LAMPIRAN

Lampiran - 1 Listing Program

0001_01_01_000000_create_users_table.php

```
<?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');
    }
};
```

0001_01_01_000001_create_cache_table.php

```
<?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('cache', function (Blueprint $table)
{
            $table->string('key')->primary();
            $table->mediumText('value');
            $table->integer('expiration');
        });
        Schema::create('cache_locks', function (Blueprint
$table) {
            $table->string('key')->primary();
            $table->string('owner');
            $table->integer('expiration');
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('cache');
        Schema::dropIfExists('cache_locks');
    }
};
```

0001_01_01_000002_create_jobs_table

```
<?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('jobs', function (Blueprint $table) {
            $table->id();
            $table->string('queue')->index();
            $table->longText('payload');
            $table->unsignedTinyInteger('attempts');
            $table->unsignedInteger('reserved_at')->nullable();
            $table->unsignedInteger('available_at');
            $table->unsignedInteger('created_at');
        });
        Schema::create('job_batches', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->string('name');
            $table->integer('total_jobs');
            $table->integer('pending_jobs');
            $table->integer('failed_jobs');
            $table->longText('failed_job_ids');
            $table->mediumText('options')->nullable();
            $table->integer('cancelled_at')->nullable();
            $table->integer('created_at');
            $table->integer('finished_at')->nullable();
        });
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('jobs');
        Schema::dropIfExists('job_batches');
        Schema::dropIfExists('failed_jobs');
    }
};
```

2025_07_09_160651_create_totals_table.php

```
<?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('totals', function (Blueprint $table) {
            $table->id();
            $table->integer('harga');
            $table->text('keterangan');
            $table->date('waktu');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('totals');
    }
};
```

2025_07_09_160710_create_asets_table.php

```
<?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('asets', function (Blueprint $table)
{
            $table->id();
            $table->string('nama_barang');
            $table->string('jenis_barang')->nullable();
            $table->integer('jumlah');
            $table->integer('harga');
            $table->text('keterangan');
            $table->integer('penyusutan')->nullable();
            $table->integer('pembelian')->nullable();
            $table->string('kategori');
            $table->date('waktu')->nullable();
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('asets');
    }
};
```

2025_07_14_161748_create_coas_table.php

```
<?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('coas', function (Blueprint $table) {
            $table->id();
            $table->string('kode_akun');
            $table->string('nama_akun');
            $table->string('penempatan_nilai');
            $table->text('penjelasan');
            $table->string('kategori');
            $table->integer('jumlah')->nullable();
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('coas');
    }
};
```

2025_07_14_184857_create_hutangs_table.php

```
<?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('hutangs', function (Blueprint
$table) {
            $table->id();
            $table->string('nama_hutang');
            $table->integer('jumlah_hutang');
            $table->text('keterangan');
            $table->date('waktu');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
```

```
        Schema::dropIfExists('hutangs');
    }
};
```

2025_07_15_204433_create_bebans_table.php

```
<?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('bebans', function (Blueprint
$table) {
            $table->id();
            $table->string('nama_beban');
            $table->integer('jumlah_beban');
            $table->text('keterangan');
            $table->date('waktu');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('bebans');
    }
};
```

2025_07_17_194251_create_piutangs_table.php

```
<?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('piutangs', function (Blueprint
$table) {
            $table->id();
            $table->decimal('jumlah', 15, 2);
            $table->date('jatuh_tempo');
            $table->string('keterangan')->nullable();
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('piutangs');
    }
};
```

2025_07_17_204848_create_coa_transaksistable.php

```
<?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('coa_transaksis', function
(Blueprint $table) {
            $table->id();
            $table->foreignId('coa_id')->constrained('coas')-
>onDelete('cascade');
            $table->integer('jumlah'); // nilai transaksi (+/-)
            $table->enum('jenis', ['debit', 'kredit']);
            $table->date('waktu'); // kapan terjadi
            $table->text('keterangan')->nullable(); // opsional
            $table->timestamps();
        });
    }
    /**
```

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

DatabaseSeeder.php

```
<?php
namespace Database\Seeders;
use App\Models\User;
// use
Illuminate\Database\Console\Seeds\WithoutModelEven
ts;
use Illuminate\Database\Seeder;
use Database\Seeders\CoaSeeder;
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        // User::factory(10)->create();
        // User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
        $this->call([
            CoaSeeder::class,
        ]);
    }
}
```

CoaSeeder.php

```
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;
use App\Models\Coa;
class CoaSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            // ---------- Aset Lancar ----------
            [
                'kode_akun' => '1101',
                'nama_akun' => 'Kas',
                'penempatan_nilai' => 'Debit',
                'penjelasan' => 'uang tunai atau simpanan di
bank (semua uang toko)',
                'kategori' => 'Aset Lancar',
            ],
            [
                'kode_akun' => '1102',
                'nama_akun' => 'Piutang Usaha',
                'penempatan_nilai' => 'Debit',
                'penjelasan' => 'hasil dari penjualan yang
dihutang pembeli (konsumen)',
                'kategori' => 'Aset Lancar',
            ],
            [
                'kode_akun' => '1103',
                'nama_akun' => 'Piutang karyawan',
                'penempatan_nilai' => 'Debit',
                'penjelasan' => 'pinjaman pribadi karyawan,
dipotong dari gaji saat gajian',
                'kategori' => 'Aset Lancar',
            ],
            [
                'kode_akun' => '1104',
                'nama_akun' => 'Persediaan Barang Dagang',
                'penempatan_nilai' => 'Debit',
                'penjelasan' => 'barang yang dimiliki
perusahaan dan siap dijual',
                'kategori' => 'Aset Lancar',
            ],
            [
                'kode_akun' => '1105',
                'nama_akun' => 'Perlengkapan',
                'penempatan_nilai' => 'Debit',
                'penjelasan' => 'penjelasan',
                'kategori' => 'Aset Lancar',
            ],
            // ---------- Aset Tetap ----------
            [
```

```
            'kode_akun' => '1201',
            'nama_akun' => 'Peralatan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'masa manfaat >1 tahun,
mengalami penyusutan (contoh: meja, kursi, lemari)',
            'kategori' => 'Aset Tetap',
        ],
        [
            'kode_akun' => '1202',
            'nama_akun' => 'Akumulasi Penyusutan
Peralatan',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'total nilai penyusutan
peralatan yang sudah dicatat',
            'kategori' => 'Aset Tetap',
        ],
        [
            'kode_akun' => '1203',
            'nama_akun' => 'Komputer',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'terdiri dari laptop dan CCTV',
            'kategori' => 'Aset Tetap',
        ],
        [
            'kode_akun' => '1204',
            'nama_akun' => 'Akumulasi Penyusutan
Komputer',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'total nilai penyusutan
komputer yang sudah dicatat',
            'kategori' => 'Aset Tetap',
        ],
        [
            'kode_akun' => '1205',
            'nama_akun' => 'Kendaraan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'terdiri dari mobil',
            'kategori' => 'Aset Tetap',
        ],
        [
            'kode_akun' => '1206',
            'nama_akun' => 'Akumulasi Penyusutan
Kendaraan',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => '',
            'kategori' => 'Aset Tetap',
        ],
        // ---------- Kewajiban ----------
        [
            'kode_akun' => '2101',
            'nama_akun' => 'Hutang Usaha',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'hutang jangka pendek
pembelian barang secara Kredit',
            'kategori' => 'Kewajiban',
        ],
        [
            'kode_akun' => '2102',
            'nama_akun' => 'Hutang bank',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'hutang jangka panjang ke
bank',
            'kategori' => 'Kewajiban',
        ],
        [
            'kode_akun' => '2103',
            'nama_akun' => 'Hutang Lain-lain',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'utang ke karyawan, pihak
ketiga non-bank, atau uang titipan',
            'kategori' => 'Kewajiban',
        ],
        // ---------- Ekuitas ----------
        [
            'kode_akun' => '3101',
            'nama_akun' => 'Modal Usaha',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'dana untuk memulai,
menjalankan, dan mengembangkan usaha',
            'kategori' => 'Ekuitas',
        ],
        [
            'kode_akun' => '3102',
            'nama_akun' => 'Prive',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'pengambilan sebagian modal
atau aset oleh pemilik',
            'kategori' => 'Ekuitas',
        ],
        [
```

```
            'kode_akun' => '3103',
            'nama_akun' => 'Ikhtisar laba rugi',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'juga dikenal sebagai laporan
laba rugi (tidak masuk di neraca)',
            'kategori' => 'Ekuitas',
        ],
        // ---------- Pendapatan ----------
        [
            'kode_akun' => '4101',
            'nama_akun' => 'Pendapatan Usaha',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'pendapatan utama dari
penjualan barang atau jasa',
            'kategori' => 'Pendapatan',
        ],
        [
            'kode_akun' => '4102',
            'nama_akun' => 'Pendapatan Lain-lain',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'penghasilan dari luar kegiatan
utama, misalnya bunga bank',
            'kategori' => 'Pendapatan',
        ],
        // ---------- Penjualan ----------
        [
            'kode_akun' => '5101',
            'nama_akun' => 'Penjualan',
            'penempatan_nilai' => 'Kredit',
            'penjelasan' => 'penghasilan dari penjualan
barang dagangan setiap hari',
            'kategori' => 'Penjualan',
        ],
        [
            'kode_akun' => '5102',
            'nama_akun' => 'Retur Penjualan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'pengembalian barang yang
dibeli dari distributor',
            'kategori' => 'Penjualan',
        ],
        [
            'kode_akun' => '5103',
            'nama_akun' => 'Potongan Penjualan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'diskon pembelian atau bonus
dari distributor',
            'kategori' => 'Penjualan',
        ],
        // ---------- Beban ----------
        [
            'kode_akun' => '6101',
            'nama_akun' => 'Beban Gaji',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'biaya untuk membayar
karyawan',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6102',
            'nama_akun' => 'Beban Transportasi',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'biaya terkait pengoperasian
kendaraan',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6103',
            'nama_akun' => 'Beban Listrik',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'biaya pembayaran listrik',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6104',
            'nama_akun' => 'Beban Asuransi',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'biaya untuk membayar
asuransi',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6105',
            'nama_akun' => 'Beban Penyusutan Peralatan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'nilai peralatan yang sudah
habis dipakai dalam periode',
            'kategori' => 'Beban',
        ],
        [
```

```
            'kode_akun' => '6106',
            'nama_akun' => 'Beban Penyusutan
Komputer',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'nilai komputer yang sudah
habis dipakai dalam periode',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6107',
            'nama_akun' => 'Beban Penyusutan
Kendaraan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'nilai kendaraan yang sudah
habis dipakai dalam periode',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6108',
            'nama_akun' => 'Beban Perlengkapan',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'dffhnfd',
            'kategori' => 'Beban',
        ],
        [
            'kode_akun' => '6201',
            'nama_akun' => 'Beban Lain-lain',
            'penempatan_nilai' => 'Debit',
            'penjelasan' => 'beban non-operasional,
seperti rugi penjualan aset tetap',
            'kategori' => 'Beban',
        ],
    ];
    foreach ($data as $item) {
        Coa::create($item);
    }
  }
}
```

Aset.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Aset extends Model
{
  protected $fillable = [
    'nama_barang',
    'jenis_barang',
    'jumlah',
    'harga',
    'keterangan',
    'penyusutan',
    'kategori',
    'pembelian',
    'waktu'
  ];
}
```

Beban.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Beban extends Model
{
  protected $fillable = [
    'nama_beban',
    'jumlah_beban',
    'keterangan',
    'waktu'
  ];
}
```

Coa.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Coa extends Model
{
    protected $fillable = [
    'kode_akun',
    'nama_akun',
    'penempatan_nilai',
    'penjelasan',
    'kategori',
    'jumlah'
  ];
  public function transaksis()
{
  return $this->hasMany(CoaTransaksi::class);
}
```

```
}
```

CoaTransaksi.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
use
Illuminate\Database\Eloquent\Factories\HasFactory;
class CoaTransaksi extends Model
{
    use HasFactory;
    protected $table = 'coa_transaksis';
    protected $fillable = [
        'coa_id',
        'jumlah',
        'jenis',
        'waktu',
        'keterangan',
    ];
    protected $casts = [
        'waktu' => 'date',
    ];
    public function coa()
    {
        return $this->belongsTo(Coa::class);
    }
}
```

Hutang.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Hutang extends Model
{
        protected $fillable = [
        'nama_hutang',
        'jumlah_hutang',
        'keterangan',
        'waktu'
    ];
}
```

Piutang.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Piutang extends Model
{
        protected $fillable = [
        'jumlah',
        'jatuh_tempo',
        'keterangan',
    ];
}
```

Total.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Total extends Model
{
    protected $fillable = [
        'harga',
        'keterangan',
        'waktu'
    ];
}
```

User.php

```
<?php
namespace App\Models;
// use Illuminate\Contracts\Auth\MustVerifyEmail;
use
Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as
Authenticatable;
use Illuminate\Notifications\Notifiable;
class User extends Authenticatable
{
    /** @use
HasFactory<\Database\Factories\UserFactory> */
    use HasFactory, Notifiable;
    /**
     * 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',
        ];
    }
}
```

PasswordController.php

```
<?php
namespace App\Http\Controllers\Settings;
use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Inertia\Inertia;
use Inertia\Response;
class PasswordController extends Controller
{
    /**
     * Show the user's password settings page.
     */
    public function edit(): Response
    {
        return Inertia::render('settings/password');
    }
    /**
     * Update the user's password.
     */
    public function update(Request $request):
RedirectResponse
    {
        $validated = $request->validate([
            'current_password' => ['required',
'current_password'],
            'password' => ['required', Password::defaults(),
'confirmed'],
        ]);
        $request->user()->update([
            'password' =>
Hash::make($validated['password']),
        ]);
        return back();
    }
}
```

ProfileController.php

```
<?php
namespace App\Http\Controllers\Settings;
use App\Http\Controllers\Controller;
use App\Http\Requests\Settings\ProfileUpdateRequest;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Inertia\Inertia;
use Inertia\Response;
class ProfileController extends Controller
{
    /**
     * Show the user's profile settings page.
     */
    public function edit(Request $request): Response
    {
        return Inertia::render('settings/profile', [
            'mustVerifyEmail' => $request->user()
instanceof MustVerifyEmail,
            'status' => $request->session()->get('status'),
        ]);
    }
    /**
```

```
     * Update the user's profile settings.
     */
    public function update(ProfileUpdateRequest
$request): RedirectResponse
    {
        $request->user()->fill($request->validated());
        if ($request->user()->isDirty('email')) {
            $request->user()->email_verified_at = null;
        }
        $request->user()->save();
        return to_route('profile.edit');
    }
    /**
     * Delete the user's account.
     */
    public function destroy(Request $request):
RedirectResponse
    {
        $request->validate([
            'password' => ['required', 'current_password'],
        ]);
        $user = $request->user();
        Auth::logout();
        $user->delete();
        $request->session()->invalidate();
        $request->session()->regenerateToken();
        return redirect('/');
    }
}
```

Controller.php

```
<?php
namespace App\Http\Controllers;
abstract class Controller
{
    //
}
```

AsetController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Aset;
use App\Models\Coa;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\DB;
use App\Models\CoaTransaksi;
use Carbon\Carbon;
class AsetController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $aset = Aset::all();
        return Inertia::render('aset_tetap/Index', [
            'aset' => $aset,
        ]);
    }
    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return Inertia::render('aset_tetap/Create');
    }
    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $validated = $request->validate([
            'nama_barang' => 'required|string',
            'jumlah' => 'required|integer|min:1',
            'harga' => 'required|numeric|min:0',
            'pembelian' => 'required',
            'jenis_barang' => 'required',
            'waktu' => 'required|date'
        ]);
        try {
            DB::transaction(function () use ($validated) {
                $tahunSekarang = 2024;
                // Jika jenis barang adalah Perlengkapan,
paksa umurAset = 1
                $umurAset = $validated['jenis_barang'] ===
'Perlengkapan'
                    ? 1
                    : max($tahunSekarang -
$validated['pembelian'], 1);
```

```
            // Hitung penyusutan berdasarkan umur aset
            $penyusutan = $validated['harga'] /
$umurAset;
            // Gunakan nilai penyusutan yang dihitung
            $penyusutanTercatat = $penyusutan;
            // === Tambah ke Aset Tetap ===
            $coaAset = Coa::where('nama_akun',
$validated['jenis_barang'])->first();
            $coaAset->jumlah += $validated['harga'];
            $coaAset->save();
            CoaTransaksi::create([
                'coa_id' => $coaAset->id,
                'jumlah' => $validated['harga'],
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan aset : ' .
$validated['nama_barang']
            ]);
            if ($validated['jenis_barang'] ===
'Perlengkapan') {
                // === Beban Perlengkapan ===
                $beban = Coa::where('nama_akun', 'Beban
Perlengkapan')->first();
                if (!$beban) {
                    throw new \Exception("Akun COA
'Beban Perlengkapan' tidak ditemukan");
                }
                $beban->jumlah += $validated['harga'];
                $beban->save();
                CoaTransaksi::create([
                    'coa_id' => $beban->id,
                    'jumlah' => $validated['harga'],
                    'jenis' => 'debit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Penambahan Beban
Perlengkapan: ' . $validated['nama_barang']
                ]);
            } else {
                // === Akumulasi Penyusutan ===
                $akunPenyusutan = 'Akumulasi Penyusutan
' . $validated['jenis_barang'];
                $coaPenyusutan =
Coa::where('nama_akun', $akunPenyusutan)->first();
                if (!$coaPenyusutan) {
                    throw new \Exception("Akun COA tidak
ditemukan untuk: $akunPenyusutan");
                }
                $coaPenyusutan->jumlah -=
$penyusutanTercatat;
                $coaPenyusutan->save();
                CoaTransaksi::create([
                    'coa_id' => $coaPenyusutan->id,
                    'jumlah' => -$penyusutanTercatat,
                    'jenis' => 'kredit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Akumulasi penyusutan
aset: ' . $validated['nama_barang']
                ]);
                // === Beban Penyusutan ===
                $akunBeban = 'Beban Penyusutan ' .
$validated['jenis_barang'];
                $coabeban = Coa::where('nama_akun',
$akunBeban)->first();
                if (!$coabeban) {
                    throw new \Exception("Akun COA tidak
ditemukan untuk: $akunBeban");
                }
                $coabeban->jumlah -=
$penyusutanTercatat;
                $coabeban->save();
                CoaTransaksi::create([
                    'coa_id' => $coabeban->id,
                    'jumlah' => -$penyusutanTercatat,
                    'jenis' => 'debit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Beban penyusutan aset: '
. $validated['nama_barang']
                ]);
            }
            // === Kurangi Kas ===
            $kass = Coa::where('nama_akun', 'Kas')-
>first();
            $kass->jumlah -= $validated['harga'];
            $kass->save();
            CoaTransaksi::create([
                'coa_id' => $kass->id,
                'jumlah' => -$validated['harga'],
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
```

```
                'keterangan' => 'Pengurangan Kas: ' .
$validated['nama_barang']
            ]);
            // === Simpan Aset ===
            Aset::create([
                'nama_barang' =>
$validated['nama_barang'],
                'jenis_barang' =>
$validated['jenis_barang'],
                'kategori' => $validated['jenis_barang'] ===
'Perlengkapan' ? 'Perlengkapan' : 'Aset Tetap',
                'jumlah' => $validated['jumlah'],
                'harga' => $validated['harga'],
                'pembelian' => $validated['pembelian'],
                'penyusutan' => $penyusutanTercatat,
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan aset '
            ]);
        });
        return redirect()->route('properti.index')-
>with('success', 'Data berhasil ditambahkan.');
    } catch (\Exception $e) {
        return redirect()->back()->with('error', $e-
>getMessage());
    }
  }
  /**
   * Display the specified resource.
   */
  // public function show(aset $aset)
  // {
  //     return Inertia::render('aset/Show', [
  //         'aset' => $aset
  //     ]);
  // }
  /**
   * Show the form for editing the specified resource.
   */
  public function edit(aset $aset)
  {
      return Inertia::render('aset_tetap/Edit', [
          'aset' => $aset
      ]);
  }
  /**
   * Update the specified resource in storage.
   */
  public function update(Request $request, Aset $aset)
  {
      $validated = $request->validate([
          'keterangan' => 'required',
      ]);
      $aset->update($validated);
      return redirect()->route('properti.index')-
>with('success', 'Data berhasil diupdate.');
  }
  /**
   * Remove the specified resource from storage.
   */
public function destroy(Aset $aset)
{
  try {
      DB::transaction(function () use ($aset) {
          $waktu = CoaTransaksi::latest('waktu')-
>value('waktu') ?? now()->format('Y-m-d');
          $tahunSekarang = 2024;
          // Hitung ulang umur aset dan nilai asli
          $umurAset = $aset->jenis_barang ===
'Perlengkapan'
              ? 1
              : max($tahunSekarang - (int) $aset-
>pembelian, 1);
          $penyusutan = $aset->penyusutan;
          $asli = $penyusutan * $umurAset;
          // 1. Kurangi akun Aset/Perlengkapan
          $coaAset = Coa::where('nama_akun', $aset-
>jenis_barang)->firstOrFail();
          $coaAset->jumlah -= $aset->harga;
          $coaAset->jumlah = max($coaAset->jumlah, 0);
          $coaAset->save();
          CoaTransaksi::create([
              'coa_id' => $coaAset->id,
              'jumlah' => -$aset->harga,
              'jenis' => 'kredit',
              'waktu' => $waktu,
              'keterangan' => 'Penghapusan aset: ' . $aset-
>nama_barang
          ]);
          if ($aset->kategori === 'Perlengkapan') {
              // === Balikkan Beban Perlengkapan ===
```

```
            $beban = Coa::where('nama_akun', 'Beban
Perlengkapan')->firstOrFail();
            $beban->jumlah -= $aset->harga;
            $beban->jumlah = max($beban->jumlah, 0);
            $beban->save();
            CoaTransaksi::create([
                'coa_id' => $beban->id,
                'jumlah' => -$aset->harga,
                'jenis' => 'kredit',
                'waktu' => $waktu,
                'keterangan' => 'Pengurangan Beban
Perlengkapan: ' . $aset->nama_barang
            ]);
        } else {
            // === Kembalikan Akumulasi Penyusutan
===
            $akunPenyusutan = 'Akumulasi Penyusutan ' .
$aset->jenis_barang;
            $coaPenyusutan = Coa::where('nama_akun',
$akunPenyusutan)->firstOrFail();
            $coaPenyusutan->jumlah += $penyusutan;
            $coaPenyusutan->save();
            CoaTransaksi::create([
                'coa_id' => $coaPenyusutan->id,
                'jumlah' => $penyusutan,
                'jenis' => 'kredit',
                'waktu' => $waktu,
                'keterangan' => 'Pemulihan akumulasi
penyusutan: ' . $aset->nama_barang
            ]);
            // === Kurangi Beban Penyusutan
            $namaBeban = 'Beban Penyusutan ' . $aset-
>jenis_barang;
            $coabeban = Coa::where('nama_akun',
$namaBeban)->firstOrFail();
            $coabeban->jumlah -= $penyusutan;
            $coabeban->jumlah = max($coabeban-
>jumlah, 0);
            $coabeban->save();
            CoaTransaksi::create([
                'coa_id' => $coabeban->id,
                'jumlah' => -$penyusutan,
                'jenis' => 'kredit',
                'waktu' => $waktu,
                'keterangan' => 'Pengurangan beban
penyusutan: ' . $aset->nama_barang
            ]);
        }
        // 4. Kembalikan Kas
        $kass = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
        $kass->jumlah += $asli;
        $kass->save();
        CoaTransaksi::create([
            'coa_id' => $kass->id,
            'jumlah' => $asli,
            'jenis' => 'kredit',
            'waktu' => $waktu,
            'keterangan' => 'Pengembalian kas karena
penghapusan aset: ' . $aset->nama_barang
        ]);
        // 5. Hapus aset
        $aset->delete();
    });
    return redirect()->route('properti.index')-
>with('success', 'Data berhasil dihapus.');
  } catch (\Exception $e) {
    return redirect()->back()->with('error', $e-
>getMessage());
  }
}
}
```

BebanController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Beban;
use App\Models\Coa;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\DB;
use App\Models\CoaTransaksi;
class BebanController extends Controller
{
  /**
   * Display a listing of the resource.
   */
public function index()
{
```

```
    // Ambil semua transaksi beban
    $bebanTransaksi = CoaTransaksi::with('coa')
        ->where('keterangan', 'Penambahan beban - Beban
Gaji')
        ->get();
$beban = Beban::all();
    // Ambil semua transaksi piutang
    $piutangTransaksi =
CoaTransaksi::where('keterangan', 'Penambahan
piutang karyawan - -')->get();
    // Gabungkan berdasarkan coa_id
$dataGabungan = $bebanTransaksi->values()-
>map(function ($beban, $index) use
($piutangTransaksi) {
    $piutang = $piutangTransaksi->values()-
>get($index); // Ambil sesuai urutan
    return [
        'id' => $beban->id,
        'coa_nama' => $beban->coa->nama_akun,
        'jumlah_beban' => $beban->jumlah,
        'waktu' => $beban->waktu,
        'keterangan_beban' => $beban->keterangan,
        'jumlah_piutang' => $piutang->jumlah ?? null,
        'keterangan_piutang' => $piutang->keterangan ??
null,
    ];
});
    return Inertia::render('Beban/Index', [
        'transaksiGabungan' => $dataGabungan,
        'beban' => $beban,
    ]);
}
    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return Inertia::render('Beban/Create');
    }
    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $validated = $request->validate([
            'nama_beban' => 'required|string|max:255',
            'jumlah_beban' => 'required|numeric|min:0',
            'keterangan' => 'required',
            'waktu' => 'required',
            'utang_karyawan' => 'nullable|numeric',
        ]);
        DB::transaction(function () use ($validated) {
            $utangKaryawan =
$validated['utang_karyawan'] ?? 0;
            $bebanReal = $validated['jumlah_beban'] -
$utangKaryawan;
            // Tambahkan ke akun beban (COA)
            $coa = Coa::where('nama_akun',
$validated['nama_beban'])->firstOrFail();
            $coa->jumlah += $bebanReal;
            $coa->save();
            CoaTransaksi::create([
                'coa_id' => $coa->id,
                'jumlah' => $bebanReal,
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan beban - ' .
$validated['nama_beban'],
            ]);
            // Tambahkan ke Piutang Karyawan (jika ada
utang karyawan)
            if ($utangKaryawan > 0) {
                $piutang = Coa::where('nama_akun', 'Piutang
karyawan')->firstOrFail();
                $piutang->jumlah += $utangKaryawan;
                $piutang->save();
                CoaTransaksi::create([
                    'coa_id' => $piutang->id,
                    'jumlah' => $utangKaryawan,
                    'jenis' => 'debit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Penambahan piutang
karyawan - ' . $validated['keterangan'],
                ]);
            }
            // Tambahkan ke Ikhtisar Laba Rugi
            $ikhtisar = Coa::where('nama_akun', 'Ikhtisar
laba rugi')->firstOrFail();
            $ikhtisar->jumlah +=
$validated['jumlah_beban'];
```

```
            $ikhtisar->save();
            CoaTransaksi::create([
                'coa_id' => $ikhtisar->id,
                'jumlah' => $validated['jumlah_beban'],
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan beban ke
Ikhtisar Laba Rugi - ' . $validated['keterangan'],
            ]);
            Beban::create($validated);
        });
        return redirect()->route('beban.index')-
>with('success', 'Data berhasil ditambahkan.');
    }
    /**
     * Display the specified resource.
     */
    // public function show(beban $beban)
    // {
    //     return Inertia::render('beban/Show', [
    //         'beban' => $beban
    //     ]);
    // }
    /**
     * Show the form for editing the specified resource.
     */
    public function edit(Beban $beban)
    {
        return Inertia::render('Beban/Edit', [
            'beban' => $beban
        ]);
    }
    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Beban
$beban)
    {
        $validated = $request->validate([
            'keterangan' => 'required',
        ]);
        $beban->update($validated);
        return redirect()->route('beban.index')-
>with('success', 'Data berhasil diupdate.');
    }
    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Beban $beban)
    {
        DB::transaction(function () use ($beban) {
            $waktu = now();

            $piutangCoa = Coa::where('nama_akun',
'Piutang karyawan')->firstOrFail();
            $transaksiPiutang =
CoaTransaksi::where('coa_id', $piutangCoa->id)
                ->where('keterangan', 'like', '%' . $beban-
>keterangan . '%')
                ->latest()
                ->first();
            $utangKaryawan = $transaksiPiutang?->jumlah
?? 0;
            $bebanReal = $beban->jumlah_beban -
$utangKaryawan;
            // Kurangi dari akun beban
            $coa = Coa::where('nama_akun', $beban-
>nama_beban)->firstOrFail();
            $coa->jumlah -= $bebanReal;
            if ($coa->jumlah < 0)
                $coa->jumlah = 0;
            $coa->save();
            CoaTransaksi::create([
                'coa_id' => $coa->id,
                'jumlah' => -$bebanReal,
                'jenis' => 'kredit',
                'waktu' => $waktu,
                'keterangan' => 'Penghapusan beban - ' .
$beban->keterangan,
            ]);
            // Kurangi Piutang Karyawan (jika ada)
            if ($utangKaryawan > 0) {
                $piutangCoa->jumlah -= $utangKaryawan;
                if ($piutangCoa->jumlah < 0)
                    $piutangCoa->jumlah = 0;
                $piutangCoa->save();
                CoaTransaksi::create([
                    'coa_id' => $piutangCoa->id,
                    'jumlah' => -$utangKaryawan,
                    'jenis' => 'kredit',
```

```
                'waktu' => $waktu,
                'keterangan' => 'Pengurangan piutang
karyawan - ' . $beban->keterangan,
            ]);
        }
        // Kurangi dari Ikhtisar Laba Rugi
        $ikhtisar = Coa::where('nama_akun', 'Ikhtisar
laba rugi')->firstOrFail();
        $ikhtisar->jumlah -= $beban->jumlah_beban;
        if ($ikhtisar->jumlah < 0)
            $ikhtisar->jumlah = 0;
        $ikhtisar->save();
        CoaTransaksi::create([
            'coa_id' => $ikhtisar->id,
            'jumlah' => -$beban->jumlah_beban,
            'jenis' => 'debit',
            'waktu' => $waktu,
            'keterangan' => 'Pengurangan dari Ikhtisar
Laba Rugi - ' . $beban->keterangan,
        ]);
        $beban->delete();
      });
      return redirect()->route('beban.index')-
>with('success', 'Data berhasil dihapus.');
   }
}
```

CoaController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Coa;
use App\Models\CoaTransaksi;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Carbon\Carbon;
class CoaController extends Controller
{
   /**
    * Display a listing of the resource.
    */
public function index(Request $request)
{
   $tahun = $request->input('tahun');
   $coas = Coa::with(['transaksis' => function ($query)
use ($tahun) {
      if ($tahun) {
         $query->whereYear('waktu', $tahun);
      }
   }])->get();
   // Hitung jumlah dari relasi CoaTransaksi
   foreach ($coas as $coa) {
      $coa->jumlah = $coa->transaksis->sum('jumlah');
   }
   // Total per kategori bawaan
   $totalPerKategori = $coas->groupBy('kategori')-
>map(fn($group) => $group->sum('jumlah'));
   // Kategori kustom
   $Totalaset = [
      'Kas',
      'Piutang Usaha',
      'Piutang karyawan',
      'Persediaan Barang Dagang',
      'Sewa Dibayar Dimuka',
      'Peralatan',
      'Akumulasi Penyusutan Peralatan',
      'Komputer',
      'Akumulasi Penyusutan Komputer',
      'Kendaraan',
      'Akumulasi Penyusutan Kendaraan',
      'Hutang Usaha',
      'Hutang bank',
      'Hutang Lain-lain',
      'Akumulasi Hutang',
      'Perlengkapan'
   ];
   $Ekuitas = [
      'Modal Usaha',
      'Prive',
      'Ikhtisar laba rugi',
      'Pendapatan Usaha',
      'Pendapatan Lain-lain',
      'Penjualan',
      'Retur Penjualan',
      'Potongan Penjualan',
      'Beban Penyusutan Peralatan',
      'Beban Penyusutan Komputer',
      'Beban Penyusutan Kendaraan'
   ];
// Temukan akun Kas
$kas = $coas->firstWhere('nama_akun', 'Kas');
```

```
// Hitung total awal (sebelum koreksi)
$totalTotalasetAwal = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Totalaset))-
>sum('jumlah');
$totalEkuitas = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Ekuitas))->sum('jumlah');

// Hitung selisih dan sesuaikan Kas agar seimbang
$selisih = $totalTotalasetAwal - $totalEkuitas;
if ($selisih !== 0 && $kas) {
    // Sesuaikan kas
    $kas->jumlah -= $selisih; // jika selisih positif -> kas
berkurang, jika negatif -> kas bertambah
    $kas->save();
    // Ambil waktu transaksi terakhir
    $waktuTerakhir =
CoaTransaksi::orderByDesc('waktu')->value('waktu') ??
now();
    // Tentukan jenis transaksi berdasarkan arah kas
    $jenisTransaksi = $selisih > 0 ? 'debit' : 'kredit';
    // Simpan transaksi penyesuaian ke kas
    CoaTransaksi::create([
        'coa_id' => $kas->id,
        'jumlah' => -$selisih, // simpan kebalikan karena
logika di atas sudah sesuaikan arah
        'jenis' => $jenisTransaksi,
        'waktu' => $waktuTerakhir,
        'keterangan' => 'Penyesuaian neraca ke kas',
    ]);
}
// Hitung ulang totalTotalaset setelah penyesuaian
$totalTotalaset = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Totalaset))-
>sum('jumlah');
// Filter ulang untuk tampilan
$coasTotalaset = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Totalaset))->values();
$coasEkuitas = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Ekuitas))->values();
    return Inertia::render('Coa/Index', [
        'coas' => $coas,
        'totalPerKategori' => $totalPerKategori,
        'totalTotalaset' => $totalTotalaset,
        'totalEkuitas' => $totalEkuitas,
        'coasTotalaset' => $coasTotalaset,
        'coasEkuitas' => $coasEkuitas,
        'tahun' => $tahun,
    ]);
}
    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return Inertia::render('Coa/Create');
    }
    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $validated = $request->validate([
            'kode_akun' => 'required|string|max:255',
            'nama_akun' => 'required|string|max:255',
            'penempatan_nilai' => 'nullable|string|max:255',
            'penjelasan' => 'nullable|string',
            'kategori' => 'required',
        ]);
        Coa::create($validated);
        return redirect()->route('coas.index')-
>with('success', 'Data berhasil ditambahkan.');
    }
    /**
     * Display the specified resource.
     */
    public function show(Coa $coa)
    {
        return Inertia::render('Coa/Show', [
            'coa' => $coa
        ]);
    }
    /**
     * Show the form for editing the specified resource.
     */
    public function edit(Coa $coa)
    {
        return Inertia::render('Coa/Edit', [
            'coa' => $coa
        ]);
    }
```

```
    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Coa $coa)
    {
        $validated = $request->validate([
            'kode_akun' => 'required|string|max:255',
            'nama_akun' => 'required|string|max:255',
            'penempatan_nilai' => 'nullable|string|max:255',
            'penjelasan' => 'nullable|string',
            'kategori' => 'required',
            'jumlah' => 'required'
        ]);
        $coa->update($validated);
        return redirect()->route('coas.index')->with('success', 'Data berhasil diupdate.');
    }
    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Coa $coa)
    {
        $coa->delete();
        return redirect()->route('coas.index')->with('success', 'Data berhasil dihapus.');
    }
}
```

HutangController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Hutang;
use App\Models\Coa;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\DB;
use App\Models\CoaTransaksi;
class HutangController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $hutang = Hutang::all();
        return Inertia::render('Hutang/Index', [
            'hutang' => $hutang,
        ]);
    }
    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        return Inertia::render('Hutang/Create');
    }
    /**
     * Store a newly created resource in storage.
     */
public function store(Request $request)
{
    $validated = $request->validate([
        'nama_hutang'    => 'required|string|max:255',
        'jumlah_hutang'  => 'required|numeric|min:0',
        'keterangan'     => 'required',
        'waktu'          => 'required'
    ]);
    try {
        DB::transaction(function () use ($validated) {
            $coa = Coa::where('nama_akun', $validated['nama_hutang'])->firstOrFail();
            $coa->jumlah += $validated['jumlah_hutang'];
            $coa->save();
            // Catat ke coa_transaksis
            CoaTransaksi::create([
                'coa_id' => $coa->id,
                'jumlah' => $validated['jumlah_hutang'],
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan hutang - ' . $validated['keterangan'],
            ]);
            $kas = Coa::where('nama_akun', 'Kas')->firstOrFail();
            $kas->jumlah += $validated['jumlah_hutang'];
            $kas->save();
            // Catat ke coa_transaksis
            CoaTransaksi::create([
                'coa_id' => $kas->id,
                'jumlah' => $validated['jumlah_hutang'],
```

```
            'jenis' => 'kredit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Penambahan Kas - ' .
$validated['keterangan'],
        ]);
        $kas =
        // Simpan hutang
        Hutang::create($validated);
    });
    return redirect()->route('hutang.index')-
>with('success', 'Data berhasil ditambahkan.');
  } catch (\Exception $e) {
    return redirect()->back()->with('error', $e-
>getMessage());
  }
}
  /**
   * Display the specified resource.
   */
  // public function show(Hutang $hutang)
  // {
  //     return Inertia::render('Hutang/Show', [
  //         'hutang' => $hutang
  //     ]);
  // }
  /**
   * Show the form for editing the specified resource.
   */
  public function edit(Hutang $hutang)
  {
    return Inertia::render('Hutang/Edit', [
      'hutang' => $hutang
    ]);
  }
  /**
   * Update the specified resource in storage.
   */
  public function update(Request $request, Hutang
$hutang)
  {
    $validated = $request->validate([
      'keterangan' => 'required',
    ]);
    $hutang->update($validated);
    return redirect()->route('hutang.index')-
>with('success', 'Data berhasil diupdate.');
  }
  /**
   * Remove the specified resource from storage.
   */
public function destroy(Request $request, Hutang
$hutang)
{
  $request->validate([
    'jumlah_hutang_dihapus' =>
'required|numeric|min:1',
    'waktu' => 'required'
  ]);
  $jumlahInput = $request-
>input('jumlah_hutang_dihapus');
  $waktu = $request->input('waktu'); // Gunakan waktu
sekarang untuk transaksi
  if ($jumlahInput > $hutang->jumlah_hutang) {
    return redirect()->back()->with('error', 'Jumlah
yang dimasukkan melebihi total hutang.');
  }
  DB::transaction(function () use ($hutang,
$jumlahInput, $waktu) {
    $coa = Coa::where('nama_akun', $hutang-
>nama_hutang)->firstOrFail();
    $coa->jumlah -= $jumlahInput;
    if ($coa->jumlah < 0) $coa->jumlah = 0;
    $coa->save();
    // Catat ke coa_transaksis
    CoaTransaksi::create([
      'coa_id' => $coa->id,
      'jumlah' => -$jumlahInput,
      'jenis' => 'debit',
      'waktu' => $waktu,
      'keterangan' => 'Pelunasan hutang
sebagian/penuh: ' . $hutang->keterangan,
    ]);
    $kas = Coa::where('nama_akun', 'Kas')->first();
    $kas->jumlah -= $jumlahInput;
    if ($kas->jumlah < 0) $kas->jumlah = 0;
    $kas->save();
    // Catat ke coa_transaksis
    CoaTransaksi::create([
      'coa_id' => $kas->id,
```

```
            'jumlah' => -$jumlahInput,
            'jenis' => 'debit',
            'waktu' => $waktu,
            'keterangan' => 'Pelunasan hutang
sebagian/penuh: ' . $hutang->keterangan,
        ]);
        if ($jumlahInput == $hutang->jumlah_hutang) {
            $hutang->delete();
        } else {
            $hutang->jumlah_hutang -= $jumlahInput;
            $hutang->save();
        }
    });
    return redirect()->route('hutang.index')-
>with('success', 'Data hutang berhasil diperbarui.');
}
}
```

ImportController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Aset;
use App\Models\Total;
use App\Models\Coa;
use App\Models\CoaTransaksi;
use App\Models\Piutang;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\DB;
class ImportController extends Controller
{
    public function halamanUploadTambahBarang()
    {
        return
Inertia::render('Aset/UploadTambahBarang');
    }
    public function importTambahi(Request $request)
    {
        \Log::info('Import request diterima', [
            'items_count' => count($request->items ?? []),
            'first_item' => $request->items[0] ?? null
        ]);
        $request->validate([
            'items' => 'required|array|min:1',
        ]);
        $errors = [];
        foreach ($request->items as $index => $item) {
            try {
                // Validasi tiap item
                $validated = validator($item, [
                    'waktu' => 'required|date',
                    'nama_barang' => 'required|string',
                    'jumlah' => 'required|integer|min:1',
                    'harga' => 'required|numeric',
                    'harga_akhir' => 'nullable|numeric',
                    'bonus' => 'nullable|numeric',
                    'keterangan_bayar_nanti' => 'nullable',
                ])->validate();
                // Jalankan logika utama
                $this->tambahiFromArray($validated);
            } catch
(\Illuminate\Validation\ValidationException $e) {
                $baris = $index + 2; // Baris ke-2 di Excel
adalah data pertama
                $pesan = "Kesalahan pada baris ke-
{$baris}:\n";
                foreach ($e->errors() as $field => $messages)
{
                    $pesan .= "- {$field}: " . implode(', ',
$messages) . "\n";
                }
                $errors[] = trim($pesan);
            } catch (\Throwable $e) {
                $baris = $index + 2;
                $errors[] = "Kesalahan pada baris ke-
{$baris}: " . $e->getMessage();
            }
        }
        if (!empty($errors)) {
            return redirect()->back()->with('error',
implode("\n\n", $errors));
        }
        return redirect()->route('aset.index')-
>with('success', 'Data berhasil ditambahkan.');
    }
    protected function tambahiFromArray(array
$validated)
    {
        DB::transaction(function () use ($validated) {
```

```
        $subtotal_normal = $validated['jumlah'] *
$validated['harga'];
        $subtotal_diskon = $validated['jumlah'] *
$validated['harga_akhir'] + $validated['bonus'];
        $total_diskon = $subtotal_normal -
$subtotal_diskon;
        $potongann = $subtotal_diskon + ($total_diskon
* 2);
        $kas = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
        if ($kas->jumlah < $subtotal_diskon &&
empty($validated['keterangan_bayar_nanti'])) {
            throw new \Exception('Saldo anda tidak
cukup');
        }
        $aset = Aset::where('nama_barang',
$validated['nama_barang'])->first();
        if ($aset) {
            $aset->jumlah += $validated['jumlah'];
            $aset->harga = $validated['harga'];
            $aset->waktu = $validated['waktu'];
            $aset->keterangan = 'pembelian';
            $aset->save();
        } else {
            Aset::create([
                'nama_barang' =>
$validated['nama_barang'],
                'kategori' => 'Aset lancar',
                'jumlah' => $validated['jumlah'],
                'harga' => $validated['harga'],
                'waktu' => $validated['waktu'],
                'keterangan' => 'pembelian'
            ]);
        }
        if
(!empty($validated['keterangan_bayar_nanti'])) {
            $hutang = Coa::where('nama_akun', 'Hutang
Usaha')->firstOrFail();
            $hutang->jumlah += $subtotal_normal;
            $hutang->save();
            CoaTransaksi::create([
                'coa_id' => $hutang->id,
                'jumlah' => $subtotal_normal,
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan hutang
pembelian',
            ]);
        } else {
            $kas->jumlah -= $subtotal_normal;
            $kas->save();
            CoaTransaksi::create([
                'coa_id' => $kas->id,
                'jumlah' => -$subtotal_normal,
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Pengeluaran kas untuk
pembelian aset --',
            ]);
            Total::create([
                'harga' => -$subtotal_normal,
                'keterangan' => 'pembelian',
                'waktu' => $validated['waktu'],
            ]);
            $persediaan = Coa::where('nama_akun',
'Persediaan Barang Dagang')->firstOrFail();
            $persediaan->jumlah += $potongann;
            $persediaan->save();
            CoaTransaksi::create([
                'coa_id' => $persediaan->id,
                'jumlah' => $potongann,
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan persediaan
dari pembelian',
            ]);
        }
        $potongan = Coa::where('nama_akun',
'Potongan Penjualan')->first();
        if ($potongan) {
            $potongan->jumlah += $total_diskon;
            $potongan->save();
            CoaTransaksi::create([
                'coa_id' => $potongan->id,
                'jumlah' => $total_diskon,
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Diskon pembelian aset',
            ]);
        }
```

```
        });
    }
    public function halamanUploadKurangBarang()
    {
        return
Inertia::render('Aset/UploadKurangBarang');
    }
    public function importKurangi(Request $request)
    {
        \Log::info('Import kurangi request diterima', [
            'items_count' => count($request->items ?? []),
            'first_item' => $request->items[0] ?? null
        ]);
        $request->validate([
            'items' => 'required|array|min:1',
        ]);
        $errors = [];
        foreach ($request->items as $index => $item) {
            try {
                $validated = validator($item, [
                    'waktu' => 'required|date',
                    'nama_barang' => 'required|string',
                    'jumlah' => 'required|integer|min:1',
                    'harga' => 'required|numeric',
                    'keterangan_bayar_nanti' => 'nullable',
                ])->validate();
                $this->kurangiFromArray($validated);
            } catch
(\Illuminate\Validation\ValidationException $e) {
                $baris = $index + 2;
                $pesan = "Kesalahan pada baris ke-
{$baris}:\n";
                foreach ($e->errors() as $field => $messages)
{
                    $pesan .= "- {$field}: " . implode(', ',
$messages) . "\n";
                }
                $errors[] = trim($pesan);
            } catch (\Throwable $e) {
                $baris = $index + 2;
                $errors[] = "Kesalahan pada baris ke-
{$baris}: " . $e->getMessage();
            }
        }
        if (!empty($errors)) {
            return redirect()->back()->with('error',
implode("\n\n", $errors));
        }
        return redirect()->route('aset.index')-
>with('success', 'Data penjualan berhasil diimpor.');
    }
    protected function kurangiFromArray(array
$validated): void
    {
        DB::transaction(function () use ($validated) {
            $sub = $validated['jumlah'] *
$validated['harga'];
            $subtotal = $sub + ($sub * 0.10);
            $aset = Aset::where('nama_barang',
$validated['nama_barang'])->firstOrFail();
            $kurangi = $aset->harga;
            $kurang = $kurangi * $validated['jumlah'];
            if ($aset->jumlah < $validated['jumlah']) {
                throw new \Exception('Aset anda tidak
cukup');
            }
            // Persediaan
            $persediaan = Coa::where('nama_akun',
'Persediaan Barang Dagang')->firstOrFail();
            // Kurangi jumlah aset
            $aset->jumlah -= $validated['jumlah'];
            $aset->keterangan = 'penjualan';
            $aset->save();
            // Kurangi persediaan
            $persediaan->jumlah -= $kurang;
            $persediaan->save();
            CoaTransaksi::create([
                'coa_id' => $persediaan->id,
                'jumlah' => -$kurang,
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Pengurangan persediaan
karena penjualan --',
            ]);
            if
(!empty($validated['keterangan_bayar_nanti'])) {
                // Tambahkan Piutang
                $piutang = Coa::where('nama_akun', 'Piutang
Usaha')->firstOrFail();
                $piutang->jumlah += $subtotal;
```

```
            $piutang->save();
            CoaTransaksi::create([
                'coa_id' => $piutang->id,
                'jumlah' => $subtotal,
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Piutang dari penjualan',
            ]);
            Piutang::create([
                'jumlah' => $subtotal,
                'jatuh_tempo' => now()->addDays((int)
$validated['keterangan_bayar_nanti']),
                'keterangan' => 'Piutang penjualan',
            ]);
        } else {
            // Tambahkan Kas
            $kas = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
            $kas->jumlah += $subtotal;
            $kas->save();
            CoaTransaksi::create([
                'coa_id' => $kas->id,
                'jumlah' => $subtotal,
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Kas masuk dari penjualan',
            ]);
            Total::create([
                'harga' => $subtotal,
                'keterangan' => 'penjualan',
                'waktu' => $validated['waktu'],
            ]);
        }
        // Penjualan
        $penjualan = Coa::where('nama_akun',
'Penjualan')->first();
        $penjualan->jumlah += $subtotal;
        $penjualan->save();
        CoaTransaksi::create([
            'coa_id' => $penjualan->id,
            'jumlah' => $subtotal,
            'jenis' => 'kredit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Penambahan akun penjualan',
        ]);
        // Pendapatan Usaha
        $usaha = Coa::where('nama_akun', 'Pendapatan
Usaha')->first();
        $usaha->jumlah += $subtotal;
        $usaha->save();
        CoaTransaksi::create([
            'coa_id' => $usaha->id,
            'jumlah' => $subtotal,
            'jenis' => 'kredit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Pendapatan usaha dari
penjualan',
        ]);
    });
  }
}
```

LabaController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Aset;
use App\Models\Coa;
use App\Models\CoaTransaksi;
use App\Models\Total;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Barryvdh\DomPDF\Facade\Pdf;
class LabaController extends Controller
{
  public function index(Request $request)
  {
    $bulan = $request->input('bulan');
    $tahun = $request->input('tahun');
    $coas = Coa::with([
      'transaksis' => function ($query) use ($bulan,
$tahun) {
        if ($bulan) {
          $query->whereMonth('waktu', $bulan);
        }
        if ($tahun) {
          $query->whereYear('waktu', $tahun);
        }
      }
    ])->get();
```

```
        // Hitung jumlah dari transaksi COA yang sudah difilter
        $coasFormatted = $coas->map(function ($coa) {
            return [
                'id' => $coa->id,
                'nama_akun' => $coa->nama_akun,
                'jumlah' => $coa->transaksis->sum('jumlah'),
            ];
        });
        $totalJumlahPembelianAset = abs(
            CoaTransaksi::where('keterangan', 'Pengeluaran kas untuk pembelian aset --')
                ->when($bulan, fn($q) => $q->whereMonth('waktu', $bulan))
                ->when($tahun, fn($q) => $q->whereYear('waktu', $tahun))
                ->get()
                ->sum('jumlah')
        );
        return Inertia::render('Laba/index', [
            'coas' => $coasFormatted,
            'totalPembelian' => $totalJumlahPembelianAset,
            'bulan' => $bulan,
            'tahun' => $tahun,
        ]);
    }
    public function exportPdf(Request $request)
    {
        $bulan = $request->input('bulan');
        $tahun = $request->input('tahun');
        $coas = Coa::with([
            'transaksis' => function ($query) use ($bulan, $tahun) {
                if ($bulan) {
                    $query->whereMonth('waktu', $bulan);
                }
                if ($tahun) {
                    $query->whereYear('waktu', $tahun);
                }
            }
        ])->get();
        // Hitung jumlah dari transaksi COA yang sudah difilter
        $coasFormatted = $coas->map(function ($coa) {
            return (object) [
                'id' => $coa->id,
                'nama_akun' => $coa->nama_akun,
                'jumlah' => $coa->transaksis->sum('jumlah'),
            ];
        });
        $totals = abs(
            CoaTransaksi::where('keterangan', 'Pengeluaran kas untuk pembelian aset --')
                ->when($bulan, fn($q) => $q->whereMonth('waktu', $bulan))
                ->when($tahun, fn($q) => $q->whereYear('waktu', $tahun))
                ->get()
                ->sum('jumlah')
        );
        $selectedAccounts = [
            'Penjualan',
            'Potongan Penjualan',
            'Beban Gaji',
            'Beban Transportasi',
            'Beban Listrik',
            'Beban Asuransi',
            'Beban Perlengkapan',
            'Beban Lain-lain',
        ];
        $getJumlah = fn($nama) => optional($coasFormatted->firstWhere('nama_akun', $nama))->jumlah ?? 0;
        $pendapatan = $getJumlah('Penjualan');
        $potongan = $getJumlah('Potongan Penjualan');
        $jumlahPendapatan = $pendapatan + $potongan ;
        $pembelianTotal = $totals;
        $labaKotor = $jumlahPendapatan - $pembelianTotal;
        $totalBeban = collect($selectedAccounts)
            ->filter(fn($a) => str_starts_with($a, 'Beban'))
            ->sum(fn($a) => $getJumlah($a));
        $labaSebelumPajak = $labaKotor - $totalBeban;
        $pajak = $pembelianTotal * 0.10;
        $pajakPPN = $pembelianTotal / 11;
        $pajakPenghasilan = $getJumlah('Beban Gaji') * 0.02;
        $labaBersih = $labaSebelumPajak - $pajakPPN - $pajakPenghasilan;
```

```
        $filteredCoas = $coasFormatted->filter(fn($coa)
=> in_array($coa->nama_akun, $selectedAccounts));
        $pdf = Pdf::loadView('laporan-laba', compact(
            'filteredCoas',
            'pembelianTotal',
            'labaKotor',
            'labaSebelumPajak',
            'pajak',
            'pajakPPN',
            'pajakPenghasilan',
            'labaBersih',
            'jumlahPendapatan',
            'totalBeban',
            'bulan',
            'tahun'
        ));
        return $pdf->stream('laporan-laba-rugi.pdf');
    }
    public function exportNeracaPdf()
    {
        $tahun = request('tahun');
        $coas = Coa::with([
            'transaksis' => function ($query) use ($tahun) {
                if ($tahun) {
                    $query->whereYear('waktu', $tahun);
                }
            }
        ])->get();
        // Hitung jumlah transaksi
        foreach ($coas as $coa) {
            $coa->jumlah = $coa->transaksis-
>sum('jumlah');
        }
        // Daftar akun sesuai kategori custom
        $Totalaset = [
            'Kas',
            'Piutang Usaha',
            'Piutang karyawan',
            'Persediaan Barang Dagang',
            'Sewa Dibayar Dimuka',
            'Peralatan',
            'Akumulasi Penyusutan Peralatan',
            'Komputer',
            'Akumulasi Penyusutan Komputer',
            'Kendaraan',
            'Akumulasi Penyusutan Kendaraan',
            'Hutang Usaha',
            'Hutang bank',
            'Hutang Lain-lain',
            'Akumulasi Hutang',
            'Perlengkapan'
        ];
        $Ekuitas = [
            'Modal Usaha',
            'Prive',
            'Ikhtisar laba rugi',
            'Pendapatan Usaha',
            'Pendapatan Lain-lain',
            'Penjualan',
            'Retur Penjualan',
            'Potongan Penjualan',
            'Beban Penyusutan Peralatan',
            'Beban Penyusutan Komputer',
            'Beban Penyusutan Kendaraan'
        ];
        // Temukan akun Kas
        $kas = $coas->firstWhere('nama_akun', 'Kas');
        // Hitung total sebelum penyesuaian
        $totalTotalasetAwal = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Totalaset))-
>sum('jumlah');
        $totalEkuitas = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Ekuitas))->sum('jumlah');
        $selisih = $totalTotalasetAwal - $totalEkuitas;
        // Koreksi kas jika tidak seimbang
        if ($selisih !== 0 && $kas) {
            $kas->jumlah -= $selisih;
        }
        // Hitung ulang total aset setelah penyesuaian
        $totalTotalaset = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Totalaset))-
>sum('jumlah');
        // Ambil data akun untuk masing-masing
kelompok
        $coasTotalaset = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Totalaset))->values();
        $coasEkuitas = $coas->filter(fn($coa) =>
in_array($coa->nama_akun, $Ekuitas))->values();
        return Pdf::loadView('neraca', compact(
            'tahun',
```

```
            'coasTotalaset',
            'coasEkuitas',
            'totalTotalaset',
            'totalEkuitas'
        ))->stream('laporan-neraca.pdf');
    }
}
```

ModalController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Aset;
use App\Models\Total;
use App\Models\Coa;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\DB;
use App\Models\CoaTransaksi;
class ModalController extends Controller
{
public function formModal()
{
    $coas = Coa::all();
    $totals = Total::all();
    return Inertia::render('Modal/index', [
        'coas' => $coas,
        'totals' => $totals,
    ]);
}
public function tambahTotal(Request $request)
{
    $validated = $request->validate([
        'harga' => 'required|numeric|min:0',
        'keterangan' => 'required|string',
        'waktu' => 'required',
    ]);
    DB::transaction(function () use ($validated) {
        // Tambahkan ke tabel Total
        Total::create([
            'harga' => $validated['harga'],
            'keterangan' => $validated['keterangan'],
            'waktu' => $validated['waktu'],
        ]);
        $kas = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
        $modal = Coa::where('nama_akun', 'Modal
Usaha')->firstOrFail();
        // Update nilai COA
        $kas->jumlah += $validated['harga'];
        $kas->save();
        $modal->jumlah += $validated['harga'];
        $modal->save();
        // Catat transaksi COA
        CoaTransaksi::create([
            'coa_id' => $kas->id,
            'jumlah' => $validated['harga'],
            'jenis' => 'debit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Modal masuk - ' .
$validated['keterangan'],
        ]);
        CoaTransaksi::create([
            'coa_id' => $modal->id,
            'jumlah' => $validated['harga'],
            'jenis' => 'kredit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Penambahan Modal Usaha - ' .
$validated['keterangan'],
        ]);
    });
    return redirect()->back()->with('success', 'Total
berhasil ditambahkan dan Coa diperbarui.');
}
public function kurangiTotal(Request $request)
{
    $validated = $request->validate([
        'harga' => 'required|numeric|min:0',
        'keterangan' => 'required|string',
        'waktu' => 'required',
    ]);
    DB::transaction(function () use ($validated) {
        Total::create([
            'harga' => -$validated['harga'] ,
            'keterangan' => $validated['keterangan'],
            'waktu' => $validated['waktu'],
        ]);
        $prive = Coa::where('nama_akun', 'Prive')-
>firstOrFail();
        $kas = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
```

```
        $modal = Coa::where('nama_akun', 'Modal
Usaha')->firstOrFail();
        if ($kas->jumlah < $validated['harga']) {
            throw new \Exception('Saldo Kas tidak
mencukupi.');
        }
        // Update nilai COA
        $prive->jumlah += $validated['harga'];
        $prive->save();
        $kas->jumlah -= $validated['harga'];
        $kas->save();
        $modal->jumlah -= $validated['harga'] * 2;
        $modal->save();
        // Catat transaksi COA
        CoaTransaksi::create([
            'coa_id' => $prive->id,
            'jumlah' => $validated['harga'],
            'jenis' => 'debit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Penarikan Prive - ' .
$validated['keterangan'],
        ]);
        CoaTransaksi::create([
            'coa_id' => $kas->id,
            'jumlah' =>  -$validated['harga'],
            'jenis' => 'kredit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Pengurangan Kas karena Prive -
' . $validated['keterangan'],
        ]);
            CoaTransaksi::create([
            'coa_id' => $modal->id,
            'jumlah' => - $validated['harga'] *2,
            'jenis' => 'kredit',
            'waktu' => $validated['waktu'],
            'keterangan' => 'Pengurangan Modal karena
Prive - ' . $validated['keterangan'],
        ]);
    });
    return redirect()->back()->with('success', 'Total
berhasil dikurangi dan Coa diperbarui.');
}
}
```

TotalController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Aset;
use App\Models\Total;
use App\Models\Coa;
use App\Models\CoaTransaksi;
use App\Models\Piutang;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\DB;
class TotalController extends Controller
{
    // Menampilkan halaman aset
    public function index()
    {
        $this->transferPiutangKeKas(); // <- panggil di
awal
        $asets = Aset::all();
        $totals = Total::all();
        return Inertia::render('Aset/index', [
            'asets' => $asets,
            'totals' => $totals,
        ]);
    }
    public function tambahi(Request $request)
    {
        $validated = $request->validate([
            'nama_barang' => 'required|string',
            'jumlah' => 'required|integer|min:1',
            'harga' => 'required|numeric|min:0',
            'harga_akhir' => 'required|numeric|min:0',
            'bonus' => 'required',
            'waktu' => 'required|date',
            'keterangan_bayar_nanti' => 'nullable',
        ]);
        DB::transaction(function () use ($validated) {
            $subtotal_normal = $validated['jumlah'] *
$validated['harga'];
            $subtotal_diskon = $validated['jumlah'] *
$validated['harga_akhir'] + $validated['bonus'];
            $total_diskon = $subtotal_normal -
$subtotal_diskon;
            $potongann = $subtotal_diskon + ($total_diskon
* 2);
```

```
            $kas = Coa::where('nama_akun', 'Kas')->firstOrFail();
            if ($kas->jumlah < $subtotal_diskon) {
                throw new \Exception('Saldo anda tidak cukup');
            }
            // Update atau buat Aset
            $aset = Aset::where('nama_barang', $validated['nama_barang'])->first();
            if ($aset) {
                $aset->jumlah += $validated['jumlah'];
                $aset->harga = $validated['harga'];
                $aset->waktu = $validated['waktu'];
                $aset->keterangan = 'pembelian';
                $aset->save();
            } else {
                Aset::create([
                    'nama_barang' => $validated['nama_barang'],
                    'kategori' => 'Aset lancar',
                    'jumlah' => $validated['jumlah'],
                    'harga' => $validated['harga'],
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'pembelian'
                ]);
            }
            if (!empty($validated['keterangan_bayar_nanti'])) {
                // Tambah Hutang
                $hutang = Coa::where('nama_akun', 'Hutang Usaha')->firstOrFail();
                $hutang->jumlah += $subtotal_normal;
                $hutang->save();
                CoaTransaksi::create([
                    'coa_id' => $hutang->id,
                    'jumlah' => $subtotal_normal,
                    'jenis' => 'kredit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Penambahan hutang pembelian',
                ]);
            } else {
                // Kurangi Kas
                $kas->jumlah -= $subtotal_normal;
                $kas->save();
                CoaTransaksi::create([
                    'coa_id' => $kas->id,
                    'jumlah' => -$subtotal_normal,
                    'jenis' => 'kredit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Pengeluaran kas untuk pembelian aset --',
                ]);
                // Total ringkasan
                Total::create([
                    'harga' => -$subtotal_normal,
                    'keterangan' => 'pembelian',
                    'waktu' => $validated['waktu'],
                ]);
            // Tambah Persediaan
            $persediaan = Coa::where('nama_akun', 'Persediaan Barang Dagang')->firstOrFail();
            $persediaan->jumlah += $potongann;
            $persediaan->save();
            CoaTransaksi::create([
                'coa_id' => $persediaan->id,
                'jumlah' => $potongann,
                'jenis' => 'debit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan persediaan dari pembelian',
            ]);
            }
            // Tambah Potongan Penjualan
            $potongan = Coa::where('nama_akun', 'Potongan Penjualan')->first();
            if ($potongan) {
                $potongan->jumlah += $total_diskon;
                $potongan->save();
                CoaTransaksi::create([
                    'coa_id' => $potongan->id,
                    'jumlah' => $total_diskon,
                    'jenis' => 'debit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Diskon pembelian aset',
                ]);
            }
        });
        return redirect()->back()->with('success', 'Aset berhasil ditambah.');
```

```
    }
    public function kurangi(Request $request)
    {
        $validated = $request->validate([
            'nama_barang' => 'required|string',
            'jumlah' => 'required|integer|min:1',
            'harga' => 'required|numeric|min:0',
            'waktu' => 'required|date',
            'keterangan_bayar_nanti' => 'nullable',
        ]);
        DB::transaction(function () use ($validated) {
            $sub = $validated['jumlah'] *
$validated['harga'];
            $subtotal = $sub + ($sub * 0.10);
            $aset = Aset::where('nama_barang',
$validated['nama_barang'])->firstOrFail();
            $kurangi = $aset->harga;
            $kurang = $kurangi * $validated['jumlah'];
            if ($aset->jumlah < $validated['jumlah']) {
                throw new \Exception('Aset anda tidak
cukup');
            }
            // Persediaan
            $persediaan = Coa::where('nama_akun',
'Persediaan Barang Dagang')->firstOrFail();
            // Kurangi jumlah aset
            $aset->jumlah -= $validated['jumlah'];
            $aset->keterangan = 'penjualan';
            $aset->save();
            // Kurangi persediaan
            $persediaan->jumlah -= $kurang;
            $persediaan->save();
            CoaTransaksi::create([
                'coa_id' => $persediaan->id,
                'jumlah' => -$kurang,
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Pengurangan persediaan
karena penjualan --',
            ]);
            // Kas atau Piutang
            if
(!empty($validated['keterangan_bayar_nanti'])) {
                // Tambahkan Piutang
                $piutang = Coa::where('nama_akun', 'Piutang
Usaha')->firstOrFail();
                $piutang->jumlah += $subtotal;
                $piutang->save();
                CoaTransaksi::create([
                    'coa_id' => $piutang->id,
                    'jumlah' => $subtotal,
                    'jenis' => 'debit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Piutang dari penjualan',
                ]);
                Piutang::create([
                    'jumlah' => $subtotal,
                    'jatuh_tempo' => now()->addDays((int)
$validated['keterangan_bayar_nanti']),
                    'keterangan' => 'Piutang penjualan',
                ]);
            } else {
                // Tambahkan Kas
                $kas = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
                $kas->jumlah += $subtotal;
                $kas->save();
                CoaTransaksi::create([
                    'coa_id' => $kas->id,
                    'jumlah' => $subtotal,
                    'jenis' => 'debit',
                    'waktu' => $validated['waktu'],
                    'keterangan' => 'Kas masuk dari penjualan',
                ]);
                // Total ringkasan
                Total::create([
                    'harga' => $subtotal,
                    'keterangan' => 'penjualan',
                    'waktu' => $validated['waktu'],
                ]);
            }
            // Penjualan
            $penjualan = Coa::where('nama_akun',
'Penjualan')->first();
            $penjualan->jumlah += $subtotal;
            $penjualan->save();
            CoaTransaksi::create([
                'coa_id' => $penjualan->id,
                'jumlah' => $subtotal,
```

```
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Penambahan akun penjualan',
            ]);
            // Pendapatan Usaha
            $usaha = Coa::where('nama_akun', 'Pendapatan
Usaha')->first();
            $usaha->jumlah += $subtotal;
            $usaha->save();
            CoaTransaksi::create([
                'coa_id' => $usaha->id,
                'jumlah' => $subtotal,
                'jenis' => 'kredit',
                'waktu' => $validated['waktu'],
                'keterangan' => 'Pendapatan usaha dari
penjualan',
            ]);
        });
        return redirect()->back()->with('success', 'Aset
berhasil dikurangi.');
    }
public function kas(Request $request)
{
    $bulan = $request->input('bulan');
    $tahun = $request->input('tahun');
    $kas = Coa::where('nama_akun', 'Kas')->first();
    if (!$kas) {
        return Inertia::render('Aset/kas', [
            'asets' => 0,
            'totals' => collect(),
            'lastTransaction' => null,
            'bulan' => $bulan,
            'tahun' => $tahun,
        ]);
    }
    $query = CoaTransaksi::where('coa_id', $kas->id);
    if ($bulan) {
        $query->whereMonth('waktu', $bulan);
    }
    if ($tahun) {
        $query->whereYear('waktu', $tahun);
    }
    $totals = $query->orderBy('waktu')->get();
    $runningTotal = 0;
    $totalsWithSaldo = $totals->map(function ($item)
use (&$runningTotal) {
        $runningTotal += $item->jumlah;
        $item->saldo = $runningTotal;
        return $item;
    });
    $lastTransaction = $totalsWithSaldo->last();
    return Inertia::render('Aset/kas', [
        'asets' => $runningTotal,
        'totals' => $totalsWithSaldo,
        'lastTransaction' => $lastTransaction,
        'bulan' => $bulan,
        'tahun' => $tahun,
    ]);
}
public function jurnal(Request $request)
{
    $bulan = $request->input('bulan');
    $tahun = $request->input('tahun');
    $kas = Coa::where('nama_akun', 'Kas')->first();
    $totals = collect();
    $totalHargaTransaksi = 0;
    if ($kas) {
        $query = CoaTransaksi::where('coa_id', $kas->id);
        if ($bulan) {
            $query->whereMonth('waktu', $bulan);
        }
        if ($tahun) {
            $query->whereYear('waktu', $tahun);
        }
        $totals = $query->orderBy('waktu')->get();
        $totalHargaTransaksi = $totals->sum('jumlah');
    }
    return Inertia::render('Aset/jurnal', [
        'totalHargaTransaksi' => $totalHargaTransaksi,
        'asets' => $kas,
        'totals' => $totals,
        'bulan' => $bulan,
        'tahun' => $tahun,
    ]);
}
    private function transferPiutangKeKas()
    {
        DB::transaction(function () {
```

```
        $piutangs = Piutang::whereDate('jatuh_tempo',
'<=', now())->get();
        if ($piutangs->isEmpty())
            return;
        $kas = Coa::where('nama_akun', 'Kas')-
>firstOrFail();
        $piutangAkun = Coa::where('nama_akun',
'Piutang Usaha')->firstOrFail();
        foreach ($piutangs as $piutang) {
            $jumlah = $piutang->jumlah;
            // Update saldo
            $kas->jumlah += $jumlah;
            $piutangAkun->jumlah -= $jumlah;
            if ($piutangAkun->jumlah < 0)
                $piutangAkun->jumlah = 0;
            // Simpan transaksi CoaTransaksi
            $waktu = now(); // Bisa juga pakai $piutang-
>jatuh_tempo kalau mau tetap pakai jadwal aslinya
            CoaTransaksi::create([
                'coa_id' => $kas->id,
                'jumlah' => $jumlah,
                'jenis' => 'debit',
                'waktu' => $waktu,
                'keterangan' => 'Pelunasan piutang masuk
ke kas',
            ]);
            CoaTransaksi::create([
                'coa_id' => $piutangAkun->id,
                'jumlah' => -$jumlah,
                'jenis' => 'kredit',
                'waktu' => $waktu,
                'keterangan' => 'Pengurangan piutang
karena pelunasan--',
            ]);
            // Total ringkasan
            Total::create([
                'harga' => $jumlah,
                'keterangan' => 'penjualan',
                'waktu' => $waktu,
            ]);
            // Hapus piutang
            $piutang->delete();
        }
        $kas->save();
        $piutangAkun->save();
```




FMG

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

Mahasiswa : Besse Syulfiani	**Pembimbing I : Andi Wafiah, S.Kom., M.Kom**
NIM : 221280036	**Pembimbing II : Mugahffir Yunus. ST., MT**
Judul Skripsi : Perencanaan Aplikasi Sistem Manajemen Toko Online Penyewaan Baju Untuk Meningkatkan Konversi Penjualan	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Lengkapi aturan dan syarat sewa baju bodo.	1.	Konsultasi 1 Perbaikai tata bahasa di latar belakang!!!	
Konsultasi 2 Use case diagram di lengkapi	1.	Konsultasi 2 - Buatkan kerangka pikir - Perbaikan use case - perbaikan penulisan judul	
Konsultasi 3 - Batasan masalah - Tempat penelitian. - Penggunaan bahasa asing di cetak miring.	1.	Konsultasi 3 - Perbaikankerangka pikir - Menggati jenis penelitian dan lokasi dan waktu - buatkan perancangan antar muka	
Konsultasi 4 - Daftar Pustaka disesuaikan dgn Pedoman Penulisan. dan diurut abjad.	1.	Konsultasi 4 - Revisi antar muka rancang. aplikasi.	
Konsultasi 5 ACC Proposal	1.	Konsultasi 5 Acc Proposal	

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
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : BESSE SYULFIANI	**Pembimbing I : Andi Wafiah, S.Kom., M.Kom**
NIM : 221280036	**Pembimbing II : Mughaffir Tunus. ST., MT**
Judul Skripsi : PERANCANGAN APLIKASI SISTEM PENYEWAAN BAJU ADAT	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Revisi Aplikasi	4.	Konsultasi 1 - Revisi Aplikasi	
Konsultasi 2 keterangan size setiap ukuran baju	4.	Konsultasi 2 Revisi Aplikasi	
Konsultasi 3 Revisi Aplikasi	4.	Konsultasi 3 Revisi Aplikasi	
Konsultasi 4 Acc	4.	Konsultasi 4 Format Penulisan	
Konsultasi 5 - Lengkapi daftar pustaka. - Revisi Aplikasi	4.	Konsultasi 5 Perjelas Gambar.	

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
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik