**VC Booking Project**

1. Create Project

composer create-project –prefer-dist Laravel/Laravel MeetingScheduler

1. Create and link database in the .env file.

DB\_CONNECTION=sqlite

DB\_HOST=/Users/durgesh/Documents/Laravel/durgeshpareva.info/database

1. Make auth

Composer require Laravel/ui --dev

Php artisan ui vue --auth

1. Npm install and npm run dev 🡪 css js etc becomes visible in the public folder
2. Create Model with migration and controller
3. Migrate the database
4. Make provision in the web.php folder
   1. Route::resource('/task','TaskController');

Make sure a folder task exist in the

1. Do the rest functionality as per your requirement.
2. Location: calendar :time: user notification: no time overlapping

Location selection -> Location tables ->id, name, data, time, purpose of meeting

Upon selection of the location-> loading of the calendar, upon selection of a particular date pop up desciribing booking with option of create, edit and delete

1. Register functionality: Disabling registration, admin to register + forgot password reset by the individual user.
2. Delete
   1. Delete confirmation by asking the password once

Route::delete('/bookroom/{bookroom}', 'BookRoomController@destroy',function () {

})->middleware(['auth', 'password.confirm']);

* 1. Delete confirmation using the alert box.

<button type="submit" onclick = "return confirm('Sure to delete ')" class="btn btn-danger btn-round">Delete</button>

* 1. Edit only if it is created by the same user

1. Status: Accept, reject, pending on the bookmark

**Migration added to the bookmark table and user table**

php artisan make:migration add\_attributes\_to\_book\_rooms\_table;

public function up()

{

Schema::table('book\_rooms', function (Blueprint $table) {

$table->integer('user\_id')->nullable();

$table->char('status')->nullable();

});

}

public function down()

{

Schema::table('book\_rooms', function (Blueprint $table) {

$table->dropColumn('user\_id');

$table->dropColumn('status');

});

}

php artisan make:migration add\_attributes\_to\_users\_table;

public function up()

{

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

$table->char('Phone')->nullable();

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

$table->string('user\_type')->nullable();

});

}

public function down()

{

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

$table->dropColumn('Phone');

$table->dropColumn('location');

$table->dropColumn('user\_type');

});

Php artisan migrate

**Auth -> Register.blade.php edited to accommodate newly added fields**

<select id="user\_type" type="text" name="user\_type" value="{{ old('user\_type') }}" autofocus>

<option value="Normal" selected>Normal</option>

<option value="Super">Super</option>

<option value="God" >God</option>

</select>

**Index.blade.php added with if else for visaibility and access list**

@foreach ($bookrooms as $bookroom)

{{-- Show the bookings that are

1. accepted and

2. the ones created by the user itself

3. The ones for which a user is super user--}}

{{-- To show all the accepted booking and the ones created by the user and booking pending for the--}}

@if($bookroom->status =='Accept' ||

$bookroom->user\_id == auth()->user()->id ||

($bookroom->user()->location = auth()->user()->location && auth()->user()->user\_type =='Super'))

<tr>

<td>{{$bookroom->conference\_details}}</td>

<td>

@foreach ($bookroom->shifts as $location)

{{$location}}

@endforeach

</td>

<td>{{$bookroom->date}}</td>

<td>{{$bookroom->startTime}}</td>

<td>{{$bookroom->endTime}}</td>

<td>{{$bookroom->agenda}}</td>

<td>{{$bookroom->status}}</td>

{{-- make the field editable when the user

1. The logged in User is self

2. When the logged in User is super, and belongs to the location made by the user ofthe location (i.e. Delhi user can crate booking for ahmedabad and the same has to be approved vy the super used of Delhi and not Ahd--}}

@if(($bookroom->user\_id == auth()->user()->id && $bookroom->status !='Accept') ||

(auth()->user()->user\_type =="Super" && auth()->user()->location==$bookroom->user->location))

{{-- (auth()->user()->user\_type == 'Super' && auth()->user()->location == $bookroom->user->location)) --}}

<td><a href= "/bookroom/{{$bookroom->id}}/edit" class="btn btn-success no-hover">Edit</a></td>

<td>

<form action="/bookroom/{{$bookroom->id}}" method="POST">

@csrf

@method('DELETE')

<button type="submit" onclick="return confirm('Sure to Delete')" class="btn btn-danger btn-round">Delete</button>

</form>

</td>

@endif

</tr>

@endif

@endforeach

**Edit.blade.php added with if else for visibility and editing**

@if (auth()->user()->user\_type =="Super")

<div class="form-group">

<label for="status">Agenda</label>

<select id ="status" type="text" class="form-control" id="status" name="status" value= {{$bookrooms->status}}>

<option value="Pending" class="success">Pending</option>

<option value="Accept" class="success">Accept</option>

<option value="Reject" class="danger" selected >Reject</option>

</select>

</div>

@endif

**One to Many relationship b/w User and BookRoom model created**

User.php

public function BookRoom(){

return $this->hasMany('App\BookRoom');

}

BookRoom.php

public function User(){

return $this->belongsTo('App\User');

}

Now it’s possible

1. $bookroom->user->location; FIndingthe location from the user who created the booking

**Functionalities required:**

In the index.blade.php

* Show the entries only when it is accepted.
* If the request is created by the user it should be shown pending/ rejected.
* If the loggedin user is the super user of the same location: pending should be shown with the option of approving the request.
* Once accepted, can only be edited or delete by the super user

Edit.blade.php

Status to be shown if the user is super for the location:

Superuser->location == bookroom->user\_id->location

One to One or many relation needs to be incorporated.

1. Messaging functionality
2. **TINKER**
3. Php artisan tinker
4. App\Bookroom::where('date',"2020-07-05")->get();
5. With and condition
   1. App\Bookroom::where('date',"2020-06-28")->where("23:00",'>',"22:00")->orWhere("23:00",'<',"23:30")->get();
6. With or confition
   1. orWhere takes precendence and has to be put first always.
   2. App\Bookroom::orWhere(function($query){

$query->where("23:00",'>',"22:00")

->where("23:00",'<',"23:30");

})

-> where('date',"2020-06-28")->get();

1. Needs to have an additional user\_id, **status** (Accept/ Reject/ Pending)column.

// Get the currently authenticated user...

$user = Auth::user();

// Get the currently authenticated user's ID...

* $id = Auth::id();

Authentication : Register needs to be modified

**User table**

1. Name
2. Email
3. **Location**
4. **Phone number**
5. **Usertype: FPR/ SuperUser, god**
6. Id and Password.

$table->char('Phone')->nullable();

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

$table->string('user\_type')->nullable();

$table->integer('user\_id')->nullable();

$table->char('status')->nullable();