4. IMPLEMENTASI SISTEM

4.1 Gambaran Umum Program

Sistem dibuat dengan menggunakan bahasa pemograman PHP dengan *framework* Laravel, Javascript, dan menggunakan *database* MySQL. Beberapa *tools* yang digunakan antara lain adalah *Browser* dan Visual Studio Code.

4.2 Pengaturan Koneksi *Database*

Dalam mengakses dan mengelola data yang tersimpan semuanya menggunakan *database* MySQL yang terhubung dengan program pada *file* ".env". *Source code* pengaturan koneksi program dengan *database* dapat terlihat pada segmen program 4

Segmen Program 4. 1 Pengaturan Koneksi Program dengan *Database*

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

4.3 Implementasi Program

Implementasi program merupakan penerapan dari rancangan sistem sesuai dengan desain yang telah dibuat sebelumnya pada bab 3 menjadi sebuah program. Pembuatan program dilakukan secara bertahap mulai dari *login* hingga menu hak akses.

4.3.1 Menu *Login* dan *Logout*

Pada menu ini, pengguna harus *login* untuk dapat mengakses aplikasi. Pengguna harus mengisi *email* dan *password* yang telag dibuat oleh admin pada *database*. Kemudian data tersebut akan dikirimkan ke dalam sistemdan divalidasi. Ketika pengguna selesai mengakses aplikasi, maka pengguna tesebut dapat melakukan *logout*. Proses *login* ke dalam aplikasi

dapat dilihat pada segmen program 4.2 dan proses *logout* dari aplikasi dapat dilihat pada segmen Program 4.3.

Segmen Program 4. 2 *Login* ke dalam Sistem Informasi

```
class LoginController extends Controller
{
        /*
        |--------------------------------------------------------------------------
        | Login Controller
        |--------------------------------------------------------------------------
        */

        use AuthenticatesUsers;

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

        /**
         * Create a new controller instance.
         */
        public function __construct()
        {
                $this->middleware('guest')->except('logout');
        }
}
```

Segmen Program 4. 3 *Logout* dari Sistem Informasi

```
        <div class="dropdown-divider"></div>
        <a href="{{ route('logout') }}" class="dropdown-item"
       onclick="event.preventDefault();
        document.getElementById('logout-form').submit();">
        <i class="ni ni-user-run"></i>
        <span>{{ __('Logout') }}</span>
        </a>
</div>
```

4.3.2 Menu *Raw*

Pada menu ini, admin dapat menambahkan data bahan baku yang ada dengan mengisi *form*. Admin harus mengisi nama dan kode barang.

Kemudian data tersebut akan dikirim ke dalam *database* yang telah dibuat. Proses penambahan data *raw* dapat dilihat pada Segmen Program 4.4

Segmen Program 4. 4 Penambahan Bahan Baku

```
        public function store(Request $request)
        {
        $request->validate([
        'name' => 'required|unique:raws',
        'item_code' => 'required|unique:raws',
        ]);

        $item = new Raw();
        $item->name = $request->input('name');
        $item->item_code = $request->input('item_code');

        $item->save();

        Alert::toast('Item Added Successfully!', 'success')->position('top-end');
        return redirect()->route('raw-item.index');
}
```

4.3.3 Menu *Feed*

Pada menu ini, admin dapat menambahkan data pakan yang ada dengan mengisi *form*. Admin harus mengisi nama dan kode barang. Kemudian data tersebut akan dikirim ke dalam *database* yang telah dibuat. Proses penambahan data *feed* dapat dilihat pada Segmen Program 4.5

Segmen Program 4. 5 Penambahan Pakan

```
        public function store(Request $request)
        {
        $request->validate([
        'name' => 'required|unique:items',
        'item_code' => 'required|unique:items',
        ]);

        $item = new Item();
        $item->name = $request->input('name');
        $item->item_code = $request->input('item_code');

        $item->save();
```

```
        Alert::toast('Item Added Successfully!', 'success')->position('top-end');
        return redirect()->route('item.index');
}
```

4.3.4 Menu *Vendor*

Pada menu ini, admin dapat menambahkan data *vendor* yang ada dengan mengisi *form*. Admin harus mengisi nama dan kode barang. Kemudian data tersebut akan dikirim ke dalam *database* yang telah dibuat. Proses penambahan data cabang dapat dilihat pada Segmen Program 4.6

Segmen Program 4. 6 Penambahan *Vendor*

```
        public function store(Request $request)
        {
                Gate::authorize('app.vendor.create');
        $supplier = new Supplier();
                $supplier->name = $request->input('name');
                $supplier->address = $request->input('address');
                $supplier->phone = $request->input('phone');
                $supplier->save();

                Alert::toast('Item Added Successfully!', 'success')->position('top-
        end');
                return redirect()->route('supplier.index');
}
```

4.3.5 Menu *Purchase Order*

Menu *Purchase Order* berfungsi sebagai menu untuk melakukan pemesanan bahan baku yang sedang dibutuhkan perusahaan untuk membuat pakan. Untuk *input*nya sendiri pengguna akan diminta untuk memilih *item*, *vendor*, jumlah, harga per kilo, kode nota, tanggal *expired*, dan *qc report*. Proses pembelian dapat dilihat pada Segmen Program 4.7

Segmen Program 4. 7 *Purchase Order*

```
        public function store(Request $request)
                {
                        Gate::authorize('app.entry.create');
```

```
                $this->validate($request, [
                        'unit_price' => 'required|numeric',
                        'amount' => 'required|numeric',
                        'receipt' => 'required',
                        'file' => 'nullable|mimes:pdf,jpg,png,jpeg',
                        'qc_report' => 'string',
                ]);

                $ingredient = Ingredient::create([
                        'raw_id' => $request->raw,
                        'supplier_id' => $request->supplier,
                        'user_id' => auth()->user()->id,
                        'unit_price' => $request->unit_price,
                        'amount' => $request->amount,
                        'stock' => $ingredient->recent,
                        'receipt' => $request->receipt,
                        'expired'       => $request->expired,
                        'file' => $request->file,
                        'qc_report' => $request->qc_report,
                ]);

                if ($request->hasFile('file')) {
                        $ingredient->addMedia($request->file)-
>toMediaCollection('file');
                }

                alert()->success('Done!', 'Entry Added successfully');

                return redirect()->route('ingredient.index');}
```

4.3.6 Menu *Pending Approval*

Menu *Pending Approval* berfungsi untuk menampilkan dan memberikan persetujuan pada *order* yang sebelumnya masih belum di *approve*. Proses *Pending Approval* dapat dilihat pada Segmen Program 4.8

Segmen Program 4. 8 *Pending Approval*

```
public function approve(Ingredient $id)
        {
                Gate::authorize('app.entry.approve');
                $ingredients = Ingredient::all()->where('stock', '>', 0);
                $current*Raw*Item = *Raw*::find($id->*raw*_id);
                $current*Raw*Amount = $id->recent; // mengambil existing
amount
```

```
                $currentTotalAmount = $current*Raw*Item-
>total_purchased_amount; //total purchased amount


                $new*Raw*Amount = $id->amount;
                $amountToShow = $current*Raw*Amount +
$new*Raw*Amount; // total amount dimasukkan ke table
                $totalPurchased = $currentTotalAmount +
$new*Raw*Amount;

                $currentCost = $current*Raw*Item->cost; // mengambil cost

                $newCost = $id->amount * $id->unit_price; // total cost

                $costToShow = $currentCost + $newCost; // total cost yang
dimasukkan ke table
                $minExpired = Ingredient::select('expired')-
>where('expired', '>', now())->where('raw_id', $id->raw_id)->min('expired');

                // updating the table
                if ($current*Raw*Amount) {
                        $current*Raw*Item->update([
                                'amount' => $amountToShow,
                                'cost' => $costToShow,
                                'total_purchased_amount' =>
$totalPurchased,
                                'minExpired'                    =>
$minExpired,
                        ]);
                }
                // $ingredients->where('raw_id', $current*Raw*Item->id)-
>first()->stock
                if (!$id->is_approved) {
                        $id->is_approved = true;
                        $id->save();
                        alert()->success('Approved', 'The entry is approved
successfully');
                } else {
                        alert()->info('This entry is already approved');
                }

                return back();
        }
```

4.3.7 Menu *Production*

Menu *production* berfungsi untuk membuat pakan ternak yang berdasarkan bahan baku yang tersedia saat ini. *Input* berupa memilih

feed, jumlah setiap bahan baku yang digunakan, dan *wastage*. Proses produksi dapat dilihat pada Segmen Program 4.9

Segmen Program 4. 9 Penambahan Pakan Hasil Produksi

```
public function store(Request $request)
{
Gate::authorize('app.feed.create');

$raws = Raw::all()->where('amount', '>', 0);
$items = Item::all();
$ingredients = Ingredient::all()->where('stock', '>', 0);
$wastage = $request->wastage;
$check=0;


$this->validate($request, [
'wastage' => 'numeric|max:99',

]);

$totalFeedAmount = 0;
$totalFeedCost = 0;
foreach ($raws as $raw) {

$this->validate($request, [
$raw->id . '-amount' => 'nullable|numeric|max:' . $raw->amount,
]);
$inputtedAmount = $request->all()[$raw->id . '-amount'];
$newRawAmount = $raw->amount - $inputtedAmount;
$avgRawCost = $raw->cost / $raw->total_purchased_amount;
$amountWithWastage = $inputtedAmount - ($inputtedAmount *
($wastage / 100));
$rawCost = $inputtedAmount * $avgRawCost;
$finalCost = $rawCost + ($rawCost * (($wastage / 100)));

$totalFeedAmount += $amountWithWastage;
$totalFeedCost += $finalCost;

$closest = new Carbon('2100-01-23');

// $minExpired = Ingredient::selectRaw('MIN(expired) as expired')-
>where(['raw_id', '=', $raw->id], ['expired', '<>', Carbon::today()]);
// whereDate('expired', '>=', Carbon::today())->get();

// $minExpire = $ingredients->where(['raw_id', '=', $raw->id],
['expired','>=', Carbon::now()->format('Y-m-d')])->min('expired');
// dd($raw->id);
```

```
$minExpire = $ingredients->where('raw_id', $raw->id)->where('stock', '>',
0)->where('expired','>=', Carbon::now()->format('Y-m-d'))->min('expired');
// dd($minExpire);

foreach ($ingredients->where('raw_id', $raw->id)->where('is_approved',
true) as $ingredient){

if($inputtedAmount > 0 && $inputtedAmount >= $ingredient->stock){
$inputtedAmount = $inputtedAmount-$ingredient->stock;
$ingredient->stock = 0;
$minExpire = $ingredients->where('raw_id', $raw->id)->where('stock', '>',
0)->where('expired','>=', Carbon::now()->format('Y-m-d'))->min('expired');
}
elseif ($inputtedAmount < $ingredient->stock) {
$ingredient->stock = $ingredient->stock-$inputtedAmount;
$inputtedAmount = 0;
}
elseif($ingredient->stock){
break;
}

$ingredient->update([
'stock' => $ingredient->stock,

]);
if($check < 1){
$hasil = $ingredients->where('raw_id', $raw->id)->where('is_approved',
true)->sortBy('expired')->first();
// dd($hasil->expired);
$check = 1;
}
}
};

// dd($minExpire);
$raw->update([
'amount' => $newRawAmount,
'minExpired' => $hasil->expired
// ->where([['is_approved', true], ['stock', '>', 0]])
]);

$feed = Feed::create([
'item_id' => $request->item,
'wastage' => $request->wastage,
'amount' => $totalFeedAmount,
'total_amount' => $totalFeedAmount,
'flock' => $request->flock,
'project_name' => $request->project_name,
'cost' => $totalFeedCost,
```

```
            'expired' => $hasil->expired//gimana caranya supaya enggak jadi string, ide
            sementara pake array atau collection untuk menyimpan raw_id yang
            dipakai dalam pembuatan feed

            ]);
            $feed->item->update([
            'minExpired' => $hasil->expired
            ]);

            Alert::toast('Feed Made Successfully!', 'success')->position('top-end');
            return redirect()->route('feed.index');
}
```

4.3.8 Menu *Distribution*

Menu *Distribution* merupakan menu yang berfungsi untuk mengatur keluarnya barang dari tempat pembuatan pakan, Menu ini terbagi menjadi 2 yaitu penjualan pakan dan penjualan bahan baku.

- *Sales*

Submenu untuk menjual pakan. Untuk peng*input*an ada *field Item*, kuantitas, serta nama, alamat, dan nomor telepon dari pembeli. Proses *Sales* dapat dilihat pada Segmen Program 4.10

Segmen Program 4. 10 Penjualan Pakan

```
            public function store(Request $data)
              {
                 Gate::authorize('app.dist.create');


                 foreach ($data['item'] as $key => $value) {
                    $feeds = Feed::all()->where('item_id', $data['item'][$key])-
            >where('expired', '>=', Carbon::now()->format('Y-m-d'))->sortBy('expired');


                    if ($feeds->sum('amount') < $data['qty'][$key]) {
                        return response()->json(['message' => 'Pakan tidak mencukupi'],
            422);
                    }

                    $distribution = new Distribution();
                    $distribution->feed_id = $data['item'][$key];
                    $distribution->customer_id = $data['client'];
                    $distribution->receipt = $data['receipt'];
                    $distribution->unit_price = $data['unit_price'][$key];
```

```
            $distribution->tax = 10;
            $distribution->amount = $data['qty'][$key];
            $distribution->description = $data['description'][$key];
            $distribution->information = $data['other_information'];
            $distribution->user_id = auth()->user()->id;



        alert()->success('Done!', 'Distribution Created Successfully');
        return redirect()->route('distribution.index');
    }

    private function updateFeedStock($feeds, $inputAmount)
    {
;

        foreach ($feeds as $feed){
            // dd($feed->amount);
        if($feed->amount >= $inputAmount){
            $feed->amount -= $inputAmount;

            $feed->update([
                'amount' => $feed->amount,
            ]);

            return;

        }

        else{
            $inputAmount = $inputAmount - $feed->amount;

            $feed->update([
                'amount' => 0,
            ]);

        }


    }
    }
```

- *Allocate*

Submenu untuk mengirim pakan ke peternakan cabang. Untuk peng*input*an ada *field* bahan baku, kuantitas, serta nama, alamat,

dan nomor telepon dari pembeli. Proses alokasi dapat dilihat pada Segmen Program 4.11

Segmen Program 4. 11 *Allocate*

```
        public function store(Request $request)
        {
                Gate::authorize('app.dist.create');
                $rawItem = Raw::find($request->raw);

                $this->validate($request, [
                        'unit_price' => 'required|numeric',
                        'amount' => 'required|numeric|max:'.$rawItem->amount,
                        'buyer_name' => 'required|string|max:255',
                        'buyer_address' => 'required',
                        'buyer_phone' => 'required'
                ]);

                $distribution = Distribution::create([
                        'raw_id' => $request->raw,
                        'unit_price' => $request->unit_price,
                        'amount' => $request->amount,
                        'buyer_name' => $request->buyer_name,
                        'buyer_address' => $request->buyer_address,
                        'buyer_phone' => $request->buyer_phone,
                ]);

                //update raw amount
                $rawAmount = $rawItem->amount;
                $soldAmount = $request->amount;
                $newRawAmount = $rawAmount - $soldAmount;
                $rawItem->update([
                        'amount' => $newRawAmount
                ]);

                alert()->success('Done!', 'Distribution Created Successfully');
                return redirect()->route('distribution.index');
}
```

4.3.9 Menu *Roles & Permission*

Menu ini berfungsi untuk membuat *roles* beserta hak aksesnya. Untuk *input*nya sendiri berupa nama dari *roles* dan *checklist* setiap hak akses yang ingin diberikan. Proses penambahan *roless* dapat dilihat pada Segmen Program 4.12

Segmen Program 4. 12 Penambahan *Roles*

```
public function store(Request $data)
  {
    Gate::authorize('app.dist.create');

    foreach ($data['item'] as $key => $value) {
      $feeds = Feed::all()->where('item_id', $data['item'][$key])-
>where('expired', '>=', Carbon::now()->format('Y-m-d'))->sortBy('expired');


      if ($feeds->sum('amount') < $data['qty'][$key]) {
          return response()->json(['message' => 'Pakan tidak mencukupi'],
422);
        }

        $transfer = new Transfer();
        $transfer->feed_id = $data['item'][$key];
        $transfer->branch_id = $data['client'];
        $transfer->receipt = $data['receipt'];
        $transfer->amount = $data['qty'][$key];


        $transfer->description = $data['description'][$key];
        $transfer->information = $data['other_information'];
        $transfer->user_id = auth()->user()->id;


    }


    alert()->success('Done!', 'Transfer Created Successfully');

    return redirect()->route('distribution.index');

  }

  private function updateFeedStock($feeds, $inputAmount)
  {


    foreach ($feeds as $feed){
      // dd($feed->amount);
    if($feed->amount >= $inputAmount){
      $feed->amount -= $inputAmount;

      $feed->update([
        'amount' => $feed->amount,
      ]);
```

```
                return;

            }

            else{
                $input Amount = $input Amount - $feed->amount;

                $feed->update([
                    'amount' => 0,
                ]);

            }
        }


            }
```

4.3.10 Menu *User Management*

Menu *User Management* berfungsi untuk mengelola *user* yang dapat mengakses sistem informasi manajemen pakan. Untuk *input*nya sendiri berupa nama, *email*, *password* pengguna, disertai peng*input*an *roles*. Proses penambahan *user* dapat dilihat pada Segmen Program 4.13.

Segmen Program 4. 13 Penambahan *User*

```
        public function store(Request $request)
        {
                Gate::authorize('app.roless.create');
        $this->validate($request, [
        'name' => 'required',
        'permissions' => 'required|array',
        'permissions.*' => 'integer'
        ]);

        Role::create([
        'name' => $request->name,
        'slug' => str_slug($request->name)
        ])->permissions()->sync($request->input('permissions', []));

        return redirect()->route('roless.index')->with('success', 'Role Created
        Successfully');
}
```