
8  {
9      //
10     public function booking()
11     {
12         return $this->belongsTo('App\Booking');
13     }
14 }
15

```

Cancellation.php:

function booking () – this function indicates that class Cancellation belongs to class Booking or has connection to class Booking

```
Hotel.php
1 |<?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Hotel extends Model
8 {
9     protected $fillable = ['name', 'description', 'address'];
10 }
11
```

Hotel.php:

\$fillable [] – this variable allows users to fill the form which is given in booking process.

```
Payment.php
1 |<?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Payment extends Model
8 {
9     //
10     public function booking()
11     {
12         return $this->belongsTo('App\Booking');
13     }
14 }
15
```

Payment.php:

function booking () – this function indicates that class Payment belongs to class Booking or has connection to class Booking.

```
Room.php
1 |<?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Room extends Model
8 {
9     //
10    public function booking()
11    {
12        return $this->belongsTo('App\Booking');
13    }
14 }
15
```

Room.php:

function booking () – this function indicates that class Room belongs to class Booking or has connection to class Booking.

```
User.php
1 |<?php
2
3 namespace App;
4
5 use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use Illuminate\Foundation\Auth\User as Authenticatable;
7 use Illuminate\Notifications\Notifiable;
8
9 class User extends Authenticatable
10 {
11     use Notifiable;
12
13     /**
14      * The attributes that are mass assignable.
15      *
16      * @var array
17      */
18     public function bookings()
19     {
20         return $this->hasMany('App\Booking');
21     }
22
23     protected $fillable = [
24         'name', 'email', 'password',
25     ];
26
27     /**
28      * The attributes that should be hidden for arrays.
29      *
30      * @var array
31      */
32     protected $hidden = [
33         'password', 'remember_token',
34     ];
35
36     /**
37      * The attributes that should be cast to native types.
38      *
39      * @var array
40      */
41     protected $casts = [
42         'email_verified_at' => 'datetime',
43     ];
44
45 }
46
47
48
49
```

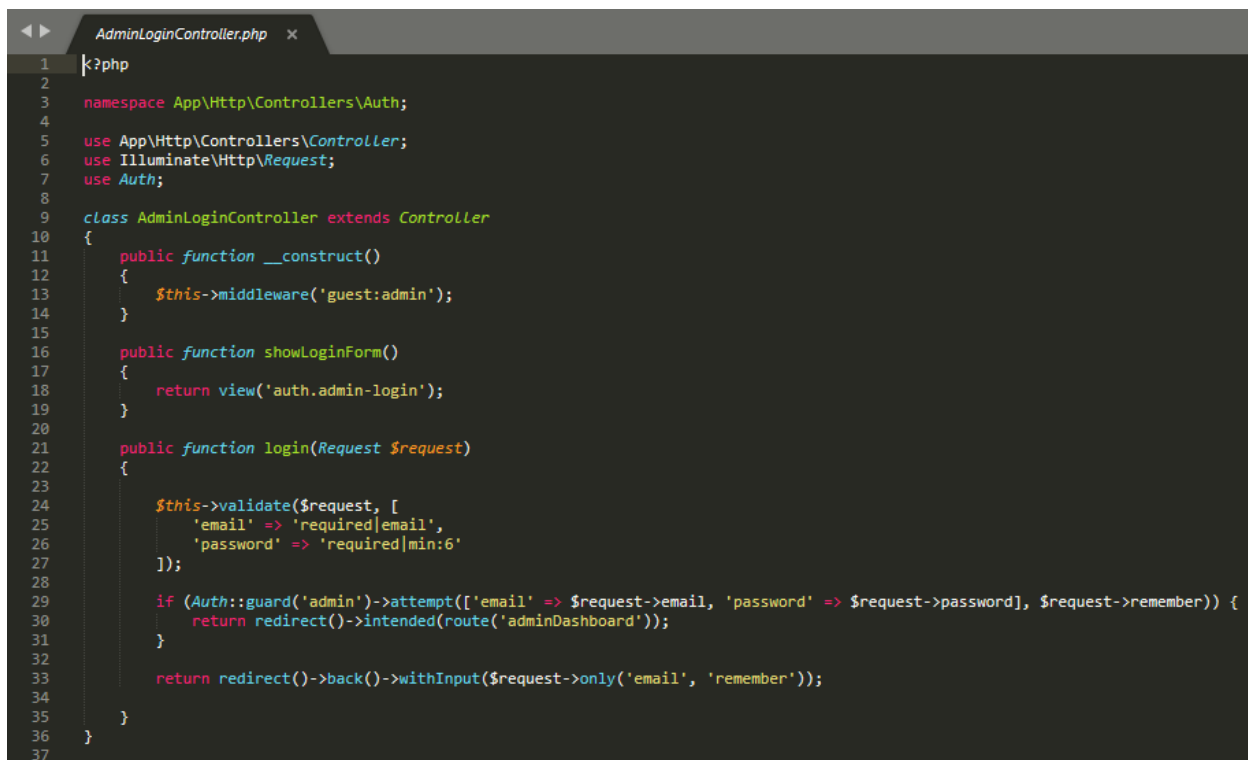
User.php:

function booking () – this function indicates that it has one to many connection (hasMany) with class Booking. hasMany () connection allows us to use class Booking in many times by users in process. For example while one user filling booking form, other users also can fill the same booking form. It means that many users can use the same booking form.

\$fillable [] – this variable allows users to fill the form which is given in booking process.

\$hidden are the fields that are hidden from passing to model json data.

\$casts property on your model provides a convenient method of converting attributes to common data types. The *\$casts* property should be an array where the key is the name of the attribute being cast and the value is the type you wish to cast the column.



```
1 <?php
2
3 namespace App\Http\Controllers\Auth;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use Auth;
8
9 class AdminLoginController extends Controller
10 {
11     public function __construct()
12     {
13         $this->middleware('guest:admin');
14     }
15
16     public function showLoginForm()
17     {
18         return view('auth.admin-login');
19     }
20
21     public function login(Request $request)
22     {
23
24         $this->validate($request, [
25             'email' => 'required|email',
26             'password' => 'required|min:6'
27         ]);
28
29         if (Auth::guard('admin')->attempt(['email' => $request->email, 'password' => $request->password], $request->remember)) {
30             return redirect()->intended(route('adminDashboard'));
31         }
32
33         return redirect()->back()->withInput($request->only('email', 'remember'));
34     }
35 }
36
37
```

AdminLoginController.php:

function __construct () – this constructor calls function *middleware* and this function checks whether guest is registered as admin or not.

function showLoginForm () – this function shows path to *admin-login.blade.php* inside the *auth* file.

function login () – this function responsible for post request and inside of function there is property *validate* which checks this entered email is in form of required email form and ensures that entered password is not less than 6 characters. If the entered email and password belong to the admin, then directed the admin profile. If the entered email and password not match with any admin's data then it will be asked re-enter the email and password.

```
ConfirmPasswordController.php x
1 |<?php
2
3 | namespace App\Http\Controllers\Auth;
4
5 | use App\Http\Controllers\Controller;
6 | use App\Providers\RouteServiceProvider;
7 | use Illuminate\Foundation\Auth\ConfirmsPasswords;
8
9 | class ConfirmPasswordController extends Controller
10 | {
11 |     /**
12 |      * Confirm Password Controller
13 |      *
14 |      * This controller is responsible for handling password confirmations and
15 |      * uses a simple trait to include the behavior. You're free to explore
16 |      * this trait and override any functions that require customization.
17 |      */
18 |
19 |     use ConfirmsPasswords;
20 |
21 |     /**
22 |      * Where to redirect users when the intended url fails.
23 |      *
24 |      * @var string
25 |      */
26 |     protected $redirectTo = RouteServiceProvider::HOME;
27 |
28 |     /**
29 |      * Create a new controller instance.
30 |      *
31 |      * @return void
32 |      */
33 |     public function __construct()
34 |     {
35 |         $this->middleware('auth');
36 |     }
37 | }
38
39
40
41
```

ConfirmPasswordController.php:

RouteServiceProvider – this property works for when the intended url fails, then it will be redirected to HOME (index.blade.php).

function __construct () – this constructor calls function *middleware*. This function checks token if there no token then directs to home.

```

1  k?php
2
3  namespace App\Http\Controllers\Auth;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Foundation\Auth\SendsPasswordResetEmails;
7
8  class ForgotPasswordController extends Controller
9  {
10
11      /*
12       |-----
13       | Password Reset Controller
14       |-----
15       |
16       | This controller is responsible for handling password reset emails and
17       | includes a trait which assists in sending these notifications from
18       | your application to your users. Feel free to explore this trait.
19       |
20       */
21
22      use SendsPasswordResetEmails;
23
24  }

```

ForgotPasswordController.php:

This class *ForgotPasswordController* works for to restore forgotten password and send it to user's email. In order to accomplish this task this works this class uses *SendsPasswordResetEmails*.

```

1  k?php
2
3  namespace App\Http\Controllers\Auth;
4
5  use App\Http\Controllers\Controller;
6  use App\Providers\RouteServiceProvider;
7  use Illuminate\Foundation\Auth\AuthenticatesUsers;
8
9  class LoginController extends Controller
10 {
11
12     /*
13      |-----
14      | Login Controller
15      |-----
16      |
17      | This controller handles authenticating users for the application and
18      | redirecting them to your home screen. The controller uses a trait
19      | to conveniently provide its functionality to your applications.
20      |
21      */
22
23     use AuthenticatesUsers;
24
25     /**
26      * Where to redirect users after login.
27      *
28      * @var string
29      */
30     protected $redirectTo = RouteServiceProvider::HOME;
31
32     /**
33      * Create a new controller instance.
34      *
35      * @return void
36      */
37     public function __construct()
38     {
39         $this->middleware('guest')->except('logout');
40     }
41 }

```

LoginControlller.php:

RouteServiceProvider – this property works for to redirected user after login.

function __construct () – this constructor calls function *middleware*. This function checks person who is using web page, user or just guest. If person is user there will be shown logout page, otherwise person assign as guest.

```

1  <?php
2
3  namespace App\Http\Controllers\Auth;
4
5  use App\Http\Controllers\Controller;
6  use App\Providers\RouteServiceProvider;
7  use App\User;
8  use Illuminate\Foundation\Auth\RegistersUsers;
9  use Illuminate\Support\Facades\Hash;
10 use Illuminate\Support\Facades\Validator;
11
12 class RegisterController extends Controller
13 {
14     /**
15      * Register Controller
16      *
17      * This controller handles the registration of new users as well as their
18      * validation and creation. By default this controller uses a trait to
19      * provide this functionality without requiring any additional code.
20      */
21
22     use RegistersUsers;
23
24     /**
25      * Where to redirect users after registration.
26      *
27      * @var string
28      */
29     protected $redirectTo = RouteServiceProvider::HOME;
30
31     /**
32      * Create a new controller instance.
33      *
34      * @return void
35      */
36     public function __construct()
37     {
38         $this->middleware('guest');
39     }
40
41
42
43
44     /**
45      * Get a validator for an incoming registration request.
46      *
47      * @param array $data
48      * @return \Illuminate\Contracts\Validation\Validator
49      */
50     protected function validator(array $data)
51     {
52         return Validator::make($data, [
53             'name' => ['required', 'string', 'max:255'],
54             'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
55             'password' => ['required', 'string', 'min:8', 'confirmed'],
56         ]);
57     }
58
59     /**
60      * Create a new user instance after a valid registration.
61      *
62      * @param array $data
63      * @return \App\User
64      */
65     protected function create(array $data)
66     {
67         return User::create([
68             'name' => $data['name'],
69             'email' => $data['email'],
70             'password' => Hash::make($data['password']),
71         ]);
72     }
73 }
74

```

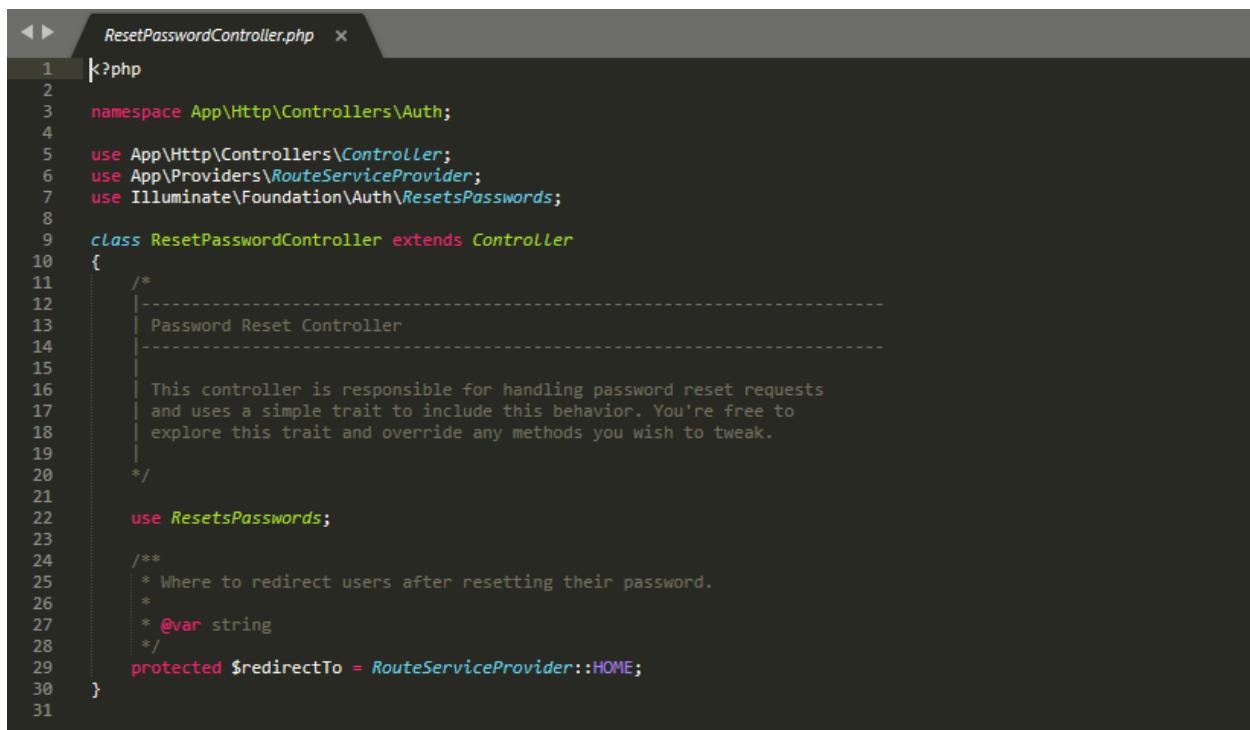
RegisterController.php:

RouteServiceProvider – this property works for to redirected user after registration.

function __construct () – this constructor calls function *middleware*. This function checks person who is using web page, user or just guest. If person is user will be entered to his/her account automatically, otherwise person will be asked for register first.

function validate () – this works for to check entered inputs are in the required form or not. For name variable, it is required, must be string and must be no more than 255 characters. For email variable, it is required, must be string, must be in the form of email, must be no more than 255 characters and must be unique. For password variable, it is required, must be string, must be no less than 8 characters, must be confirmed.

function create () – this function works for to create user. In order to create user, function get required input such as name, email and password in the required form.



```
1 |<?php
2 |
3 | namespace App\Http\Controllers\Auth;
4 |
5 | use App\Http\Controllers\Controller;
6 | use App\Providers\RouteServiceProvider;
7 | use Illuminate\Foundation\Auth\ResetsPasswords;
8 |
9 | class ResetPasswordController extends Controller
10 | {
11 |     /**
12 |      * -----
13 |      | Password Reset Controller
14 |      * -----
15 |      |
16 |      | This controller is responsible for handling password reset requests
17 |      | and uses a simple trait to include this behavior. You're free to
18 |      | explore this trait and override any methods you wish to tweak.
19 |      |
20 |      */
21 |
22 |     use ResetsPasswords;
23 |
24 |     /**
25 |      * Where to redirect users after resetting their password.
26 |      *
27 |      * @var string
28 |      */
29 |     protected $redirectTo = RouteServiceProvider::HOME;
30 | }
31 |
```

ResetPasswordController.php:

This class *ResetPasswordController* works for to reset password. In order to accomplish this task this works this class uses *ResetsPasswords*.

RouteServiceProvider – this property works after resetting the password, then it will be redirected to HOME (index.blade.php).

```

1  <?php
2
3  namespace App\Http\Controllers\Auth;
4
5  use App\Http\Controllers\Controller;
6  use App\Providers\RouteServiceProvider;
7  use Illuminate\Foundation\Auth\VerifiesEmails;
8
9  class VerificationController extends Controller
10 {
11     /**
12      * -----
13      * Email Verification Controller
14      * -----
15      *
16      * This controller is responsible for handling email verification for any
17      * user that recently registered with the application. Emails may also
18      * be re-sent if the user didn't receive the original email message.
19      */
20
21     use VerifiesEmails;
22
23     /**
24      * Where to redirect users after verification.
25      *
26      * @var string
27      */
28     protected $redirectTo = RouteServiceProvider::HOME;
29
30     /**
31      * Create a new controller instance.
32      *
33      * @return void
34      */
35     public function __construct()
36     {
37         $this->middleware('auth');
38         $this->middleware('signed')->only('verify');
39         $this->middleware('throttle:6,1')->only('verify', 'resend');
40     }
41 }
42
43

```

VerificationController.php:

This class VerificationController works work for to check the user's email is real or fake.

RouteServiceProvider – this property works for to redirected user after Verification.

function __construct () – this constructor calls function *middleware*. This function checks the user is registered before or not. And it checks does user fill the required form, if not it will redirect to user required page. If the user does not receive the verification code, then this function ensures to resent the verification code to email.

```

1 |<?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use App\Customer;
6 | use Illuminate\Http\Request;
7 |
8 | class AdminController extends Controller
9 | {
10 |     /**
11 |      * Create a new controller instance.
12 |      *
13 |      * @return void
14 |      */
15 |     public function __construct()
16 |     {
17 |         $this->middleware('auth:admin');
18 |     }
19 |
20 |     /**
21 |      * Show the application dashboard.
22 |      *
23 |      * @return \Illuminate\Contracts\Support\Renderable
24 |      */
25 |     //get
26 |     public function index()
27 |     {
28 |         return view('admin.index');
29 |     }
30 |
31 |
32 | }
33 |

```

AdminController.php:

function __construct () – this constructor calls function *middleware*. This function checks user is admin or not, through token or cookie.

function index () – this function works for so send post request to *index.blade.php* inside the *admin* file through function *view*.

```

1 |<?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
6 | use Illuminate\Foundation\Bus\DispatchesJobs;
7 | use Illuminate\Foundation\Validation\ValidatesRequests;
8 | use Illuminate\Routing\Controller as BaseController;
9 |
10 | class Controller extends BaseController
11 | {
12 |     use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
13 | }
14 |

```

Controller.php:

class Controller uses AuthorizesRequests, DispatchesJobs and ValidatesRequests

```
1 k?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class HomeController extends Controller
8 {
9     /**
10      * Create a new controller instance.
11      *
12      * @return void
13      */
14     public function __construct()
15     {
16         $this->middleware('auth');
17     }
18
19     /**
20      * Show the application dashboard.
21      *
22      * @return \Illuminate\Contracts\Support\Renderable
23      */
24     public function index()
25     {
26         return view('home');
27     }
28 }
29
```

HomeController.php:

function __construct () – this constructor calls function *middleware*. This function checks auth is created or not. If not, function *index ()* redirect the page to *home.blade.php* through function *view*.

```
1 k?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Hotel;
7
8 class HotelController extends Controller
9 {
10     public function getHotel() {
11
12         return view('home.index', [
13
14             'hotel' => Hotel::first()
15
16         ]);
17     }
18
19
20 }
```

HotelController.php:

function getHotel () works for to send request to *index.blade.php* inside the *home* file through *view* function, and create variable *hotel* then get first line of data from Hotel Table.

```

RoomController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7 use App\Room;
8
9 class RoomController extends Controller
10 {
11     public function getRoom() {
12
13         return view('home.rooms', [
14
15             'rooms' => Room::all()
16
17         ]);
18     }
19
20 }
21

```

RoomController.php:

function getRoom () works for to send request to *rooms.blade.php* inside the *home* file through *view function*, and create variable *room* then get all data from *Room Table*.

```

Authenticate.php
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Illuminate\Auth\Middleware\Authenticate as Middleware;
6
7 class Authenticate extends Middleware
8 {
9     /**
10      * Get the path the user should be redirected to when they are not authenticated.
11      *
12      * @param  \Illuminate\Http\Request  $request
13      * @return string|null
14      */
15     protected function redirectTo($request)
16     {
17         if (! $request->expectsJson()) {
18             return route('login');
19         }
20     }
21 }
22

```

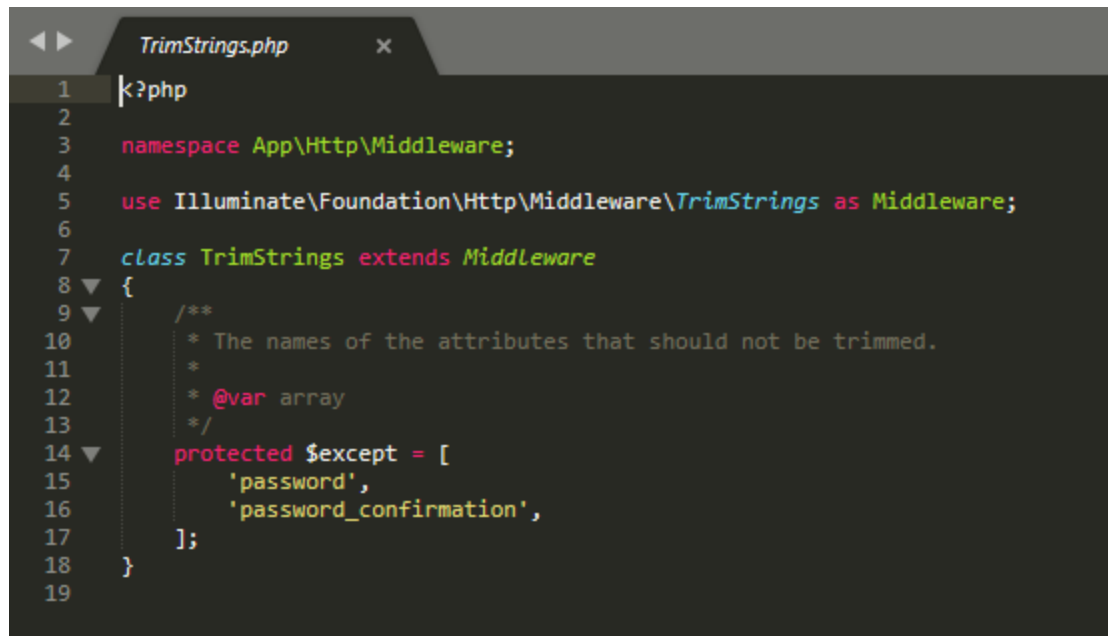
Authenticate.php:

class *Authenticate* works for to check coming requests if request out of *Json* then page will be redirected to login page.

```
RedirectIfAuthenticated.php x
1 |<?php
2
3 namespace App\Http\Middleware;
4
5 use App\Providers\RouteServiceProvider;
6 use Closure;
7 use Illuminate\Support\Facades\Auth;
8
9 class RedirectIfAuthenticated
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param \Illuminate\Http\Request $request
15      * @param \Closure $next
16      * @param string|null $guard
17      * @return mixed
18      */
19     public function handle($request, Closure $next, $guard = null)
20     {
21         switch ($guard) {
22             case 'admin':
23                 if (Auth::guard($guard)->check()) {
24                     return redirect()->route('adminDashboard');
25                 }
26                 break;
27
28             default:
29                 if (Auth::guard($guard)->check()) {
30                     return redirect()->route('Index');
31                 }
32                 break;
33         }
34
35         return $next($request);
36     }
37 }
38
39
```

RedirectIfAuthenticated.php:

class RedirectIfAuthenticated works for to check request post if the request belongs to admin then page will be redirected to admin dash board, otherwise page will be redirected to home (index.blade.php) page.



```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Illuminate\Foundation\Http\Middleware\TrimStrings as Middleware;
6
7 class TrimStrings extends Middleware
8 {
9     /**
10      * The names of the attributes that should not be trimmed.
11      *
12      * @var array
13      */
14     protected $except = [
15         'password',
16         'password_confirmation',
17     ];
18 }
19
```

TrimStrings.php:

class TrimString works for to remove spaces in the entered password.

```
2020_04_17_184829_create_book_types_table.php x
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateBookTypesTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('book_types', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->string('type');
20             $table->integer('booking_id');
21         });
22     }
23
24     /**
25      * Reverse the migrations.
26      *
27      * @return void
28      */
29     public function down()
30     {
31         Schema::dropIfExists('book_types');
32     }
33 }
34
```

create_book_types_table.php:

class *CreateBookTypesTable* works for to create Book Type table with 4 column and each column has its own functionality. These columns are id () creates different id numbers for users, timestamps () records when the column is created or updated, string ('type') identifies book type and integer('booking_id') works for to access to booking ID.

function down () – this function checks current Book_Type table created before or not. if Book_Type table created before then it will assign current Book_type table to old one.

```
2020_05_01_095645_create_admins_table.php x
1 |<?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateAdminsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('admins', function (Blueprint $table) {
17             $table->id();
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->timestamp('email_verified_at')->nullable();
21             $table->string('password');
22             $table->rememberToken();
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31      */
32     public function down()
33     {
34         Schema::dropIfExists('admins');
35     }
36 }
37
38
```

create_admins_table.php:

class *CreateAdminsTable* works for to create Admin Table with 7 column and each column has its own functionality. These columns are id () which creates different id numbers for users, string('name') get

name, string('email') get unique email, timestamp('email_verified_at'), string('password') get password, rememberToken () checks whether guest is user or admin and timestamps () records when the column is created or updated.

function down () – this function checks current admin table created before or not. if admins table created before then it will assign current table admins to old one.

```
2020_04_17_184814_create_bookings_table.php x
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateBookingsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('bookings', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->dateTime('arrival');
20             $table->dateTime('checkout');
21             $table->integer('breakfast');
22             $table->integer('night');
23             $table->string('comment');
24             $table->dateTime('book_time');
25             $table->integer('customer_id');
26         });
27     }
28
29     /**
30      * Reverse the migrations.
31      *
32      * @return void
33      */
34     public function down()
35     {
36         Schema::dropIfExists('bookings');
37     }
38 }
39
```

create_bookings_table.php:

class *CreateBookingsTable* works for to create Bookings Table with 9 column and each column has its own functionality. These columns are id () which creates different id numbers for users, timestamps () records when the column is created or updated, dateTime('arrival') gets arrival date and time, dateTime('checkout') gets checkout date and time, integer(night) identifies that user books for a night or more, integer('breakfast') identifies that user required breakfast or not, string('comment') receive comment from user, dateTime('Book_time') gets booking date and time and integer('costumer_id') gets consumer ID.

function down () – this function checks current bookings table created before or not. if bookings tables created before then it will assign current bookings tables to old one.

```
2020_04_17_184854_create_cancellations_table.php x
1  k?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateCancellationsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('cancellations', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->dateTime('arrival');
20             $table->dateTime('checkout');
21             $table->integer('breakfast');
22             $table->integer('night');
23             $table->dateTime('book_time');
24             $table->dateTime('cancel_time');
25             $table->integer('booking_id');
26         });
27     }
28
29     /**
30      * Reverse the migrations.
31      *
32      * @return void
33      */
34     public function down()
35     {
36         Schema::dropIfExists('cancellations');
37     }
38 }
39
```

create_cancellations_table.php:

class *CreateCancellationsTable* works for to create Cancellations Table with 9 column and each column has its own functionality. These columns are id () which creates different id numbers for users, timestamps () records when the column is created or updated, dateTime('arrival') gets arrival date and time, dateTime('checkout') gets checkout date and time, integer(night) identifies that user books for a night or more, integer('breakfast') identifies that user required breakfast or not, dateTime('Book_time') gets booking date dateTime('cancel_time') gets cancellation date and time and integer('booking_id') gets Booking ID.

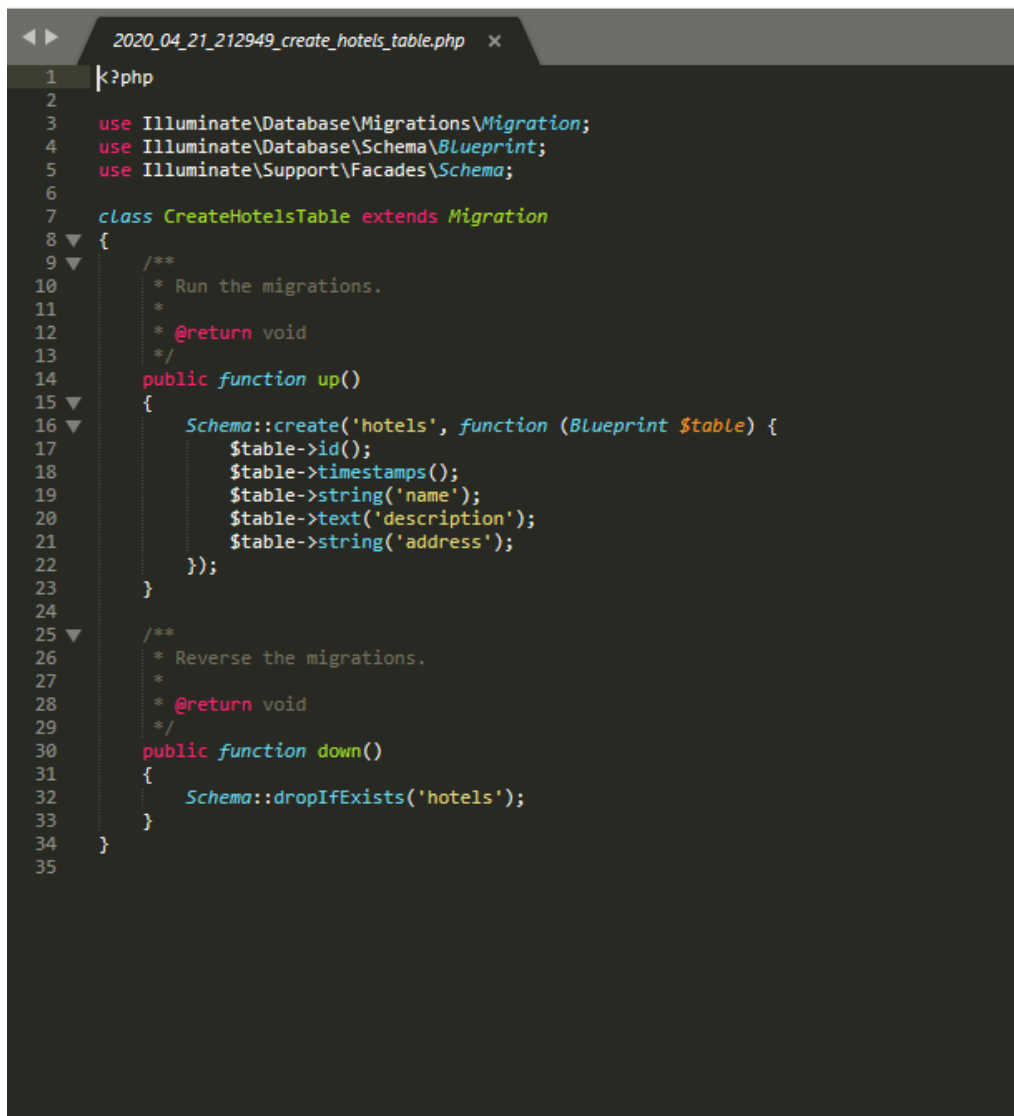
function down () – this function checks current cancellation table created before or not. if cancellation table created before then it will assign current cancellation table to old one.

```
2020_04_21_084659_create_customers_table.php x
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateCustomersTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('customers', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->string('first_name');
20             $table->string('last_name');
21             $table->string('address');
22             $table->string('postal_code');
23             $table->string('city');
24             $table->string('country');
25             $table->string('phone');
26             $table->string('email');
27             $table->dateTime('reg_time');
28         });
29     }
30
31
32     /**
33      * Reverse the migrations.
34      *
35      * @return void
36      */
37     public function down()
38     {
39         Schema::dropIfExists('customers');
40     }
41 }
42
```

create_consumers_table.php

class *CreateConsumerTable* works for to create Consumers Table with 11 column and each column has its own functionality. These columns are id () which creates different id numbers for users, timestamps () records when the column is created or updated, string('first_name') gets first name of user, string('last_name') gets last name of user, string('address') gets the address of user, string('postal_code') gets postal code of users country, string('city') gets info about which city user come from, string('country') gets info about where user come from, string('phone') gets the phone number of user, string('email') gets the email of user and dateTime('reg_time') gets registration date and time.

function down () – this function checks current consumers table created before or not. If consumers table created before then it will assign current table consumers to old one.



```
1  k?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateHotelsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('hotels', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->string('name');
20             $table->text('description');
21             $table->string('address');
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('hotels');
33     }
34 }
35
```

create_hotels_table.php

class *CreateHotelTable* works for to create Hotel Table with 5 column and each column has its own functionality. These columns are id () which creates different id numbers for users, timestamps () records when the column is created or updated, string('name') gets name of hotel, text('description') get the description about hotel and string('address') gets address of hotel.

function down () – this function checks current hotel table created before or not. If hotel table created before then it will assign current hotel table to old one.

```
2014_10_12_100000_create_password_resets_table.php x
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreatePasswordResetsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('password_resets', function (Blueprint $table) {
17             $table->string('email')->index();
18             $table->string('token');
19             $table->timestamp('created_at')->nullable();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      *
26      * @return void
27      */
28     public function down()
29     {
30         Schema::dropIfExists('password_resets');
31     }
32 }
33
```

create_password_resets_table.php:

class *CreatePasswordResetTable* works for to create Password Reset Table with 3 column and each column has its own functionality. These columns are string('email'), string('token') and timestamp('created_at').

function down() – this function checks current password reset table created before or not. If password reset table created before then it will assign current password reset table to old one.

```
2020_04_17_184759_create_payments_table.php x
1  k?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreatePaymentsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('payments', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->integer('amount');
20             $table->string('paid');
21             $table->dateTime('pay_time');
22             $table->string('invoice');
23             $table->string('cancelled');
24             $table->integer('booking_id');
25         });
26     }
27
28     /**
29      * Reverse the migrations.
30      *
31      * @return void
32      */
33     public function down()
34     {
35         Schema::dropIfExists('payments');
36     }
37
38 }
39
40
41
```

create_payments_table.php:

class *CreatePaymentsTable* works for to create Payments Table with 8 column and each column has its own functionality. These columns are id () which creates different id numbers for users, timestamps () records when the column is created or updated, integer('amount'), string('paid'), dateTime('pay_time'), string('invoice'), string('cancelled') and integer('booking_id').

function down () – this function checks current payment table created before or not. If payment table created before then it will assign current room table to old one.

```
2020_04_17_184840_create_rooms_table.php x
1  k?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateRoomsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('rooms', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->integer('room_number');
20             $table->string('room_type');
21             $table->integer('max_person');
22             $table->string('locked');
23             $table->decimal('price_per_night');
24             $table->integer('booking_id');
25         });
26     }
27
28     /**
29      * Reverse the migrations.
30      *
31      * @return void
32      */
33     public function down()
34     {
35         Schema::dropIfExists('rooms');
36     }
37 }
38
```

create_rooms_table.php:

class *CreateRoomsTable* works for to create Rooms Table with 8 column and each column has its own functionality. These columns are id () which creates different id numbers for users, timestamps () records when the column is created or updated, integer('room_number'), string(room_type), integer('max_person'), string('locked') decimal('price_per_night') and integer('booking_id')

function down () – this function checks current room table created before or not. If room table created before then it will assign current room table to old one.

```
2014_10_12_000000_create_users_table.php x
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateUsersTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->id();
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->timestamp('email_verified_at')->nullable();
21             $table->string('password');
22             $table->rememberToken();
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31      */
32     public function down()
33     {
34         Schema::dropIfExists('users');
35     }
36 }
37
```

create_users_table.php:

class *CreateUserTable* works for to create User Table with 7 column and each column has its own functionality. These columns are id () creates different id numbers for users, string('name') get name, string('email') get unique email, timestamp('email_verified_at'), string('password') get password, rememberToken () checks whether guest is user or admin and timestamps () records when the column is created or updated.

function down () – this function checks current user table created before or not. if user table created before then it will assign current table user to old one.

Contribution of our members

Islam Akhrarov – he was mainly responsible for server-side development. And he is one who initialized the project and made initial changes. He also structured the logic of our project, in other words, he made relationships of models. And took part in coding the controllers. Finally, he deployed the project to hosting.

Abduvohid Isroilov – he was one who provided interactivity to our website by using javascript and jquery. He also developed UI for Login and Register pages. He also worked on pages of admin dashboard.

Akbar Olimov – he is the one who contributed to frontend development by using js, jquery code. And made validations for forms to prevent user from requesting to database with invalid data.

Masrur Bekmirzayev – he mainly worked on html/css part, and styled some pages of our website, also he made use of bootstrap in our project to ensure good look and feel of our website.

Farrukh Zokirov - he also worked in backend side. He was responsible for taking data from models and generating the values in frontend side by working on controllers and routes.

Solih Abrollov – He contributed to admin pages and designed that pages by using bootstrap. He is one who showed data in front which is taken by database.

Authentication and Authorization

We used laravel's own authentication for authenticating our customers and admins. We used guards to differentiate user types. New provider for new guard called admin also used by using driver called eloquent. As result, we needed two different passwords for different types of users(customers) and admins. All of configurations made in auth.php file.

We used middleware to allow a particular user for accessing admin pages or not. If user authenticated as admin, then they can access to admin/ pages.