

LAMPIRAN

Lampiran 1 Surat keterangan Observasi



No. : 068/I-BAAIK/AMIKOM SKA/V/2026
Hal. : Permohonan Ijin/Rekomendasi Penelitian

Kepada Yth.
Pemilik
Apotek Ridha Farma
Klewer, Sragen, Kec. Gatak, Kabupaten Sukoharjo
Di Tempat.

Salam Sejahtera, semoga limpahan rizki dan hidayah-Nya selalu menyertai dalam kegiatan rutin kita, amin.

Kami adalah Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) AMIKOM Surakarta sebuah perguruan tinggi swasta di Surakarta yang beralamat di Veteran Notosuman Singopuran Kartasura Sukoharjo Telp. (0271) 7851501. 7851507 fax. (0271) 7851507.

Sehubungan dengan pengerjaan tugas mata kuliah Skripsi pada Semester VIII, mahasiswa/i STMIK AMIKOM Surakarta dengan data dibawah ini, berkeinginan untuk mengadakan observasi di instansi yang Bapak/Ibu pimpin.

Adapun data mahasiswa/i adalah sebagai berikut :

Nama	: BAYU RIZKY UTOMO
NIM	: 2213010423
Program Studi	: Informatika
Jenjang	: S1

Demikian surat permohonan ini, terima kasih atas kerja samanya.

Surakarta, 18 Mei 2026



AMIKOM SURAKARTA
Yeni Fitriati, S.Kom.
BAAIK

Lampiran 2 listing program point of sale

Listing program halaman dashboard

```
<?php

namespace App\Application\UseCases\Dashboard;

use App\Domain\Contracts\MedicineRepository;
use App\Domain\Contracts\SaleRepository;
use App\Domain\Contracts\SupplierRepository;
use App\Models\Medicine;
use App\Models\Sale;
use Carbon\CarbonPeriod;
use Illuminate\Support\Collection;

class GetDashboard
{
    public function __construct(
        private readonly MedicineRepository $medicines,
        private readonly SupplierRepository $suppliers,
        private readonly SaleRepository $sales,
    ) {}

    public function execute(): array
    {
        $lowStock = $this->medicines->lowStock(6);
        $nearExpired = $this->medicines->nearExpired(now()->addDays(30), 6);

        return [
            'stats' => [
                'total_obat' => $this->medicines->count(),
                'total_supplier' => $this->suppliers->count(),
                'transaksi_hari_ini' => $this->sales->countOnDate(today()),
                'penjualan_bulan_ini' => $this->sales->sumForMonth(now()->month, now()->year),
            ],
            'sales_trend' => $this->salesTrend(),
            'notifications' => [
                'stok_menipis' => $lowStock,
                'expired_dekat' => $nearExpired,
            ],
        ],
    }
}
```

```

        'activities' => $this->activities($lowStock,
$nearExpired),
    ];
}

private function salesTrend(): array
{
    $startDate = now()->subDays(29)->startOfDay();
    $endDate = now()->endOfDay();
    $totals = $this->sales->dailyTrend($startDate, $endDate)
        ->keyBy(fn ($row): string => $row->tanggal-
>toDateString());

    return collect(CarbonPeriod::create($startDate, $endDate))
        ->map(function ($date) use ($totals): array {
            $dateString = $date->toDateString();

            return [
                'date' => $dateString,
                'label' => $date->locale('id')-
>translatedFormat('j M'),
                'total' => (float) ($totals->get($dateString)?-
>total ?? 0),
            ];
        })
        ->values()
        ->all();
}

private function activities(Collection $lowStock, Collection
$nearExpired): array
{
    $sales = $this->sales->recent(3)->map(fn (Sale $sale): array
=> [
        'id' => "sale-{$sale->id}",
        'type' => 'sale',
        'title' => 'Penjualan baru senilai Rp
'.number_format((float) $sale->total_harga, 0, ',', '.'),
        'description' => 'Kasir: ' . ($sale->user?->name ?? 'User
terhapus'),
        'occurred_at' => $sale->tanggal->toISOString(),
    ]);

    $stockActivities = $lowStock->take(2)->map(fn (Medicine
$medicine): array => [

```

```

        'id' => "stock-{$medicine->id}",
        'type' => 'stock',
        'title' => "Stok menipis: {$medicine->nama_obat}",
        'description' => "Sisa {$medicine->stok} {$medicine->satuan}",
        'occurred_at' => $medicine->updated_at->toISOString(),
    ]);

    $expiredActivities = $nearExpired->take(2)->map(fn (Medicine $medicine): array => [
        'id' => "expired-{$medicine->id}",
        'type' => 'expired',
        'title' => "Produk akan kedaluwarsa: {$medicine->nama_obat}",
        'description' => 'Exp: '.$medicine->tanggal_expired->locale('id')->translatedFormat('j M Y'),
        'occurred_at' => $medicine->updated_at->toISOString(),
    ]);

    return $sales
        ->concat($stockActivities)
        ->concat($expiredActivities)
        ->sortByDesc('occurred_at')
        ->take(6)
        ->values()
        ->all();
    }
}

```

Listing program halaman data obat

```

<?php

namespace App\Application\UseCases\Medicines;

use App\Domain\Contracts\MedicineRepository;
use App\Models\Medicine;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Validation\ValidationException;

class ManageMedicines
{

```

```

        public function __construct(private readonly MedicineRepository
$medicines) {}

        public function list(array $filters, int $perPage):
LengthAwarePaginator
        {
            return $this->medicines->paginate($filters,
min(max($perPage, 1), 100));
        }

        public function create(array $data): Medicine
        {
            return $this->medicines->create($data);
        }

        public function show(Medicine $medicine): Medicine
        {
            return $medicine->load('supplier');
        }

        public function update(Medicine $medicine, array $data):
Medicine
        {
            return $this->medicines->update($medicine, $data);
        }

        public function delete(Medicine $medicine): void
        {
            if ($this->medicines->hasSales($medicine)) {
                throw ValidationException::withMessages([
                    'medicine' => 'Obat yang sudah dipakai transaksi
tidak dapat dihapus.',
                ]);
            }

            $this->medicines->delete($medicine);
        }
    }
}

```

Listing program bagian pembelian

```
<?php
```

```

namespace App\Application\UseCases\Purchases;

use App\Domain\Contracts\MedicineRepository;
use App\Domain\Contracts\PurchaseRepository;
use App\Models\Purchase;
use App\Models\PurchaseDetail;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Support\Facades\DB;
use Illuminate\Validation\ValidationException;

class ManagePurchases
{
    public function __construct(
        private readonly PurchaseRepository $purchases,
        private readonly MedicineRepository $medicines,
    ) {}

    public function list(array $filters, int $perPage):
    LengthAwarePaginator
    {
        return $this->purchases->paginate($filters,
min(max($perPage, 1), 100));
    }

    public function show(Purchase $purchase): Purchase
    {
        return $purchase->load(['supplier', 'user',
'details.medicine.supplier']);
    }

    public function create(array $payload, int $userId): Purchase
    {
        return DB::transaction(function () use ($payload, $userId):
Purchase {
            $purchase = $this->purchases->create([
                'supplier_id' => $payload['supplier_id'],
                'user_id' => $userId,
                'nomor_faktur' => $payload['nomor_faktur'],
                'tanggal_faktur' => $payload['tanggal_faktur'],
                'total_harga' => 0,
                'file_faktur' => $payload['file_faktur'] ?? null,
                'catatan' => $payload['catatan'] ?? null,
            ]);

```

```

        $total = $this->syncDetails($purchase,
        $payload['details']);
        $this->purchases->update($purchase, ['total_harga' =>
        $total]);

        return $this->show($purchase);
    });
}

    public function update(Purchase $purchase, array $payload):
    Purchase
    {
        return DB::transaction(function () use ($purchase,
        $payload): Purchase {
            $purchase->load('details');
            $this->reverseStockImpact($purchase);
            $purchase->details()->delete();

            $this->purchases->update($purchase, [
                'supplier_id' => $payload['supplier_id'],
                'nomor_faktur' => $payload['nomor_faktur'],
                'tanggal_faktur' => $payload['tanggal_faktur'],
                'file_faktur' => $payload['file_faktur'] ??
        $purchase->file_faktur,
                'catatan' => $payload['catatan'] ?? null,
            ]);

            $total = $this->syncDetails($purchase,
        $payload['details']);
            $this->purchases->update($purchase, ['total_harga' =>
        $total]);

            return $this->show($purchase);
        });
    }

    public function delete(Purchase $purchase): void
    {
        DB::transaction(function () use ($purchase): void {
            $purchase->load('details');
            $this->reverseStockImpact($purchase);
            $this->purchases->delete($purchase);
        });
    }
}

```

```

    public function markReturn(Purchase $purchase, PurchaseDetail
    $purchaseDetail, array $payload): Purchase
    {
        return DB::transaction(function () use ($purchase,
    $purchaseDetail, $payload): Purchase {
            $detail = $this->purchases-
    >findDetailForUpdate($purchaseDetail->id);

            if ($detail->purchase_id !== $purchase->id) {
                throw ValidationException::withMessages([
                    'detail' => 'Detail faktur tidak sesuai dengan
    faktur pembelian.',
                ]);
            }

            $qtyRetur = (int) ($payload['qty_retur'] ?? 0);
            if ($qtyRetur > $detail->qty) {
                throw ValidationException::withMessages([
                    'qty_retur' => 'Qty retur tidak boleh melebihi
    qty pembelian.',
                ]);
            }

            $oldReturnedQty = $detail->status_retur === 'diretur' ?
    $detail->qty_retur : 0;
            $newReturnedQty = $payload['status_retur'] === 'diretur'
    ? $qtyRetur : 0;
            $stockDelta = $newReturnedQty - $oldReturnedQty;

            if ($stockDelta !== 0) {
                $medicine = $this->medicines->findForUpdate($detail->
    medicine_id);

                if ($stockDelta > 0 && $medicine->stok <
    $stockDelta) {
                    throw ValidationException::withMessages([
                        'qty_retur' => "Stok {$medicine->nama_obat}
    tidak cukup untuk retur.",
                    ]);
                }

                $stockDelta > 0
                    ? $medicine->decrement('stok', $stockDelta)
                    : $medicine->increment('stok',
    abs($stockDelta));
            }
        });
    }

```

```

    }

    $detail->update([
        'qty_retur' => $qtyRetur,
        'status_retur' => $payload['status_retur'],
        'tanggal_retur' => $payload['tanggal_retur'] ??
null,
        'catatan_retur' => $payload['catatan_retur'] ??
null,
    ]);

    return $this->show($purchase->refresh());
});
}

private function syncDetails(Purchase $purchase, array
$details): float
{
    $total = 0.0;

    foreach ($details as $item) {
        $medicine = $this->medicines-
>findForUpdate($item['medicine_id']);
        $qty = (int) $item['qty'];
        $hargaBeli = (float) $item['harga_beli'];
        $subtotal = $qty * $hargaBeli;

        $purchase->details()->create([
            'medicine_id' => $medicine->id,
            'nomor_batch' => $item['nomor_batch'] ?? null,
            'qty' => $qty,
            'harga_beli' => $hargaBeli,
            'subtotal' => $subtotal,
            'tanggal_expired' => $item['tanggal_expired'],
        ]);

        $medicine->increment('stok', $qty);
        $medicine->update([
            'supplier_id' => $purchase->supplier_id,
            'harga_beli' => $hargaBeli,
            'tanggal_expired' => $item['tanggal_expired'],
        ]);

        $total += $subtotal;
    }
}

```

```

        return $total;
    }

    private function reverseStockImpact(Purchase $purchase): void
    {
        foreach ($purchase->details as $detail) {
            $remainingStockImpact = $detail->qty - ($detail->status_retur === 'diretur' ? $detail->qty_retur : 0);
            if ($remainingStockImpact <= 0) {
                continue;
            }

            $medicine = $this->medicines->findForUpdate($detail->medicine_id);
            if ($medicine->stok < $remainingStockImpact) {
                throw ValidationException::withMessages([
                    'details' => "Stok {$medicine->nama_obat} tidak cukup untuk membatalkan faktur.",
                ]);
            }

            $medicine->decrement('stok', $remainingStockImpact);
        }
    }
}

```

Listing program laporan penjualan

```

<?php

namespace App\Application\UseCases\Reports;

use App\Domain\Contracts\SaleRepository;
use App\Models\Sale;
use Carbon\CarbonInterface;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Support\Facades\DB;

class GenerateSalesReport
{
    public function __construct(private readonly SaleRepository $sales) {}
}

```

```

public function daily(CarbonInterface $date): array
{
    return $this->summary(
        $this->sales->query()->whereDate('tanggal', $date),
        'Laporan harian '.$date->toDateString()
    );
}

public function monthly(int $month, int $year): array
{
    return $this->summary(
        $this->sales->query()->whereMonth('tanggal', $month)-
>whereYear('tanggal', $year),
        "Laporan bulanan {$month}/{ $year}"
    );
}

public function yearly(int $year): array
{
    return $this->summary(
        $this->sales->query()->whereYear('tanggal', $year),
        "Laporan tahunan {$year}"
    );
}

private function summary(Builder $query, string $title): array
{
    $sales = (clone $query)->with(['details', 'user'])-
>latest('tanggal')->get();

    return [
        'title' => $title,
        'total_transaksi' => $sales->count(),
        'total_penjualan' => (float) $sales->sum('total_harga'),
        'total_item_terjual' => (int) $sales->flatMap->details-
>sum('qty'),
        'top_medicines' => DB::table('sale_details')
            ->join('sales', 'sales.id', '=',
'sale_details.sale_id')
            ->join('medicines', 'medicines.id', '=',
'sale_details.medicine_id')
            ->whereIn('sales.id', $sales->pluck('id'))
            ->select('medicines.id', 'medicines.kode_obat',
'medicines.nama_obat', DB::raw('SUM(sale_details.qty) as qty'))
    ];
}

```

```

        ->groupBy('medicines.id', 'medicines.kode_obat',
'medicines.nama_obat')
        ->orderByDesc('qty')
        ->limit(5)
        ->get(),
'sales' => $sales->map(fn (Sale $sale): array => [
    'id' => $sale->id,
    'nomor_transaksi' => $sale->nomor_transaksi,
    'tanggal' => $sale->tanggal->toISOString(),
    'total_harga' => (float) $sale->total_harga,
    'kasir' => $sale->user?->name ?? 'Kasir',
    'jumlah_item' => (int) $sale->details->sum('qty'),
]),
];
    }
}

```

Listing program penjualan

```

<?php

namespace App\Application\UseCases\Sales;

use App\Domain\Contracts\MedicineRepository;
use App\Domain\Contracts\SaleRepository;
use App\Models\Sale;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;
use Illuminate\Validation\ValidationException;

class ManageSales
{
    public function __construct(
        private readonly SaleRepository $sales,
        private readonly MedicineRepository $medicines,
    ) {}

    public function list(array $filters, int $perPage):
LengthAwarePaginator

```

```

    {
        return $this->sales->paginate($filters, min(max($perPage,
1), 100));
    }

    public function show(Sale $sale): Sale
    {
        return $sale->load(['user', 'details.medicine.supplier']);
    }

    public function create(array $payload, int $userId): Sale
    {
        return DB::transaction(function () use ($payload, $userId):
Sale {
            $sale = $this->sales->create([
                'user_id' => $userId,
                'nomor_transaksi' => $this->
>generateTransactionNumber(),
                'tanggal' => $payload['tanggal'],
                'total_harga' => 0,
            ]);

            $total = $this->syncDetails($sale, $payload['details']);
            $this->sales->update($sale, ['total_harga' => $total]);

            return $sale->load(['user',
'details.medicine.supplier']);
        });
    }

    public function update(Sale $sale, array $payload): Sale
    {
        return DB::transaction(function () use ($sale, $payload):
Sale {
            foreach ($sale->details as $detail) {
                $detail->medicine()->increment('stok', $detail->
>qty);
            }

            $sale->details()->delete();
            $this->sales->update($sale, ['tanggal' =>
$payload['tanggal']]);

            $total = $this->syncDetails($sale, $payload['details']);
            $this->sales->update($sale, ['total_harga' => $total]);

```

```

        return $sale->load(['user',
'details.medicine.supplier']);
    });
}

public function delete(Sale $sale): void
{
    DB::transaction(function () use ($sale): void {
        foreach ($sale->details as $detail) {
            $detail->medicine()->increment('stok', $detail-
>qty);
        }

        $this->sales->delete($sale);
    });
}

private function syncDetails(Sale $sale, array $details): float
{
    $total = 0.0;

    foreach ($details as $item) {
        $medicine = $this->medicines-
>findForUpdate($item['medicine_id']);

        if ($medicine->stok < $item['qty']) {
            throw ValidationException::withMessages([
                'details' => "Stok {$medicine->nama_obat} tidak
mencukupi.",
            ]);
        }

        $subtotal = (float) $medicine->harga_jual * (int)
$item['qty'];
        $sale->details()->create([
            'medicine_id' => $medicine->id,
            'qty' => $item['qty'],
            'harga' => $medicine->harga_jual,
            'subtotal' => $subtotal,
        ]);

        $medicine->decrement('stok', $item['qty']);
        $total += $subtotal;
    }
}

```

```

        return $total;
    }

    private function generateTransactionNumber(): string
    {
        return 'TRX-'.now()->format('Ymd').'-'.Str::upper(substr((string) Str::ulid(), -8));
    }
}

```

Listing program SAW

```

<?php

namespace App\Application\UseCases\Saw;

use App\Domain\Contracts\MedicineRepository;
use App\Models\Medicine;
use Carbon\CarbonImmutable;
use Carbon\CarbonInterface;
use Illuminate\Support\Collection;

class CalculateRestockRanking
{
    private const WEIGHTS = [
        'penjualan' => 0.30,
        'stok' => 0.25,
        'expired' => 0.20,
        'prioritas_owner' => 0.15,
        'harga' => 0.10,
    ];

    public function __construct(private readonly MedicineRepository $medicines) {}

    public function execute(?CarbonInterface $startDate = null, ?CarbonInterface $endDate = null): Collection
    {
        $startDate ??= now()->subDays(30)->startOfDay();
        $endDate ??= now()->endOfDay();
        $today = CarbonImmutable::today();
    }
}

```

```

        $rows = $this->medicines
            ->rankingCandidates($startDate, $endDate)
            ->map(function (Medicine $medicine) use ($today): array
{
            $expiredDate = $medicine->nearest_batch_expired
                ? CarbonImmutable::parse($medicine-
>nearest_batch_expired)
                : CarbonImmutable::parse($medicine-
>tanggal_expired);
            $rawDaysUntilExpired = (int) $today-
>diffInDays($expiredDate, false);
            $daysUntilExpired = max(1, $rawDaysUntilExpired);

            return [
                'medicine' => $medicine,
                'criteria' => [
                    'penjualan' => (float) ($medicine->sales_qty
?? 0),

                    'stok' => max(1, (float) $medicine->stok),
                    'expired' => (float) $daysUntilExpired,
                    'prioritas_owner' => (float) min(5, max(1,
$medicine->prioritas_owner)),
                    'harga' => (float) $medicine->harga_jual,
                ],
                'tanggal_expired_digunakan' => $expiredDate-
>toDateString(),
                'sisahari_expired' => $rawDaysUntilExpired,
            ];
        });

        if ($rows->isEmpty()) {
            return collect();
        }

        $max = [
            'penjualan' => max(1, $rows->max('criteria.penjualan')),
            'stok' => max(1, $rows->max('criteria.stok')),
            'expired' => max(1, $rows->max('criteria.expired')),
            'prioritas_owner' => max(1, $rows-
>max('criteria.prioritas_owner')),
            'harga' => max(1, $rows->max('criteria.harga')),
        ];
        $min = [
            'penjualan' => max(0, $rows->min('criteria.penjualan')),
            'stok' => max(1, $rows->min('criteria.stok')),

```

```

        'expired' => max(1, $rows->min('criteria.expired')),
        'prioritas_owner' => max(1, $rows->
min('criteria.prioritas_owner')),
        'harga' => max(0, $rows->min('criteria.harga')),
    ];
    $benefitMax = [
        'penjualan' => $max['penjualan'],
        'harga' => $max['harga'],
        'prioritas_owner' => max(1, $rows->
max('criteria.prioritas_owner')),
    ];
    $costMin = [
        'stok' => $min['stok'],
        'expired' => $min['expired'],
    ];

    return $rows
        ->map(function (array $row) use ($max, $min,
$benefitMax, $costMin): array {
            $criteria = $row['criteria'];
            $normalized = [
                'penjualan' => $criteria['penjualan'] /
$benefitMax['penjualan'],
                'stok' => $costMin['stok'] / $criteria['stok'],
                'expired' => $costMin['expired'] /
$criteria['expired'],
                'prioritas_owner' =>
$criteria['prioritas_owner'] / $benefitMax['prioritas_owner'],
                'harga' => $criteria['harga'] /
$benefitMax['harga'],
            ];

            $weighted = collect($normalized)
                ->mapWithKeys(fn (float $value, string $key):
array => [$key => $value * self::WEIGHTS[$key]])
                ->all();
            $score = collect($normalized)
                ->reduce(fn (float $total, float $value, string
$key): float => $total + ($value * self::WEIGHTS[$key]), 0.0);

            return [
                ...$row,
                'normalized' => $normalized,
                'weighted' => $weighted,
                'score' => $score,
            ];
        });

```

```

        ];
    })
    ->sortByDesc('score')
    ->values()
    ->map(function (array $row, int $index): array {
        $row['ranking'] = $index + 1;

        return $row;
    });
}

public function weights(): array
{
    return self::WEIGHTS;
}
}

```

Listing program Suppliers

```

<?php

namespace App\Application\UseCases\Suppliers;

use App\Domain\Contracts\SupplierRepository;
use App\Models\Supplier;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;

class ManageSuppliers
{
    public function __construct(private readonly SupplierRepository
$suppliers) {}

    public function list(array $filters, int $perPage):
LengthAwarePaginator
    {
        return $this->suppliers->paginate($filters,
min(max($perPage, 1), 100));
    }

    public function create(array $data): Supplier
    {

```

```

        return $this->suppliers->create($data);
    }

    public function show(Supplier $supplier): Supplier
    {
        return $supplier->loadCount('medicines');
    }

    public function update(Supplier $supplier, array $data):
Supplier
    {
        return $this->suppliers->update($supplier, $data);
    }

    public function delete(Supplier $supplier): void
    {
        $this->suppliers->delete($supplier);
    }
}

```

Listing program USERS

```

<?php

namespace App\Application\UseCases\Users;

use App\Domain\Contracts\UserRepository;
use App\Models\User;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Validation\ValidationException;

class ManageUsers
{
    public function __construct(private readonly UserRepository
$users) {}

    public function list(int $perPage): LengthAwarePaginator
    {
        return $this->users->paginate(min(max($perPage, 1), 100));
    }

    public function create(array $data): User
    {
        return $this->users->create($data);
    }
}

```

```

    }

    public function update(User $user, array $data): User
    {
        if (blank($data['password'] ?? null)) {
            unset($data['password']);
        }

        return $this->users->update($user, $data);
    }

    public function delete(User $actor, User $target): void
    {
        if ($actor->is($target)) {
            throw ValidationException::withMessages([
                'user' => 'User aktif tidak dapat menghapus akunnya
sendiri.',
            ]);
        }

        $this->users->delete($target);
    }
}

```

Listing program database obat

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Medicine extends Model
{
    use HasFactory;

    protected $fillable = [
        'supplier_id',
        'kode_obat',
    ];
}

```

```

        'nama_obat',
        'kategori',
        'satuan',
        'harga_beli',
        'harga_jual',
        'stok',
        'stok_minimum',
        'tanggal_expired',
        'prioritas_owner',
    ];

    protected function casts(): array
    {
        return [
            'harga_beli' => 'decimal:2',
            'harga_jual' => 'decimal:2',
            'stok' => 'integer',
            'stok_minimum' => 'integer',
            'tanggal_expired' => 'date',
            'prioritas_owner' => 'integer',
        ];
    }

    public function supplier(): BelongsTo
    {
        return $this->belongsTo(Supplier::class);
    }

    public function saleDetails(): HasMany
    {
        return $this->hasMany(SaleDetail::class);
    }

    public function purchaseDetails(): HasMany
    {
        return $this->hasMany(PurchaseDetail::class);
    }

    public function scopeLowStock(Builder $query): Builder
    {
        return $query->whereColumn('stok', '<=', 'stok_minimum');
    }
}

```

Listing program Database Pembelian

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Purchase extends Model
{
    use HasFactory;

    protected $fillable = [
        'supplier_id',
        'user_id',
        'nomor_faktur',
        'tanggal_faktur',
        'total_harga',
        'file_faktur',
        'catatan',
    ];

    protected function casts(): array
    {
        return [
            'tanggal_faktur' => 'date',
            'total_harga' => 'decimal:2',
        ];
    }

    public function supplier(): BelongsTo
    {
        return $this->belongsTo(Supplier::class);
    }

    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }

    public function details(): HasMany
    {

```

```

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

```

Listing program Database detail pembelian

```

<?php

namespace App\Models;

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

class PurchaseDetail extends Model
{
    use HasFactory;

    protected $fillable = [
        'medicine_id',
        'nomor_batch',
        'qty',
        'harga_beli',
        'subtotal',
        'tanggal_expired',
        'qty_retur',
        'status_retur',
        'tanggal_retur',
        'catatan_retur',
    ];

    protected function casts(): array
    {
        return [
            'qty' => 'integer',
            'harga_beli' => 'decimal:2',
            'subtotal' => 'decimal:2',
            'tanggal_expired' => 'date',
            'qty_retur' => 'integer',
            'tanggal_retur' => 'date',
        ];
    }
}

```

```

    public function purchase(): BelongsTo
    {
        return $this->belongsTo(Purchase::class);
    }

    public function medicine(): BelongsTo
    {
        return $this->belongsTo(Medicine::class);
    }
}

```

Listing program Database Penjualan

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Sale extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'nomor_transaksi',
        'tanggal',
        'total_harga',
    ];

    protected function casts(): array
    {
        return [
            'tanggal' => 'datetime',
            'total_harga' => 'decimal:2',
        ];
    }

    public function user(): BelongsTo
    {

```

```

        return $this->belongsTo(User::class);
    }

    public function details(): HasMany
    {
        return $this->hasMany(SaleDetail::class);
    }
}

```

Listing program Database detail penjualan

```

<?php

namespace App\Models;

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

class SaleDetail extends Model
{
    use HasFactory;

    protected $fillable = [
        'sale_id',
        'medicine_id',
        'qty',
        'harga',
        'subtotal',
    ];

    protected function casts(): array
    {
        return [
            'qty' => 'integer',
            'harga' => 'decimal:2',
            'subtotal' => 'decimal:2',
        ];
    }

    public function sale(): BelongsTo
    {
        return $this->belongsTo(Sale::class);
    }
}

```

```

    }

    public function medicine(): BelongsTo
    {
        return $this->belongsTo(Medicine::class);
    }
}

```

DATABASE SUPPLIER

```

<?php

namespace App\Models;

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

class Supplier extends Model
{
    use HasFactory;

    protected $fillable = [
        'nama_supplier',
        'alamat',
        'telepon',
    ];

    public function medicines(): HasMany
    {
        return $this->hasMany(Medicine::class);
    }

    public function purchases(): HasMany
    {
        return $this->hasMany(Purchase::class);
    }
}

```

Listing program Database pengguna

```

<?php

```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

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

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

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

    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }

    public function isSuperAdmin(): bool
    {
        return $this->role === 'super_admin';
    }
}
```

Lampiran 3 Lembar Revisi penguji



SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
AMIKOM SURAKARTA
www.amikomsolo.ac.id amikomsolo@amikomsolo.ac.id

LEMBAR REVISI
UJIAN SKRIPSI
TA. 2025/2026

NAMA : BAYU RIZKY UTOMO
NOMOR MAHASISWA : 2213010423
PROGRAM STUDI : Informatika
HARI, TGL UJIAN : Senin, 24 Agustus 2026
BATAS AKHIR REVISI : **Senin, 7 September 2026**
JUDUL SKRIPSI : Implementasi Metode SAW Penentuan Prioritas Restock Obat Berdasarkan MADM pada Sistem Pos Apotek Berbasis Web (Studi Kasus: Apotek Ridha Farma Sraten Kecamatan Gatak)
PEMBIMBING SKRIPSI : 1. Lilik Sugiarto, M.Kom
2. Syam Kurniawan Hidayat, ST, M.Kom

NO	URAIAN	KET
	Tambahkan Use case description	/
	Bab IV disesuaikan dengan Alur yang disetujui	/
	Perancangan Database didetailkan	/
	Program : Penjualan , filter , cetak	/

Penguji : Indrawan Ady S, M.Kom ()*)

*) Ditandatangani oleh penguji setelah selesai Ujian Skripsi

Telah selesai dilakukan revisi pada tanggal : Senin, 7 September 2026

Penguji : Indrawan Ady S, M.Kom ()**)

**) Ditandatangani oleh penguji setelah mahasiswa selesai melakukan bimbingan revisi ke penguji
Jika revisi dilakukan setelah batas waktu revisi, maka akan diujikan ulang



SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
AMIKOM SURAKARTA
www.amikomsolo.ac.id amikomsolo@amikomsolo.ac.id

LEMBAR REVISI
UJIAN SKRIPSI
TA. 2025/2026

NAMA : BAYU RIZKY UTOMO
NOMOR MAHASISWA : 2213010423
PROGRAM STUDI : Informatika
HARI, TGL UJIAN : Senin, 24 Agustus 2026
BATAS AKHIR REVISI : **Senin, 7 September 2026**
JUDUL SKRIPSI : Implementasi Metode SAW Penentuan Prioritas Restock Obat Berdasarkan MADM pada Sistem Pos Apotek Berbasis Web (Studi Kasus: Apotek Ridha Farma Sragen Kecamatan Gatak)
PEMBIMBING SKRIPSI : 1. Lilik Sugiarto, M.Kom
2. Syam Kurniawan Hidayat, ST, M.Kom

NO	URAIAN	KET
1	Penulisan : - cek pd naskah yg ditinjau - jika salah ubah → iyalis - tabel → format font 10 spasi 1	1
2, 3.	Landasan teori : → tabel jangan di screenshot, buat tabel nya Pembahasan → sesuaikan bahasan dg alur penelitian	1
	- Tabel piece tidak perlu ditampilkan, ambil dari tabel 3.1 saja - Uskcam sesuaikan lagi → include lagi - Aktiviti Diagram → sesuaikan lagi dg prosesnya	1
	- Sequence Diagram → sesuaikan dg proses - UML Class Diagram / ERD → pilih salah 1	1
	- Teks & gambar pd desain jelaskan dg detail - jelaskan bahasan blackbox sekiranya, berpakailah, apa halnya - UML Use Case Diagram → breakdown per indikator	1

4. Kesimpulan → buat 2-3 kalimat paragraf.

Penguji : M. Sugiarto

*)

*) Ditandatangani oleh penguji setelah selesai Ujian Skripsi

Telah selesai dilakukan revisi pada tanggal : 3 September 2026

Penguji : M. Sugiarto

**)

**) Ditandatangani oleh penguji setelah mahasiswa selesai melakukan bimbingan revisi ke penguji
Jika revisi dilakukan setelah batas waktu revisi, maka akan diujikan ulang



SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
AMIKOM SURAKARTA
www.amikomsolo.ac.id amikomsolo@amikomsolo.ac.id

LEMBAR REVISI
UJIAN SKRIPSI
TA. 2025/2026

NAMA : BAYU RIZKY UTOMO
NOMOR MAHASISWA : 2213010423
PROGRAM STUDI : Informatika
HARI, TGL UJIAN : Senin, 24 Agustus 2026
BATAS AKHIR REVISI : **Senin, 7 September 2026**
JUDUL SKRIPSI : Implementasi Metode SAW Penentuan Prioritas Restock Obat Berdasarkan MADM pada Sistem Pos Apotek Berbasis Web (Studi Kasus: Apotek Ridha Farma Sragen Kecamatan Gatak)
PEMBIMBING SKRIPSI : 1. Lilik Sugiarto, M.Kom
2. Syam Kurniawan Hidayat, ST, M.Kom

NO	URAIAN	KET
1	rev. penulisan	

Penguji : Lilik SUGIARTO, M.Kom (Sugarto) *)

*) Ditandatangani oleh penguji setelah selesai Ujian Skripsi

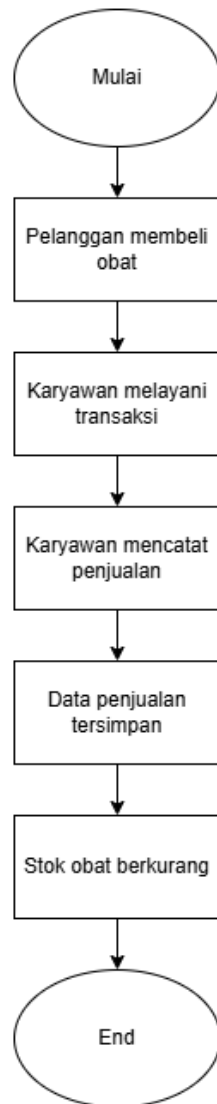
Telah selesai dilakukan revisi pada tanggal : 4 September 2026

Penguji : Lilik SUGIARTO, M.Kom (Sugarto) **)

**) Ditandatangani oleh penguji setelah mahasiswa selesai melakukan bimbingan revisi ke penguji
Jika revisi dilakukan setelah batas waktu revisi, maka akan diujikan ulang

Lampiran 4 Proses bisnis lama yang terjadi pada apotek

Proses Transaksi Penjualan



Proses Pengecekan Stok



Proses Pemesanan Obat

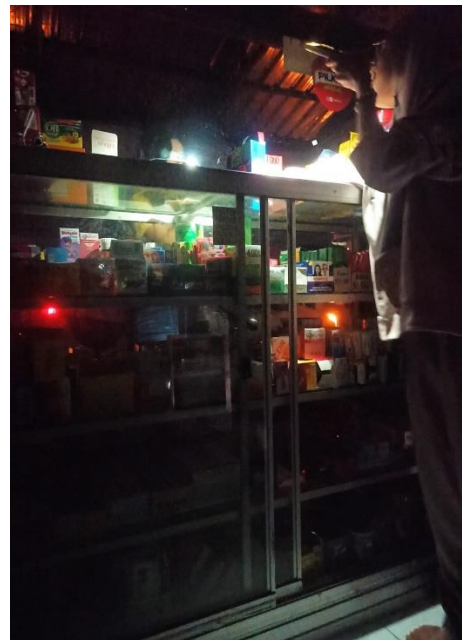


Lampiran 5 Foto buku pengelolaan stok dan buku faktur

		23.00	23.00	271 7123493234M
		26.00	26.00	271 8015602324M
271	100	27.00	27.00	271 2478002324M
		28.00	28.00	271 2710002324M
		29.00	29.00	271 2710002324M
		30.00	30.00	271 2710002324M
		31.00	31.00	271 2710002324M
		32.00	32.00	271 2710002324M
		33.00	33.00	271 2710002324M
		34.00	34.00	271 2710002324M
		35.00	35.00	271 2710002324M
		36.00	36.00	271 2710002324M
		37.00	37.00	271 2710002324M
		38.00	38.00	271 2710002324M
		39.00	39.00	271 2710002324M
		40.00	40.00	271 2710002324M
		41.00	41.00	271 2710002324M
		42.00	42.00	271 2710002324M
		43.00	43.00	271 2710002324M
		44.00	44.00	271 2710002324M
		45.00	45.00	271 2710002324M
		46.00	46.00	271 2710002324M
		47.00	47.00	271 2710002324M
		48.00	48.00	271 2710002324M
		49.00	49.00	271 2710002324M
		50.00	50.00	271 2710002324M
		51.00	51.00	271 2710002324M
		52.00	52.00	271 2710002324M
		53.00	53.00	271 2710002324M
		54.00	54.00	271 2710002324M
		55.00	55.00	271 2710002324M
		56.00	56.00	271 2710002324M
		57.00	57.00	271 2710002324M
		58.00	58.00	271 2710002324M
		59.00	59.00	271 2710002324M
		60.00	60.00	271 2710002324M
		61.00	61.00	271 2710002324M
		62.00	62.00	271 2710002324M
		63.00	63.00	271 2710002324M
		64.00	64.00	271 2710002324M
		65.00	65.00	271 2710002324M
		66.00	66.00	271 2710002324M
		67.00	67.00	271 2710002324M
		68.00	68.00	271 2710002324M
		69.00	69.00	271 2710002324M
		70.00	70.00	271 2710002324M
		71.00	71.00	271 2710002324M
		72.00	72.00	271 2710002324M
		73.00	73.00	271 2710002324M
		74.00	74.00	271 2710002324M
		75.00	75.00	271 2710002324M
		76.00	76.00	271 2710002324M
		77.00	77.00	271 2710002324M
		78.00	78.00	271 2710002324M
		79.00	79.00	271 2710002324M
		80.00	80.00	271 2710002324M
		81.00	81.00	271 2710002324M
		82.00	82.00	271 2710002324M
		83.00	83.00	271 2710002324M
		84.00	84.00	271 2710002324M
		85.00	85.00	271 2710002324M
		86.00	86.00	271 2710002324M
		87.00	87.00	271 2710002324M
		88.00	88.00	271 2710002324M
		89.00	89.00	271 2710002324M
		90.00	90.00	271 2710002324M
		91.00	91.00	271 2710002324M
		92.00	92.00	271 2710002324M
		93.00	93.00	271 2710002324M
		94.00	94.00	271 2710002324M
		95.00	95.00	271 2710002324M
		96.00	96.00	271 2710002324M
		97.00	97.00	271 2710002324M
		98.00	98.00	271 2710002324M
		99.00	99.00	271 2710002324M
		100.00	100.00	

[illegible][illegible][illegible][illegible]

Lampiran 6 Foto Observasi di apotek ridha farma



Lampiran 7 Foto kartu pengurangan stok obat di apotek ridha farma

Tanggal	Terima Dari	Masuk	Keluar	Sisa	Harga	No. Stok	Barang
15/6		10	10	100	100		
16/6		10	10	100	100		
17/6		10	10	100	100		
18/6		10	10	100	100		
19/6		10	10	100	100		
20/6		10	10	100	100		
21/6		10	10	100	100		
22/6		10	10	100	100		
23/6		10	10	100	100		
24/6		10	10	100	100		
25/6		10	10	100	100		
26/6		10	10	100	100		
27/6		10	10	100	100		
28/6		10	10	100	100		
29/6		10	10	100	100		
30/6		10	10	100	100		
1/7		10	10	100	100		
2/7		10	10	100	100		
3/7		10	10	100	100		
4/7		10	10	100	100		
5/7		10	10	100	100		
6/7		10	10	100	100		
7/7		10	10	100	100		
8/7		10	10	100	100		
9/7		10	10	100	100		
10/7		10	10	100	100		
11/7		10	10	100	100		
12/7		10	10	100	100		
13/7		10	10	100	100		
14/7		10	10	100	100		
15/7		10	10	100	100		
16/7		10	10	100	100		
17/7		10	10	100	100		
18/7		10	10	100	100		
19/7		10	10	100	100		
20/7		10	10	100	100		
21/7		10	10	100	100		
22/7		10	10	100	100		
23/7		10	10	100	100		
24/7		10	10	100	100		
25/7		10	10	100	100		
26/7		10	10	100	100		
27/7		10	10	100	100		
28/7		10	10	100	100		
29/7		10	10	100	100		
30/7		10	10	100	100		
1/8		10	10	100	100		
2/8		10	10	100	100		
3/8		10	10	100	100		
4/8		10	10	100	100		
5/8		10	10	100	100		
6/8		10	10	100	100		
7/8		10	10	100	100		
8/8		10	10	100	100		
9/8		10	10	100	100		
10/8		10	10	100	100		
11/8		10	10	100	100		
12/8		10	10	100	100		
13/8		10	10	100	100		
14/8		10	10	100	100		
15/8		10	10	100	100		
16/8		10	10	100	100		
17/8		10	10	100	100		
18/8		10	10	100	100		
19/8		10	10	100	100		
20/8		10	10	100	100		
21/8		10	10	100	100		
22/8		10	10	100	100		
23/8		10	10	100	100		
24/8		10	10	100	100		
25/8		10	10	100	100		
26/8		10	10	100	100		
27/8		10	10	100	100		
28/8		10	10	100	100		
29/8		10	10	100	100		
30/8		10	10	100	100		
1/9		10	10	100	100		
2/9		10	10	100	100		
3/9		10	10	100	100		
4/9		10	10	100	100		
5/9		10	10	100	100		
6/9		10	10	100	100		
7/9		10	10	100	100		
8/9		10	10	100	100		
9/9		10	10	100	100		
10/9		10	10	100	100		
11/9		10	10	100	100		
12/9		10	10	100	100		
13/9		10	10	100	100		
14/9		10	10	100	100		
15/9		10	10	100	100		
16/9		10	10	100	100		
17/9		10	10	100	100		
18/9		10	10	100	100		
19/9		10	10	100	100		
20/9		10	10	100	100		
21/9		10	10	100	100		
22/9		10	10	100	100		
23/9		10	10	100	100		
24/9		10	10	100	100		
25/9		10	10	100	100		
26/9		10	10	100	100		
27/9		10	10	100	100		
28/9		10	10	100	100		
29/9		10	10	100	100		
30/9		10	10	100	100		
1/10		10	10	100	100		
2/10		10	10	100	100		
3/10		10	10	100	100		
4/10		10	10	100	100		
5/10		10	10	100	100		
6/10		10	10	100	100		
7/10		10	10	100	100		
8/10		10	10	100	100		
9/10		10	10	100	100		
10/10		10	10	100	100		
11/10		10	10	100	100		
12/10		10	10	100	100		
13/10		10	10	100	100		
14/10		10	10	100	100		
15/10		10	10	100	100		
16/10		10	10	100	100		
17/10		10	10	100	100		
18/10		10	10	100	100		
19/10		10	10	100	100		
20/10		10	10	100	100		
21/10		10	10	100	100		
22/10		10	10	100	100		
23/10		10	10	100	100		
24/10		10	10	100	100		
25/10		10	10	100	100		
26/10		10	10	100	100		
27/10		10	10	100	100		
28/10		10	10	100	100		
29/10		10	10	100	100		
30/10		10	10	100	100		
1/11		10	10	100	100		
2/11		10	10	100	100		
3/11		10	10	100	100		
4/11		10	10	100	100		
5/11		10	10	100	100		
6/11		10	10	100	100		
7/11		10	10	100	100		
8/11		10	10	100	100		
9/11		10	10	100	100		
10/11		10	10	100	100		
11/11		10	10	100	100		
12/11		10	10	100	100		
13/11		10	10	100	100		
14/11		10	10	100	100		
15/11		10	10	100	100		
16/11		10	10	100	100		
17/11		10	10	100	100		
18/11		10	10	100	100		
19/11		10	10	100	100		
20/11		10	10	100	100		
21/11		10	10	100	100		
22/11		10	10	100	100		
23/11		10	10	100	100		
24/11		10	10	100	100		
25/11		10	10	100	100		
26/11		10	10	100	100		
27/11		10	10	100	100		
28/11		10	10	100	100		
29/11		10	10	100	100		
30/11		10	10	100	100		
1/12		10	10	100	100		
2/12		10	10	100	100		
3/12		10	10	100	100		
4/12		10	10	100	100		
5/12		10	10	100	100		
6/12		10	10	100	100		
7/12		10	10	100	100		
8/12		10	10	100	100		
9/12		10	10	100	100		
10/12		10	10	100	100		
11/12		10	10	100	100		
12/12		10	10	100	100		
13/12		10	10	100	100		
14/12		10	10	100	100		
15/12		10	10	100	100		
16/12		10	10	100	100		
17/12		10	10	100	100		
18/12		10	10	100	100		
19/12		10	10	100	100		
20/12		10	10	100	100		
21/12		10	10	100	100		
22/12		10	10	100	100		
23/12		10	10	100	100		
24/12		10	10	100	100		
25/12		10	10	100	100		
26/12		10	10	100	100		
27/12		10	10	100	100		
28/12		10	10	100	100		
29/12		10	10	100	100		
30/12		10	10	100	100		
1/1		10	10	100	100		
2/1		10	10	100	100		
3/1		10	10	100	100		
4/1		10	10	100	100		
5/1		10	10	100	100		
6/1		10	10	100	100		
7/1		10	10	100	100		
8/1		10	10	100	100		
9/1		10	10	100	100		
10/1		10	10	100	100		
11/1		10	10	100	100		
12/1		10	10	100	100		
13/1		10	10	100	100		
14/1		10	10	100	100		
15/1		10	10	100	100		
16/1		10	10	100	100		
17/1		10	10	100	100		
18/1		10	10	100	100		
19/1							

Lampiran 8 Foto pada saat evaluasi sistem di apotek ridha farma

