Contents

SYSTEM ARCHITECTURE DOCUMENTATION

Application Overview

Application Name: Koperasi Karyawan Digital Platform
Architecture Pattern: MVC (Model-View-Controller)
Framework: Laravel 11.x
Deployment Model: Monolithic Web Application
Platform: Web-based (Responsive PWA-ready)

HIGH-LEVEL ARCHITECTURE

```
                    CLIENT LAYER

  Desktop        Tablet         Mobile
  Browser        Browser        Browser
```

```
               HTTPS (SSL/TLS)




               WEB SERVER LAYER

            Nginx / Apache HTTP Server
- SSL Termination
- Static Asset Serving
- Reverse Proxy to PHP-FPM
- Gzip Compression





             APPLICATION LAYER (Laravel)


                  ROUTING LAYER

  web.php                 api.php
  (Web Routes)          (API Routes)





                MIDDLEWARE LAYER
- Authentication (Sanctum)
- Authorization (Policy, Gates)
- CSRF Protection
- Rate Limiting
- Session Management




            CONTROLLER LAYER (MVC)

  Member            Finance           Commerce
  Controller        Controller        Controller


           BUSINESS LOGIC LAYER
  - Validation
  - Data Processing
  - Business Rules Enforcement





                MODEL LAYER (Eloquent ORM)
```

```
        Member          Loan         Product
        Model          Model          Model
                                ...
     - Relationships
     - Accessors & Mutators
     - Eloquent Scopes



                      VIEW LAYER (Blade)
     - Template Rendering
     - Alpine.js (Frontend Reactivity)
     - Tailwind CSS (Styling)







   DATABASE         STORAGE       CACHE
   LAYER            LAYER         LAYER
  (MySQL 8.0)       (Files)     (File/Redis)
```

DETAILED ARCHITECTURE COMPONENTS

1. CLIENT LAYER

Supported Browsers: - Chrome 90+ - Firefox 88+ - Safari 14+ - Edge 90+

Technologies: - **HTML5** - Semantic markup - **CSS3** - Tailwind CSS framework - **JavaScript (ES6+)** - Alpine.js for reactivity - **Responsive Design** - Mobile-first approach

Features: - Progressive Web App (PWA) ready - Offline capability (service workers) - Push notifications support

---

2. WEB SERVER LAYER

Nginx Configuration (Production):

```
server {
    listen 80;
    server_name kopkarskf.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name kopkarskf.com;
    root /var/www/koperasi/public;

    ssl_certificate /etc/letsencrypt/live/kopkarskf.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/kopkarskf.com/privkey.pem;

    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    # Static assets caching
    location ~* \.(jpg|jpeg|png|gif|ico|css|js|svg|woff|woff2)$ {
        expires 365d;
        add_header Cache-Control "public, immutable";
    }
}
```

---

3. APPLICATION LAYER (Laravel MVC)

A. Routing Layer

```php
// routes/web.php
Route::middleware(['auth', 'verified'])->group(function () {
    Route::get('/dashboard', [DashboardController::class, 'index'])
        ->name('dashboard');

    Route::resource('members', MemberController::class);
    Route::resource('loans', LoanController::class);
    Route::resource('products', ProductController::class);
    // ... more routes
});
```

Route Groups: - `auth` - Authenticated users only - `verified` - Email verified users - `role:admin` - Admin-only access - `throttle` - Rate limiting

B. Middleware Stack

```php
// app/Http/Kernel.php
protected $middlewareGroups = [
    'web' => [
        \App\Http\Middleware\EncryptCookies::class,
        \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
        \Illuminate\Session\Middleware\StartSession::class,
        \Illuminate\View\Middleware\ShareErrorsFromSession::class,
        \App\Http\Middleware\VerifyCsrfToken::class,
        \Illuminate\Routing\Middleware\SubstituteBindings::class,
        \App\Http\Middleware\GlobalSettings::class, // Custom
    ],
];
```

Custom Middleware: - `GlobalSettings` - Load koperasi settings for all requests - `RoleMiddleware` - Role-based access control - `AuditMiddleware` - Log user activities

C. Controller Layer Controller Structure:

```
  app/Http/Controllers/
     Auth/
        LoginController.php
        RegisterController.php
     MemberController.php
     SavingController.php
     LoanController.php
     TransactionController.php
     ProductController.php
     JournalController.php
     DocumentController.php
     ... (50+ controllers)
```

Example Controller Pattern:

```
namespace App\Http\Controllers;

class LoanController extends Controller
{
    public function index()
    {
        // Authorization check
        $this->authorize('viewAny', Loan::class);

        // Business logic
        $loans = Loan::with(['member', 'payments'])
            ->when(request('status'), function($q, $status) {
                $q->where('status', $status);
            })
            ->paginate(20);

        // Return view
        return view('loans.index', compact('loans'));
    }

    public function store(Request $request)
    {
        // Validation
        $validated = $request->validate([...]);

        // Create loan with transaction
        DB::transaction(function() use ($validated) {
            $loan = Loan::create($validated);
            $loan->generatePayments();
            $loan->createJournalEntry();
        });

        // Notification
        event(new LoanCreated($loan));
        return redirect()->route('loans.index');
    }
}
```

D. Model Layer (Eloquent ORM) **Model Structure:**

```
  app/Models/
     User.php
     Member.php
     Saving.php
     Loan.php
     LoanPayment.php
     Product.php
     Transaction.php
     JournalEntry.php
     ... (60+ models)
```

Example Model:

```
namespace App\Models;

class Loan extends Model
{
    protected $fillable = [...];
    protected $casts = [
        'amount' => 'decimal:2',
        'approved_at' => 'datetime',
    ];

    // Relationships
    public function member() {
        return $this->belongsTo(Member::class);
    }

    public function payments() {
        return $this->hasMany(LoanPayment::class);
    }

    // Accessors
    public function getRemainingBalanceAttribute() {
        return $this->amount - $this->payments->sum('amount');
    }

    // Business Logic
    public function generatePayments() {
        // Complex calculation...
    }
}
```

E. View Layer (Blade Templates) **View Structure:**

```
  resources/views/
     layouts/
        app.blade.php (Master template)
        sidebar.blade.php
        navbar.blade.php
```

```
    dashboard.blade.php
    members/
       index.blade.php
       create.blade.php
       show.blade.php
    loans/
    products/
    ... (100+ blade files)
```

Technology Stack: - **Blade** - Laravel templating engine - **Alpine.js** - Lightweight reactive framework - **Tailwind CSS** - Utility-first CSS - **Chart.js** - Data visualization

---

4. DATA LAYER

A. Database (MySQL)

Connection Pool:

```
// config/database.php
'mysql' => [
    'driver' => 'mysql',
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'koperasi'),
    'username' => env('DB_USERNAME', 'root'),
    'password' => env('DB_PASSWORD', ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'strict' => true,
    'engine' => 'InnoDB',
],
```

Query Optimization: - Eager loading (N+1 prevention) - Index on foreign keys - Query result caching

B. File Storage

Storage Structure:

```
storage/app/public/
  members/          → Member photos
  products/         → Product images
  documents/        → Uploaded docs
  logos/            → Koperasi logos
  announcements/    → Announcement images
  backups/          → Database backups
```

Storage Configuration:

```
// config/filesystems.php
'public' => [
    'driver' => 'local',
    'root' => storage_path('app/public'),
    'url' => env('APP_URL').'/storage',
    'visibility' => 'public',
],
```

C. Caching Layer

Cache Strategy: - **Config Cache:** `php artisan config:cache` - **Route Cache:** `php artisan route:cache` - **View Cache:** `php artisan view:cache` - **Query Cache:** Result caching for expensive queries

```
// Example cached query
$members = Cache::remember('active_members', 3600, function() {
    return Member::where('status', 'active')->get();
});
```

EXTERNAL INTEGRATIONS

```
                  LARAVEL APPLICATION







 Midtrans     SMTP    WhatsApp    QR Code     Google
 Payment     Email    API         Server      Drive
 Gateway              (Fonnte)                Backup
```

Integration Details:

Service	Purpose	Protocol	Config Location
Midtrans	Payment processing	REST API	`.env` (MIDTRANS_*)
SMTP (Gmail)	Email notifications	SMTP	`.env` (MAIL_*)
Fonnte/Twilio	WhatsApp messaging	REST API	`.env` (WHATSAPP_*)
QR Server	QR code generation	HTTP GET	Hardcoded URL
Google Drive	Cloud backup	OAuth 2.0	`config/backup.php`

SECURITY ARCHITECTURE

```
                    SECURITY LAYERS


  1. TRANSPORT SECURITY
       HTTPS/TLS 1.3
       SSL Certificate (Let's Encrypt)

  2. APPLICATION SECURITY
       CSRF Token Protection
       XSS Prevention (Blade escaping)
       SQL Injection Prevention (Eloquent)
       Password Hashing (bcrypt)
       Rate Limiting (throttle middleware)

  3. AUTHENTICATION & AUTHORIZATION
       Laravel Sanctum (Session-based)
       Role-Based Access Control (RBAC)
       Policies & Gates
       Multi-factor Authentication (planned)

   4. DATA SECURITY
       Encrypted Sensitive Fields
       Audit Logging (all CRUD operations)
       Daily Encrypted Backups
       GDPR Compliance Ready
```

Security Best Practices: - All forms protected with CSRF token - User input sanitized & validated - Passwords never stored plain-text (bcrypt) - File uploads: type & size validation - API rate limiting: 60 requests/minute - Session timeout: 120 minutes

---

PERFORMANCE OPTIMIZATION

1. Frontend Optimization

```
ASSETS PIPELINE:

  Source Files   (CSS, JS, Images)




   Laravel Mix   (Asset compilation)
   - Minify CSS
   - Minify JS
   - Image Opt




 public/build/   (Optimized assets)
```

Techniques: - CSS/JS minification - Image lazy loading - WebP image format - Gzip compression - Browser caching (365 days for static)

2. Backend Optimization

Query Optimization:

```
//  Bad: N+1 Problem
$members = Member::all();
foreach($members as $member) {
    echo $member->user->name; // N queries
}

//  Good: Eager Loading
$members = Member::with('user')->get();
foreach($members as $member) {
    echo $member->user->name; // 2 queries only
}
```

Indexing: - All foreign keys indexed - Composite indexes for common queries - Unique constraints for business keys

3. Caching Strategy

```
// Configuration cache (production only)
php artisan config:cache
php artisan route:cache
php artisan view:cache

// Query result cache
Cache::remember('dashboard_stats', 600, function() {
    return [
        'total_members' => Member::count(),
        'total_savings' => Saving::sum('amount'),
        // ... expensive queries
    ];
});
```

SCALABILITY ARCHITECTURE

Current Setup (Single Server)

```
         SINGLE VPS SERVER

     Nginx + PHP-FPM + MySQL
     RAM: 4GB
     CPU: 2 cores
     Storage: 50GB SSD


  Capacity: ~100 concurrent users
```

Scalability Roadmap (Future)

Stage 1: Vertical Scaling - Upgrade VPS (4→8GB RAM, 2→4 CPU cores) - Capacity: ~500 concurrent users

Stage 2: Horizontal Scaling

```
  Web Server         Web Server         Web Server
   (Nginx)            (Nginx)            (Nginx)






                       Load
                     Balancer
```

```
                        Database
                         Master



                        Database
                        Replica
```

Stage 3: Microservices (Long-term) - Separate service for POS transactions - Separate service for reporting - Message queue (Redis/RabbitMQ) - API Gateway

---

DEPLOYMENT ARCHITECTURE

Development Environment

```
     LARAGON (Windows)
  - PHP 8.2
  - MySQL 8.0
  - Apache/Nginx
  - Redis (optional)
```

Production Environment

```
      VPS (Ubuntu 22.04 LTS)

     Application Stack
       Nginx 1.24
       PHP 8.2 (FPM)
       MySQL 8.0
       Redis 7.0 (cache & sessions)
       Supervisor (queue workers)


  Security:
    UFW Firewall
    Fail2ban
    SSL via Let's Encrypt

  Backup:
    Daily cron backup to Google Drive
```

CI/CD Pipeline (Git-based Deploy)

```
Local Dev

     git push

GitHub/GitLab
```

```
     git pull (on server)

Production Server

      composer install
      php artisan migrate
      php artisan config:cache
      php artisan route:cache
      sudo systemctl reload php8.2-fpm
```

MONITORING & LOGGING

Application Monitoring

```
          MONITORING STACK


  1. Application Logs
       storage/logs/laravel.log

  2. Audit Logs
       Database: audit_logs table

  3. Error Tracking (Recommended)
       Sentry / Bugsnag

  4. Uptime Monitoring
       UptimeRobot / Pingdom

  5. Analytics
       Google Analytics
```

Log Rotation:

```
# Laravel daily log rotation
'daily' => [
    'driver' => 'daily',
    'path' => storage_path('logs/laravel.log'),
    'level' => env('LOG_LEVEL', 'debug'),
    'days' => 14,
],
```

TECHNOLOGY STACK SUMMARY

Layer	Technology	Version
Backend	PHP	8.2+
Framework	Laravel	11.x
Database	MySQL	8.0+
Cache	File/Redis	7.0+

Layer	Technology	Version
Web Server	Nginx	1.24+
Frontend	Alpine.js	3.x
CSS	Tailwind CSS	3.x
PDF	DomPDF	2.x
Payments	Midtrans SDK	Latest
Version Control	Git	2.x
Deployment	Manual/Forge	-

SYSTEM REQUIREMENTS

Server Minimum Requirements:

- **OS:** Ubuntu 20.04+ / Debian 11+
- **PHP:** 8.2+
- **MySQL:** 8.0+ / MariaDB 10.6+
- **RAM:** 2GB minimum, 4GB recommended
- **Storage:** 50GB SSD
- **Bandwidth:** 100GB/month

PHP Extensions Required:

```
- OpenSSL
- PDO (MySQL)
- Mbstring
- Tokenizer
- XML
- Ctype
- JSON
- BCMath
- GD / Imagick
- Zip
```

BACKUP & DISASTER RECOVERY

```
      BACKUP ARCHITECTURE


 Daily Backup (Automated - 02:00 AM)

   1. Database Dump (SQL)
   2. File System (storage/)
   3. .env file





   Encrypt with GPG
```

```
     Upload to Google Drive


  Retention Policy:
  - Daily: 30 days
  - Weekly: 3 months
  - Monthly: 1 year
```

Recovery Time Objective (RTO): < 4 hours
Recovery Point Objective (RPO): < 24 hours

DOCUMENTATION & SUPPORT

Architecture Documentation: - System Architecture: `ARCHITECTURE.md` (this file) - Database Schema: `DATABASE_SCHEMA.md` - Features List: `FEATURES.md` - Deployment Guide: `DEPLOYMENT.md` - API Reference: `API_DOCUMENTATION.md` (if needed)

Technical Support: - Email: dev@kopkarskf.com - Issue Tracker: GitHub Issues (if open-source)

Document Version: 1.0
Last Updated: 17 January 2026
Maintainer: Architecture Team
Review Cycle: Quarterly