LAMPIRAN

Migrasi:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password_reset_tokens', function (Blueprint $table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
        Schema::dropIfExists('password_reset_tokens');
        Schema::dropIfExists('sessions');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use function Laravel\Prompts\table;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('depots', function (Blueprint $table) {
            $table->id();
            $table->integer('no_struk');
            $table->string('kasir')->nullable();
            $table->string('nama')->nullable();
```

```
            $table->string('no_hp')->nullable();
            $table->string('alamat')->nullable();
            $table->string('jenis_pembayaran');
            $table->date('waktu');
            $table->integer('harga');
            $table->integer('jumlah');
            $table->integer('total');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('depots');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('depotproduks', function (Blueprint
            $table) {
            $table->id();
            $table->string('nama')->nullable();
            $table->integer('harga');
            $table->integer('jumlah')->nullable();
            $table->text('deskripsi')->nullable();
            $table->string('foto')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('depotproduks');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('fotocopies', function (Blueprint
            $table) {
            $table->id();
            $table->integer('no_struk');
            $table->string('kasir')->nullable();
            $table->string('nama')->nullable();
            $table->string('no_hp')->nullable();
            $table->string('alamat')->nullable();
            $table->string('jenis_pembayaran');
            $table->date('waktu');
            $table->integer('harga');
```

```
            $table->integer('jumlah');
            $table->integer('total');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('fotocopies');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('fotocopyproduks', function (Blueprint $table) {
            $table->id();
            $table->string('nama')->nullable();
            $table->string('kategori');
            $table->integer('harga');
            $table->integer('jumlah');
            $table->text('deskripsi');
            $table->string('foto')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('fotocopyproduks');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('laundries', function (Blueprint $table) {
            $table->id();
            $table->integer('no_struk');
            $table->string('kasir')->nullable();
            $table->string('nama')->nullable();
            $table->string('no_hp')->nullable();
            $table->string('alamat')->nullable();
            $table->string('status');
            $table->string('pengerjaan');
            $table->string('jenis_pembayaran');
            $table->date('waktu');
            $table->integer('harga');
            $table->float('jumlah');
            $table->integer('total');
```

```
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('laundries');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('laundryproduks', function
            (Blueprint $table) {
            $table->id();
            $table->string('nama')->nullable();
            $table->integer('harga');
            $table->text('deskripsi')->nullable();
            $table->string('foto')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('laundryproduks');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('pengeluarans', function (Blueprint
            $table) {
            $table->id();
            $table->string('nama_pengeluaran');
            $table->string('nama_permintaan');
            $table->date('waktu');
            $table->integer('harga');
            $table->string('foto')->nullable();

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('pengeluarans');
```

```
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('laporans', function (Blueprint $table) {
            $table->id();
            $table->date('tanggal'); // Untuk harian, bulanan, tahunan bisa diproses dari sini
            $table->enum('kategori', ['harian', 'bulanan', 'tahunan']);
            $table->integer('total_pemasukan'); // dari 3 produk
            $table->integer('total_pengeluaran'); // dari tabel pengeluaran
            $table->integer('total_keuntungan'); // pemasukan - pengeluaran
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('laporans');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('depot_items', function (Blueprint $table) {
            $table->id();
            $table->foreignId('depot_id')->constrained()->onDelete('cascade');
            $table->foreignId('depotproduk_id')->constrained()->onDelete('cascade');
            $table->integer('quantity');
            $table->integer('price');
            $table->integer('subtotal');
            $table->timestamps();
        });

    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('depot_items');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('fotocopy_items', function (Blueprint $table) {
            $table->id();
            $table->foreignId('fotocopy_id')->constrained()->onDelete('cascade');
            $table->foreignId('fotocopyproduk_id')->constrained()->onDelete('cascade');
            $table->integer('quantity');
            $table->integer('price');
            $table->integer('subtotal');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('fotocopy_items');
    }
};
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        $teams = config('permission.teams');
        $tableNames = config('permission.table_names');
        $columnNames = config('permission.column_names');
        $pivotRole = $columnNames['role_pivot_key'] ?? 'role_id';
        $pivotPermission = $columnNames['permission_pivot_key'] ?? 'permission_id';

        throw_if(empty($tableNames), new Exception('Error: config/permission.php not loaded. Run [php artisan config:clear] and try again.'));
        throw_if($teams && empty($columnNames['team_foreign_key'] ?? null), new Exception('Error: team_foreign_key on config/permission.php not loaded. Run [php artisan config:clear] and try again.'));

        Schema::create($tableNames['permissions'], static function (Blueprint $table) {
            // $table->engine('InnoDB');
            $table->bigIncrements('id'); // permission id
            $table->string('name');       // For MyISAM use string('name', 225); // (or 166 for InnoDB with Redundant/Compact row format)
            $table->string('guard_name'); // For MyISAM use string('guard_name', 25);
            $table->timestamps();

            $table->unique(['name', 'guard_name']);
        });

        Schema::create($tableNames['roles'], static function (Blueprint $table) use ($teams, $columnNames) {
            // $table->engine('InnoDB');
            $table->bigIncrements('id'); // role id
```

```
    if ($teams || config('permission.testing')) { // permission.testing is a fix for sqlite testing
        $table->unsignedBigInteger($columnNames['team_foreign_key'])->nullable();
        $table->index($columnNames['team_foreign_key'], 'roles_team_foreign_key_index');
    }
    $table->string('name');       // For MyISAM use string('name', 225); // (or 166 for InnoDB with Redundant/Compact row format)
    $table->string('guard_name'); // For MyISAM use string('guard_name', 25);
    $table->timestamps();
    if ($teams || config('permission.testing')) {
        $table->unique([$columnNames['team_foreign_key'], 'name', 'guard_name']);
    } else {
        $table->unique(['name', 'guard_name']);
    }
});

        Schema::create($tableNames['model_has_permissions'], static function (Blueprint $table) use ($tableNames, $columnNames, $pivotPermission, $teams) {
    $table->unsignedBigInteger($pivotPermission);

    $table->string('model_type');
    $table->unsignedBigInteger($columnNames['model_morph_key']);
    $table->index([$columnNames['model_morph_key'], 'model_type'], 'model_has_permissions_model_id_model_type_index');

    $table->foreign($pivotPermission)
        ->references('id') // permission id
        ->on($tableNames['permissions'])
        ->onDelete('cascade');
    if ($teams) {
        $table->unsignedBigInteger($columnNames['team_foreign_key']);
        $table->index($columnNames['team_foreign_key'], 'model_has_permissions_team_foreign_key_index');

        $table->primary([$columnNames['team_foreign_key'], $pivotPermission, $columnNames['model_morph_key'], 'model_type'],
            'model_has_permissions_permission_model_type_primary');
    } else {
        $table->primary([$pivotPermission, $columnNames['model_morph_key'], 'model_type'],
            'model_has_permissions_permission_model_type_primary');
    }
});

Schema::create($tableNames['model_has_roles'], static function (Blueprint $table) use ($tableNames, $columnNames, $pivotRole, $teams) {
    $table->unsignedBigInteger($pivotRole);

    $table->string('model_type');
    $table->unsignedBigInteger($columnNames['model_morph_key']);
    $table->index([$columnNames['model_morph_key'], 'model_type'], 'model_has_roles_model_id_model_type_index');

    $table->foreign($pivotRole)
        ->references('id') // role id
        ->on($tableNames['roles'])
        ->onDelete('cascade');
    if ($teams) {
```

```
            $table->unsignedBigInteger($columnNames['team_foreign_key']);

            $table->index($columnNames['team_foreign_key'], 'model_has_roles_team_foreign_key_index');

            $table->primary([$columnNames['team_foreign_key'], $pivotRole, $columnNames['model_morph_key'], 'model_type'], 'model_has_roles_role_model_type_primary');
        } else {
            $table->primary([$pivotRole, $columnNames['model_morph_key'], 'model_type'], 'model_has_roles_role_model_type_primary');
        }
    });

        Schema::create($tableNames['role_has_permissions'], static function (Blueprint $table) use ($tableNames, $pivotRole, $pivotPermission) {
        $table->unsignedBigInteger($pivotPermission);
        $table->unsignedBigInteger($pivotRole);

        $table->foreign($pivotPermission)
            ->references('id') // permission id
            ->on($tableNames['permissions'])
            ->onDelete('cascade');

        $table->foreign($pivotRole)
            ->references('id') // role id
            ->on($tableNames['roles'])
            ->onDelete('cascade');

        $table->primary([$pivotPermission, $pivotRole], 'role_has_permissions_permission_id_role_id_primary');
    });

    app('cache')
        ->store(config('permission.cache.store') != 'default' ? config('permission.cache.store') : null)
        ->forget(config('permission.cache.key'));
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    $tableNames = config('permission.table_names');

    if (empty($tableNames)) {
        throw new \Exception('Error: config/permission.php not found and defaults could not be merged. Please publish the package configuration before proceeding, or drop the tables manually.');
    }

        Schema::drop($tableNames['role_has_permissions']);
    Schema::drop($tableNames['model_has_roles']);
        Schema::drop($tableNames['model_has_permissions']);
    Schema::drop($tableNames['roles']);
    Schema::drop($tableNames['permissions']);
}
};
```

Model:

. <?php

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Depot extends Model
{
    protected $fillable = ['no_struk','kasir','nama','no_hp','alamat','jenis_pembayaran','jumlah','waktu','harga','total',];

    public function items()
    {
        return $this->hasMany(DepotItem::class, 'depot_id');
    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class DepotItem extends Model
{
    use HasFactory;

    protected $fillable = [
        'depot_id',
        'depotproduk_id',
        'quantity',
        'price',
        'subtotal'
    ];

    public function depot()
    {
        return $this->belongsTo(Depot::class);
    }

    public function produk()
    {
        return $this->belongsTo(Depotproduk::class, 'depotproduk_id');
    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Depotproduk extends Model
{
    protected $fillable = ['nama', 'jumlah', 'harga', 'deskripsi', 'foto'];
// app/Models/Depotproduk.php
public function items()
{
    return $this->hasMany(DepotItem::class);
}

}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Fotocopy extends Model
{
    protected $fillable = ['no_struk','kasir','nama','no_hp','alamat','jenis_pembayaran','jumlah','waktu','harga','total',];

    public function items()
{
```

```
    return $this->hasMany(FotocopyItem::class);
}

}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class FotocopyItem extends Model
{
        use HasFactory;

    protected $fillable = [
        'fotocopy_id',
        'fotocopyproduk_id',
        'quantity',
        'price',
        'subtotal'
    ];

    public function fotocopy()
    {
        return $this->belongsTo(Fotocopy::class);
    }

    public function produk()
    {
        return      $this->belongsTo(Fotocopyproduk::class,
                'fotocopyproduk_id');
    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Fotocopyproduk extends Model
{
    protected                $fillable                =
                ['nama','kategori','jumlah','harga','deskripsi','f
                oto'];
    public function items()
{
    return $this->hasMany(FotocopyItem::class);
}

}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Laporan extends Model
{
        protected $table = 'laporans';

    protected $fillable = [
        'tanggal',
        'kategori',
        'total_pemasukan',
        'total_pengeluaran',
        'total_keuntungan',
    ];

    protected $casts = [
        'tanggal' => 'date',
    ];
}
<?php

namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Model;


class Laundry extends Model
{
    protected $fillable = ['no_struk','kasir','nama','no_hp','alamat','status','pengerjaan','jenis_pembayaran','waktu','jumlah','harga','total',];
}
<?php


namespace App\Models;


use Illuminate\Database\Eloquent\Model;


class Laundryproduk extends Model
{
    protected $fillable = ['nama','harga','deskripsi','foto'];
}
<?php


namespace App\Models;


use Illuminate\Database\Eloquent\Model;


class Pengeluaran extends Model
{
    protected $fillable = ['nama_pengeluaran','nama_permintaan','waktu','harga','foto'];
}
<?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 HasFactory<\Database\Factories\UserFactory> */
    use HasFactory, Notifiable,HasApiTokens,HasRoles;


    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];


    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];


    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
```

```
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```

CONTROLLER:

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;

class RegisteredUserController extends Controller
{
    /**
     * Handle an incoming registration request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): Response
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:'.User::class],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->string('password')),
        ]);

        event(new Registered($user));

        Auth::login($user);

        return response()->noContent();
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Depot;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class DepotController extends Controller
{
public function index()
{
    $depot = Cache::remember('depot_all_data', 60, function () {
        return Depot::with(['items.produk'])->get();
    });

    return response()->json($depot, 200);
```

```
}

public function store(Request $request)
{
    $validated = $request->validate([
        'nama' => 'nullable|string|max:255',
        'no_hp' => 'nullable|string|max:20',
        'alamat' => 'nullable|string|max:255',
        'jenis_pembayaran' => 'required',
        'jumlah' => 'required|integer|min:1',
        'harga' => 'required|numeric|min:0',
        'total' => 'required|numeric|min:0',
        'items' => 'required|array|min:1',
        'items.*.depotproduk_id' =>
            'required|integer|exists:depotproduks,id',
        'items.*.quantity' => 'required|integer|min:1',
        'items.*.price' => 'required|numeric|min:0',
        'items.*.subtotal' => 'required|numeric|min:0',
    ]);

    $depot = DB::transaction(function () use ($validated)
        {
        $today = Carbon::today();
        $lastNumber = Depot::whereDate('created_at',
            $today)->max('no_struk') ?? 0;
        $nextNumber = $lastNumber + 1;

        // Buat header
        $depot = Depot::create([
            'no_struk' => $nextNumber,
            'kasir' => \Auth::user()->name ?? \Auth::user()-
                >username ?? \Auth::id(),
            'nama' => $validated['nama'],
            'no_hp' => $validated['no_hp'],
            'alamat' => $validated['alamat'] ?? null,
            'jenis_pembayaran' =>
                $validated['jenis_pembayaran'],
            'waktu' => Carbon::now(),
            'harga' => $validated['harga'],
            'jumlah' => $validated['jumlah'],
            'total' => $validated['total'],
        ]);

        // Buat detail items
        foreach ($validated['items'] as $item) {
            $depot->items()->create([
                'depotproduk_id' => $item['depotproduk_id'],
                'quantity' => $item['quantity'],
                'price' => $item['price'],
                'subtotal' => $item['subtotal'],
            ]);
        }

        return $depot->load('items'); // biar respons sudah
            ada itemnya juga
    });

    Cache::forget('depot_all_data');

    return response()->json($depot, 201);
}

public function show(Depot $depot)
{
    $depot->load('items.produk'); // lazy load relasi
    return response()->json($depot);
}

    public function destroy(Depot $depot)
    {
        // Hapus data
        $depot->delete();

        // Kosongkan cache
        Cache::forget('depot_all_data');

        // Respons sukses
        return response()->json([
```

```
        'message' => 'Data berhasil dihapus'
    ], 200);
  }

}
<?php

namespace App\Http\Controllers;

use App\Models\Depotproduk;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;

class DepotprodukController extends Controller
{
  public function index()
  {
    $depotproduk = Cache::remember('depotproduk_all_data', 60, function () {
      return Depotproduk::all();
    });

    return response()->json($depotproduk, 200);
  }

  public function store(Request $request)
  {

    $validated = $request->validate([
      'nama' => 'required|string|max:255',

      'harga' => 'nullable|numeric|min:0',

      'foto' => 'nullable|image|mimes:png,jpeg,jpg|max:10048',
    ]);

if ($request->hasFile('foto')) {
  try {
    $path = $request->file('foto')->store('Depotproduk/foto', 'public');
    $validated['foto'] = $path;
    \Log::info("Foto berhasil diupload: " . $path); // Log path foto
  } catch (\Exception $e) {
    \Log::error("Gagal upload foto: " . $e->getMessage()); // Log error
    return response()->json(['error' => 'Gagal upload foto'], 400);
  }
}

    $depotproduk = Depotproduk::create($validated);

    Cache::forget('depotproduk_all_data');

    return response()->json($depotproduk, 201);
  }

  public function show(Depotproduk $depotproduk)
  {
    return response()->json($depotproduk, 200);
  }

  public function update(Request $request, Depotproduk $depotproduk)
  {
    $validated = $request->validate([
      'nama' => 'required|string|max:255',
      'jumlah' => 'nullable|numeric|min:1',
      'harga' => 'required|numeric|min:0',
      'deskripsi' => 'nullable|string',
      'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
    ]);
```

```
        if ($request->hasFile('foto')) {
            if ($depotproduk->foto && Storage::disk('public')->exists($depotproduk->foto)) {
                Storage::disk('public')->delete($depotproduk->foto);
            }
            $validated['foto'] = $request->file('foto')->store('Depotproduk/foto', 'public');
        }

        $depotproduk->update($validated);

        Cache::forget('depotproduk_all_data');

        return response()->json($depotproduk, 200);
    }

    public function destroy(Depotproduk $depotproduk)
    {
        if ($depotproduk->items()->exists()) {
            return response()->json([
                'message' => 'Produk tidak dapat dihapus karena masih digunakan dalam transaksi.'
            ], 400);
        }

        if ($depotproduk->foto && Storage::disk('public')->exists($depotproduk->foto)) {
            Storage::disk('public')->delete($depotproduk->foto);
        }

        $depotproduk->delete();

        Cache::forget('depotproduk_all_data');

        return response()->json(['message' => 'Data berhasil dihapus'], 200);
    }
<?php

namespace App\Http\Controllers;

use App\Models\Fotocopy;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class FotocopyController extends Controller
{
    public function index()
    {
        $fotocopy = Cache::remember('fotocopy_all_data', 60, function () {
            return Fotocopy::with(['items.produk'])->get();
        });

        return response()->json($fotocopy, 200);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'nama' => 'nullable|string|max:255',
            'no_hp' => 'nullable|string|max:20',
            'alamat' => 'nullable|string|max:255',
            'jenis_pembayaran' => 'required',
            'jumlah' => 'required|integer|min:1',
            'harga' => 'required|numeric|min:0',
            'total' => 'required|numeric|min:0',
            'items' => 'required|array|min:1',
```

```
    'items.*.fotocopyproduk_id' => 'required|integer|exists:fotocopyproduks,id',
    'items.*.quantity' => 'required|integer|min:1',
    'items.*.price' => 'required|numeric|min:0',
    'items.*.subtotal' => 'required|numeric|min:0',
]);

$fotocopy = DB::transaction(function () use ($validated) {
    $today = Carbon::today();
    $lastNumber = Fotocopy::whereDate('created_at', $today)->max('no_struk') ?? 0;
    $nextNumber = $lastNumber + 1;

    // Buat header
    $fotocopy = Fotocopy::create([
        'no_struk' => $nextNumber,
        'kasir' => \Auth::user()->name ?? \Auth::user()->username ?? \Auth::id(),
        'nama' => $validated['nama'],
        'no_hp' => $validated['no_hp'],
        'alamat' => $validated['alamat'] ?? null,
        'jenis_pembayaran' => $validated['jenis_pembayaran'],
        'waktu' => Carbon::now(),
        'harga' => $validated['harga'],
        'jumlah' => $validated['jumlah'],
        'total' => $validated['total'],
    ]);

    // Buat detail items
    foreach ($validated['items'] as $item) {
        $fotocopy->items()->create([
            'fotocopyproduk_id' => $item['fotocopyproduk_id'],
            'quantity' => $item['quantity'],
            'price' => $item['price'],
            'subtotal' => $item['subtotal'],
        ]);
    }

    return $fotocopy->load('items'); // biar respons sudah ada itemnya juga
});

Cache::forget('fotocopy_all_data');

return response()->json($fotocopy, 201);
}
    public function show(Fotocopy $fotocopy)
    {
        $fotocopy->load('items.produk'); // lazy load relasi
    return response()->json($fotocopy);
    }


    public function destroy(Fotocopy $fotocopy)
    {
        // Hapus data
        $fotocopy->delete();

        // Kosongkan cache
        Cache::forget('fotocopy_all_data');

        // Respons sukses
        return response()->json([
            'message' => 'Data berhasil dihapus'
        ], 200);
    }

}
<?php

namespace App\Http\Controllers;

use App\Models\Fotocopyproduk;
use Illuminate\Http\Request;
```

```
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;

class FotocopyprodukController extends Controller
{
    public function index()
    {
        $fotocopyproduk = Cache::remember('fotocopyproduk_all_data', 60, function () {
            return Fotocopyproduk::all();
        });

        return response()->json($fotocopyproduk, 200);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'nama' => 'required|string|max:255',
            'kategori' => 'required',
            'jumlah' => 'required|numeric|min:1',
            'harga' => 'required|numeric|min:0',
            'deskripsi' => 'required|string',
            'foto' => 'nullable|image|mimes:png,jpeg,jpg|max:10048',
        ]);

        if ($request->hasFile('foto')) {
            $validated['foto'] = $request->file('foto')->store('fotocopyproduk/foto', 'public');
        }

        $fotocopyproduk = Fotocopyproduk::create($validated);

        Cache::forget('fotocopyproduk_all_data');

        return response()->json($fotocopyproduk, 201);
    }

    public function show(Fotocopyproduk $fotocopyproduk)
    {
        return response()->json($fotocopyproduk, 200);
    }

    public function update(Request $request, Fotocopyproduk $fotocopyproduk)
    {
        $validated = $request->validate([
            'nama' => 'required|string|max:255',
            'jumlah' => 'required|numeric|min:1',
            'harga' => 'required|numeric|min:0',
            'deskripsi' => 'required|string',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
        ]);

        if ($request->hasFile('foto')) {
            if ($fotocopyproduk->foto && Storage::disk('public')->exists($fotocopyproduk->foto)) {
                Storage::disk('public')->delete($fotocopyproduk->foto);
            }
            $validated['foto'] = $request->file('foto')->store('fotocopyproduk/foto', 'public');
        }

        $fotocopyproduk->update($validated);

        Cache::forget('fotocopyproduk_all_data');

        return response()->json($fotocopyproduk, 200);
    }

    public function destroy(Fotocopyproduk $fotocopyproduk)
    {
```

```
    if ($fotocopyproduk->items()->exists()) {
        return response()->json([
            'message' => 'Produk tidak dapat dihapus karena masih digunakan dalam transaksi.'
        ], 400);
    }
        if ($fotocopyproduk->foto && Storage::disk('public')->exists($fotocopyproduk->foto)) {
            Storage::disk('public')->delete($fotocopyproduk->foto);
        }

        $fotocopyproduk->delete();

        Cache::forget('fotocopyproduk_all_data');

        return response()->json(['message' => 'Data deleted successfully'], 200);
    }
}<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;
use App\Models\FotocopyItem;

class LaporanController extends Controller
{
    // Laporan Harian
    public function laporanHarian(Request $request)
    {
        $tanggal = $request->input('tanggal', Carbon::today()->toDateString());

        $pemasukan = $this->getPemasukanDetail($tanggal, $tanggal);
        $totalPengeluaran = $this->getTotalPengeluaran($tanggal, $tanggal);

        return response()->json([
            'tanggal' => $tanggal,
            'kategori' => 'harian',
            'pemasukan' => $pemasukan,
            'total_pengeluaran' => $totalPengeluaran,
            'total_keuntungan' => $pemasukan['total'] - $totalPengeluaran,
        ]);
    }

    // Laporan Bulanan
    public function laporanBulanan(Request $request)
    {
        $start = Carbon::createFromDate(
            $request->input('start_year'),
            $request->input('start_month'),
            1
        )->startOfMonth();

        $end = Carbon::createFromDate(
            $request->input('end_year'),
            $request->input('end_month'),
            1
        )->endOfMonth();

        $pemasukan = $this->getPemasukanDetail($start, $end);
        $totalPengeluaran = $this->getTotalPengeluaran($start, $end);

        return response()->json([
            'periode' => $start->toDateString() . ' - ' . $end->toDateString(),
            'kategori' => 'bulanan',
            'pemasukan' => $pemasukan,
            'total_pengeluaran' => $totalPengeluaran,
            'total_keuntungan' => $pemasukan['total'] - $totalPengeluaran,
        ]);
```

```
}

// Laporan Tahunan
public function laporanTahunan(Request $request)
{
    $start = Carbon::createFromDate($request->input('start_year'), 1, 1)->startOfYear();
    $end = Carbon::createFromDate($request->input('end_year'), 12, 31)->endOfYear();

    $pemasukan = $this->getPemasukanDetail($start, $end);
    $totalPengeluaran = $this->getTotalPengeluaran($start, $end);

    return response()->json([
        'periode' => $start->toDateString() . ' - ' . $end->toDateString(),
        'kategori' => 'tahunan',
        'pemasukan' => $pemasukan,
        'total_pengeluaran' => $totalPengeluaran,
        'total_keuntungan' => $pemasukan['total'] - $totalPengeluaran,
    ]);
}

// Fungsi pemasukan detail per tabel
private function getPemasukanDetail($startDate, $endDate)
{
    $totalLaundry = DB::table('laundries')
        ->whereBetween('waktu', [$startDate, $endDate])
        ->sum('total');

    $totalFotocopy = DB::table('fotocopies')
        ->whereBetween('waktu', [$startDate, $endDate])
        ->sum('total');

    $totalDepot = DB::table('depots')
        ->whereBetween('waktu', [$startDate, $endDate])
        ->sum('total');

    return [
        'laundry' => $totalLaundry,
        'fotocopy' => $totalFotocopy,
        'depot' => $totalDepot,
        'total' => $totalLaundry + $totalFotocopy + $totalDepot,
    ];
}

// Fungsi total pengeluaran
private function getTotalPengeluaran($startDate, $endDate)
{
    return DB::table('pengeluarans')
        ->whereBetween('waktu', [$startDate, $endDate])
        ->sum('harga');
}

public function getProdukTerlaris()
{
    // Mengambil semua fotocopy item dan mengelompokkan berdasarkan fotocopyproduk_id untuk menghitung quantity-nya
    $produkTerlaris = FotocopyItem::with('produk')
        ->selectRaw('fotocopyproduk_id, sum(quantity) as total_quantity')
        ->groupBy('fotocopyproduk_id')
        ->orderByDesc('total_quantity')
        ->limit(5)  // Jika ingin mengambil 5 produk terlaris
        ->get();

    return response()->json($produkTerlaris, 200);
}

}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Laundry;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class LaundryController extends Controller
{
public function index()
{
   $laundry = Cache::remember('laundry_all_data', 60, function () {
      return Laundry::all(); // laundry memang tidak punya relasi items
   });

   return response()->json($laundry, 200);
}


   public function store(Request $request)
   {
      // 1. Validasi—sesuaikan rule sesuai kebutuhan bisnis Anda
      $validated = $request->validate([
         'nama' => 'required|string|max:255',
         'no_hp' => 'required|string|max:20',
         'alamat' => 'nullable|string|max:255',
         'jenis_pembayaran' => 'required', // contoh opsi
         'jumlah' => 'required',
         'status' => 'required',
         'pengerjaan' => 'required',
         'harga' => 'required|numeric|min:0',
         'total' => 'required|numeric|min:0',
      ]);

      // 2. Gunakan transaksi agar penomoran no_struk tetap konsisten
      $laundry = DB::transaction(function () use ($validated) {

         // Ambil nomor struk terakhir untuk hari ini; jika tidak ada, mulai dari 0
         $today = Carbon::today();                  // pakai timezone default aplikasi (config/app.php)
         $lastNumber = Laundry::whereDate('created_at', $today)
            ->max('no_struk') ?? 0;

         $nextNumber = $lastNumber + 1;             // reset otomatis ke 1 tiap hari baru

         // Buat entri
         return Laundry::create([
            'no_struk' => $nextNumber,
            'kasir' => Auth::user()->name ?? Auth::user()->username ?? Auth::id(),
            'nama' => $validated['nama'],
            'status' => $validated['status'],
            'no_hp' => $validated['no_hp'],
            'alamat' => $validated['alamat'] ?? null,
            'jenis_pembayaran' => $validated['jenis_pembayaran'],
            'pengerjaan' => $validated['pengerjaan'],
            'waktu' => Carbon::now(),          // waktu saat input
            'harga' => $validated['harga'],
            'jumlah' => $validated['jumlah'],
            'total' => $validated['total'],
         ]);
      });

      // 3. Kosongkan cache karena ada data baru
      Cache::forget('laundry_all_data');

      // 4. Beri respons
```

```
        return response()->json($laundry, 201);
    }
    public function show(Laundry $laundry)
    {
        return response()->json($laundry, 200);
    }
    public function update(Request $request, Laundry $laundry)
    {
        $validated = $request->validate([
            'status' => 'required|string|max:255',

        ]);

        $laundry->update($validated);

        Cache::forget('laundry_all_data');

        return response()->json($laundry, 200);
    }
    public function destroy(Laundry $laundry)
    {
        // Hapus data
        $laundry->delete();

        // Kosongkan cache
        Cache::forget('laundry_all_data');

        // Respons sukses
        return response()->json([
            'message' => 'Data berhasil dihapus'
        ], 200);
    }

}
<?php

namespace App\Http\Controllers;

use App\Models\Laundryproduk;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;

class LaundryprodukController extends Controller
{
    public function index()
    {
        $laundryproduk = Cache::remember('laundryproduk_all_data', 60, function () {
            return Laundryproduk::all();
        });

        return response()->json($laundryproduk, 200);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'nama' => 'required|string|max:255',
            'harga' => 'required|numeric|min:0',
            'deskripsi' => 'required',
            'foto' => 'nullable|image|mimes:png,jpeg,jpg|max:10048',
        ]);

        if ($request->hasFile('foto')) {
            $validated['foto'] = $request->file('foto')->store('Laundryproduk/foto', 'public');
        }

        $laundryproduk = Laundryproduk::create($validated);

        Cache::forget('laundryproduk_all_data');
```

```
    return response()->json($laundryproduk, 201);
}

public function show(Laundryproduk $laundryproduk)
{
    return response()->json($laundryproduk, 200);
}

public function update(Request $request, Laundryproduk $laundryproduk)
{
    $validated = $request->validate([
        'nama' => 'required|string|max:255',
        'harga' => 'required|numeric|min:0',
        'deskripsi' => 'nullable|string',
        'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
    ]);

    if ($request->hasFile('foto')) {
        if ($laundryproduk->foto && Storage::disk('public')->exists($laundryproduk->foto)) {
            Storage::disk('public')->delete($laundryproduk->foto);
        }
        $validated['foto'] = $request->file('foto')->store('Laundryproduk/foto', 'public');
    }

    $laundryproduk->update($validated);

    Cache::forget('laundryproduk_all_data');

    return response()->json($laundryproduk, 200);
}

public function destroy(Laundryproduk $laundryproduk)
{
    if ($laundryproduk->foto && Storage::disk('public')->exists($laundryproduk->foto)) {
        Storage::disk('public')->delete($laundryproduk->foto);
    }

    $laundryproduk->delete();

    Cache::forget('laundryproduk_all_data');

    return response()->json(['message' => 'Data deleted successfully'], 200);
  }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Pengeluaran;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;
use Carbon\Carbon;

class PengeluaranController extends Controller
{
  public function index()
  {
    $pengeluaran = Cache::remember('pengeluaran_all_data', 60, function () {
      return Pengeluaran::all();
    });

    return response()->json($pengeluaran, 200);
  }

public function store(Request $request)
```

```
{
    $validated = $request->validate([
        'nama_pengeluaran' => 'required|string|max:255',
        'nama_permintaan' => 'required|string|max:255',
        // waktu tidak perlu divalidasi dari request
        'harga' => 'required|numeric|min:0',
        'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
    ]);

    if ($request->hasFile('foto')) {
        $validated['foto'] = $request->file('foto')->store('Pengeluaran/foto', 'public');
    }

    // waktu diisi otomatis sekarang
    $validated['waktu'] = Carbon::now();

    $pengeluaran = Pengeluaran::create($validated);

    Cache::forget('pengeluaran_all_data');

    return response()->json($pengeluaran, 201);
}

    public function show(Pengeluaran $pengeluaran)
    {
        return response()->json($pengeluaran, 200);
    }

public function update(Request $request, Pengeluaran $pengeluaran)
{
    $validated = $request->validate([
        'nama_pengeluaran' => 'required|string|max:255',
        'nama_permintaan' => 'required|string|max:255',
        // waktu tidak diambil dari request
        'harga' => 'required|numeric|min:0',
        'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
    ]);

    if ($request->hasFile('foto')) {
        if ($pengeluaran->foto && Storage::disk('public')->exists($pengeluaran->foto)) {
            Storage::disk('public')->delete($pengeluaran->foto);
        }
        $validated['foto'] = $request->file('foto')->store('Pengeluaran/foto', 'public');
    }

    $validated['waktu'] = Carbon::now(); // di-update otomatis

    $pengeluaran->update($validated);

    Cache::forget('pengeluaran_all_data');

    return response()->json($pengeluaran, 200);
}

    public function destroy(Pengeluaran $pengeluaran)
    {
        if ($pengeluaran->foto && Storage::disk('public')->exists($pengeluaran->foto)) {
            Storage::disk('public')->delete($pengeluaran->foto);
        }

        $pengeluaran->delete();

        Cache::forget('pengeluaran_all_data');

        return response()->json(['message' => 'Data deleted successfully'], 200);
    }
```

```
}
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;
use Illuminate\Support\Facades\Auth;
class ProfileController extends Controller
{
   /**
    * Update the authenticated user's profile.
    */
   public function update(Request $request)
   {
      $user = $request->user();

      $validated = $request->validate([
         'name' => ['required', 'string', 'max:255'],
         'email' => ['required', 'email', 'max:255',
            Rule::unique('users')->ignore($user->id)],
         'password' => ['nullable', 'string', 'min:8',
            'confirmed'], // use password_confirmation
      ]);

      $user->name = $validated['name'];
      $user->email = $validated['email'];
      if (!empty($validated['password'])) {
         $user->password =
            Hash::make($validated['password']);
      }

      $user->save();

      return response()->json([
         'message' => 'Profile updated successfully.',
         'user' => $user,
      ]);
   }

}
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Spatie\Permission\Models\Role;
use Illuminate\Support\Facades\DB;

class UserController extends Controller
{
   // Menampilkan semua user
public function index()
{
   $users = User::with('roles')
      ->whereDoesntHave('roles', function ($query) {
         $query->where('name', 'admin');
      })
      ->get();

   return response()->json($users);
}

   // Menyimpan user baru
   public function store(Request $request)
   {
      $validator = Validator::make($request->all(), [
         'name'     => 'required|string|max:255',
         'email'    => 'required|email|unique:users,email',
         'password' => 'required|string|min:6',
```

```
        'role'     => 'required|exists:roles,name',
    ]);

    if ($validator->fails()) {
        return response()->json($validator->errors(), 422);
    }

    $user = User::create([
        'name'     => $request->name,
        'email'    => $request->email,
        'password' => Hash::make($request->password),
    ]);

    $user->assignRole($request->role);

    return response()->json([
        'message' => 'User berhasil dibuat',
        'data'    => $user->load('roles')
    ], 201);
}

// Menampilkan detail user (pakai model binding)
public function show(User $user)
{
    return response()->json($user->load('roles'));
}

// Mengupdate data user
public function update(Request $request, User $user)
{
    $validator = Validator::make($request->all(), [
        'name'     => 'sometimes|required|string|max:255',
        'email'    => 'sometimes|required|email|unique:users,email,' . $user->id,
        'password' => 'sometimes|required|string|min:6',
        'role'     => 'sometimes|required|exists:roles,name',
    ]);

    if ($validator->fails()) {
        return response()->json($validator->errors(), 422);
    }

    if ($request->filled('name')) $user->name = $request->name;
    if ($request->filled('email')) $user->email = $request->email;
    if ($request->filled('password')) $user->password = Hash::make($request->password);

    $user->save();

    if ($request->filled('role')) {
        $user->syncRoles([$request->role]);
    }

    return response()->json([
        'message' => 'User berhasil diupdate',
        'data'    => $user->load('roles')
    ]);
}

// Menghapus user
public function destroy(User $user)
{
    $user->delete();
    return response()->json(['message' => 'User berhasil dihapus']);
}
}
```

ROUTE:

```
<?php


use Illuminate\Http\Request;

use
        App\Http\Controllers\Auth\AuthenticatedSessionController;

use
        App\Http\Controllers\Auth\EmailVerificationNotificationController;

use
        App\Http\Controllers\Auth\NewPasswordController;

use
        App\Http\Controllers\Auth\PasswordResetLinkController;

use
        App\Http\Controllers\Auth\RegisteredUserController;

use App\Http\Controllers\Auth\VerifyEmailController;

use App\Http\Controllers\ProfileController;

use App\Http\Controllers\LaporanController;

use App\Http\Controllers\LaundryprodukController;

use App\Http\Controllers\DepotprodukController;

use App\Http\Controllers\FotocopyprodukController;

use App\Http\Controllers\DepotController;

use App\Http\Controllers\FotocopyController;

use App\Http\Controllers\LaundryController;

use App\Http\Controllers\PengeluaranController;

use App\Http\Controllers\UserController;

use Illuminate\Support\Facades\Route;


Route::middleware(['auth:sanctum'])->get('/user',
        function (Request $request) {

    $user = $request->user();

    $roles = $user->getRoleNames(); // ambil roles dari
        Spatie


    return response()->json([

        'id' => $user->id,

        'name' => $user->name,

        'email' => $user->email,

        'roles' => $roles,  // <<< ini WAJIB dikirim agar
            frontend bisa baca

    ]);

});


Route::middleware('auth:sanctum')->put('/profile',
        [ProfileController::class, 'update']);

Route::post('/register', [RegisteredUserController::class,
        'store'])

    ->middleware('guest')

    ->name('register');


Route::post('/login',
        [AuthenticatedSessionController::class,
        'store'])

    ->middleware('guest')

    ->name('login');


Route::post('/forgot-password',
        [PasswordResetLinkController::class,
        'store'])

    ->middleware('guest')

    ->name('password.email');


Route::post('/reset-password',
        [NewPasswordController::class, 'store'])

    ->middleware('guest')

    ->name('password.store');


Route::get('/verify-email/{id}/{hash}',
        VerifyEmailController::class)

    ->middleware(['auth', 'signed', 'throttle:6,1'])

    ->name('verification.verify');


Route::post('/email/verification-notification',
        [EmailVerificationNotificationController::cl
        ass, 'store'])

    ->middleware(['auth', 'throttle:6,1'])

    ->name('verification.send');


Route::post('/logout',
        [AuthenticatedSessionController::class,
        'destroy'])
```

```
    ->middleware('auth')

    ->name('logout');


//laporan

Route::get('/laporan/harian',    [LaporanController::class,
            'laporanHarian']);

Route::get('/laporan/bulanan', [LaporanController::class,
            'laporanBulanan']);

Route::get('/laporan/tahunan', [LaporanController::class,
            'laporanTahunan']);

Route::get('/fotocopy/terlaris',
            [LaporanController::class,
            'getProdukTerlaris']);



//depotproduk

Route::get('/depotproduk',
            [DepotprodukController::class, 'index']);

Route::post('/depotproduk',
            [DepotprodukController::class, 'store']);

Route::get('/depotproduk/{depotproduk}',
            [DepotprodukController::class, 'show']);

Route::post('/depotproduk/{depotproduk}',
            [DepotprodukController::class, 'update']);

Route::delete('/depotproduk/{depotproduk}',
            [DepotprodukController::class, 'destroy']);



//Fotocopyproduk

Route::get('/fotocopyproduk',
            [FotocopyprodukController::class, 'index']);

Route::post('/fotocopyproduk',
            [FotocopyprodukController::class, 'store']);

Route::get('/fotocopyproduk/{fotocopyproduk}',
            [FotocopyprodukController::class, 'show']);

Route::post('/fotocopyproduk/{fotocopyproduk}',
            [FotocopyprodukController::class, 'update']);

Route::delete('/fotocopyproduk/{fotocopyproduk}',
            [FotocopyprodukController::class,
            'destroy']);



//laundryproduk

Route::get('/laundryproduk',
            [LaundryprodukController::class, 'index']);

Route::post('/laundryproduk',
            [LaundryprodukController::class, 'store']);

Route::get('/laundryproduk/{laundryproduk}',
            [LaundryprodukController::class, 'show']);

Route::post('/laundryproduk/{laundryproduk}',
            [LaundryprodukController::class, 'update']);

Route::delete('/laundryproduk/{laundryproduk}',
            [LaundryprodukController::class, 'destroy']);



//depot transaksi


Route::get('/depot',    [DepotController::class,    'index']);
            // Get semua data depot

Route::get('/depot/{depot}',       [DepotController::class,
            'show']);

Route::post('/depot',    [DepotController::class,    'store']);
            // Simpan data baru

Route::delete('/depot/{depot}',    [DepotController::class,
            'destroy']); // Hapus data berdasarkan ID
            (pakai model binding)



//Fotocopy transaksi


Route::get('/fotocopy',          [FotocopyController::class,
            'index']);          // Get semua data fotocopy

Route::get('/fotocopy/{fotocopy}',
            [FotocopyController::class, 'show']);

Route::post('/fotocopy',         [FotocopyController::class,
            'store']);          // Simpan data baru

Route::delete('/fotocopy/{fotocopy}',
            [FotocopyController::class,    'destroy']);    //
            Hapus data berdasarkan ID (pakai model
            binding)



//laundry transaksi


Route::get('/laundry',           [LaundryController::class,
            'index']);          // Get semua data Laundry

Route::get('/laundry/{laundry}',
            [LaundryController::class, 'show']);

Route::post('/laundry',          [LaundryController::class,
            'store']);          // Simpan data baru

Route::post('/laundry/{laundry}',
            [LaundryController::class, 'update']);

Route::delete('/laundry/{laundry}',
            [LaundryController::class,     'destroy']);    //
```

```
            Hapus data berdasarkan ID (pakai model
            binding)


//pengeluaran

Route::get('/pengeluaran',
            [PengeluaranController::class, 'index']);

Route::post('/pengeluaran',
            [PengeluaranController::class, 'store']);

Route::get('/pengeluaran/{pengeluaran}',
            [PengeluaranController::class, 'show']);

Route::put('/pengeluaran/{pengeluaran}',
            [PengeluaranController::class, 'update']);

Route::delete('/pengeluaran/{pengeluaran}',
            [PengeluaranController::class, 'destroy']);


//manajemen user


Route::prefix('users')->group(function () {

   Route::get('/', [UserController::class, 'index']);       //
            GET /api/users

   Route::post('/', [UserController::class, 'store']);      //
            POST /api/users

   Route::get('{user}', [UserController::class, 'show']);
            // GET /api/users/{user}

   Route::put('{user}', [UserController::class, 'update']);
            // PUT /api/users/{user}

   Route::delete('{user}', [UserController::class,
            'destroy']); // DELETE /api/users/{user}

});
```

APP:

```
import axios from 'axios';

import AsyncStorage from '@react-native-async-
            storage/async-storage';


const API_URL = 'https://tokoulyaa.com/api';

// const API_URL = 'http://tokoulyaa.com/api'; // Jika
            ingin tes di lokal


// Konfigurasi instance axios
const api = axios.create({

   baseURL: API_URL,

   headers: {

      'Content-Type': 'application/json',

      'Accept': 'application/json'

   }

});



// Interceptor untuk menambahkan token otomatis di
            setiap request

api.interceptors.response.use(response => {

   if (response.data.token) {

      AsyncStorage.setItem('userToken',
            response.data.token);

      api.defaults.headers.common['Authorization']     =
            `Bearer ${response.data.token}`;

   }

   return response;

}, error => Promise.reject(error));



// 🔷 Registrasi

export const register = async (data) => {

   try {

      const response = await api.post('/register', data);

      return response.data;

   } catch (error) {

      throw error.response?.data || { message: 'Registrasi
            gagal!' };

   }

};


// 🔷 Login

export const login = async (data) => {

   try {

      console.log('🔍 Memulai proses login...', data);


      const response = await api.post('/login', data);
```

```
      const token = response.data.token;

      if (!token) throw { message: 'Token tidak diterima
          dari server' };

      // Simpan token ke AsyncStorage
      await AsyncStorage.setItem('userToken', token);
      console.log('🔐 Token tersimpan:', token);

      return response.data;
    } catch (error) {
      console.error('❌ Error saat login:',
          error.response?.data || error);
      throw error.response?.data || { message: 'Login
          gagal!' };
    }
};

// 🔷 Logout
export const logout = async () => {
  try {
    const token = await
        AsyncStorage.getItem('userToken');
    if (!token) {
      console.log('Tidak ada token, langsung logout
          lokal.');
      return { message: 'Logout lokal' };
    }

    try {
      // Coba logout ke server
      await api.post('/logout', {}, {
        headers: { Authorization: `Bearer ${token}` }
      });
      console.log('Logout server berhasil');
    } catch (err) {
      // Kalau token expired/invalid tetap hapus token
      if (err?.response?.status === 401) {
        console.warn('Token invalid saat logout, hapus
            token lokal.');
      } else {
        console.error('Error logout server:', err);
      }
    }
  } finally {
    // Pastikan token lokal dihapus
    await AsyncStorage.removeItem('userToken');
  }

  return { message: 'Logout berhasil (lokal)' };
};

// 🔷 Mendapatkan data user
export const getUser = async (token = null) => {
  try {
    const headers = token ? { Authorization: `Bearer
        ${token}` } : {};
    const response = await api.get('/user', { headers });
    return response.data;
  } catch (error) {
    throw error.response?.data || { message: 'Gagal
        mendapatkan data user!' };
  }
};

// 🔷 Update profil user (name, email, password)
export const updateProfile = async (data) => {
  try {
    const token = await
        AsyncStorage.getItem('userToken');
    if (!token) throw { message: 'Token tidak tersedia'
        };

    const response = await api.put('/profile', data, {
      headers: {
        Authorization: `Bearer ${token}`
      }
```

```
    });

    return response.data;
  } catch (error) {
    console.error('❌ Error saat update profile:',
        error.response?.data || error);
    throw error.response?.data || { message: 'Gagal
        memperbarui profil!' };
  }
};

// ====================
// LAPORAN
// ====================

// Default (tanpa filter)
export const fetchDefaultLaporanHarian = () =>
    api.get('/laporan/harian');
export const fetchDefaultLaporanBulanan = () =>
    api.get('/laporan/bulanan');
export const fetchDefaultLaporanTahunan = () =>
    api.get('/laporan/tahunan');

// Dengan filter tanggal/parameter
export const fetchLaporanHarianByTanggal = (tanggal)
    => {
 return api.get(`/laporan/harian?tanggal=${tanggal}`);
};

export const fetchLaporanBulananByRange =
    (startMonth, startYear, endMonth, endYear)
    => {
 return api.get(
    `/laporan/bulanan?start_month=${startMont
    h}&start_year=${startYear}&end_month=$
    {endMonth}&end_year=${endYear}`
 );
};

export const fetchLaporanTahunanByRange = (startYear,
    endYear) => {
 return
    api.get(`/laporan/tahunan?start_year=${start
    Year}&end_year=${endYear}`);
};

export const getProdukTerlaris = async () => {
  try {
    const response = await api.get('/fotocopy/terlaris');
    return response.data;
  } catch (error) {
    console.error("Error getting best-selling products",
        error);
    throw error;
  }
};

// ====================
// DEPOT PRODUK
// ====================

export const getDepotproduk = () =>
    api.get('/depotproduk');
export const getDepotprodukById = (id) =>
    api.get(`/depotproduk/${id}`);
export const deleteDepotproduk = (id) =>
    api.delete(`/depotproduk/${id}`);

export const createDepotproduk = async (formData) =>
    {
 try {
  const response = await api.post('/depotproduk',
      formData, {
   headers: {
    'Content-Type': 'multipart/form-data',
   },
  });
  console.log('Status:', response.status, 'Response:',
      response.data);
  return response.data;
 } catch (error) {
```

```
  console.error('Gagal mengirim data:', error.response?.data || error.message);
  throw error;
 }
};

export const updateDepotproduk = async (id, data) => {
 return api.post(`/depotproduk/${id}?_method=POST`, data, {
  headers: {
   'Content-Type': 'multipart/form-data',
  },
 });
};

// ====================
// FOTOCOPY PRODUK
// ====================

export const getFotocopyproduk = () => api.get('/fotocopyproduk');
export const getFotocopyprodukById = (id) => api.get(`/fotocopyproduk/${id}`);
export const deleteFotocopyproduk = (id) => api.delete(`/fotocopyproduk/${id}`);

export const createFotocopyproduk = async (formData) => {
 try {
  const response = await api.post('/fotocopyproduk', formData, {
   headers: {
    'Content-Type': 'multipart/form-data',
   },
  });
  console.log('Status:', response.status, 'Response:', response.data);
  return response.data;
 } catch (error) {
  console.error('Gagal mengirim data:', error.response?.data || error.message);
  throw error;
 }
};

export const updateFotocopyproduk = async (id, data) => {
 return api.post(`/fotocopyproduk/${id}?_method=POST`, data, {
  headers: {
   'Content-Type': 'multipart/form-data',
  },
 });
};

// ====================
// LAUNDRY PRODUK
// ====================

export const getLaundryproduk = () => api.get('/laundryproduk');
export const getLaundryprodukById = (id) => api.get(`/laundryproduk/${id}`);
export const deleteLaundryproduk = (id) => api.delete(`/laundryproduk/${id}`);

export const createLaundryproduk = async (formData) => {
 try {
  const response = await api.post('/laundryproduk', formData, {
   headers: {
    'Content-Type': 'multipart/form-data',
   },
  });
  console.log('Status:', response.status, 'Response:', response.data);
  return response.data;
 } catch (error) {
  console.error('Gagal mengirim data:', error.response?.data || error.message);
  throw error;
```

```
 }
};


export const updateLaundryproduk = async (id, data) =>
            {
 return
            api.post(`/laundryproduk/${id}?_method=P
            OST`, data, {
   headers: {
    'Content-Type': 'multipart/form-data',
   },
 });
};



// ====================
// DEPOT TRANSAKSI
// ====================


export const getDepot = () => api.get('/depot');
export      const      getDepotById      =      (id)      =>
            api.get(`/depot/${id}`);
export const createDepot = (data) => api.post('/depot',
            data);
export      const      deleteDepot      =      (id)      =>
            api.delete(`/depot/${id}`);


// ====================
// FOTOCOPY TRANSAKSI
// ====================


export const getFotocopy = () => api.get('/fotocopy');
export     const     getFotocopyById     =     (id)     =>
            api.get(`/fotocopy/${id}`);
export     const     createFotocopy     =     (data)     =>
            api.post('/fotocopy', data);
export     const     deleteFotocopy     =     (id)     =>
            api.delete(`/fotocopy/${id}`);


// ====================
// LAUNDRY TRANSAKSI
// ====================


export const getLaundry = () => api.get('/laundry');
export     const     getLaundryById     =     (id)     =>
            api.get(`/laundry/${id}`);
export     const     createLaundry     =     (data)     =>
            api.post('/laundry', data);
export     const     deleteLaundry     =     (id)     =>
            api.delete(`/laundry/${id}`);
export const updateLaundry = async (id, data) => {
 return api.post(`/laundry/${id}?_method=POST`, data,
            {
  headers: {
   'Content-Type': 'multipart/form-data',
  },
 });
};


// Pengeluaran


export     const     getPengeluaran     =     ()     =>
            api.get('/pengeluaran');
export    const    getPengeluaranById    =    (id)    =>
            api.get(`/pengeluaran/${id}`);
export     const     deletePengeluaran     =     (id)     =>
            api.delete(`/pengeluaran/${id}`);
export const createPengeluaran = async (formData) => {
 try {
  const response = await api.post('/pengeluaran',
            formData, {
   headers: {
    'Content-Type': 'multipart/form-data',
   },
  });
  console.log('Status:', response.status, 'Response:',
            response.data);
  return response.data;
 } catch (error) {
  console.error('Gagal mengirim data:',
            error.response?.data || error.message);
  throw error;
 }
};
```

```
export const updatePengeluaran = async (id, data) => {
 return api.post(`/pengeluaran/${id}?_method=PUT`, data, {
  headers: {
   'Content-Type': 'multipart/form-data',
  },
 });
};


// ====================
// USERS
// ====================

// Ambil semua user
export const getUsers = () => api.get('/users');

// Ambil 1 user berdasarkan ID
export const getUserById = (id) => api.get(`/users/${id}`);

// Tambah user baru
export const createUser = async (data) => {
 try {
  const response = await api.post('/users', data);
  return response.data;
 } catch (error) {
  console.error('Gagal menambahkan user:', error.response?.data || error.message);
  throw error.response?.data || { message: 'Gagal menambahkan user!' };
 }
};

// Update user
export const updateUser = async (id, data) => {
 try {
  const response = await api.put(`/users/${id}`, data);
  return response.data;
 } catch (error) {
  console.error('Gagal update user:', error.response?.data || error.message);
  throw error.response?.data || { message: 'Gagal update user!' };
 }
};

// Hapus user
export const deleteUser = async (id) => {
 try {
  const response = await api.delete(`/users/${id}`);
  return response.data;
 } catch (error) {
  console.error('Gagal menghapus user:', error.response?.data || error.message);
  throw error.response?.data || { message: 'Gagal menghapus user!' };
 }
};

export default api;
```

```
import { enableScreens } from 'react-native-screens';
enableScreens();
import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { createContext, useEffect, useState } from 'react';
import { getUser } from './api';
import { View, ActivityIndicator, Text } from 'react-native';
import tw from 'twrnc';
import { LinearGradient } from 'expo-linear-gradient';
import LoginScreen from './Page/Auth/LoginScreen';
import RegisterScreen from './Page/Auth/RegisterScreen';
```

```
import BottomNavbar from './Component/BottomNavbar';

import AsyncStorage from '@react-native-async-storage/async-storage';

import Header from './Component/Header';

import WelcomeScreen from './Page/WelcomeScreen';

import GlobalErrorBoundary from './Component/GlobalErrorBoundary';

import { AuthContext } from './Component/AuthContext';

import DepotprodukIndex from './Page/Depot/Produk/DepotprodukIndex';

import DepotprodukCreate from './Page/Depot/Produk/DepotprodukCreate';

import DepotprodukEdit from './Page/Depot/Produk/DepotprodukEdit';

import DepotprodukDetail from './Page/Depot/Produk/DepotprodukDetail';

import DepotTransaksi from './Page/Depot/Transaksi/DepotTransaksi';

import DepotIndex from './Page/Depot/Transaksi/DepotIndex';

import DepotDetail from './Page/Depot/Transaksi/DepotDetail';

import LaundryprodukIndex from './Page/Laundry/Produk/LaundryprodukIndex';

import LaundryprodukCreate from './Page/Laundry/Produk/LaundryprodukCreate';

import LaundryprodukEdit from './Page/Laundry/Produk/LaundryprodukEdit';

import LaundryprodukDetail from './Page/Laundry/Produk/LaundryprodukDetail';

import LaundryTransaksi from './Page/Laundry/Transaksi/Laundrytransaksi';

import LaundryIndex from './Page/Laundry/Transaksi/Laundryindex';

import LaundryDetail from './Page/Laundry/Transaksi/LaundryDetail';

import FotocopyprodukIndex from './Page/Fotocopy/Produk/FotocopyprodukIndex';

import FotocopyprodukCreate from './Page/Fotocopy/Produk/FotocopyprodukCreate';

import FotocopyprodukEdit from './Page/Fotocopy/Produk/FotocopyprodukEdit';

import FotocopyprodukDetail from './Page/Fotocopy/Produk/FotocopyprodukDetail';

import FotocopyTransaksi from './Page/Fotocopy/Transaksi/FotocopyTransaksi';

import FotocopyIndex from './Page/Fotocopy/Transaksi/FotocopyIndex';

import FotocopyDetail from './Page/Fotocopy/Transaksi/FotocopyDetail';

import Laporan from './Page/Laporan';

import PengeluaranIndex from './Page/Pengeluaran/Pengeluaranindex';

import PengeluaranCreate from './Page/Pengeluaran/PengeluaranCreate';

import PengeluaranEdit from './Page/Pengeluaran/PengeluaranEdit';

import PengeluaranDetail from './Page/Pengeluaran/PengeluaranDetail';

import AddUser from './Page/Admin/AddUser';

import EditUser from './Page/Admin/EditUser';

import UserList from './Page/Admin/UserList';

import Riyawat from './Page/Riwayat';


const Stack = createNativeStackNavigator();


export default function App() {

  const [user, setUser] = useState(null);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  const [checkingToken, setCheckingToken] = useState(true);


const checkAuth = async () => {

  setLoading(true);

  setError(null);


  try {

    const token = await AsyncStorage.getItem('userToken');
```

```
      if (!token) {
        // Tidak ada token → user dianggap belum login
        setUser(null);
      } else {
        try {
          // Coba ambil data user
          const userData = await getUser(token);
          setUser(userData);
        } catch (err) {
          // Jika gagal karena Unauthenticated → hapus token & set user null
          if (err?.message?.includes('Unauthenticated') || err?.status === 401) {
            console.warn('Token expired/invalid, menghapus token...');
            await AsyncStorage.removeItem('userToken');
          }
          setUser(null);
        }
      }
    } catch (err) {
      console.error('Error saat checkAuth:', err);
      setError(err.message || 'Terjadi kesalahan saat memuat data.');
      setUser(null);
    } finally {
      setLoading(false);
      setCheckingToken(false);
    }
  };

  // Gunakan useEffect hanya sekali saat aplikasi pertama kali dijalankan
  useEffect(() => {
    checkAuth();
  }, []);

  if (checkingToken) {
    return (
      <View style={tw`flex-1 justify-center items-center bg-gray-100`}>
        <ActivityIndicator size="large" color="#0000ff" />
        <Text style={tw`mt-4 text-lg text-gray-700`}>Memuat aplikasi...</Text>
      </View>
    );
  }

  return (
    <GlobalErrorBoundary>
      <AuthContext.Provider value={{ user, setUser }}>
        <NavigationContainer>
          <LinearGradient
            colors={['#43cea2', '#185a9d']}
            start={{ x: 0, y: 0 }}
            end={{ x: 1, y: 1 }}
            style={{ flex: 1 }}
          >
            <Header />
            <Stack.Navigator screenOptions={{ headerShown: false }} >
              {!user ? (
                <>
                  <Stack.Screen name="Welcome" component={WelcomeScreen} />
                  <Stack.Screen name="Login" component={LoginScreen} />
                  <Stack.Screen name="Register" component={RegisterScreen} />
                </>
              ) : (
                <>
                  <Stack.Screen name="Main" component={BottomNavbar} />
```

```
            </>

        )}

        <>

            <Stack.Screen name="Tambah
Depot" component={DepotprodukCreate} />

            <Stack.Screen name="edit Depot"
component={DepotprodukEdit} />

            <Stack.Screen
name="DashboardDepot"
component={DepotprodukIndex} />

            <Stack.Screen name="Detail Depot"
component={DepotprodukDetail} />



            <Stack.Screen name="Depot
Transaksi" component={DepotTransaksi} />

            <Stack.Screen name="Depot
Transaksi Index" component={DepotIndex}
/>

            <Stack.Screen name="Depot
Transaksi Detail" component={DepotDetail}
/>



            <Stack.Screen name="Tambah
Laundry"
component={LaundryprodukCreate} />

            <Stack.Screen name="edit Laundry"
component={LaundryprodukEdit} />

            <Stack.Screen
name="DashboardLaundry"
component={LaundryprodukIndex} />

            <Stack.Screen name="Detail
Laundry"
component={LaundryprodukDetail} />



            <Stack.Screen name="Laundry
Transaksi" component={LaundryTransaksi}
/>

            <Stack.Screen name="Laundry
Transaksi Index"
component={LaundryIndex} />

            <Stack.Screen name="Laundry
Transaksi Detail"
component={LaundryDetail} />



            <Stack.Screen name="Tambah
Fotocopy"
component={FotocopyprodukCreate} />

            <Stack.Screen name="edit
Fotocopy"
component={FotocopyprodukEdit} />

            <Stack.Screen
name="DashboardFotocopy"
component={FotocopyprodukIndex} />

            <Stack.Screen name="Detail
Fotocopy"
component={FotocopyprodukDetail} />



            <Stack.Screen name="Fotocopy
Transaksi" component={FotocopyTransaksi}
/>

            <Stack.Screen name="Fotocopy
Transaksi Index"
component={FotocopyIndex} />

            <Stack.Screen name="Fotocopy
Transaksi Detail"
component={FotocopyDetail} />



            <Stack.Screen name="Tambah
Pengeluaran"
component={PengeluaranCreate} />

            <Stack.Screen name="edit
Pengeluaran"
component={PengeluaranEdit} />

            <Stack.Screen
name="DashboardPengeluaran"
component={PengeluaranIndex} />

            <Stack.Screen name="Detail
Pengeluaran"
component={PengeluaranDetail} />

            <Stack.Screen name="Tambah user"
component={AddUser} />

            <Stack.Screen name="List user"
component={UserList} />

            <Stack.Screen name="Edit user"
component={EditUser} />



            <Stack.Screen name="Laporan"
component={Laporan} />

            <Stack.Screen name="Riwayat"
component={Riyawat} />

        </>



    </Stack.Navigator>

    <StatusBar style="dark"
backgroundColor="#ffffff"
translucent={false} />

  </LinearGradient>

 </NavigationContainer>

</AuthContext.Provider>
```

```
      </GlobalErrorBoundary>
    );
}

import { useState, useContext } from 'react';
import { View, Text, TextInput, Button, Alert, TouchableOpacity, ActivityIndicator } from 'react-native';
import { login } from '../../api';
import { AuthContext } from '../../Component/AuthContext';
import tw from 'twrnc';
import { FontAwesome } from '@expo/vector-icons';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { useNavigation } from '@react-navigation/native';

export default function LoginScreen() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [isLoading, setIsLoading] = useState(false); // ⬅️ State loading
  const { setUser } = useContext(AuthContext);
  const [showPassword, setShowPassword] = useState(false);
  const navigation = useNavigation();

  const handleLogin = async () => {
    if (!email.trim() || !password.trim()) {
      Alert.alert('Error', 'Email dan password tidak boleh kosong!');
      return;
    }

    setIsLoading(true); // Mulai loading
    try {
      const response = await login({ email, password });

      await AsyncStorage.setItem('userToken', response.token);
      setUser(response.user);

      Alert.alert('Berhasil', 'Login sukses!');
    } catch (error) {
      console.error('❌ Login gagal:', error);
      let message = 'Login gagal!';
      const msg = error?.message;

      if (msg === 'These credentials do not match our records.') {
        message = 'Email atau kata sandi Anda salah';
      }

      Alert.alert('Gagal', message);
    } finally {
      setIsLoading(false); // Selesai loading
    }
  };

  return (
    <View style={tw`flex-1 items-center justify-center bg-gray-100 p-6`}>
      <Text style={tw`text-2xl font-bold mb-6 text-red-500`}>Login</Text>

      <TextInput
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        style={tw`w-full bg-white p-4 mb-4 rounded-xl shadow-sm`}
        keyboardType="email-address"
        autoCapitalize="none"
      />
      <View style={tw`w-full bg-white flex-row items-center p-4 mb-4 rounded-xl shadow-sm`}>
        <TextInput
          placeholder="Password"
          value={password}
```

```
            onChangeText={setPassword}
            style={tw`flex-1`}
            secureTextEntry={!showPassword}
          />
          <TouchableOpacity onPress={() => setShowPassword(!showPassword)}>
            <FontAwesome name={showPassword ? "eye-slash" : "eye"} size={20} />
          </TouchableOpacity>
        </View>

        <View style={tw`w-full mt-2`}>
          {isLoading ? (
            <ActivityIndicator size="large" color="#4CAF50" />
          ) : (
            <Button title="Login" onPress={handleLogin} color="#4CAF50" />
          )}
        </View>

        {/* <TouchableOpacity onPress={() => navigation.navigate('Register')}>
          <Text style={tw`text-sm mt-4 text-blue-600`}>Belum punya akun? Daftar di sini</Text>
        </TouchableOpacity> */}
      </View>
    );
}
import { useState } from 'react';
import { View, Text, TextInput, Button, Alert, TouchableOpacity, ActivityIndicator } from 'react-native';
import { register } from '../../api';
import tw from 'twrnc';
import { FontAwesome } from '@expo/vector-icons';

export default function RegisterScreen({ navigation }) {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [passwordConfirmation, setPasswordConfirmation] = useState('');
  const [showPassword, setShowPassword] = useState(false);
  const [showPasswordConfirmation, setShowPasswordConfirmation] = useState(false);
  const [isLoading, setIsLoading] = useState(false); // ⬅️ Tambahkan ini

  const handleRegister = async () => {
    if (password !== passwordConfirmation) {
      Alert.alert('Gagal', 'Password dan Konfirmasi Password tidak sama!');
      return;
    }

    setIsLoading(true); // ⬅️ Mulai loading
    try {
      await register({
        name,
        email,
        password,
        password_confirmation: passwordConfirmation
      });
      Alert.alert('Berhasil', 'Registrasi sukses!');
      navigation.navigate('Login');
    } catch (error) {
      Alert.alert('Gagal', error.message || 'Registrasi gagal!');
    } finally {
      setIsLoading(false); // ⬅️ Selesai loading
    }
  };

  return (
    <View style={tw`flex-1 items-center justify-center bg-gray-100 p-6`}>
      <Text style={tw`text-2xl font-bold mb-6`}>Register</Text>
```

```
<TextInput
  placeholder="Nama"
  value={name}
  onChangeText={setName}
  style={tw`w-full bg-white p-4 mb-4 rounded-xl shadow-sm`}
/>

<TextInput
  placeholder="Email"
  value={email}
  onChangeText={setEmail}
  style={tw`w-full bg-white p-4 mb-4 rounded-xl shadow-sm`}
  keyboardType="email-address"
  autoCapitalize="none"
/>

<View style={tw`w-full bg-white flex-row items-center p-4 mb-4 rounded-xl shadow-sm`}>
  <TextInput
    placeholder="Password"
    value={password}
    onChangeText={setPassword}
    style={tw`flex-1`}
    secureTextEntry={!showPassword}
  />
  <TouchableOpacity onPress={() => setShowPassword(!showPassword)}>
    <FontAwesome name={showPassword ? "eye-slash" : "eye"} size={20} />
  </TouchableOpacity>
</View>

<View style={tw`w-full bg-white flex-row items-center p-4 mb-6 rounded-xl shadow-sm`}>
  <TextInput
    placeholder="Konfirmasi Password"
    value={passwordConfirmation}
    onChangeText={setPasswordConfirmation}
    style={tw`flex-1`}
    secureTextEntry={!showPasswordConfirmation}
  />
  <TouchableOpacity onPress={() => setShowPasswordConfirmation(!showPasswordConfirmation)}>
    <FontAwesome name={showPasswordConfirmation ? "eye-slash" : "eye"} size={20} />
  </TouchableOpacity>
</View>

<View style={tw`w-full mt-2`}>
  {isLoading ? (
    <ActivityIndicator size="large" color="#4CAF50" />
  ) : (
    <Button title="Register" onPress={handleRegister} color="#4CAF50" />
  )}
</View>

<TouchableOpacity onPress={() => navigation.navigate('Login')}>
  <Text style={tw`text-sm mt-4 text-blue-600`}>Sudah punya akun? Login di sini</Text>
</TouchableOpacity>
</View>
);
}

import React, { useEffect, useState, useContext } from 'react';
import { View, Text, Dimensions, ScrollView, Image } from 'react-native';
import { getUser, getProdukTerlaris } from '../api';
import tw from 'twrnc';
```

```
import { AuthContext } from '../Component/AuthContext';
import { LinearGradient } from 'expo-linear-gradient';
import { BlurView } from 'expo-blur';
import LottieView from 'lottie-react-native';

const { width } = Dimensions.get('window');

export default function DashboardScreen({ navigation }) {
  const { user, setUser } = useContext(AuthContext);
  const [produkTerlaris, setProdukTerlaris] = useState([]);

  useEffect(() => {
    const fetchUser = async () => {
      try {
        const userData = await getUser();
        setUser(userData);
      } catch (error) {
        setUser(null);
      }
    };

    const fetchProdukTerlaris = async () => {
      try {
        const data = await getProdukTerlaris();
        setProdukTerlaris(data);
      } catch (error) {
        console.error("Error fetching best-selling products:", error);
      }
    };

    fetchUser();
    fetchProdukTerlaris();
  }, []);

  return (
    <LinearGradient
      colors={['#43cea2', '#185a9d']}
      style={tw`flex-1 px-4`}
    >
      <ScrollView contentContainerStyle={tw`flex-grow items-center justify-center`}>
        <LottieView
          source={require('../assets/kasir.json')} // ganti sesuai file animasimu
          autoPlay
          loop
          style={{ width: width * 0.7, height: width * 0.7, marginBottom: 10 }}
        />

        <Text style={tw`text-white text-3xl font-bold mb-4`}>
          Dashboard
        </Text>

        <BlurView intensity={90} style={tw`w-full rounded-3xl px-6 py-5 mb-6`}>
          {user ? (
            <View>
              <Text style={tw`text-xl font-bold text-gray-800 mb-1`}>
                👋 Halo, {user.name}
              </Text>
              <Text style={tw`text-sm text-gray-500`}>
                📧 {user.email}
              </Text>
              <Text style={tw`text-sm text-gray-500 mt-1`}>
                📜 Role: {user.roles?.join(', ') || 'Tidak tersedia'}
              </Text>
            </View>
          ) : (
            <Text style={tw`text-base text-gray-700`}>
              Anda belum login.
```

```
      </Text>
    )}
</BlurView>

{/* Produk Terlaris */}
<Text style={tw`text-white text-2xl font-bold mb-4`}>Produk Terlaris</Text>
<ScrollView contentContainerStyle={tw`w-full mb-20`}>
  {produkTerlaris.length > 0 ? (
    produkTerlaris.map((item, index) => (
      <View key={index} style={tw`bg-white p-4 rounded-lg mb-6 shadow-lg`}>
        {/* Produk Image (if available) */}
        {item.produk?.image && (
          <Image
            source={{ uri: item.produk.image }}
            style={tw`w-full h-40 rounded-lg mb-4`}
            resizeMode="cover"
          />
        )}
        <Text style={tw`text-xl font-semibold text-gray-800`}>
          {item.produk.nama}
        </Text>
        <Text style={tw`text-sm text-gray-600 mb-2`}>
          Terjual: {item.total_quantity}
        </Text>
        <Text style={tw`text-sm text-gray-500`}>
          Harga: Rp {item.produk.harga.toLocaleString()}
        </Text>
      </View>
    ))
  ) : (
    <Text style={tw`text-sm text-gray-600`}>Tidak ada data produk terlaris.</Text>
  )}
      </ScrollView>
    </ScrollView>
  </LinearGradient>
 );
}import React, { useEffect, useState, useCallback } from 'react';
import { View, Text, FlatList, ActivityIndicator, RefreshControl, TouchableOpacity, Alert } from 'react-native';
import tw from 'twrnc';
import { getUsers, deleteUser } from '../../api'; // sesuaikan path

const UserList = ({ navigation }) => {
 const [users, setUsers] = useState([]);
 const [loading, setLoading] = useState(false);
 const [refreshing, setRefreshing] = useState(false);
 const [deletingId, setDeletingId] = useState(null); // untuk menandai user yang sedang dihapus

 const fetchUsers = async () => {
  setLoading(true);
  try {
   const response = await getUsers();
   setUsers(response.data);
  } catch (error) {
   Alert.alert('Error', 'Gagal mengambil data user');
  } finally {
   setLoading(false);
  }
 };

 const onRefresh = useCallback(async () => {
  setRefreshing(true);
  try {
   const response = await getUsers();
   setUsers(response.data);
  } catch (error) {
   Alert.alert('Error', 'Gagal mengambil data user');
  } finally {
```

```
    setRefreshing(false);
  }
}, []);

useEffect(() => {
  fetchUsers();
}, []);

const confirmDelete = (id, name) => {
  Alert.alert(
    'Konfirmasi Hapus',
    `Yakin ingin menghapus user "${name}"?`,
    [
      { text: 'Batal', style: 'cancel' },
      { text: 'Hapus', style: 'destructive', onPress: () =>
          handleDelete(id) },
    ],
    { cancelable: true }
  );
};

const handleDelete = async (id) => {
  setDeletingId(id);
  try {
    await deleteUser(id);
    Alert.alert('Sukses', 'User berhasil dihapus');
    // Refresh data setelah hapus
    fetchUsers();
  } catch (error) {
    Alert.alert('Error', error.message || 'Gagal menghapus
        user');
  } finally {
    setDeletingId(null);
  }
};

const renderItem = ({ item }) => (
  <View style={tw`border-b border-gray-200 p-4 flex-
      row justify-between items-center`}>
    <TouchableOpacity
      style={tw`flex-1`}
      onPress={() => navigation.navigate('Edit user', {
          userId: item.id })}
    >
      <Text style={tw`text-lg font-
          semibold`}>{item.name}</Text>
      <Text style={tw`text-gray-
          600`}>{item.email}</Text>
      <Text style={tw`text-gray-500 italic`}>
{item.roles && item.roles.length > 0 ?
          item.roles[0].name : 'Role belum diisi'}
</Text>

    </TouchableOpacity>
    <TouchableOpacity
      onPress={() => confirmDelete(item.id,
          item.name)}
      disabled={deletingId === item.id}
      style={tw`bg-red-600 px-3 py-1 rounded ml-3`}
    >
      <Text style={tw`text-white font-semibold`}>
        {deletingId === item.id ? 'Menghapus...' : 'Hapus'}
      </Text>
    </TouchableOpacity>
  </View>
);

if (loading) {
  return (
    <View style={tw`flex-1 justify-center items-
        center`}>
      <ActivityIndicator size="large" color="#3b82f6" />
    </View>
  );
}

return (
  <View style={tw`flex-1 bg-white`}>
    {/* Tombol tambah */}
```

```
    <View style={tw`p-4 border-b border-gray-300 flex-row justify-end`}>
      <TouchableOpacity
        style={tw`bg-blue-600 px-4 py-2 rounded`}
        onPress={() => navigation.navigate('Tambah user')}
      >
        <Text style={tw`text-white font-semibold`}>Tambah User</Text>
      </TouchableOpacity>
    </View>

    <FlatList
      data={users}
      keyExtractor={(item) => item.id.toString()}
      renderItem={renderItem}
      refreshControl={<RefreshControl refreshing={refreshing} onRefresh={onRefresh} />}
      ListEmptyComponent={
        <View style={tw`p-4`}>
          <Text style={tw`text-center text-gray-400`}>Tidak ada data user.</Text>
        </View>
      }
    />
  </View>
 );
};

export default UserList;
import React, { useEffect, useState, useContext } from 'react';
import {
 View,
 Text,
 ScrollView,
 ActivityIndicator,
 TextInput,
 TouchableOpacity,
 Alert,
 ImageBackground,
} from 'react-native';
import tw from 'twrnc';
import { getDepotproduk, deleteDepotproduk } from '../../../api';
import { AuthContext } from "../../../Component/AuthContext";
import { LinearGradient } from 'expo-linear-gradient';

const DepotprodukIndex = ({ navigation }) => {
 const { user } = useContext(AuthContext) || { user: null };
 const [depotproduk, setDepotproduk] = useState([]);
 const [filteredDepotproduk, setFilteredDepotproduk] = useState([]);
 const [search, setSearch] = useState('');
 const [loading, setLoading] = useState(true);
 const role = user?.roles || "";

 useEffect(() => {
  fetchDepotproduk();
 }, []);

 useEffect(() => {
  handleSearch(search);
 }, [search, depotproduk]);

 const fetchDepotproduk = async () => {
  setLoading(true);
  try {
   const response = await getDepotproduk();
   setDepotproduk(response.data);
   setFilteredDepotproduk(response.data);
  } catch (error) {
   console.error('Gagal memuat data depot produk:', error);
   Alert.alert('Error', 'Gagal memuat data depot produk.');
  } finally {
   setLoading(false);
```

```
  }
};

const handleSearch = (text) => {
  setSearch(text);
  const filtered = depotproduk.filter((item) =>
        item.nama.toLowerCase().includes(text.toLowerCase())
  );
  setFilteredDepotproduk(filtered);
};

const handleDelete = (id) => {
  Alert.alert(
    'Konfirmasi Hapus',
    'Apakah Anda yakin ingin menghapus item ini?',
    [
      { text: 'Batal', style: 'cancel' },
      {
        text: 'Hapus',
        style: 'destructive',
        onPress: async () => {
          try {
            await deleteDepotproduk(id);
            fetchDepotproduk();
          } catch (error) {
            console.error('Gagal menghapus:', error);

            // Ambil pesan error dari response jika tersedia
            const errorMessage =
              error?.response?.data?.message || 'Gagal menghapus data.';

            Alert.alert('Error', errorMessage);
          }
        },
      },
    ]
  );
};

if (loading) {
  return (
    <View style={tw`flex-1 justify-center items-center bg-black`}>
      <ActivityIndicator size="large" color="#FACC15" />
      <Text style={tw`mt-4 text-green-400`}>Memuat data depot produk...</Text>
    </View>
  );
}

return (
    <LinearGradient
      colors={['#43cea2', '#185a9d']}
      style={tw`flex-1 items-center justify-center px-4`}
    >
    <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
      <Text style={tw`text-2xl font-bold text-green-400 text-center mb-6`}>
        Daftar Barang
      </Text>

      {role.includes("admin") && (
        <>
          <TouchableOpacity
            style={tw`bg-green-500 py-3 rounded-full mb-4`}
            onPress={() => navigation.navigate("Tambah Depot")}
          >
            <Text style={tw`text-black text-center font-bold`}>
              + Tambah Barang
            </Text>
          </TouchableOpacity>
```

```
  </>
)
}

<TextInput
  placeholder="Cari berdasarkan nama produk..."
  value={search}
  onChangeText={handleSearch}
  style={tw`bg-white/80 border border-green-400 rounded-xl px-4 py-2 mb-4 text-gray-900`}
  placeholderTextColor="#6B7280"
/>

<ScrollView>
  {filteredDepotproduk.length > 0 ? (
    filteredDepotproduk.map((item, index) => (
      <View
        key={index}
        style={tw`mb-4 p-4 bg-white/80 rounded-xl shadow`}
      >
        <Text style={tw`text-lg font-semibold text-black`}>
          {item.nama}
        </Text>
        <Text style={tw`mt-1 text-gray-700`}>jumlah: {item.jumlah}</Text>
        <Text style={tw`mt-1 text-gray-700`}>Harga: Rp {item.harga}</Text>

        <View style={tw`flex-row gap-4 mt-3 flex-wrap`}>
          {role.includes("admin") && (
            <>
              <TouchableOpacity
                onPress={() => handleDelete(item.id)}
                style={tw`bg-red-600 px-4 py-2 rounded-full`}
              >
                <Text style={tw`text-white font-semibold`}>Hapus</Text>
              </TouchableOpacity>

              <TouchableOpacity
                onPress={() =>
                  navigation.navigate("edit Depot", { id: item.id })
                }
                style={tw`bg-blue-600 px-4 py-2 rounded-full`}
              >
                <Text style={tw`text-white font-semibold`}>Edit</Text>
              </TouchableOpacity>
            </>
          )}
          <TouchableOpacity
            onPress={() =>
              navigation.navigate("Detail Depot", { id: item.id })
            }
            style={tw`bg-green-600 px-4 py-2 rounded-full`}
          >
            <Text style={tw`text-white font-semibold`}>Detail</Text>
          </TouchableOpacity>
        </View>
      </View>
    ))
  ) : (
    <Text style={tw`text-center text-green-200`}>
      Tidak ada data depot produk yang cocok.
    </Text>
  )}
</ScrollView>
</View>
</LinearGradient>
);
};
```

```
export default DepotprodukIndex;

import React, { useEffect, useState,useContext } from
        "react";

import {

  View,

  Text,

  ScrollView,

  TouchableOpacity,

  ActivityIndicator,

  Alert,

} from "react-native";

import tw from "twrnc";

import { getDepot, deleteDepot } from "../../../api";

import { FontAwesome5 } from "@expo/vector-icons";

import { LinearGradient } from "expo-linear-gradient";

import      {      AuthContext      }      from
        "../../../Component/AuthContext";



const DepotIndex = ({ navigation }) => {

  const { user } = useContext(AuthContext);


  const role = user?.roles || "";

  const [data, setData] = useState([]);

  const [loading, setLoading] = useState(true);


  const fetchData = async () => {

    try {

      const res = await getDepot();

      setData(res.data);

    } catch (err) {

      console.log(err);

      Alert.alert("Error", "Gagal mengambil data
        transaksi");

    } finally {

      setLoading(false);

    }

  };
```

```
const handleDelete = (id) => {

  Alert.alert("Konfirmasi", "Yakin ingin menghapus
      transaksi ini?", [

    { text: "Batal" },

    {

      text: "Hapus",

      onPress: async () => {

        try {

          await deleteDepot(id);

          Alert.alert("Berhasil", "Transaksi
        dihapus");

          fetchData();

        } catch (e) {

          console.log(e);

          Alert.alert("Error", "Gagal menghapus
        transaksi");

        }

      },

      style: "destructive",

    },

  ]);

};


useEffect(() => {

  fetchData();

}, []);


if (loading) {

  return (

    <View style={tw`flex-1 justify-center items-
      center bg-black`}>

      <ActivityIndicator size="large"
      color="#FACC15" />

      <Text style={tw`text-green-500 mt-
      4`}>Memuat transaksi...</Text>

    </View>

  );

}


return (
```

```
<LinearGradient
  colors={["rgba(0,0,0,0.9)",
    "rgba(30,30,50,0.95)"]}
  style={tw`flex-1`}
>
  <ScrollView contentContainerStyle={tw`p-5 pb-20`}>
    <View style={tw`flex-row justify-between items-center mb-4`}>
      <Text style={tw`text-green-500 text-2xl font-bold`}>
        Data Transaksi Depot
      </Text>
      <TouchableOpacity
        style={tw`bg-green-500 px-3 py-2 rounded-xl`}
        onPress={() => navigation.navigate("Depot Transaksi")}
      >
        <Text style={tw`text-black font-bold`}>
          + Tambah
        </Text>
      </TouchableOpacity>
    </View>

    {data.length === 0 && (
      <Text style={tw`text-white text-center`}>
        Belum ada transaksi
      </Text>
    )}

    {data.map((item) => (
      <View
        key={item.id}
        style={tw`bg-black/40 p-4 rounded-xl mb-4`}
      >
        <Text style={tw`text-green-500 text-lg font-bold`}>
          No Struk: {item.no_struk}
        </Text>
        <Text style={tw`text-white`}>
          Kasir: {item.kasir}
        </Text>

        <Text style={tw`text-white`}>
          Jenis Pembayaran: {item.jenis_pembayaran}
        </Text>
        <Text style={tw`text-white`}>
          Jumlah: {item.jumlah.toLocaleString()}
        </Text>
        {/* <Text style={tw`text-white`}>
          Harga: Rp {item.harga.toLocaleString()}
        </Text> */}
        <Text style={tw`text-white`}>
          Total: Rp {item.total.toLocaleString()}
        </Text>
        <Text style={tw`text-white`}>
          Waktu: {item.waktu}
        </Text>

        <View style={tw`flex-row justify-between mt-4`}>
          {role.includes("admin") && (
            <>
              <TouchableOpacity
                style={tw`bg-red-600 px-4 py-2 rounded-xl`}
                onPress={() => handleDelete(item.id)}
              >
                <Text style={tw`text-white font-bold`}>Hapus</Text>
              </TouchableOpacity>
            </>
          )
          }

          <TouchableOpacity
```

```
                    style={tw`bg-green-500 px-4 py-2
            rounded-xl`}
                    onPress={() =>
                      navigation.navigate("Depot
            Transaksi Detail", {
                        id: item.id,
                      })
                    }
                  >
                    <Text style={tw`text-black font-
            bold`}>
                      Detail
                    </Text>
                  </TouchableOpacity>
                </View>
              </View>
            ))}
          </ScrollView>
        </LinearGradient>
    );
};

export default DepotIndex;
```

```
import React, { useEffect, useState, useContext } from
        'react';
import {
  View,
  Text,
  ScrollView,
  ActivityIndicator,
  TextInput,
  TouchableOpacity,
  Alert,
  ImageBackground,
} from 'react-native';
import tw from 'twrnc';
import { getFotocopyproduk, deleteFotocopyproduk }
        from '../../../api';
import { AuthContext } from
        "../../../Component/AuthContext";
import { LinearGradient } from 'expo-linear-gradient';

const FotocopyprodukIndex = ({ navigation }) => {
  const { user } = useContext(AuthContext) || { user: null
        };
  const [fotocopyproduk, setFotocopyproduk] =
        useState([]);
  const [filteredFotocopyproduk,
        setFilteredFotocopyproduk] = useState([]);
  const [search, setSearch] = useState('');
  const [loading, setLoading] = useState(true);
  const [selectedTab, setSelectedTab] = useState('ATK');
        // default ATK

  const role = user?.roles || "";

  useEffect(() => {
    fetchFotocopyproduk();
  }, []);

  useEffect(() => {
    handleSearch(search);
  }, [search, fotocopyproduk]);

  const fetchFotocopyproduk = async () => {
    setLoading(true);
    try {
      const response = await getFotocopyproduk();
      setFotocopyproduk(response.data);
      setFilteredFotocopyproduk(response.data);
    } catch (error) {
      console.error('Gagal memuat data Fotocopy
        produk:', error);
      Alert.alert('Error', 'Gagal memuat data Fotocopy
        produk.');
    } finally {
      setLoading(false);
    }
  };
```

```
const handleSearch = (text) => {
  setSearch(text);
  const filtered = fotocopyproduk.filter((item) =>
            item.nama.toLowerCase().includes(text.toLowerCase())
  );
  setFilteredFotocopyproduk(filtered);
};

const handleDelete = (id) => {
  Alert.alert(
    'Konfirmasi Hapus',
    'Apakah Anda yakin ingin menghapus item ini?',
    [
      { text: 'Batal', style: 'cancel' },
      {
        text: 'Hapus',
        style: 'destructive',
        onPress: async () => {
          try {
            await deleteFotocopyproduk(id);
            fetchFotocopyproduk();
          } catch (error) {
            console.error('Gagal menghapus:', error);
            // Ambil pesan error dari response jika tersedia
            const errorMessage =
              error?.response?.data?.message || 'Gagal menghapus data.';

            Alert.alert('Error', errorMessage);

          }
        },
      },
    ]
  );
};

if (loading) {
  return (
    <View style={tw`flex-1 justify-center items-center bg-black`}>
      <ActivityIndicator size="large" color="#FACC15" />
      <Text style={tw`mt-4 text-green-400`}>Memuat data Fotocopy barang...</Text>
    </View>
  );
}

return (
    <LinearGradient
      colors={['#43cea2', '#185a9d']}
      style={tw`flex-1 items-center justify-center px-4`}
    >
   <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
    <Text style={tw`text-2xl font-bold text-green-400 text-center mb-6`}>
      Daftar Barang dan jasa
    </Text>

    {role.includes("admin") && (
      <>
        <TouchableOpacity
          style={tw`bg-green-500 py-3 rounded-full mb-4`}
          onPress={() => navigation.navigate("Tambah Fotocopy")}
        >
          <Text style={tw`text-black text-center font-bold`}>
            + Tambah Barang dan jasa
          </Text>
        </TouchableOpacity>
      </>
    )
    }
```

```
<TextInput
 placeholder="Cari berdasarkan nama barang atau
      jasa..."
 value={search}
 onChangeText={handleSearch}
 style={tw`bg-white/80 border border-green-400
      rounded-xl px-4 py-2 mb-4 text-gray-900`}
 placeholderTextColor="#6B7280"
/>
<View style={tw`flex-row justify-center mb-4`}>
 <TouchableOpacity
  onPress={() => setSelectedTab('ATK')}
  style={tw`px-4 py-2 rounded-l-full
      ${selectedTab === 'ATK' ? 'bg-green-400' :
      'bg-green-200'}`}
 >
  <Text style={tw`${selectedTab === 'ATK' ?
      'text-black font-bold' : 'text-gray-
      700'}`}>ATK</Text>
 </TouchableOpacity>
 <TouchableOpacity
  onPress={() => setSelectedTab('Jasa')}
  style={tw`px-4 py-2 rounded-r-full
      ${selectedTab === 'Jasa' ? 'bg-green-400' :
      'bg-green-200'}`}
 >
  <Text style={tw`${selectedTab === 'Jasa' ? 'text-
      black font-bold' : 'text-gray-
      700'}`}>Jasa</Text>
 </TouchableOpacity>
</View>

<ScrollView>
 {filteredFotocopyproduk.filter(item =>
      item.kategori === selectedTab).length > 0 ? (
  filteredFotocopyproduk
   .filter(item => item.kategori === selectedTab)
   .map((item, index) => (
    <View
     key={index}
     style={tw`mb-4 p-4 bg-white/80 rounded-xl
shadow`}
    >
     <Text style={tw`text-lg font-semibold text-
black`}>
      {item.nama}
     </Text>
     <Text style={tw`mt-1 text-gray-
700`}>Jumlah: {item.jumlah}</Text>
     <Text style={tw`mt-1 text-gray-
700`}>Harga: Rp {item.harga}</Text>

     <View style={tw`flex-row gap-4 mt-3 flex-
wrap`}>
      {role.includes("admin") && (
       <>
        <TouchableOpacity
         onPress={() => handleDelete(item.id)}
         style={tw`bg-red-600 px-4 py-2
rounded-full`}
        >
         <Text style={tw`text-white font-
semibold`}>Hapus</Text>
        </TouchableOpacity>

        <TouchableOpacity
         onPress={() =>
navigation.navigate("edit Fotocopy", { id:
item.id })}
         style={tw`bg-blue-600 px-4 py-2
rounded-full`}
        >
         <Text style={tw`text-white font-
semibold`}>Edit</Text>
        </TouchableOpacity>

       </>
      )
     }

     <TouchableOpacity
```

```
            onPress={() => navigation.navigate("Detail Fotocopy", { id: item.id })}
            style={tw`bg-green-600 px-4 py-2 rounded-full`}
           >
            <Text style={tw`text-white font-semibold`}>Detail</Text>
           </TouchableOpacity>
          </View>
         </View>
        ))
       ) : (
        <Text style={tw`text-center text-green-200`}>
         Tidak ada data kategori {selectedTab}.
        </Text>
       )}
      </ScrollView>
     </View>
    </LinearGradient>
  );
};

export default FotocopyprodukIndex;
import React, { useEffect, useState, useContext } from 'react';
import {
 View,
 Text,
 ScrollView,
 ActivityIndicator,
 TextInput,
 TouchableOpacity,
 Alert,
 ImageBackground,
} from 'react-native';
import tw from 'twrnc';
import { getLaundryproduk, deleteLaundryproduk } from '../../../api';
import { LinearGradient } from 'expo-linear-gradient';
import { AuthContext } from "../../../Component/AuthContext";

const LaundryprodukIndex = ({ navigation }) => {
 const { user } = useContext(AuthContext) || { user: null };

 const role = user?.roles || "";
 const [laundryproduk, setLaundryproduk] = useState([]);
 const [filteredLaundryproduk, setFilteredLaundryproduk] = useState([]);
 const [search, setSearch] = useState('');
 const [loading, setLoading] = useState(true);

 useEffect(() => {
  fetchLaundryproduk();
 }, []);

 useEffect(() => {
  handleSearch(search);
 }, [search, laundryproduk]);

 const fetchLaundryproduk = async () => {
  setLoading(true);
  try {
   const response = await getLaundryproduk();
   setLaundryproduk(response.data);
   setFilteredLaundryproduk(response.data);
  } catch (error) {
   console.error('Gagal memuat data Laundry produk:', error);
   Alert.alert('Error', 'Gagal memuat data Laundry produk.');
  } finally {
   setLoading(false);
  }
 };
```

```
const handleSearch = (text) => {
 setSearch(text);
 const filtered = laundryproduk.filter((item) =>
        item.nama.toLowerCase().includes(text.toLowerCase())
 );
 setFilteredLaundryproduk(filtered);
};

const handleDelete = (id) => {
 Alert.alert(
  'Konfirmasi Hapus',
  'Apakah Anda yakin ingin menghapus item ini?',
  [
   { text: 'Batal', style: 'cancel' },
   {
    text: 'Hapus',
    style: 'destructive',
    onPress: async () => {
     try {
      await deleteLaundryproduk(id);
      fetchLaundryproduk();
     } catch (error) {
      console.error('Gagal menghapus:', error);
      Alert.alert('Error', 'Gagal menghapus data.');
     }
    },
   },
  ]
 );
};

if (loading) {
 return (
  <View style={tw`flex-1 justify-center items-center bg-black`}>
   <ActivityIndicator size="large" color="#FACC15" />
   <Text style={tw`mt-4 text-green-400`}>Memuat data Laundry produk...</Text>
  </View>
 );
}

return (
  <LinearGradient
   colors={['#43cea2', '#185a9d']}
   style={tw`flex-1 items-center justify-center px-4`}
  >
  <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
   <Text style={tw`text-2xl font-bold text-green-400 text-center mb-6`}>
    Daftar Jasa Laundry
   </Text>

   {role.includes("admin") && (
    <>
     <TouchableOpacity
      style={tw`bg-green-500 py-3 rounded-full mb-4`}
      onPress={() => navigation.navigate("Tambah Laundry")}
     >
      <Text style={tw`text-black text-center font-bold`}>
       + Tambah Jasa Laundry
      </Text>
     </TouchableOpacity>
    </>
   )}

   <TextInput
    placeholder="Cari berdasarkan nama produk..."
    value={search}
```

```
 onChangeText={handleSearch}
 style={tw`bg-white/80 border border-green-400 rounded-xl px-4 py-2 mb-4 text-gray-900`}
 placeholderTextColor="#6B7280"
/>

<ScrollView>
 {filteredLaundryproduk.length > 0 ? (
  filteredLaundryproduk.map((item, index) => (
   <View
    key={index}
    style={tw`mb-4 p-4 bg-white/80 rounded-xl shadow`}
   >
    <Text style={tw`text-lg font-semibold text-black`}>
     {item.nama}
    </Text>

    <Text style={tw`mt-1 text-gray-700`}>Harga: Rp {item.harga} Per {item.deskripsi}</Text>

    <View style={tw`flex-row gap-4 mt-3 flex-wrap`}>
     {role.includes("admin") && (
      <>
       <TouchableOpacity
        onPress={() => handleDelete(item.id)}
        style={tw`bg-red-600 px-4 py-2 rounded-full`}
       >
        <Text style={tw`text-white font-semibold`}>Hapus</Text>
       </TouchableOpacity>

       <TouchableOpacity
        onPress={() =>
         navigation.navigate("edit Laundry", { id: item.id })
        }
        style={tw`bg-blue-600 px-4 py-2 rounded-full`}
       >
        <Text style={tw`text-white font-semibold`}>Edit</Text>
       </TouchableOpacity>
      </>
     )}
     <TouchableOpacity
      onPress={() =>
       navigation.navigate("Detail Laundry", { id: item.id })
      }
      style={tw`bg-green-600 px-4 py-2 rounded-full`}
     >
      <Text style={tw`text-white font-semibold`}>Detail</Text>
     </TouchableOpacity>
    </View>
   </View>
  ))
 ) : (
  <Text style={tw`text-center text-green-200`}>
   Tidak ada data Laundry produk yang cocok.
  </Text>
 )}
</ScrollView>
</View>
</LinearGradient>
);
};

export default LaundryprodukIndex;

import React, { useEffect, useState, useContext } from 'react';
import {
 View,
 Text,
 ScrollView,
 ActivityIndicator,
```

```
  TextInput,
  TouchableOpacity,
  Alert,
  Image,
  ImageBackground,
} from 'react-native';
import tw from 'twrnc';
import { getPengeluaran, deletePengeluaran } from '../../api';
import { AuthContext } from '../../Component/AuthContext';
import { LinearGradient } from 'expo-linear-gradient';

const Pengeluaranindex = ({ navigation }) => {
  const { user } = useContext(AuthContext) || { user: null };
  const [pengeluaran, setPengeluaran] = useState([]);
  const [filteredPengeluaran, setFilteredPengeluaran] = useState([]);
  const [search, setSearch] = useState('');
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchPengeluaran();
  }, []);

  useEffect(() => {
    handleSearch(search);
  }, [search, pengeluaran]);

  const fetchPengeluaran = async () => {
    setLoading(true);
    try {
      const response = await getPengeluaran();
      setPengeluaran(response.data);
      setFilteredPengeluaran(response.data);
    } catch (error) {
      console.error('Gagal memuat data Pengeluaran:', error);
      Alert.alert('Error', 'Gagal memuat data pengeluaran.');
    } finally {
      setLoading(false);
    }
  };

  const handleSearch = (text) => {
    setSearch(text);
    const filtered = pengeluaran.filter((item) =>
      item.nama_pengeluaran.toLowerCase().includes(text.toLowerCase())
    );
    setFilteredPengeluaran(filtered);
  };

  const handleDelete = (id) => {
    Alert.alert(
      'Konfirmasi Hapus',
      'Apakah Anda yakin ingin menghapus pengeluaran ini?',
      [
        { text: 'Batal', style: 'cancel' },
        {
          text: 'Hapus',
          style: 'destructive',
          onPress: async () => {
            try {
              await deletePengeluaran(id);
              fetchPengeluaran();
            } catch (error) {
              console.error('Gagal menghapus:', error);
              Alert.alert('Error', 'Gagal menghapus data.');
            }
          },
        },
      ]
    );
```

```
};

if (loading) {
 return (
  <View style={tw`flex-1 justify-center items-center bg-black`}>
   <ActivityIndicator size="large" color="#FACC15" />
   <Text style={tw`mt-4 text-green-400`}>Memuat data pengeluaran...</Text>
  </View>
 );
}

return (
    <LinearGradient
      colors={['#43cea2', '#185a9d']}
      style={tw`flex-1 items-center justify-center px-4`}
    >
  <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
   <Text style={tw`text-2xl font-bold text-green-400 text-center mb-6`}>
    Daftar Pengeluaran
   </Text>

   <TouchableOpacity
    style={tw`bg-green-500 py-3 rounded-full mb-4`}
    onPress={() => navigation.navigate('Tambah Pengeluaran')}
   >
    <Text style={tw`text-black text-center font-bold`}>
     + Tambah Pengeluaran
    </Text>
   </TouchableOpacity>

   <TextInput
    placeholder="Cari berdasarkan nama pengeluaran..."
    value={search}
    onChangeText={handleSearch}
    style={tw`bg-white/80 border border-green-400 rounded-xl px-4 py-2 mb-4 text-gray-900`}
    placeholderTextColor="#6B7280"
   />

<ScrollView>
 {filteredPengeluaran.length > 0 ? (
  filteredPengeluaran.map((item, index) => (
   <View
    key={index}
    style={tw`mb-4 p-4 bg-white/80 rounded-xl shadow`}
   >
    <Text style={tw`text-lg font-semibold text-black`}>
     {item.nama_pengeluaran}
    </Text>
    <Text style={tw`mt-1 text-gray-700`}>
     Kategori usaha: {item.nama_permintaan}
    </Text>
    <Text style={tw`mt-1 text-gray-700`}>
     Waktu: {item.waktu}
    </Text>
    <Text style={tw`mt-1 text-gray-700`}>
     Jumlah pengeluaran Rp {item.harga}
    </Text>
    {item.foto && (
     <Image
      source={{ uri: item.foto }}
      style={tw`w-full h-40 mt-2 rounded-lg`}
      resizeMode="cover"
     />
    )}

    <View style={tw`flex-row gap-4 mt-3 flex-wrap`}>
     <TouchableOpacity
      onPress={() => handleDelete(item.id)}
```

```
            style={tw`bg-red-600 px-4 py-2 rounded-full`}
            >
              <Text style={tw`text-white font-semibold`}>Hapus</Text>
            </TouchableOpacity>

            <TouchableOpacity
              onPress={() =>
                navigation.navigate('edit Pengeluaran', { id: item.id })
              }
              style={tw`bg-blue-600 px-4 py-2 rounded-full`}
            >
              <Text style={tw`text-white font-semibold`}>Edit</Text>
            </TouchableOpacity>

            <TouchableOpacity
              onPress={() =>
                navigation.navigate('Detail Pengeluaran', { id: item.id })
              }
              style={tw`bg-green-600 px-4 py-2 rounded-full`}
            >
              <Text style={tw`text-white font-semibold`}>Detail</Text>
            </TouchableOpacity>
          </View>
        </View>
      ))
    ) : (
      <Text style={tw`text-center text-green-200`}>
        Tidak ada data pengeluaran yang cocok.
      </Text>
    )}
  </ScrollView>
 </View>
</LinearGradient>
  );
};

export default Pengeluaranindex;
import React, { useEffect, useState } from "react";
import {
 View,
 Text,
 ScrollView,
 ActivityIndicator,
 TouchableOpacity,
 Platform
} from "react-native";
import DateTimePicker from "@react-native-community/datetimepicker";
import tw from "twrnc";
import {
 fetchDefaultLaporanHarian,
 fetchDefaultLaporanBulanan,
 fetchDefaultLaporanTahunan,
 fetchLaporanHarianByTanggal,
 fetchLaporanBulananByRange,
 fetchLaporanTahunanByRange
} from "../api";
import { LinearGradient } from "expo-linear-gradient";
import { Picker } from "@react-native-picker/picker";


const Laporan = ({ navigation }) => {
 const [tab, setTab] = useState("harian");
 const [dataHarian, setDataHarian] = useState(null);
 const [dataBulanan, setDataBulanan] = useState(null);
 const [dataTahunan, setDataTahunan] = useState(null);
 const [loading, setLoading] = useState(false);
 const [isFiltered, setIsFiltered] = useState(false);
 const [showDatePicker, setShowDatePicker] = useState(false);
```

```
// FILTER STATE
const [tanggalHarian, setTanggalHarian] = useState(new Date());
const [startMonth, setStartMonth] = useState(6);
const [startYear, setStartYear] = useState(2025);
const [endMonth, setEndMonth] = useState(6);
const [endYear, setEndYear] = useState(2025);
const [startYearTahunan, setStartYearTahunan] = useState(2024);
const [endYearTahunan, setEndYearTahunan] = useState(2025);

const fetchDefaultData = async () => {
 try {
  setLoading(true);
  const [resHarian, resBulanan, resTahunan] = await Promise.all([
   fetchDefaultLaporanHarian(),
   fetchDefaultLaporanBulanan(),
   fetchDefaultLaporanTahunan()
  ]);
  setDataHarian(resHarian.data);
  setDataBulanan(resBulanan.data);
  setDataTahunan(resTahunan.data);
  setIsFiltered(false);
 } catch (err) {
  console.log(err);
  alert("Gagal mengambil data laporan");
 } finally {
  setLoading(false);
 }
};

const fetchFilteredData = async () => {
 try {
  setLoading(true);
  const [resHarian, resBulanan, resTahunan] = await Promise.all([
   fetchLaporanHarianByTanggal(tanggalHarian.toISOString().split('T')[0]),
   fetchLaporanBulananByRange(startMonth, startYear, endMonth, endYear),
   fetchLaporanTahunanByRange(startYearTahunan, endYearTahunan)
  ]);
  setDataHarian(resHarian.data);
  setDataBulanan(resBulanan.data);
  setDataTahunan(resTahunan.data);
  setIsFiltered(true);
 } catch (err) {
  console.log(err);
  alert("Gagal mengambil data laporan");
 } finally {
  setLoading(false);
 }
};

useEffect(() => {
 fetchDefaultData();
}, []);
useEffect(() => {
 if (tab === "bulanan" || tab === "tahunan") {
  fetchFilteredData();
 }
}, [tab, startMonth, startYear, endMonth, endYear, startYearTahunan, endYearTahunan]);

const renderLaporan = (data) => {
 if (!data) return <Text style={tw`text-white`}>Tidak ada data</Text>;

 return (
  <View style={tw`bg-white/10 p-4 rounded-xl mb-4`}>
   <Text style={tw`text-green-400 font-bold text-lg`}>
    Kategori: {data.kategori}
```

```
</Text>

{data.tanggal && (

 <Text style={tw`text-white`}>Tanggal: {data.tanggal}</Text>

)}

{data.periode && (

 <Text style={tw`text-white`}>Periode: {data.periode}</Text>

)}


<View style={tw`mt-2 mb-1`}>

 <Text style={tw`text-green-300 font-semibold`}>Pemasukan per layanan:</Text>

 <Text style={tw`text-white`}>

  Laundry: Rp {data.pemasukan?.laundry?.toLocaleString('id-ID')}

 </Text>

 <Text style={tw`text-white`}>

  Fotocopy: Rp {data.pemasukan?.fotocopy?.toLocaleString('id-ID')}

 </Text>

 <Text style={tw`text-white`}>

  Depot: Rp {data.pemasukan?.depot?.toLocaleString('id-ID')}

 </Text>

</View>


<Text style={tw`text-white font-semibold mt-2`}>

 Total Pemasukan: Rp {data.pemasukan?.total?.toLocaleString('id-ID')}

</Text>

<Text style={tw`text-white`}>

 Total Pengeluaran: Rp {data.total_pengeluaran?.toLocaleString('id-ID')}

</Text>

<Text style={tw`text-white`}>

 Total Keuntungan: Rp {data.total_keuntungan?.toLocaleString('id-ID')}

  </Text>

 </View>

 );

};


return (

 <LinearGradient

  colors={["rgba(0,0,0,0.9)", "rgba(30,30,50,0.95)"]}

  style={tw`flex-1`}

 >

  <View style={tw`flex-row justify-around p-4`}>

   {["harian", "bulanan", "tahunan"].map((item) => (

    <TouchableOpacity

     key={item}

     style={[

      tw`px-4 py-2 rounded-full`,

      tab === item ? tw`bg-green-500` : tw`bg-green-400/20`

     ]}

     onPress={() => setTab(item)}

    >

     <Text style={tw`text-black font-bold capitalize`}>{item}</Text>

    </TouchableOpacity>

   ))}

  </View>


  <View style={tw`px-4`}>

   {tab === "harian" && (

    <>

     <Text style={tw`text-white mb-2`}>Pilih Tanggal:</Text>


     <TouchableOpacity

      onPress={() => setShowDatePicker(true)}

      style={tw`bg-white/10 px-4 py-2 rounded-lg mb-2`}

     >

      <Text style={tw`text-white`}>
```

```
        {tanggalHarian.toLocaleDateString("id-ID")}
      </Text>
    </TouchableOpacity>

    {showDatePicker && (
      <DateTimePicker
        value={tanggalHarian}
        mode="date"
        display={Platform.OS === "ios" ? "spinner" :
          "default"}
        onChange={(event, selectedDate) => {
          if (event.type === "set" && selectedDate) {
            setTanggalHarian(selectedDate);
          }
          setShowDatePicker(false); // Tutup picker
          setelah pilih / batal
        }}
      />
    )}
  </>
)}

{tab === "bulanan" && (
  <View>
    <Text style={tw`text-white mt-2`}>Periode
      Bulan & Tahun:</Text>

    <View style={tw`flex-row gap-2`}>
      <View style={tw`flex-1`}>
        <Text style={tw`text-white text-xs`}>Mulai
          Bulan</Text>
        <Picker
          selectedValue={startMonth}
          onValueChange={(itemValue) =>
          setStartMonth(itemValue)}
          style={tw`bg-white/10 text-white`}
        >
          {[...Array(12)].map((_, i) => (
            <Picker.Item key={i} label={`${i + 1}`}
          value={i + 1} />
          ))}
        </Picker>
      </View>

      <View style={tw`flex-1`}>
        <Text style={tw`text-white text-xs`}>Mulai
          Tahun</Text>
        <Picker
          selectedValue={startYear}
          onValueChange={(itemValue) =>
          setStartYear(itemValue)}
          style={tw`bg-white/10 text-white`}
        >
          {[2023, 2024, 2025].map((year) => (
            <Picker.Item key={year} label={`${year}`}
          value={year} />
          ))}
        </Picker>
      </View>
    </View>

    <View style={tw`flex-row gap-2 mt-2`}>
      <View style={tw`flex-1`}>
        <Text style={tw`text-white text-xs`}>Akhir
          Bulan</Text>
        <Picker
          selectedValue={endMonth}
          onValueChange={(itemValue) =>
          setEndMonth(itemValue)}
          style={tw`bg-white/10 text-white`}
        >
          {[...Array(12)].map((_, i) => (
            <Picker.Item key={i} label={`${i + 1}`}
          value={i + 1} />
          ))}
        </Picker>
      </View>

      <View style={tw`flex-1`}>
        <Text style={tw`text-white text-xs`}>Akhir
          Tahun</Text>
```

```
      <Picker
        selectedValue={endYear}
        onValueChange={(itemValue) => setEndYear(itemValue)}
        style={tw`bg-white/10 text-white`}
      >
        {[2023, 2024, 2025].map((year) => (
          <Picker.Item key={year} label={`${year}`} value={year} />
        ))}
      </Picker>
    </View>
  </View>
 </View>
)}

{tab === "tahunan" && (
 <View>
  <Text style={tw`text-white mt-2`}>Periode Tahun:</Text>

  <View style={tw`flex-row gap-2`}>
   <View style={tw`flex-1`}>
    <Text style={tw`text-white text-xs`}>Tahun Awal</Text>
    <Picker
     selectedValue={startYearTahunan}
     onValueChange={(itemValue) => setStartYearTahunan(itemValue)}
     style={tw`bg-white/10 text-white`}
    >
     {[2023, 2024, 2025].map((year) => (
      <Picker.Item key={year} label={`${year}`} value={year} />
     ))}
    </Picker>
   </View>

   <View style={tw`flex-1`}>
    <Text style={tw`text-white text-xs`}>Tahun Akhir</Text>
    <Picker
     selectedValue={endYearTahunan}
     onValueChange={(itemValue) => setEndYearTahunan(itemValue)}
     style={tw`bg-white/10 text-white`}
    >
     {[2023, 2024, 2025].map((year) => (
      <Picker.Item key={year} label={`${year}`} value={year} />
     ))}
    </Picker>
   </View>
  </View>
 </View>
)}

<TouchableOpacity
 onPress={fetchFilteredData}
 style={tw`mt-4 bg-green-500 py-2 rounded-full items-center`}
>
 <Text style={tw`text-black font-bold`}>Tampilkan Laporan</Text>
</TouchableOpacity>

{isFiltered && (
 <TouchableOpacity
  onPress={fetchDefaultData}
  style={tw`mt-2 bg-green-300 py-2 rounded-full items-center`}
 >
  <Text style={tw`text-black font-bold`}>Kembali ke Data Real Time</Text>
 </TouchableOpacity>
)}
</View>

{loading ? (
 <View style={tw`flex-1 justify-center items-center`}>
```

```
        <ActivityIndicator size="large"
            color="#FACC15" />

        <Text style={tw`text-green-500 mt-2`}>Memuat
            laporan...</Text>

      </View>

    ) : (

      <ScrollView style={tw`p-4`}>

        {tab === "harian" &&
            renderLaporan(dataHarian)}

        {tab === "bulanan" &&
            renderLaporan(dataBulanan)}

        {tab === "tahunan" &&
            renderLaporan(dataTahunan)}

      </ScrollView>

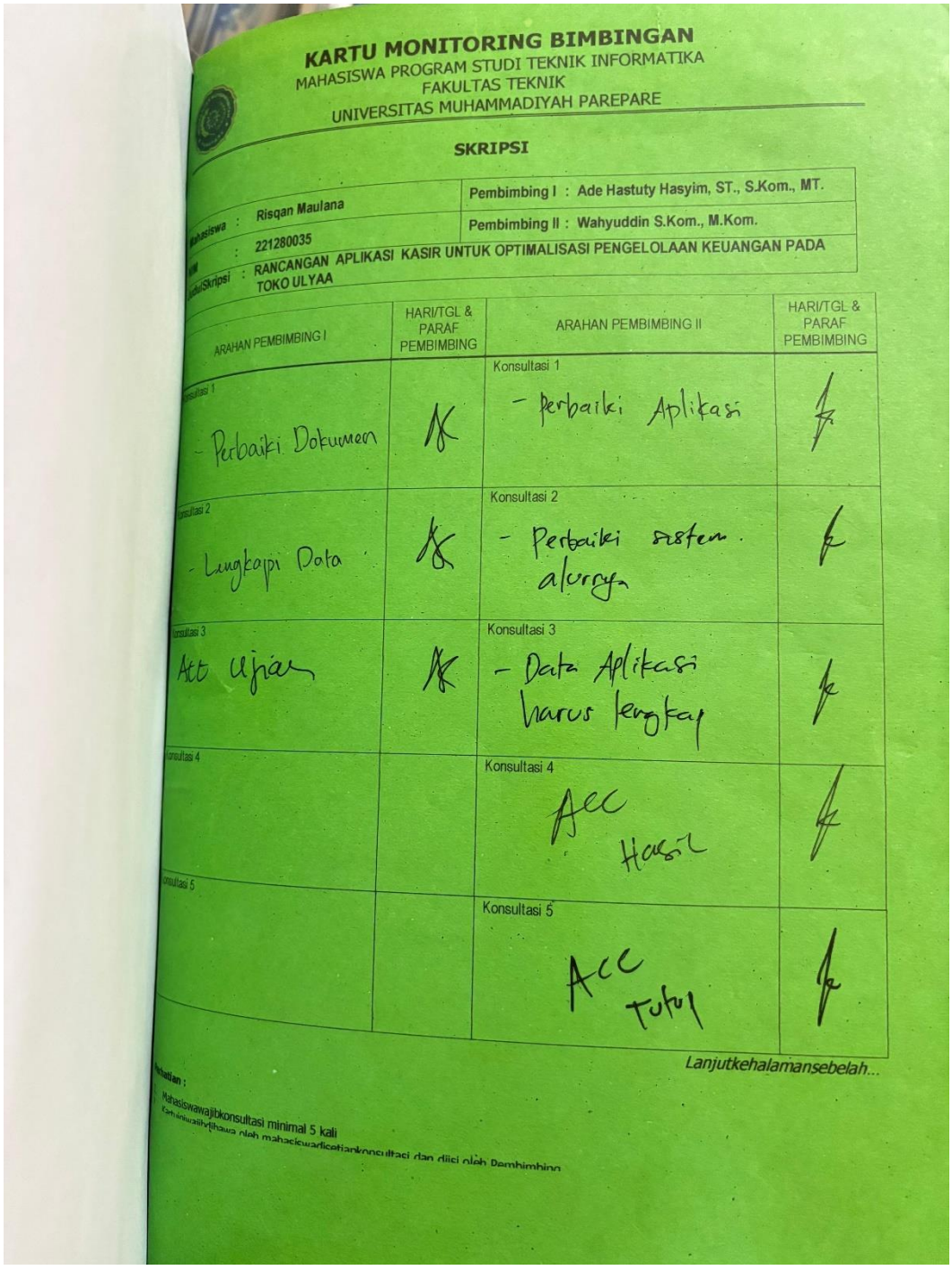
    )}

  </LinearGradient>

 );

};


export default Laporan;
```



KARTU MONITORING BIMBINGAN
MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

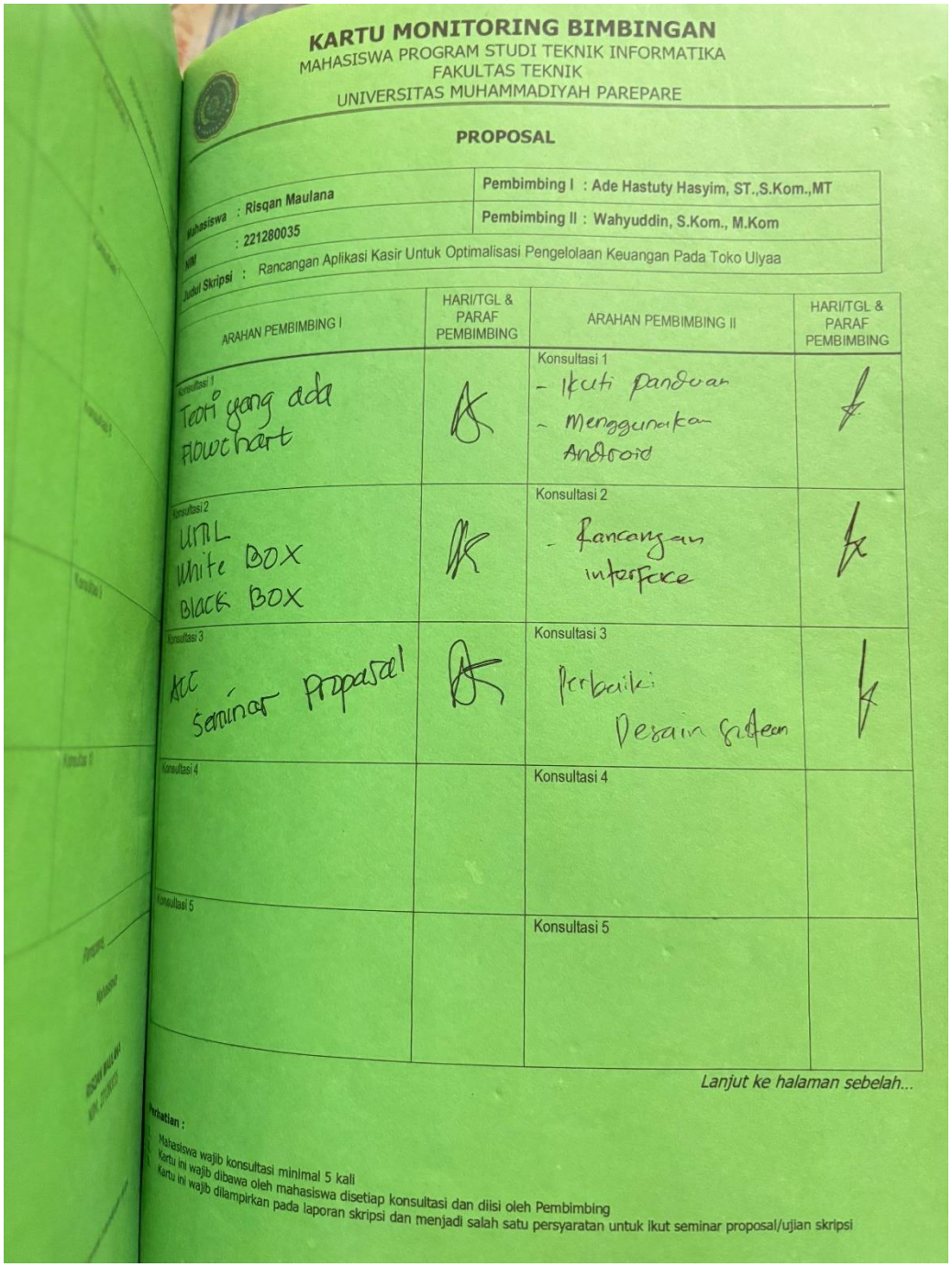
SKRIPSI

Mahasiswa : Risqan Maulana	Pembimbing I : Ade Hastuty Hasyim, ST., S.Kom., MT.
NIM : 221280035	Pembimbing II : Wahyuddin S.Kom., M.Kom.
Judul Skripsi : RANCANGAN APLIKASI KASIR UNTUK OPTIMALISASI PENGELOLAAN KEUANGAN PADA TOKO ULYAA	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 - Perbaiki Dokumen		Konsultasi 1 - perbaiki Aplikasi	
Konsultasi 2 - Lengkapi Data		Konsultasi 2 - Perbaiki sistem alurnya	
Konsultasi 3 Att Ujian		Konsultasi 3 - Data Aplikasi harus lengkap	
Konsultasi 4		Konsultasi 4 Acc Hasil	
Konsultasi 5		Konsultasi 5 Acc Tutup	

Lanjutkehalamansebelah...

Ketentuan :
1. Mahasiswawajibkonsultasi minimal 5 kali
2. Kemajuanbimbingan diisi oleh mahasiswadicetakdankonsultasi dan diisi oleh Pembimbing



KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

Mahasiswa : Risqan Maulana	Pembimbing I : Ade Hastuty Hasyim, ST.,S.Kom.,MT
NIM : 221280035	Pembimbing II : Wahyuddin, S.Kom., M.Kom
Judul Skripsi : Rancangan Aplikasi Kasir Untuk Optimalisasi Pengelolaan Keuangan Pada Toko Ulyaa	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Teori yang ada Flowchart		Konsultasi 1 - Ikuti panduan - Menggunakan Android	
Konsultasi 2 UML White BOX Black BOX		Konsultasi 2 - Rancangan interface	
Konsultasi 3 Acc Seminar Proposal		Konsultasi 3 Perbaiki: Desain sistem	
Konsultasi 4		Konsultasi 4	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi