Lampiran 1 Daftar Riwayat Hidup

Data Diri

Nama Lengkap : Indra Subroto

Tempat Tanggal Lahir : Sidomukti, 22 Mei 1996

Jenis Kelamin : Laki – Laki

Agama : Islam

Alamat : Sidomukti RT. 1 / RW. 1, Abung Timur, Lampung Utara, Lampung

Email :indra22inazuto@gmail.com, indraesgul16@gmail.com

No. Telpon & WA : 081218466350

Kewarganegaraan : Indonesia

Nama Ayah : Sutrisno

Nama Ibu : Atmini

Riwayat Pendidikan

- 2002 – 2008 SD Negeri 04 Sidomukti
- 2008 – 2011 SMP Negeri 1 Abung Semuli
- 2011 – 2014 SMA Negeri 1 Abung Semuli
  Jurusan Ilmu Pengetahuan Alam (IPA)
- 2016 – 2020 Universitas Esa Unggul
  Jurusan Teknik Informatika

Lampiran 2 *Full Source Coding*

Visit this url :

https://github.com/IndraSubroto/tournament_16

Lampiran 3 *Source Coding Controller*

app > Http > Controllers >

ComplainController.php

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use App\Model\Team;
6. use App\Model\Complain;
7. use Illuminate\Http\Request;
8.
9. class ComplainController extends Controller
10.{
11.    public function index()
12.    {
13.        $complains = Complain::all();
14.        return view('complain.table',compact('complains'));
15.    }
16.
17.    public function create($id)
18.    {
19.        $team = Team::with('tournament.user')
20.                ->findOrFail($id);
21.        return view('complain.complain',compact('team'));
22.    }
23.
24.    public function store(Request $request,$id)
25.    {
26.        $team = Team::with('tournament')->findOrFail($id);
27.        Complain::create([
28.            'user_id' => auth()->id(),
29.            'team_id' => $id,
30.            'tournament_id' => $team->tournament_id,
31.            'subject' => $request['subject'],
32.            'text' => $request['text'],
33.        ]);
34.
35.        return redirect(route('list.Teams'));
36.    }
37.}
38.
```

ModifyController.php

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use App\Model\Province;
6. use App\Model\Team;
7. use App\Model\Tournament;
8. use Illuminate\Http\Request;
9. use Illuminate\Support\Facades\DB;
10.
11.class ModifyController extends Controller
12.{
13.    public function getDataProvince()
14.    {
15.        $provinces = Province::all()->pluck('name','id');
16.        return response()->json($provinces);
17.    }
18.
19.    public function selectCity($id)
20.    {
21.        $cities = DB::table("cities")
22.            ->where("province_id",$id)->pluck('name','id');
23.        return json_encode($cities);
24.    }
25.
26.    public function selectDistrict($id)
27.    {
28.        $districts = DB::table("districts")
29.            ->where("city_id",$id)->pluck('name','id');
30.        return json_encode($districts);
31.    }
32.
33.    public function getDataChampion($id)
34.    {
35.        $teams = Team::where('tournament_id',$id)
36.                    ->pluck('name','id');
37.        return json_encode($teams);
38.    }
39.
40.    public function selectFirstPlace($champion_id,$tournament_id)
41.    {
42.        $teams = Team::where('tournament_id',$tournament_id)
43.        ->whereNotIn('id',[$champion_id])->pluck('name','id');
44.        return json_encode($teams);
45.    }
```

```
46.
47.     public function selectSecondPlace($first_place_id,$champion_i
   d,$tournament_id)
48.     {
49.         $teams = DB::table("teams")
50.             ->where('tournament_id',$tournament_id)
51.             ->whereNotIn("id",[$first_place_id,$champion_id])
52.             ->pluck('name','id');
53.         return json_encode($teams);
54.     }
55.
56.     public function selectThirdPlace($second_place_id,$first_plac
   e_id,$champion_id,$tournament_id)
57.     {
58.         $teams = DB::table("teams")
59.             ->where('tournament_id',$tournament_id)
60.             -
   >whereNotIn("id",[$second_place_id,$first_place_id,$champion_id])
61.             ->pluck('name','id');
62.         return json_encode($teams);
63.     }
64.
65.     public function searchTournament(Request $request)
66.     {
67.         $output = '';
68.         if($request->ajax()) {
69.             if($request->title !== null ){
70.                 $data = Tournament::where('title', 'LIKE', '%'.$r
   equest->title.'%')->limit(8)->get();
71.                 if (count($data)>0) {
72.                     $output = '<li class="nav-item dropdown mt-
   4"><ul aria-labelledby="dropdownSubMenu1" class="dropdown-
   menu border-
   0 shadow" style="display: block; position: absolute; z-
   index: 1">';
73.                     foreach ($data as $row){
74.                         $output .= '<li><a class="dropdown-
   item search" href="/tournament/detail/'.$row->id.'">'.$row-
   >title.'</a></li>';
75.                     }
76.                     $output .= '</ul></li>';
77.                 }
78.                 else {
79.                     $output = '<li class="nav-item dropdown mt-
   5"><ul aria-labelledby="dropdownSubMenu1" class="dropdown-
   menu border-
   0 shadow" style="display: block; position: absolute; z-
```

```
   index: 1">';
80.                     $output .= '<li>'.'<p class="dropdown-
   item">No results</p>'.'</li></ul></li>';
81.                 }
82.                 return $output;
83.             }else{
84.                 return null;
85.             }
86.         }
87.     }
88.}
89.
```

PaymentController.php

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use App\Model\Team;
6. use Veritrans_Snap;
7. use Veritrans_Config;
8. use App\Model\Payment;
9. use Veritrans_Transaction;
10.use Illuminate\Http\Request;
11.use Illuminate\Support\Facades\DB;
12.
13.class PaymentController extends Controller
14.{
15.  protected $request;
16.
17.  public function __construct(Request $request)
18.  {
19.    $this->request = $request;
20.
21.    Veritrans_Config::$serverKey = config('services.midtrans.serv
   erKey');
22.    Veritrans_Config::$isProduction = config('services.midtrans.i
   sProduction');
23.    Veritrans_Config::$isSanitized = config('services.midtrans.is
   Sanitized');
24.    Veritrans_Config::$is3ds = config('services.midtrans.is3ds');
25.  }
26.
27.  public function getDataPayment( $id)
28.  {
```

```
29.      $team = Team::with('tournament.user','payment')
30.            ->findOrFail($id);
31.      $user_id = $team->user->id;
32.      $team_id = $team->id;
33.      $tournament_id = $team->tournament->id;
34.      $user_name = $team->user->name;
35.      $user_email = $team->user->email;
36.      $amount = $team->tournament->fee;
37.      $payment_type = 'Registration';
38.
39.      return response()->json([
40.          'user_id' => $user_id,
41.          'user_name' => $user_name,
42.          'user_email' => $user_email,
43.          'tournament_id' => $tournament_id,
44.          'amount' => $amount,
45.          'team_id' => $team_id,
46.          'payment_type' => $payment_type,
47.          ]);
48.  }
49.
50.  public function submitPayment()
51.  {
52.    DB::transaction(function(){
53.      $payment = Payment::create([
54.        'user_id' => $this->request->user_id,
55.        'team_id' => $this->request->team_id,
56.        'tournament_id' => $this->request->tournament_id,
57.        'user_name' => $this->request->user_name,
58.        'user_email' => $this->request->user_email,
59.        'payment_type' => $this->request->payment_type,
60.        'amount' => floatval($this->request->amount),
61.      ]);
62.
63.      $payload = [
64.        'transaction_details' => [
65.          'order_id'      => $payment->id,
66.          'gross_amount'  => $payment->amount,
67.        ],
68.
69.        'customer_details' => [
70.          'first_name'    => $payment->user_name,
71.          'email'         => $payment->user_email,
72.        ],
73.
74.        'item_details' => [
75.          [
```

```
76.             'id' => $payment->payment_type,
77.             'price'    => $payment->amount,
78.             'quantity' => 1,
79.             'name' => ucwords(str_replace('_', ' ',
80.                       $payment->payment_type))
81.           ]
82.         ]
83.       ];
84.
85.       $snapToken = Veritrans_Snap::getSnapToken($payload);
86.       $payment->snap_token = $snapToken;
87.       $payment->save();
88.
89.       $this->response['snap_token'] = $snapToken;
90.     });
91.     return response()->json($this->response);
92.   }
93.
94.   private $response;
95.
96.   public function notificationHandler(Request $request)
97.   {
98.     $input_source = "php://input";
99.     $raw_notification = json_decode(stripslashes(trim
100.   (file_get_contents($input_source), '"')), true);
101.       $status_response = Veritrans_Transaction::status($raw_noti
    fication['transaction_id']);
102.       $this->response = $status_response;
103.       $notif = new V();
104.
105.       DB::transaction(function() use($raw_notification) {
106.         $transaction =
107.                 $raw_notification['transaction_status'];
108.         $type = $raw_notification['payment_type'];
109.         $orderId = $raw_notification['order_id'];
110.         $fraud = $raw_notification['fraud_status'];
111.         $payment = Payment::findOrFail($orderId);
112.
113.         if ($transaction == 'capture') {
114.           if ($type == 'credit_card') {
115.             if($fraud == 'challenge') {
116.               $payment->setPending();
117.             } else {
118.               $payment->setSuccess();
119.             }
120.           }
121.
```

```
122.         } elseif ($transaction == 'settlement') {
123.               $payment->setSuccess();
124.         } elseif($transaction == 'pending'){
125.               $payment->setPending();
126.         } elseif ($transaction == 'deny') {
127.               $payment->setFailed();
128.         } elseif ($transaction == 'expire') {
129.               $payment->setExpired();
130.         } elseif ($transaction == 'cancel') {
131.               $payment->setFailed();
132.         }
133.       });
134.       return;
135.     }
136.   }
137.
138.   class V {
139.     private $response;
140.     public function __construct($input_source = "php://input")
141.     {
142.       $raw_notification = json_decode(stripslashes(trim(file_get
   _contents($input_source), '"')), true);
143.       $status_response = Veritrans_Transaction::status($raw_noti
   fication['transaction_id']);
144.       $this->response = $status_response;
145.     }
146.     public function __get($name)
147.     {
148.       if (array_key_exists($name, $this->response)) {
149.         return $this->response['transaction_id']->name;
150.       }
151.     }
152.   }
153.
```

TeamController.php

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use App\Model\Team;
6. use App\Model\Athlete;
7. use App\Model\Tournament;
8. use Illuminate\Http\Request;
9. use Illuminate\Support\Facades\Gate;
```

```
10.use Yajra\DataTables\Facades\DataTables;
11.use Illuminate\Support\Facades\Validator;
12.
13.class TeamController extends Controller
14.{
15.  public function index()
16.    {
17.       if (Gate::allows('isAdmin')) {
18.           $teams = Team::orderBy('id','desc')
19.       ->with('tournament.user','payment','complain')->get();
20.        } else {
21.           $teams = Team::orderBy('id','desc')
22.       ->with('tournament.user','payment','complain')
23.       ->where('user_id',auth()->id())->get();
24.        }
25.        return view('team.table',compact('teams'));
26.    }
27.
28.    public function store(Request $request)
29.    {
30.        $this->validate($request, [
31.            'name' => 'required|string',
32.        ]);
33.
34.        Team::create([
35.            'name' => $request['name'],
36.            'tournament_id' => $request['tournament_id'],
37.            'user_id' => auth()->id(),
38.        ]);
39.        return redirect('listTeams');
40.    }
41.
42.    public function destroyTeam($id)
43.    {
44.        $team = Team::find($id);
45.        $team->delete();
46.        return response()->json(['success'=>'done']);
47.    }
48.
49.    public function destroyAthlete($id)
50.    {
51.        $athlete = Athlete::with('tournament.team.user')
52.                    ->find($id);
53.        $team_id = $athlete->team->id;
54.        $team_name = $athlete->team->name;
55.        $tournament_id = $athlete->tournament->id;
56.        $tournament_athlete = $athlete->tournament->athlete;
```

```
57.         $tournament_title = $athlete->tournament->title;
58.         $athlete->delete();
59.         $count = Athlete::where('team_id',$team_id)->count();
60.         return response()->json([
61.             'count'=>$count,
62.             'team_id' => $team_id,
63.             'user_id' => auth()->id(),
64.             'tournament_id' => $tournament_id,
65.             'athleteTotal' => $tournament_athlete,
66.             'title' => $tournament_title,
67.             'teamName' => $team_name]);
68.     }
69.
70.     public function addAthlete(Request $request)
71.     {
72.         $rules = [];
73.
74.         foreach($request
75.           ->input('first','last') as $key => $value) {
76.             $rules["first.{$key}"] = 'required';
77.             $rules["last.{$key}"] = 'required';
78.         }
79.
80.         $validator = Validator::make($request->all(), $rules);
81.
82.         if ($validator->validated()) {
83.             $first = $request->input('first');
84.             foreach($request
85.                 ->input('last') as $key => $value) {
86.                  Athlete::create([
87.                      'user_id'=>$request->user_id,
88.                      'team_id'=>$request->team_id,
89.                      'tournament_id'=>$request->tournament_id,
90.                      'first'=>$first[$key],
91.                      'last'=>$value,
92.                      ]);
93.                      $athlete = Athlete::with('team.tournament.use
   r')->where('tournament_id',$request->tournament_id)-
   >where('team_id',$request->team_id)->where('user_id',auth()-
   >id())->get();
94.                      $tournament_athlete = Tournament::where('id',
   $request->tournament_id)->pluck('athlete');
95.                      $tournament_name = Tournament::where('id',$re
   quest->tournament_id)->pluck('title');
96.                      $team_name = Team::where('id',$request-
   >team_id)->pluck('name');
97.                      $count = $athlete->count();
```

```
98.                     $team_id = $request->team_id;
99.                     $tournament_id = $request->tournament_id;
100.                   }
101.               return response()->json([
102.                   'count'=>$count,
103.                   'key' => $key,
104.                   'value' => $value,
105.                   'team_id' => $team_id,
106.                   'user_id' => auth()->id(),
107.                   'tournament_id' => $tournament_id,
108.                   'athleteTotal' => $tournament_athlete,
109.                   'title' => $tournament_name,
110.                   'teamName' => $team_name]);
111.           }
112.           return response()->json(['error'=>$validator
113.            ->errors()->all()]);
114.       }
115.
116.       public function athleteByTeam(Request $request, $id)
117.       {
118.           if($request->ajax())
119.           {
120.               $data = Athlete::where('team_id',$id)
121.               ->get();
122.               return DataTables::of($data)
123.                   ->addColumn('action', function($data){
124.                     $button = '<span type="button" name="edit" i
   d="'.$data->id.'" data-id="'.$data->id.'" class="delete-
   athlete btn btn-danger btn-sm m-1"><i class="fas fa-
   trash"></i></span>';
125.                       return $button;
126.                   })
127.                   ->rawColumns(['action'])
128.                   ->addIndexColumn()
129.                   ->make(true);
130.           }
131.       }
132.   }
133.
```

TournamentController.php

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
```

```
5. use App\User;
6. use App\Model\Age;
7. use App\Model\Team;
8. use App\Model\Winner;
9. use App\Model\Payment;
10.use App\Model\Tournament;
11.use Illuminate\Http\Request;
12.use Illuminate\Support\Facades\Gate;
13.
14.class TournamentController extends Controller
15.{
16.    public function index()
17.    {
18.        if (Gate::allows('isAdmin')) {
19.            $tournaments = Tournament::orderBy('id','desc')
20.           ->get();
21.        } else {
22.            $tournaments = Tournament::orderBy('id','desc')
23.           ->where('user_id',auth()->id())->get();
24.        }
25.        return view('tournament.table',compact('tournaments'));
26.    }
27.
28.    public function create()
29.    {
30.        $ages = Age::all();
31.        $user = User::with('promotor')->findOrFail(auth()
32.               ->id());
33.        return view('tournament.create',compact('ages'),
34.              ['user' => $user]);
35.    }
36.
37.    public function store(Request $request)
38.    {
39.        if($request->hasFile('poster')){
40.            $resource = $request->file('poster');
41.            $poster   = $resource->getClientOriginalName();
42.            $tujuan_upload = 'images/posters';
43.            $resource->move($tujuan_upload,$poster);
44.            Tournament::create([
45.                'title' => $request['title'],
46.                'poster' => $poster,
47.                'description' => $request['description'],
48.                'pricepool' => $request['pricepool'],
49.                'slot' => $request['slot'],
50.                'fee' => $request['fee'],
51.                'athlete' => $request['athlete'],
```

```
52.                 'date_regis_limit' => $request['dateRegisLimit'],
53.                 'date_initial' => $request['dateInitial'],
54.                 'date_final' => $request['dateFinal'],
55.                 'user_id' => auth()->id(),
56.             ]);
57.         }else{
58.             echo "Nothing";
59.         }
60.
61.         return redirect('listTournaments');
62.     }
63.
64.     public function show($id)
65.     {
66.         $pending = Payment::where('tournament_id',$id)
67.          ->where('status','pending')->get()->count();
68.         $success = Payment::where('tournament_id',$id)
69.          ->where('status','success')->get()->count();
70.         $check = Team::where('tournament_id',$id)
71.          ->where('user_id',auth()->id())->get()->count();
72.         if(Gate::allows('isFinish')){
73.           $winner = Winner::where('tournament_id',$id)->get();
74.             foreach ($winner as $win) {
75.                 $champion = Team::where('tournament_id',$id)
76.                 ->findOrFail($win->champion_team_id);
77.                 $first = Team::where('tournament_id',$id)
78.                 ->findOrFail($win->first_place_team_id);
79.                 $second = Team::where('tournament_id',$id)
80.                 ->findOrFail($win->second_place_team_id);
81.                 $third = Team::where('tournament_id',$id)
82.                 ->findOrFail($win->third_place_team_id);
83.             }
84.             return view('tournament.detail', compact('first','sec
    ond','third','champion','check','pending','success') ,['tournamen
    t' => Tournament::with('user.promotor','team')-
    >findOrFail($id)]);
85.         }else{
86.             return view('tournament.detail', compact('check','pen
    ding','success') ,['tournament' => Tournament::with('user.promoto
    r','team')->findOrFail($id)]);
87.         }
88.     }
89.
90.     public function edit($id)
91.     {
92.         $ages = Age::all();
93.         $tournament = Tournament::with('user.promotor')
```

```
94.             ->findOrFail($id);
95.         return view('tournament.edit',compact('ages'),
96.         ['tournament' => Tournament::with('user.promotor')
97.             ->findOrFail($id)]);
98.     }
99.
100.        public function update(Request $request, $id)
101.        {
102.            $tournament = Tournament::find($id);
103.            if($request->hasFile('poster')){
104.             $resource = $request->file('poster');
105.             $poster   = $resource->getClientOriginalName();
106.             $tujuan_upload = 'images/posters';
107.             $resource->move($tujuan_upload,$poster);
108.             $tournament->update([
109.                 'title' => $request['title'],
110.                 'poster' => $poster,
111.                 'description' => $request['description'],
112.                 'pricepool' => $request['pricepool'],
113.                 'slot' => $request['slot'],
114.                 'fee' => $request['fee'],
115.               'athlete' => $request['athlete'],
116.           'date_regis_limit' => $request['dateRegisLimit'],
117.                'date_initial' => $request['dateInitial'],
118.                 'date_final' => $request['dateFinal'],
119.                 'user_id' => auth()->id(),
120.               ]);
121.            }else{
122.            $tournament->update([
123.                'title' => $request->title,
124.                'description' => $request->description,
125.                'pricepool' => $request->pricePool,
126.                'slot' => $request->slot,
127.                'fee' => $request->fee,
128.                'athlete' => $request->athlete,
129.                'date_regis_limit' => $request
130.               ->dateRegisLimit,
131.                'date_initial' => $request->dateInitial,
132.                'date_final' => $request->dateFinal,
133.                'user_id' => auth()->id(),
134.                ]);
135.            }
136.            return redirect('listTournaments');
137.        }
138.
139.        public function destroy($id)
140.        {
```

```
141.          $tournament = Tournament::find($id);
142.          $tournament->delete();
143.          return response()->json(['success'=>'done']);
144.      }
145.
146.      public function winner($id)
147.      {
148.         $teams = Team::where('tournament_id',$id)->get();
149.          $tournament = Tournament::with('user.promotor')
150.              ->findOrFail($id);
151.          return view('tournament.winner',compact('teams','tourn
   ament'));
152.      }
153.
154.      public function storeWinner(Request $request, $id)
155.      {
156.          $this->validate($request, [
157.              'champion' => 'required',
158.              'first' => 'required',
159.          ]);
160.
161.          $tournament = Tournament::findOrFail($id);
162.
163.          Winner::create([
164.              'user_id' => auth()->id(),
165.              'tournament_id' => $id,
166.              'first_place_team_id' => $request['first'],
167.             'second_place_team_id' => $request['second'],
168.              'third_place_team_id' => $request['third'],
169.              'champion_team_id' => $request['champion'],
170.          ]);
171.
172.          $tournament->update([
173.              'tournament_status' => 1,
174.          ]);
175.          return redirect('listTournaments');
176.      }
177.  }
178.
```

UserController.php

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
```

```
5. use App\User;
6. use App\Model\Team;
7. use App\Model\Promotor;
8. use App\Model\Tournament;
9. use Illuminate\Http\Request;
10.use Illuminate\Support\Facades\Hash;
11.use Illuminate\Validation\ValidationException;
12.use Illuminate\Foundation\Auth\AuthenticatesUsers;
13.use Illuminate\Http\Response;
14.
15.class UserController extends Controller
16.{
17.    use AuthenticatesUsers;
18.
19.    public function index()
20.    {
21.        $tournaments = Tournament::where('user_id',auth()
22.         ->id())->with('user.promotor','payment')->get();
23.        $team = Team::all();
24.       return view('user.profile', compact('tournaments'));
25.    }
26.
27.    public function userTable()
28.    {
29.        $users = User::with('promotor')->all();
30.        return view('user.table',compact('users'));
31.    }
32.
33.    public function contact()
34.    {
35.        return view('errors.coming-soon');
36.    }
37.
38.    public function validatePromotor(Request $request){
39.        $this->validate($request, [
40.            'email' => ['required', 'string', 'email', 'max:255',
   'unique:users'],
41.            'password' => 'required|string',
42.            'company' => 'required',
43.            'phone' => 'required|numeric',
44.            'wa' => 'required|numeric',
45.            'district_id' => 'required',
46.            'terms' => 'required',
47.            'ktp' => ['required','size:file:120'],
48.        ]);
49.    }
50.
```

```
51.     public function storePromotor(Request $request)
52.     {
53.         $user = User::findOrFail(auth()->id());
54.
55.         $this->validatePromotor($request);
56.
57.         if ($user->email == $request['email'] &&
58.        Hash::check($request['password'],$user->password)) {
59.             if($request->hasFile('ktp')){
60.                 $resource = $request->file('ktp');
61.                 $ktp = $resource->getClientOriginalName();
62.                 $tujuan_upload = 'images/ktp';
63.                 $resource->move($tujuan_upload,$ktp);
64.                 Promotor::create([
65.                     'user_id' => Auth()->id(),
66.                     'ktp' => $ktp,
67.                     'company' => $request['company'],
68.                   'district_id' => $request['district_id'],
69.                     'phone' => $request['phone'],
70.                     'wa' => $request['wa'],
71.                 ]);
72.             }
73.             return redirect('profile');
74.         }else{
75.             throw ValidationException::withMessages([
76.             $request['email'] => [trans('auth.failed')],
77.             ]);
78.         }
79.     }
80.
81.     public function approvePromotor($id)
82.     {
83.         $prmtr = Promotor::findOrFail($id);
84.         $prmtr->update([
85.             'promotor_status' => 1,
86.         ]);
87.         $user = User::find($user);
88.         $user->update([
89.             'role_id' => '2',
90.         ]);
91.         return $prmtr;
92.     }
93. }
94.
```

Lampiran 4 *Source Coding Middleware*

app > Http > Middleware >

CheckRole.php

```
1. <?php
2.
3. namespace App\Http\Middleware;
4.
5. use Closure;
6.
7. class CheckRole
8. {
9.     public function handle($request, Closure $next)
10.    {
11.        $roles = $this->CheckRoute($request->route());
12.
13.        if($request->user()->hasRole($roles) || !$roles)
14.        {
15.            return $next($request);
16.        }
17.        return abort(403);
18.    }
19.
20.    private function CheckRoute ($route)
21.    {
22.        $actions = $route->getAction();
23.
24.        return isset($actions['roles']) ? $actions['roles'] : nul
   l;
25.    }
26.}
27.
```

ForceSSL.php

```
1. <?php
2.
3. namespace App\Http\Middleware;
4.
5. use Closure;
6. use Illuminate\Support\Facades\App;
7.
8. class ForceSSL
9. {
```

```
10.     public function handle($request, Closure $next)
11.     {
12.         if (!$request->secure() &&
13.                 App::environment() == 'production') {
14.             return redirect()->secure($request
15.                 ->getRequestUri());
16.         }
17.         return $next($request);
18.     }
19.}
20.
```

Lampiran 5 *Source Coding Model*

app > Http > Model >

Payment.php

```
1. <?php
2.
3. namespace App\Model;
4.
5. use Illuminate\Database\Eloquent\Model;
6.
7. class Payment extends Model
8. {
9.     protected $fillable = ['user_id','team_id','tournament_id',
10.         'user_name','user_email','payment_type','amount',];
11.
12.     public function setPending()
13.     {
14.         $this->attributes['status'] = 'pending';
15.         self::save();
16.     }
17.
18.     public function setSuccess()
19.     {
20.         $this->attributes['status'] = 'success';
21.         self::save();
22.     }
23.
24.     public function setFailed()
25.     {
26.         $this->attributes['status'] = 'failed';
27.         self::save();
28.     }
29.
30.     public function setExpired()
31.     {
32.         $this->attributes['status'] = 'expired';
33.         self::save();
34.     }
35.
36.     public function user()
37.     {
38.         return $this->belongsTo('App\User','user_id');
39.     }
40.
41.     public function team()
```

```
42.     {
43.         return $this->belongsTo('App\Model\Team','team_id');
44.     }
45.
46.     public function tournament()
47.     {
48.         return $this-
   >belongsTo('App\Model\Tournament','tournament_id');
49.     }
50. }
51.
```

Team.php

```
1. <?php
2.
3. namespace App\Model;
4.
5. use Illuminate\Database\Eloquent\Model;
6.
7. class Team extends Model
8. {
9.     protected $fillable = [
10.         'name','tournament_id','user_id',
11.     ];
12.     public function tournament()
13.     {
14.         return $this-
   >belongsTo('App\Model\Tournament', 'tournament_id');
15.     }
16.
17.     public function user()
18.     {
19.         return $this->belongsTo('App\User','user_id');
20.     }
21.
22.     public function payment()
23.     {
24.         return $this->hasOne('App\Model\Payment', 'team_id');
25.     }
26.
27.     public function athlete()
28.     {
29.         return $this->hasMany('App\Model\Athlete', 'team_id');
30.     }
31.     public function complain()
```

```
32.     {
33.         return $this->hasOne('App\Model\Complain', 'team_id');
34.     }
35.}
36.
```

Tournament.php

```
1. <?php
2.
3. namespace App\Model;
4.
5. use Illuminate\Database\Eloquent\Model;
6.
7. class Tournament extends Model
8. {
9.     protected $fillable = [
10.         'title','description','fee','pricepool','slot','date_regi
   s_limit','date_initial','date_final','user_id','athlete','poster'
   ,
11.     ];
12.
13.     public function user()
14.     {
15.         return $this->belongsTo('App\User', 'user_id');
16.     }
17.
18.     public function district()
19.     {
20.         return $this-
   >belongsTo('App\Model\district', 'district_id');
21.     }
22.
23.     public function team()
24.     {
25.         return $this->hasMany('App\Model\Team', 'tournament_id');
26.     }
27.
28.     public function payment()
29.     {
30.         return $this-
   >hasMany('App\Model\Payment', 'tournament_id');
31.     }
32.}
33.
```

app >

User.php

```php
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable implements MustVerifyEmail
{
    use Notifiable;

    protected $fillable = [
        'name', 'email', 'password','role_id',
        'verified_status', 'registerToken',
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'verified_status' => true,
    ];

    public function role()
    {
        return $this->belongsTo('App\Model\Role','role_id');
    }

    public function promotor()
    {
        return $this->hasOne('App\Model\Promotor','user_id');
    }

    public function tournament()
    {
        return $this->hasMany('App\Model\Tournament','user_id');
    }

    public function payment()
    {
```

```
44.         return $this->hasMany('App\Model\Payment','user_id');
45.     }
46.
47.     public function hasRole($roles)
48.     {
49.         $this->have_role = $this->getUserRole();
50.
51.         if (is_array($roles)) {
52.             foreach($roles as $need_role){
53.                 if ($this->checkUserRole($need_role)) {
54.                     return true;
55.                 }
56.             }
57.         } else {
58.             return $this->checkUserRole($roles);
59.         }
60.         return false;
61.     }
62.
63.     private function getUserRole()
64.     {
65.         return $this->role()->getResults();
66.     }
67.
68.     private function checkUserRole($role)
69.     {
70.         return (strtolower($role)==strtolower($this->have_role-
   >name)) ? true : false;
71.     }
72.}
73.
```

Lampiran 6 *Source Coding Providers*

app > Http > Providers >

AppServiceProvider.php

```
1. <?php
2.
3. namespace App\Providers;
4.
5. use App\User;
6. use Illuminate\Support\Facades\Gate;
7. use Illuminate\Support\Facades\Blade;
8. use Illuminate\Support\Facades\Schema;
9. use Illuminate\Support\ServiceProvider;
10.use Illuminate\Support\Facades\Validator;
11.
12.class AppServiceProvider extends ServiceProvider
13.{
14.    public function register(){}
15.
16.    public function boot()
17.    {
18.        Gate::define('isFinish',function($tournament){
19.            return $tournament->tournament_status == 1;
20.        });
21.        Gate::define('isActive',function($user){
22.            return $user->verified_status == '1';
23.        });
24.        Gate::define('isAdmin',function($user){
25.            return $user->role_id == '1';
26.        });
27.        Gate::define('isMember',function($user){
28.            return $user->role_id == '3';
29.        });
30.
31.        Blade::directive('money', function ($money) {
32.            return "<?php echo number_format($money, 0); ?>";
33.        });
34.
35.        Blade::directive('created_at', function ($date) {
36.            return "<?php echo date_format($date,'d M y -
    h : i : s'); ?>";
37.        });
38.
39.        Blade::directive('date', function ($date) {
40.            return "<?php echo date('d M Y',strtotime($date)); ?>
```

```
    ";
41.         });
42.
43.         Schema::defaultStringLength(191);
44.
45.         date_default_timezone_set('Asia/Jakarta');
46.     }
47. }
48.
```

Lampiran 7 *Source Coding config*

config >

database.php

```
1. // ubah sintaks didalam file database.php menjadi seperti ini
2. 'mysql' => [
3.             'driver' => 'mysql',
4.             'url' => env('DATABASE_URL'),
5.             'host' => env('DB_HOST', 'yourdbhost'),
6.             'port' => env('DB_PORT', '3306'),
7.             'database' => env('DB_DATABASE', 'dbname'),
8.             'username' => env('DB_USERNAME', 'yourusername'),
9.             'password' => env('DB_PASSWORD', 'yourpassword'),
10.            'unix_socket' => env('DB_SOCKET', ''),
11.            'charset' => 'utf8mb4',
12.            'collation' => 'utf8mb4_unicode_ci',
13.            'prefix' => '',
14.            'prefix_indexes' => true,
15.            'strict' => true,
16.            'engine' => null,
17.            'options' => extension_loaded('pdo_mysql') ? array_fi
   lter([
18.                PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'
   ),
19.            ]) : [],
20.        ],
21.
```

mail.php

```
1. // ubah sintaks didalam file mail.php menjadi seperti ini
2. 'smtp' => [
3.             'transport' => 'smtp',
4.             'host' => env('MAIL_HOST', 'smtp.gmail.com'),
5.             'port' => env('MAIL_PORT', 587),
6.             'encryption' => env('MAIL_ENCRYPTION', 'tls'),
7.             'username' => env('MAIL_USERNAME','youremail@gmail.co
   m'),
8.             'password' => env('MAIL_PASSWORD','yourpassword'),
9.             'timeout' => null,
10.        ],
11.
12.'from' => [
13. 'address' => env('MAIL_FROM_ADDRESS', 'youremail@gmail.com'),
```

```
14. 'name' => env('MAIL_FROM_NAME', 'Tournament 16 Pak Broto'),
15.   ],
16.
```

service.php

```
1. /*
2.     | Dibagian atas ini ada konfigurasi untuk service lainnya sep
   erti: email, stripe, dsb.
3.     */
4.
5.     'midtrans' => [
6.         // Midtrans server key
7.         'serverKey'     => ('yourserverkey'),
8.         // Midtrans client key
9.         'clientKey'     => ('yourclientkey'),
10.        // Isi false jika masih tahap development dan true jika s
   udah di production, default false (development)
11.        'isProduction'  => env('MIDTRANS_IS_PRODUCTION', false),
12.        'isSanitized'   => env('MIDTRANS_IS_SANITIZED', true),
13.        'is3ds'         => env('MIDTRANS_IS_3DS', true),

14.    ],
15.
```

Lampiran 8 *Source Coding Routes*

routes >

web.php

```
1. <?php
2.
3. use App\User;
4. use App\Model\Payment;
5. use App\Model\Promotor;
6. use App\Model\Tournament;
7. use Illuminate\Support\Facades\Auth;
8. use Illuminate\Support\Facades\Route;
9.
10./*
11.|--------------------------------------------------------------------
   ----------
12.| Modify Routes
13.|--------------------------------------------------------------------
   ----------
14.|
15.| Disini adalah routes untuk memanggil Chain Dropdown Modify
16.|
17.*/
18.Route::get('/getDataProvince','ModifyController@getDataProvince')
   ->name('getDataProvince');
19.Route::get('/select/city/{id}',array('as'=>'select.city', 'uses'=
   >'ModifyController@selectCity'));
20.Route::get('/select/district/{id}',array('as'=>'select.district',
    'uses'=>'ModifyController@selectDistrict'));
21.Route::get('/search', 'ModifyController@searchTournament');
22.
23./*
24.|--------------------------------------------------------------------
   ----------
25.| Winner Routes
26.|--------------------------------------------------------------------
   ----------
27.|
28.| Disini adalah routes untuk memanggil Chain Dropdown Modify
29.| Querynya menggunakan whereNotIn
30.|
31.*/
32.Route::get('getDataChampion/{tournament}', array('as'=>'select.ch
   ampion', 'uses'=>'ModifyController@getDataChampion'));
33.Route::get('/select/first/{champion}/{tournament}',array('as'=>'s
```

```
   elect.first', 'uses'=>'ModifyController@selectFirstPlace'));
34.Route::get('/select/second/{first}/{champion}/{tournament}',array
   ('as'=>'select.second', 'uses'=>'ModifyController@selectSecondPla
   ce'));
35.Route::get('/select/third/{second}/{first}/{champion}/{tournament
   }',array('as'=>'select.third', 'uses'=>'ModifyController@selectTh
   irdPlace'));
36.
37./*
38.|--------------------------------------------------------------------
   ----------
39.| Payment Routes
40.|--------------------------------------------------------------------
   ----------
41.|
42.| Disini adalah routes untuk memangunakan fungsi Payment Midtrans
43.|
44.*/
45.Route::post('/payment/store', 'PaymentController@submitPayment')-
   >name('payment.store');
46.Route::post('/notification/handler', 'PaymentController@notificat
   ionHandler')->name('notification.handler');
47.
48.Route::post('/finish', function(){
49.    return redirect('listTeams');
50.})->name('payment.finish');
51.
52./*
53.|--------------------------------------------------------------------
   ----------
54.| Web Routes
55.|--------------------------------------------------------------------
   ----------
56.|
57.| Here is where you can register web routes for your application.
    These
58.| routes are loaded by the RouteServiceProvider within a group wh
   ich
59.| contains the "web" middleware group. Now create something great
   !
60.|
61.*/
62.Auth::routes(['verify' => true]);
63.
64.Route::get('/', function() {
65.    $user = User::all()->count();
66.    $users = User::all();
```

```
67.     $tournaments = Tournament::orderBy('id','desc')->paginate(6);
68.     $count = Tournament::all()->count();
69.     $promotor = Promotor::all()->count();
70.     $payment = Payment::where('status','success')->count();
71.     return view('home',compact('user','tournaments','promotor','p
   ayment','count','users'));
72. })->name('home');
73.
74. Route::get('/contact', 'UserController@contact');
75.
76. Route::group(['middleware' => ['web', 'auth', 'roles', 'verified'
   ]], function () {
77.
78.     Route::group(['roles' => ['Admin', 'promotor', 'Member']], fu
   nction () {
79.         Route::get('/tournament/detail/{id}', 'TournamentControll
   er@show')->name('tournament.detail');      // R dari CRUD
80.         Route::get('/complain/{id}', 'ComplainController@create')
   ->name('complain.create');
81.         Route::post('/complain/{id}', 'ComplainController@store')
   ->name('complain.store');
82.
83.         Route::get('/profile', 'UserController@index')-
   >name('profile.index');
84.
85.         Route::get('/getDataPayment/{id}', 'PaymentController@get
   DataPayment')->name('getDataPayment');
86.         Route::get('/listTeams', 'TeamController@index')-
   >name('list.Teams');
87.         Route::get('/listTeams/athlete/{id}', 'TeamController@ath
   leteByTeam')->name('list.AthleteByTeam');
88.         Route::post('/team', 'TeamController@store')-
   >name('team.store');
89.         Route::post('/athlete', 'TeamController@addAthlete')-
   >name('athlete.store');
90.         Route::get('/team/{id}', 'TeamController@destroyTeam')
     ->name('team.destroy');
91.         Route::get('/athlete/{id}', 'TeamController@destroyAthlet
   e')->name('athlete.destroy');
92.     });
93.
94.     Route::group(['roles' => 'Member'], function () {
95.         Route::post('upgrade', 'UserController@storePromotor')-
   >name('store.promotor');
96.     });
97.
98.     Route::group(['roles' => 'Admin'], function () {
```

```
99.         Route::get('/approve/{id}', 'UserController@approvePromot
   or')->name('approve.promotor');
100.          Route::get('/listUsers', 'HomeController@userTable')-
   >name('list.Users');
101.          Route::get('/listComplain', 'ComplainController@index'
   )->name('complain.index');
102.      });
103.
104.      Route::group(['roles' => ['Promotor', 'Admin']], function
   () {
105.          Route::get('/listTournaments', 'TournamentController@i
   ndex')->name('tournaments.list');         // R dari CRUD
106.          Route::get('/tournament/{id}', 'TournamentController@d
   estroy')->name('tournament.destroy');     // D dari CRUD
107.          Route::get('/tournament', 'TournamentController@create
   ')->name('tournament.create');            // C dari CRUD
108.          Route::post('/tournament', 'TournamentController@store
   ')->name('turnament.store');              // C dari CRUD
109.          Route::get('/tournament/edit/{id}', 'TournamentControl
   ler@edit')->name('tournament.edit');      // U dari CRUD
110.          Route::put('/tournament/{id}', 'TournamentController@u
   pdate')->name('tournament.update');       // U dari CRUD
111.
112.          Route::get('/tournament/winner/{id}', 'TournamentContr
   oller@winner')->name('tournament.winner');
113.          Route::post('/tournament/winner/{id}', 'TournamentCont
   roller@storeWinner')->name('tournament.storeWinner');
114.      });
115.  });
   116.
```