Listing Program

1. PenjualanController.php

```
<?php

namespace App\Http\Controllers;


use App\Models\DetailPenjualan;
use App\Models\Produk;
use App\Models\Penjualan;
use Barryvdh\DomPDF\Facade\Pdf;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Yajra\DataTables\DataTables;
use Illuminate\Support\Facades\Log;


class PenjualanController extends Controller
{




    public function index(Request $request)
    {
        if ($request->ajax()) {
            // dd($request->all());
            $query_data = Penjualan::select('penjualans.*');



            if ($request->sSearch) {
                $search_value = '%' . $request->sSearch . '%';
                $query_data = $query_data->where(function
($query) use ($search_value) {
                    $query
                        ->where('keterangan', 'like', $search_value)
                        ->orWhere('created_at', 'like',
$search_value)
                        ->orWhere('created_by', 'like',
$search_value);
                });
            }
```

```
            return Datatables::of($query_data)
                ->addIndexColumn()
                ->addColumn('total', function ($row) {
                    return number_format($row->get_total(), 0, '.',
',');
                })
                ->addColumn('action', function ($row) {
                    $btn = '<form action="' .
route('penjualans.destroy', $row->id) . '" method="POST">
                <a class="btn btn-info" href="' .
route('penjualans.show', $row->id) . '"> <i class="fa fa-eye"></i>
</a>  ';
                    // dd(Auth::user());
                    $btn = $btn . '<a class="btn btn-primary"
href="' .
                        route('cetak_invoice', ['id' => $row->id]) .
'"><i class="fa fa-print mx-3"></i></a>  ';
                    $confirm = 'onclick="return confirm(\'Hapus
data?\')"';
                    $btn = $btn . csrf_field() .
method_field('DELETE') . '<button type="submit" class="btn
btn-danger"' . $confirm . '><i class="fa fa-trash"></i></button>
 ';
                    $btn = $btn . '</form>';


                    return $btn;
                })
                ->rawColumns(['action'])
                ->make(true);
        }


        return view('penjualans.index');
    }



    public function put_order(Request $request)
    {
        try {
            $input = $request->input('data');
            Log::info($input);


            $pecah = explode('|', $input);
            $berat = $pecah[1];
            $kode_barang = trim($pecah[2]);
```

```
            $produk = Produk::where('kode_produk', 'like',
"%$kode_barang%")->firstOrFail();
            $harga_gram = $produk->harga_produk;
            $data_detail = [
                "id_penjualan" => 0,
                "id_produk" => $produk->id,
                "berat" => $berat,
                "harga_gram" => $harga_gram,
                "harga_total" => $berat * $harga_gram
            ];
            DetailPenjualan::create($data_detail);


            return json_encode(["success" => 1]);
        } catch (\Exception $e) {
            Log::info($e->getMessage());
            return response()->json([
                'success' => 0
            ]);
        }
    }



    public function create()
    {
        $product = Produk::all();
        $detail_penjualan =
DetailPenjualan::where('id_penjualan', '=', 0)->get();
        return view('penjualans.create', compact('product',
'detail_penjualan'));
    }


    public function store(Request $request)
    {
        // dd($request->all());
        try {


            DB::beginTransaction();
            $input = $request->all();



            $penjualan = Penjualan::create($input);
```

```
            $detail = DetailPenjualan::where('id_penjualan', '=',
0)->update(['id_penjualan' => $penjualan->id]);


            // $produk = $request->input('detail_produk', []);
            // $berat = $request->input('berat', []);
            // $satuan = $request->input('satuan', []);


            // for ($index = 0; $index < count($produk);
$index++) {
            //       if ($produk[$index] != '') {
            //           $harga_gram =
str_replace(".","",$satuan[$index]);
            //           DetailPenjualan::create([
            //               'id_penjualan' => $penjualan->id,
            //               'id_produk' => $produk[$index],
            //               'berat' => $berat[$index],
            //               'harga_gram' => $harga_gram,
            //               'harga_total' => floatval($berat[$index])
* floatval($harga_gram) ,
            //           ]);
            //       }
            // }


            DB::commit();
            return redirect()->route('penjualans.index')
                ->with('success', 'PenjualanController created
successfully.');
        } catch (\Exception $e) {
            DB::rollBack();
            return redirect()->back()->withInput()
                ->with('error', $e->getMessage());
        }
    }



    public function cetak_invoice(Request $request)
    {
        $id = $request->id;
        $penjualan = Penjualan::find($id);
        return view('penjualans.invoice', compact('penjualan'));
    }



    public function show($id)
```

```
    {
        $penjualan = Penjualan::find($id);
        return view('penjualans.show', compact('penjualan'));
    }


    public function destroy(Penjualan $penjualan)
    {
        $penjualan->delete();


        return redirect()->route('penjualans.index')
            ->with('success', 'PenjualanController deleted
successfully');
    }


    public function fetchTableData()
    {
        $detail_penjualan =
DetailPenjualan::where('id_penjualan', '=', 0)->get();
        $total = 0;
        $tableData = '';


        foreach ($detail_penjualan as $dp) {
            $tableData .= '<tr>';
            $tableData .= '<td>' . $dp->produk?->nama_produk .
'</td>';
            $tableData .= '<td>' . $dp->berat . '</td>';
            $tableData .= '<td>' .
number_format($dp->harga_gram) . '</td>';
            $tableData .= '<td>' .
number_format($dp->harga_total) . '</td>';
            $tableData .= '</tr>';
            $total += $dp->harga_total;
        }


        return response()->json([
            'tableData' => $tableData,
            'total' => number_format($total),
        ]);
    }


    public function filter_by_date(Request $request)
    {
        $input = $request->all();
```

```
        $penjualans = Penjualan::whereBetween('created_at',
[$input['start_date'], $input['end_date']])
                ->get();
                $pdf =
PDF::loadview('penjualans.exportpenjalanbulanan-pdf',
['penjualans' => $penjualans, 'start_date' => $input['start_date'],
'end_date' => $input['end_date']])->setPaper('a4', 'landscape');
                return $pdf->stream('Laporanpenjualan.pdf');
    }
}
```

2. ProdukController.php

```
<?php

namespace App\Http\Controllers;


use App\Models\Gudang;
use App\Models\Produk;
use App\Models\DetailProduk;
use App\Models\Kategori_produk;
use App\Models\Stock;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Yajra\DataTables\DataTables;
use Barryvdh\DomPDF\Facade\Pdf;
use Illuminate\Support\Facades\DB;


class ProdukController extends Controller
{
    function __construct()
    {
        $this->middleware('permission:produk-list|produk-
create|produk-edit|produk-delete', ['only' => ['index', 'store']]);
        $this->middleware('permission:produk-create', ['only' =>
['create', 'store']]);
        $this->middleware('permission:produk-edit', ['only' =>
['edit', 'update']]);
        $this->middleware('permission:produk-delete', ['only' =>
['destroy']]);
    }


    public function index(Request $request)
    {
        if ($request->ajax()) {
```

```
            $query_data = new Produk();

            // dd($query_data->toSql());


            if ($request->sSearch) {
                $search_value = '%' . $request->sSearch . '%';
                $query_data = $query_data->where(function
($query) use ($search_value) {
                    $query
                        ->where('kode_produk', 'like',
$search_value)
                        ->orWhere('nama_produk', 'like',
$search_value)
                        ->orWhere('barcode_produk', 'like',
$search_value)
                        ->orWhere('harga_produk', 'like',
$search_value);
                });
            }


            // $data = $query_data->orderBy('nama_produk',
'asc');
            $data = $query_data->whereNull('deleted_at');
            // dd($data);
            return Datatables::of($data)
                ->addIndexColumn()


                ->addColumn('action', function ($row) {
                    $btn = '<form action="' .
route('produks.destroy', $row->id) . '" method="POST">
                    <a class="btn btn-info" href="' .
route('produks.show', $row->id) . '"><i class="fa fa-
eye"></i></a>  ';


                    if (Auth::user()->can('produk-edit')) {
                        $btn = $btn . '<a class="btn btn-primary"
href="' .
                            route('produks.edit', $row->id) . '"><i
class="fa fa-edit"></i></a>  ';
                    }
                    if (Auth::user()->can('produk-delete')) {
                        $confirm = 'onclick="return
confirm(\'Hapus data?\')"';
```

```
                        $btn = $btn . csrf_field() .
method_field('DELETE') . '<button type="submit" class="btn
btn-danger"' . $confirm . '><i class="fa fa-trash"></button>';
                    }
                    $btn = $btn . '</form>';


                    return $btn;
                })
                ->rawColumns(['action'])
                ->make(true);
        }

        return view('produks.index');
    }

    public function stock()
    {
        $stocks = Stock::all();
        return view('produks.stock', compact('stocks'));
    }

    public function create()
    {
        return view('produks.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'kode_produk' => 'required',
            'nama_produk' => 'required',
            'harga_produk' => 'required',


        ]);
        DB::beginTransaction();
        $input = $request->all();
        // dd($input);
        $produk = Produk::create($input);


        DB::commit();
        return redirect()->route('produks.index')
            ->with('success', 'Produk created successfully.');
    }
```

```
public function show($id)
{
    $produk = Produk::where('id', $id)->first();
    return view('produks.show', compact('produk'));
}


public function edit(Produk $produk)
{
    return view('produks.edit', compact('produk'));
}


public function update(Request $request, $produk)
{
    $request->validate([
        'kode_produk' => 'required',
        'nama_produk' => 'required',
        'harga_produk' => 'required',


    ]);
    try {
        DB::beginTransaction();
        $input = $request->all();



        $item = Produk::where('id', $produk)->first();


        $item->update($input);



        DB::commit();
        return redirect()->route('produks.index')
            ->with('success', 'Produk created successfully.');
    } catch (\Exception $e) {
        DB::rollBack();
        return redirect()->back()->withInput()
            ->with('error', $e->getMessage());
    }
}


public function destroy(Produk $produk)
{
```

```
        $produk->delete();


        return redirect()->route('produks.index')
            ->with('success', 'Produk deleted successfully');
    }
    function exportPDF()
    {
        $produks = Produk::all();
        $pdf = PDF::loadview('produks.exportproduk-pdf',
['produks' => $produks])->setPaper('a4', 'landscape');
        return $pdf->stream('Laporanproduk.pdf');
    }


    public function cetak_reportProduk()
    {
        return view('produks.reportProduk');
    }


    public function proses_cetak_reportProduk(Request $request)
    {
        $input = $request->all();
        $produks = Produk::whereBetween('created_at',
[$input['start_date'], $input['end_date']])
            ->get();
        $pdf = PDF::loadview('produks.exportprodukbulanan-
pdf', ['produks' => $produks, 'start_date' => $input['start_date'],
'end_date' => $input['end_date']])->setPaper('a4', 'landscape');
        return $pdf->stream('Laporanproduk.pdf');
    }
}
```

3. RoleController.php

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Spatie\Permission\Models\Role;
use Spatie\Permission\Models\Permission;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;


class RoleController extends Controller
```

```
{
    /**
     * * Display a listing of the resource.
     * *
     * * @return \Illuminate\Http\Response
     * */


    function __construct()
    {
        $this->middleware(
            'permission:role-list|role-create|role-edit|role-delete',
            ['only' => ['index', 'store']]
        );
        $this->middleware('permission:role-create', [
            'only' => ['create', 'store'],
        ]);
        $this->middleware('permission:role-edit', [
            'only' => ['edit', 'update'],
        ]);
        $this->middleware('permission:role-delete', ['only' =>
['destroy']]);
    }
    /**
     * * Display a listing of the resource.
     * *
     * * @return \Illuminate\Http\Response
     * */


    public function index(Request $request)
    {
        // $roles = Role::orderBy('id', 'DESC')->paginate(5);
        // return view('roles.index', compact('roles'))->with(
        //        'i',
        //        ($request->input('page', 1) - 1) * 5
        // );


        $roles = Role::orderBy('id', 'DESC')->get();
        return view('roles.index', compact('roles'));
    }


    /**
     * * Show the form for creating a new resource.
     * *
     * * @return \Illuminate\Http\Response
```

```
 * */
public function create()
{
    $permission = Permission::get();
    return view('roles.create', compact('permission'));
}


/**
 * * Store a newly created resource in storage.
 * *
 * * @param \Illuminate\Http\Request $request
 * * @return \Illuminate\Http\Response
 * */


public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|unique:roles,name',
        'permission' => 'required',
    ]);
    $role = Role::create(['name' => $request->input('name')]);
    $role->syncPermissions($request->input('permission'));
    return redirect()
        ->route('roles.index')
        ->with('success', 'Role created successfully');
}


/**
 * * Display the specified resource.
 * *
 * * @param int $id
 * * @return \Illuminate\Http\Response
 * */
public function show($id)
{
    $role = Role::find($id);
    $rolePermissions = Permission::join(
        'role_has_permissions',
        'role_has_permissions.permission_id',
        '=',
        'permissions.id'
    )
        ->where('role_has_permissions.role_id', $id)
        ->get();
```

```
        return view('roles.show', compact('role',
'rolePermissions'));
    }


    /**
     * * Show the form for editing the specified resource.
     * *
     * * @param int $id
     * * @return \Illuminate\Http\Response
     * */


    public function edit($id)
    {
        $role = Role::find($id);
        $permission = Permission::get();
        $rolePermissions = DB::table('role_has_permissions')
            ->where('role_has_permissions.role_id', $id)
            ->pluck(
                'role_has_permissions.permission_id',
                'role_has_permissions.permission_id'
            )
            ->all();
        return view(
            'roles.edit',
            compact('role', 'permission', 'rolePermissions')
        );
    }


    /**
     * * Update the specified resource in storage.
     * *
     * * @param \Illuminate\Http\Request $request
     * * @param int $id
     * * @return \Illuminate\Http\Response
     * */
    public function update(Request $request, $id)
    {
        $this->validate($request, [
            'name' => 'required',
            'permission' => 'required',
        ]);


        $role = Role::find($id);
        $role->name = $request->input('name');
```

```
        $role->save();
        $role->syncPermissions($request->input('permission'));
        return redirect()
            ->route('roles.index')
            ->with('success', 'Role updated successfully');
    }


    /**
     * * Remove the specified resource from storage.
     * *
     * * @param int $id
     * * @return \Illuminate\Http\Response
     * */
    public function destroy($id)
    {
        DB::table('roles')
            ->where('id', $id)
            ->delete();
        return redirect()
            ->route('roles.index')
            ->with('success', 'Role deleted successfully');
    }
}
```

4. Controller.php

```
<?php

namespace App\Http\Controllers;


use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;


class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

5. UserController.php

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
```

```
use App\Models\User;
use Spatie\Permission\Models\Role;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Arr;


class UserController extends Controller
{
    //
    public function index(Request $request)
    {
        // $data = User::orderBy('id', 'DESC')->paginate(5);
        // return view('users.index', compact('data'))->with(
        //        'i',
        //        ($request->input('page', 1) - 1) * 5
        // );


        $data = User::orderBy('id', 'DESC')->get();
        $i=0;
        return view('users.index', compact('data','i'));
    }
    /**
     * * Show the form for creating a new resource.
     * *
     * * @return \Illuminate\Http\Response
     * */


    public function create()
    {
        $roles = Role::pluck('name', 'name')->all();
        return view('users.create', compact('roles'));
    }
    /**
     * * Store a newly created resource in storage.
     * *
     * * @param \Illuminate\Http\Request $request
     * * @return \Illuminate\Http\Response
     * */


    public function store(Request $request)
    {
        $this->validate($request, [
            'name' => 'required',
```

```
            'email' => 'required|email|unique:users,email',
            'password' => 'required|same:confirm-password',
            'roles' => 'required',
        ]);
        $input = $request->all();
        $input['password'] = Hash::make($input['password']);
        $user = User::create($input);
        $user->assignRole($request->input('roles'));
        return redirect()
            ->route('users.index')
            ->with('success', 'User created successfully');
    }
    /**
     * * Display the specified resource.
     * *
     * * @param int $id
     * * @return \Illuminate\Http\Response
     * */


    public function show($id)
    {
        $user = User::find($id);
        return view('users.show', compact('user'));
    }
    /**
     * * Show the form for editing the specified resource.
     * *
     * * @param int $id
     * * @return \Illuminate\Http\Response
     * */


    public function edit($id)
    {
        $user = User::find($id);
        $roles = Role::pluck('name', 'name')->all();
        $userRole = $user->roles->pluck('name', 'name')->all();
        return view('users.edit', compact('user', 'roles',
'userRole'));
    }
    /**
     * * Update the specified resource in storage.
     * *
     * * @param \Illuminate\Http\Request $request
     * * @param int $id
     ** @return \Illuminate\Http\Response
```

```
     */

    public function update(Request $request, $id)
    {
        $this->validate($request, [
            'name' => 'required',
            'email' => 'required|email|unique:users,email,' . $id,
            'password' => 'same:confirm-password',
            'roles' => 'required',
        ]);
        $input = $request->all();
        if (!empty($input['password'])) {
            $input['password'] = Hash::make($input['password']);
        } else {
            $input = Arr::except($input, ['password']);
        }
        $user = User::find($id);
        $user->update($input);
        DB::table('model_has_roles')
            ->where('model_id', $id)
            ->delete();
        $user->assignRole($request->input('roles'));
        return redirect()
            ->route('users.index')
            ->with('success', 'User updated successfully');
    }
    /**
     * * Remove the specified resource from storage.
     * *
     * * @param int $id
     * * @return \Illuminate\Http\Response
     * */

    public function destroy($id)
    {
        User::find($id)->delete();
        return redirect()
            ->route('users.index')
            ->with('success', 'User deleted successfully');
    }
}
```

6. DetailPenjualan.php

```
<?php

namespace App\Models;


use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Sis\TrackHistory\HasTrackHistory;


class DetailPenjualan extends Model
{
    use HasFactory, SoftDeletes;


    protected $table= 'detail_penjualans';
    protected $fillable = [
        'id_penjualan',
        'id_produk',
        'berat',
        'harga_gram',
        'harga_total'
    ];



    public function penjualan(){
        return $this->hasOne(Penjualan::class,'id','id_penjualan');
    }


    public function produk(){
        return $this->hasOne(Produk::class,'id','id_produk');
    }



    public function user()
    {
        return $this->hasOne(User::class, 'id', 'created_by');
    }


}
```

7. Produk.php

```
<?php

namespace App\Models;


use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Sis\TrackHistory\HasTrackHistory;


class Produk extends Model
{
    use HasFactory, SoftDeletes;
    protected $fillable = [
        'kode_produk',
        'nama_produk',
        'barcode_produk',
        'harga_produk',

    ];


    public function user(){
        return $this->hasOne(User::class, 'id', 'created_by');
    }


}
```

8. Penjualan.php

```
<?php

namespace App\Models;


use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Sis\TrackHistory\HasTrackHistory;


class Penjualan extends Model
{
    use HasFactory, SoftDeletes;
    protected $fillable = [
```

```
        'keterangan',
        'created_by',
    ];


    public static function generateNumber()
    {
        // Get the current month
        $currentYear = Carbon::now()->format('Y');
        $currentMonth = Carbon::now()->format('m');


        // Increment the counter for the current month
        $counter = DB::table('penjualans')
        ->whereMonth('created_at', $currentMonth)
        ->whereYear('created_at', $currentYear)
        ->max('po_counter') + 1;


        // Generate the PO number
        $poNumber = 'PO-' . $currentMonth . '-' .
str_pad($counter, 3, '0', STR_PAD_LEFT);


        return $poNumber;
    }


    public function get_total(){
        $total =
DetailPenjualan::where('id_penjualan','=',$this->id)->sum('harga_
total');
        return $total;

    }


    public function detail_penjualan(){
        return
$this->hasMany(DetailPenjualan::class,'id_penjualan','id');
    }




    public function user()
    {
        return $this->hasOne(User::class, 'id', 'created_by');
```

```
    }
}
```

9. User.php

```
<?php

namespace App\Models;


// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;


class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;


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


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


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


    public function pegawai(){
        return $this->hasOne(Pegawai::class,'email','email');
    }
}
```

Program Esp32

```
#include <SoftwareSerial.h>
#include <WiFi.h>
#include <HTTPClient.h>

// Define the RX and TX pins for SoftwareSerial
#define RX_PIN 16
#define TX_PIN 17

SoftwareSerial mySerial(RX_PIN, TX_PIN);

// Wi-Fi credentials
const char* ssid = "vina";
const char* password = "12341234";

// Server endpoint
const char* serverName =
"http://192.168.240.240:8000/api/put_order";

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);    // Adjust the baud rate as needed

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to Wi-Fi");
```

```
    // sendDataToServer("KIRIM|100|12345");
}

void loop() {
    if (mySerial.available()) {
        String data = mySerial.readStringUntil('\n');
        Serial.println(data);
        if (data.indexOf("KIRIM") >= 0) {
            sendDataToServer(data);
        }
    }
}

void sendDataToServer(String data) {
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin(serverName);

        // Tentukan tipe konten dan buat payload
        http.addHeader("Content-Type", "application/x-www-form-
urlencoded");
        String payload = "data=" + data;

        // Kirim request POST dengan payload
        int httpResponseCode = http.POST(payload);

        if (httpResponseCode > 0) {
            String response = http.getString();
            Serial.println(httpResponseCode);
            Serial.println(response);
        } else {
```

```
      Serial.print("Error on sending POST request: ");
      Serial.println(httpResponseCode);
    }

    http.end();
  } else {
    Serial.println("Wi-Fi Disconnected");
  }
}
```

Program Arduino

```
#include <DFRobot_HX711.h>
DFRobot_HX711 MyScale(10, 11);

// #include <MultitapKeypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>

// Define the number of rows and columns on the keypad
const byte ROWS = 4;
const byte COLS = 4;

// Define the keymap
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};
```

```
LiquidCrystal_I2C lcd(0x27, 20, 4);  // set the LCD address to 0x27 for a 16 chars and 2 line display

const byte ROW0 = A0;
const byte ROW1 = A1;
const byte ROW2 = A2;
const byte ROW3 = A3;
const byte COL0 = 5;
const byte COL1 = 4;
const byte COL2 = 3;
const byte COL3 = 2;
const byte BEEP = 13;

byte rowPins[ROWS] = { ROW0, ROW1, ROW2, ROW3 };

// Connect keypad COL0, COL1, COL2, COL3 to Arduino pins
byte colPins[COLS] = { COL0, COL1, COL2, COL3 };

// Create the Keypad object
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// MultitapKeypad kpd(ROW0, ROW1, ROW2, ROW3, COL0, COL1, COL2, COL3);
// Key key;
String kode_barang = "";

#include <SoftwareSerial.h>

SoftwareSerial barcode_reader(6, 7);  // RX, TX
```

```
String inputString = "";    // a String to hold incoming data
bool stringComplete = false;

void setup() {
   Serial.begin(9600);
   barcode_reader.begin(9600);
   lcd.init();
   lcd.backlight();
   lcd.setCursor(0, 0);
   lcd.print("Selamat Datang");
   lcd.setCursor(0, 1);
   lcd.print("Tk.Lelly Frends");
   delay(1000);
   lcd.clear();
   MyScale.setCalibration(-285); //-1992

   // while(1){
   //      baca_barcode();
   // }
}

float bobot = 0;
bool selesai = false;
void loop() {
   bobot = 0;
   long waktu_stabil = millis();
   while (bobot < 10 || millis() - waktu_stabil < 10000) {
      lcd.setCursor(0, 0);
      lcd.print("Berat:");
      bobot = hitung_berat();
      lcd.print(bobot);
```

```
        lcd.print("        ");
        if (bobot < 10) {
            waktu_stabil = millis();
        }
    }

    //selesai baca berat
    lcd.setCursor(0, 1);
    lcd.print("kode:");
    kode_barang = "";
    selesai = false;
    while (!selesai) {
        baca_keypad();
        baca_barcode();
        //baca serial;
    }
}

void baca_barcode() {
    barcodeEvent();
    if (stringComplete) {
        Serial.println(inputString);
        // clear the string:
        kode_barang += inputString;
        lcd.setCursor(0, 1);
        lcd.print("kode:");
        lcd.print(kode_barang);
        inputString = "";
        stringComplete = false;
    }
}
```

```
void barcodeEvent() {
   while (barcode_reader.available()) {
      // get the new byte:
      char inChar = (char)barcode_reader.read();
      // add it to the inputString:
      // if the incoming character is a newline, set a flag so the main
loop can
      // do something about it:
      if (inChar == '\n') {
         stringComplete = true;
      } else {
         inputString += inChar;
      }
   }
}

float hitung_berat() {    // Get the weight of the object
   float berat = MyScale.readWeight();
   if (berat <0){
      berat = 0;
   }
   Serial.print(berat, 1);
   Serial.println(" g");
   // delay(200);

   return berat;
}


void baca_keypad() {
```

```
char key = keypad.getKey();    // Read the key

if (key) {                          // If a key is pressed
   Serial.println(key);    // Print the key to the serial monitor
   if (key == '#') {
      Serial.println("POST API");

      Serial.print("KIRIM");
      Serial.print("|");
      Serial.print(int(bobot));
      Serial.print("|");
      Serial.print(kode_barang);
      Serial.println("|");

      kode_barang = "";
      lcd.clear();
      selesai = true;

   } else if (key == "") {

   } else {
      kode_barang += key;
      lcd.setCursor(0, 1);
      lcd.print("kode:");
      lcd.print(kode_barang);
      Serial.println(kode_barang);
   }
}
// char karakter = "";
// key = kpd.getKey();
// Serial.print(F("Key "));
```

```
// // delay(200);
// if (key.character > 0) {
//     Serial.print(char(key.character));
//     // kode_barang = kode_barang + char(key.character);
//     karakter = char(key.character);
// } else
//     Serial.print(F("??"));
// switch (key.state) {
//     case KEY_DOWN:
//     case MULTI_TAP:
//         tone(BEEP, 5000, 20);
//         Serial.println(F(" down"));
//         if (key.state == MULTI_TAP) {
//             Serial.print(F("TapCounter: "));
//             if (key.tapCounter < 10)
//                 Serial.println(key.tapCounter, DEC);
//             else {
//                 kpd.resetTapCounter();
//                 Serial.println(0);
//             }
//         }
//         break;
//     case LONG_TAP:
//         tone(BEEP, 5000, 20);
//         Serial.println(F(" hold"));
//         break;
//     case MULTI_KEY_DOWN:
//         tone(BEEP, 4000, 100);
//         Serial.println(F(" down"));

//         break;
```

```
    //      case KEY_UP:
    //          Serial.println(F(" up"));
    //          if (karakter == '#') {
    //              Serial.println("POST API");
    //          } else if (karakter == ""){

    //          }else{
    //              kode_barang += karakter;

    //          }
    //          Serial.println(kode_barang);
    // }
}
```

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REKOMENDASI UJIAN TUGAS AKHIR

Pembimbing Laporan Tugas Akhir, memberikan rekomendasi ujian laporan tugas akhir kepada,

Nama Mahasiswa	: Fina Meysita
NIM	: 062130700204
Jurusan/Program Studi	: Teknik Komputer/D-III Teknik Komputer
Judul Tugas Akhir	: Rancang Bangun Timbangan Digital Berbasis *Website* Pada Toko Bahan Pangan Lelly Frends Menggunakan Sensor *Barcode*

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Laporan Tugas Akhir, pada Tahun Akademik 2023/ 2024

Palembang, 30 Juli 2024

Disetujui oleh,

Pembimbing I

Ema Laila, S.Kom., M.Kom
NIP. 197703292001122002

Pembimbing II

Adi Sutrisman,S.Kom., M.Kom
NIP. 197503052001121005

KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website: www.polsri.ac.id **Email:**info@polsri.ac.id

REVISI TUGAS AKHIR (TA)

Dosen Penguji	: Yulian Mirza, S.T., M.Kom
Nama Mahasiswa	: Fina Meysita
NIM	: 062130700204
Jurusan/Program Studi	: Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	: Rancang Bangun Timbangan Digital Berbasis *Website* Pada Toko Bahan Pangan Lelly Frends Menggunakan Sensor *Barcode*

No	Uraian Revisi	Paraf
	tata tulis Pembahasan Kesimpulan Perbaikan alat	

Palembang, 31 Juli 2024
Dosen Penguji

Yulian Mirza, S.T., M.Kom
NIP. 196607121990031003

KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website: www.polsri.ac.id **Email**:info@polsri.ac.id

REVISI TUGAS AKHIR (TA)

Dosen Penguji	: Ir. Alan Tompunu, S.T., M.T., IPM., ASEAN Eng
Nama Mahasiswa	: Fina Meysita
NIM	: 062130700204
Jurusan/Program Studi	: Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	: Rancang Bangun Timbangan Digital Berbasis *Website* Pada Toko Bahan Pangan Lelly Frends Menggunakan Sensor *Barcode*

No	Uraian Revisi	Paraf
	Rancang Alat datala.kp@gmail.com	

Palembang, 31 Juli 2024
Dosen Penguji

Ir. Alan Tompunu, S.T., M.T., IPM., ASEAN Eng
NIP. 196611082000031002

KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website: www.polsri.ac.id **Email**:info@polsri.ac.id

REVISI TUGAS AKHIR (TA)

Dosen Penguji	: Hartati Deviana, S.T., M.Kom
Nama Mahasiswa	: Fina Meysita
NIM	: 062130700204
Jurusan/Program Studi	: Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	: Rancang Bangun Timbangan Digital Berbasis *Website* Pada Toko Bahan Pangan Lelly Frends Menggunakan Sensor *Barcode*

No	Uraian Revisi	Paraf
	Perbaiki alat, diagram blok. flowchart, pengujian	

Palembang, 31 Juli 2024
Dosen Penguji

Hartati Deviana, S.T., M.Kom
NIP. 197405262008122001

KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website: www.polsri.ac.id Email:info@polsri.ac.id

REVISI TUGAS AKHIR (TA)

Dosen Penguji	: Rian Rahmanda, S.Kom., M.Kom
Nama Mahasiswa	: Fina Meysita
NIM	: 062130700204
Jurusan/Program Studi	: Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	: Rancang Bangun Timbangan Digital Berbasis *Website* Pada Toko Bahan Pangan Lelly Frends Menggunakan Sensor *Barcode*

No	Uraian Revisi	Paraf
1.	Data pengujian perlu ditambahkan dengan tujuan yang ingin dicapai.	
2.	Data harus divalidasi	

Palembang, 31 Juli 2024
Dosen Penguji

Rian Rahmanda, S.Kom., M.Kom
NIP. 198901252019031013

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

PELAKSANAAN REVISI UJIAN TUGAS AKHIR

Nama Mahasiswa : Fina Meysita

NIM : 06213070204

Jurusan/Program Studi : Teknik Komputer / DIII Teknik Komputer

Judul Tugas Akhir : Rancang Bangun Timbangan Digital Berbasis Website Pada Toko Bahan Pangan Lelly Frends Menggunakan Sensor Barcode

Telah melaksanakan revisi terhadap Laporan Tugas Akhir yang diujikan pada hari Kamis tanggal 17 bulan Oktober tahun 2024 Pelaksanaan revisi terhadap Laporan Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi :

No.	Komentar	Nama Dosen Penguji	Tanggal /Bulan	Tanda Tangan
1.	Acc	Yulian Mirza, S.T., M.Kom	22/11/24	
2.	Acc	Ir. Alan Novi Tompunu, S.T., M.T., IPM., ASEAN Eng	25/11/24	
3.	Acc	Hartati Deviana, S.T., M.Kom	21/11/24	
4.	Acc	Rian Rahmanda Putra, S.Kom., M.Kom	25/11/24	

Palembang, Oktober 2024

Ketua Penguji

(Yulian Mirza, S.T., M.Kom)