**Smart Notification System**

**Software Requirements Specification (SRS)**

**Version: 1.0**

**Date: May 29, 2025**

1. Introduction

1.1 Purpose

เอกสารนี้มีวัตถุประสงค์เพื่อกำหนดข้อกำหนดสำหรับระบบ Smart Notification System ที่พัฒนาด้วย PHP Laravel สำหรับการส่งการแจ้งเตือนอัตโนมัติในองค์กร

1.2 Scope

ระบบ Smart Notification จะเป็นระบบแจ้งเตือนแบบอัตโนมัติที่สามารถ:

* เชื่อมต่อกับระบบ LDAP เพื่อจัดการผู้ใช้
* ส่งการแจ้งเตือนผ่าน Microsoft Teams และ Email
* ใช้ RabbitMQ ในการจัดการคิวข้อความ
* เก็บข้อมูลใน PostgreSQL
* บันทึกและติดตาม Log การใช้งาน

1.3 Definitions and Acronyms

* LDAP: Lightweight Directory Access Protocol
* MQ: Message Queue (RabbitMQ)
* API: Application Programming Interface
* JWT: JSON Web Token
* CRUD: Create, Read, Update, Delete

1.4 References

* Laravel Framework Documentation
* RabbitMQ Documentation
* Microsoft Graph API Documentation
* PostgreSQL Documentation

2. Overall Description

2.1 Product Perspective

ระบบ Smart Notification เป็นระบบแจ้งเตือนแบบรวมศูนย์ที่ทำงานร่วมกับ:

* ระบบ LDAP ขององค์กร
* Microsoft Teams
* Email Server
* RabbitMQ Message Broker
* PostgreSQL Database

2.2 Product Functions

* การจัดการผู้ใช้ผ่าน LDAP Authentication
* สร้างและจัดการกลุ่มการแจ้งเตือน
* กำหนดเงื่อนไขและตารางเวลาการแจ้งเตือน
* ส่งการแจ้งเตือนผ่านหลายช่องทาง
* API Gateway สำหรับระบบภายนอก
* การจัดการ API Keys และ Authentication
* ติดตามสถานะการส่งข้อความ
* รายงานและสถิติการใช้งาน

2.3 User Classes

* System Administrator: จัดการระบบและผู้ใช้
* Notification Manager: สร้างและจัดการรูปแบบการแจ้งเตือน
* End User: รับการแจ้งเตือนและปรับแต่งการตั้งค่า
* IT Support: ตรวจสอบ logs และแก้ไขปัญหา
* API Consumer: ระบบภายนอกที่เรียกใช้งานผ่าน API Gateway
* API Administrator: จัดการ API Keys และสิทธิการเข้าถึง

2.4 Operating Environment

* Server: Linux/Windows Server
* Web Server: Apache/Nginx
* Database: PostgreSQL 13+
* Message Queue: RabbitMQ 3.8+
* PHP: PHP 8.1+ with Laravel 10+
* Frontend: Vue.js/React (Optional)

2.5 Constraints

* ต้องใช้ LDAP ขององค์กรในการ Authentication
* จำกัดจำนวนการส่งข้อความต่อนาทีตาม API Limits
* ข้อมูลส่วนบุคคลต้องปฏิบัติตาม PDPA

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Authentication and Management

FR-01: LDAP Authentication

* ระบบต้องสามารถเชื่อมต่อกับ LDAP Server เพื่อตรวจสอบข้อมูลผู้ใช้
* รองรับการ Login ผ่าน LDAP credentials
* ดึงข้อมูลผู้ใช้จาก LDAP (ชื่อ, อีเมล, แผนก, ตำแหน่ง)

FR-02: User Profile Management

* แสดงข้อมูลโปรไฟล์จาก LDAP
* อนุญาตให้ผู้ใช้ปรับแต่งการตั้งค่าการแจ้งเตือน
* จัดการ Preference การรับข้อความ

3.1.2 Notification Management

FR-03: Notification Template Creation

* สร้างเทมเพลตการแจ้งเตือนแบบต่างๆ
* รองรับ Rich Text และ Markdown
* กำหนดตัวแปรแบบ Dynamic (placeholder)

FR-04: Notification Scheduling

* กำหนดเวลาส่งการแจ้งเตือน (ทันที, กำหนดเวลา, เป็นงวด)
* รองรับ Cron Expression สำหรับการกำหนดเวลาซับซ้อน
* จัดการ Time Zone

FR-05: Recipient Management

* เลือกผู้รับจาก LDAP (บุคคล, กลุ่ม, แผนก)
* สร้างกลุ่มการแจ้งเตือนแบบกำหนดเอง
* รองรับการกรองผู้รับตามเงื่อนไข

3.1.3 Message Delivery

FR-06: Microsoft Teams Integration

* ส่งข้อความไปยัง Teams Channel
* ส่งข้อความส่วนตัวใน Teams
* รองรับ Adaptive Cards

FR-07: Email Notification

* ส่งอีเมลผ่าน SMTP
* รองรับ HTML และ Plain Text
* จัดการ Attachment

FR-08: Message Queue Processing

* ใช้ RabbitMQ ในการจัดการคิวข้อความ
* รองรับ Retry Mechanism
* Priority Queue สำหรับข้อความเร่งด่วน

3.1.5 API Gateway Management

FR-12: API Key Management

* สร้างและจัดการ API Keys สำหรับระบบภายนอก
* กำหนดสิทธิการเข้าถึงและข้อจำกัดต่อ API Key
* Expire Date และ Auto-renewal สำหรับ API Keys
* Revoke API Keys เมื่อจำเป็น

FR-13: External API Endpoints

* API สำหรับส่งการแจ้งเตือนจากระบบภายนอก
* API สำหรับตรวจสอบสถานะการส่ง
* API สำหรับดึงรายการผู้ใช้จาก LDAP
* API สำหรับสร้างและจัดการกลุ่มการแจ้งเตือน

FR-14: API Rate Limiting

* จำกัดจำนวนการเรียกใช้ API ต่อ API Key
* แยกแยะ Rate Limit ตาม API Key Level
* Throttling และ Queue สำหรับ Request ที่เกินขีดจำกัด

FR-15: API Request Validation

* ตรวจสอบ API Key และสิทธิการเข้าถึง
* Validate Request Format และ Required Fields
* Error Response ที่มีมาตรฐาน (JSON format)

3.1.6 Dual Usage Mode Support

FR-09: Delivery Status Tracking

* ติดตามสถานะการส่งข้อความ (Pending, Sent, Delivered, Failed)
* แสดงเหตุผลของการส่งที่ล้มเหลว
* Notification History
* API สำหรับ Query สถานะจากระบบภายนอก

FR-10: System Logging

* บันทึก User Activities
* บันทึกการส่งข้อความทั้งหมด
* บันทึก System Errors และ Exceptions
* บันทึกการเข้าสู่ระบบ
* บันทึก API Usage และ API Key Activities

FR-11: Reporting and Analytics

* รายงานสถิติการส่งข้อความ
* รายงานการใช้งานของผู้ใช้
* Dashboard แสดงข้อมูลแบบ Real-time
* Export รายงานเป็น PDF/Excel
* รายงานการใช้งาน API แยกตาม API Key
* API Usage Analytics และ Performance Metrics

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

NFR-01: Response Time

* การตอบสนองของ Web Interface ไม่เกิน 3 วินาที
* การประมวลผลข้อความในคิวไม่เกิน 30 วินาที
* การค้นหาข้อมูล Log ไม่เกิน 5 วินาที

NFR-02: Throughput

* รองรับการส่งข้อความ 1,000 ข้อความต่อนาที
* รองรับผู้ใช้งานพร้อมกัน 100 คน (Web Interface)
* รองรับ API Calls 5,000 requests ต่อนาที
* ประมวลผลคิวได้ 10,000 jobs ต่อชั่วโมง

3.2.2 Security Requirements

NFR-03: Authentication and Authorization

* ใช้ LDAP สำหรับ Authentication (Web Interface)
* ใช้ API Key Authentication สำหรับ External Systems
* ใช้ JWT Token สำหรับ Session Management (Web Interface)
* Role-based Access Control
* Session Timeout 8 ชั่วโมง
* API Rate Limiting per API Key

NFR-04: Data Security

* เข้ารหัสข้อมูลสำคัญใน Database
* ใช้ HTTPS สำหรับการสื่อสาร
* Audit Trail สำหรับการเปลี่ยนแปลงข้อมูล
* เข้ารหัส API Keys ใน Database
* API Request/Response Logging สำหรับ Security Audit

3.2.3 Reliability Requirements

NFR-05: Availability

* ระบบต้องพร้อมใช้งาน 99.5% ของเวลา
* Downtime สำหรับ Maintenance ไม่เกิน 4 ชั่วโมงต่อเดือน

NFR-06: Fault Tolerance

* Auto-retry สำหรับการส่งข้อความที่ล้มเหลว
* Graceful Degradation เมื่อบริการภายนอกไม่พร้อมใช้งาน
* Database Connection Pooling

3.2.4 Scalability Requirements

NFR-07: Horizontal Scaling

* รองรับการเพิ่ม Worker Process สำหรับ Queue
* Database Connection Scaling
* Load Balancing Ready

4. System Architecture

4.1 High-Level Architecture

┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐

│ Web Browser │────│ Laravel App │────│ PostgreSQL │

└─────────────────┘ └──────────────────┘ └─────────────────┘

│

┌────────┴────────┐

┌─────────────────┐ │ RabbitMQ │

│ External Systems│────│ │

│ (API Calls) │ └────────┬────────┘

└─────────────────┘ │

│ ┌───────┼───────┐

│ │ │ │

│ ┌────────▼──┐ ┌──▼───┐ ┌─▼────┐

│ │ MS Teams │ │Email │ │ LDAP │

│ │ API │ │Server│ │Server│

│ └───────────┘ └──────┘ └──────┘

│

▼

┌──────────────────┐

│ API Gateway │

│ (Laravel API) │

└──────────────────┘

4.2 Component Description

* Laravel Application: Core business logic และ Web Interface
* API Gateway: RESTful API สำหรับระบบภายนอก
* Queue Workers: ประมวลผลข้อความจาก RabbitMQ
* Notification Services: จัดการการส่งข้อความ
* LDAP Integration: การเชื่อมต่อกับ LDAP
* Database Layer: จัดการข้อมูลใน PostgreSQL
* API Key Management: จัดการ Authentication สำหรับ External Systems

5. User Interface Requirements

5.1 Web Interface

* Responsive Design รองรับ Desktop และ Mobile
* Modern UI Framework (Bootstrap/Tailwind CSS)
* Dashboard แสดงสถิติและสถานะ
* Form Builder สำหรับสร้าง Notification Template

5.2 Key Screens

1. Login Page: LDAP Authentication
2. Dashboard: สรุปสถิติและกิจกรรมล่าสุด
3. Notification Management: จัดการเทมเพลตและการส่ง
4. User Management: จัดการผู้ใช้และกลุ่ม
5. API Key Management: จัดการ API Keys สำหรับระบบภายนอก
6. API Usage Analytics: สถิติการใช้งาน API
7. Reports: รายงานและสถิติ
8. System Logs: ตรวจสอบ Log

6. Data Requirements

6.1 Database Schema

6.1.1 Core Tables

* users: ข้อมูลผู้ใช้จาก LDAP
* notification\_templates: เทมเพลตการแจ้งเตือน
* notification\_groups: กลุ่มผู้รับ
* notifications: รายการการแจ้งเตือน
* notification\_logs: ประวัติการส่ง
* user\_preferences: การตั้งค่าของผู้ใช้
* api\_keys: API Keys สำหรับระบบภายนอก
* api\_usage\_logs: การใช้งาน API

6.1.2 Logging Tables

* activity\_logs: กิจกรรมของผู้ใช้
* system\_logs: Log ระบบ
* delivery\_logs: สถานะการส่งข้อความ

6.2 Data Retention Policy

* Notification Logs: เก็บ 1 ปี
* Activity Logs: เก็บ 6 เดือน
* System Logs: เก็บ 3 เดือน

7. External Interface Requirements

7.1 LDAP Integration

* Protocol: LDAP v3
* Authentication: Bind Authentication
* Data Sync: ทุก 6 ชั่วโมง
* Attributes: cn, mail, department, title

7.2 Microsoft Teams API

* Protocol: Microsoft Graph API
* Authentication: OAuth 2.0
* Permissions: ChannelMessage.Send, Chat.ReadWrite
* Rate Limits: 300 requests per minute

7.3 Email Integration

* Protocol: SMTP/IMAP
* Security: TLS/SSL
* Authentication: Username/Password หรือ OAuth

7.4 RabbitMQ Configuration

* Exchange Type: Topic Exchange
* Queue Durability: Persistent
* Message TTL: 24 hours
* Dead Letter Queue: Enabled

8. Quality Assurance

8.1 Testing Requirements

* Unit Testing: Coverage 80%+
* Integration Testing: API และ Database
* Load Testing: 1,000 concurrent users
* Security Testing: Penetration Testing

8.2 Acceptance Criteria

* ระบบส่งข้อความได้สำเร็จ 99% ของครั้ง
* ผู้ใช้สามารถเข้าสู่ระบบผ่าน LDAP ได้
* การแจ้งเตือนส่งถึงผู้รับภายใน 5 นาที
* ระบบแสดงสถานะการส่งแบบ Real-time

8.3 Performance Benchmarks

* Page Load Time: < 2 วินาที
* API Response Time: < 1 วินาที
* Database Query Time: < 500ms
* Queue Processing Rate: 100 messages/minute

9. Implementation Timeline

Phase 1 (4 สัปดาห์)

* Setup Laravel Framework
* LDAP Integration
* Basic Authentication

Phase 2 (6 สัปดาห์)

* Database Design และ Models
* RabbitMQ Integration
* Basic Notification System

Phase 3 (4 สัปดาห์)

* Microsoft Teams Integration
* Email Integration
* Web Interface

Phase 4 (3 สัปดาห์)

* Logging และ Monitoring
* Reporting System
* Testing และ Bug Fixes

Phase 5 (2 สัปดาห์)

* Performance Optimization
* Documentation
* Deployment

10. Risk Assessment

10.1 Technical Risks

* LDAP Connection Issues: จัดเตรียม Fallback Authentication
* External API Rate Limits: Implement Queue และ Retry Logic
* Database Performance: Indexing และ Query Optimization

10.2 Operational Risks

* Server Downtime: High Availability Setup
* Data Loss: Regular Backup และ Disaster Recovery
* Security Breach: Regular Security Audit

11. Appendices

11.1 API Endpoints

Authentication APIs

* POST /api/auth/login - LDAP Authentication (Web Interface)
* POST /api/auth/validate-key - Validate API Key

Notification APIs (External)

* POST /api/v1/notifications/send - Send notification via API
* POST /api/v1/notifications/bulk - Send bulk notifications
* GET /api/v1/notifications/{id}/status - Check delivery status
* GET /api/v1/notifications/history - Get notification history

User Management APIs (External)

* GET /api/v1/users - List users from LDAP
* GET /api/v1/groups - List notification groups
* POST /api/v1/groups - Create notification group

Internal APIs (Web Interface)

* GET /api/notifications - List notifications (Web UI)
* POST /api/notifications - Create notification (Web UI)
* GET /api/reports/delivery - Delivery report
* GET /api/reports/api-usage - API usage statistics

API Key Management

* GET /api/admin/api-keys - List API keys
* POST /api/admin/api-keys - Create new API key
* PUT /api/admin/api-keys/{id} - Update API key
* DELETE /api/admin/api-keys/{id} - Revoke API key

11.2 Environment Variables

* LDAP\_HOST - LDAP Server Address
* RABBITMQ\_HOST - RabbitMQ Server
* TEAMS\_CLIENT\_ID - Microsoft Teams App ID
* DB\_CONNECTION - PostgreSQL Connection
* API\_RATE\_LIMIT - API Rate Limit per minute
* API\_KEY\_EXPIRE\_DAYS - API Key expiration period

11.4 API Request/Response Examples

Send Notification Request

POST /api/v1/notifications/send

Headers:

X-API-Key: your-api-key-here

Content-Type: application/json

{

"template\_id": "urgent-alert",

"recipients": ["john.doe@company.com", "team-leads"],

"channels": ["teams", "email"],

"data": {

"title": "System Alert",

"message": "Database maintenance scheduled",

"priority": "high"

},

"schedule\_at": "2025-05-30T10:00:00Z"

}

Send Notification Response

{

"success": true,

"notification\_id": "noti\_12345",

"status": "queued",

"estimated\_delivery": "2025-05-30T10:02:00Z",

"recipients\_count": 15,

"message": "Notification queued successfully"

}

Check Status Request

GET /api/v1/notifications/noti\_12345/status

Headers:

X-API-Key: your-api-key-here

Check Status Response

{

"notification\_id": "noti\_12345",

"status": "delivered",

"sent\_at": "2025-05-30T10:01:30Z",

"delivery\_stats": {

"total": 15,

"delivered": 14,

"failed": 1,

"pending": 0

},

"channels": {

"teams": {"delivered": 14, "failed": 0},

"email": {"delivered": 14, "failed": 1}

}

}

* SendTeamsNotification - ส่งข้อความไป Teams
* SendEmailNotification - ส่งอีเมล
* SyncLdapUsers - ซิงค์ข้อมูลจาก LDAP
* GenerateReports - สร้างรายงาน

*# Smart Notification System - Laravel Project Setup*

*# 1. สร้างโปรเจค Laravel ใหม่*

composer create-project laravel/laravel smart-notification-system

cd smart-notification-system

*# 2. ติดตั้ง packages ที่จำเป็น*

composer require adldap2/adldap2-laravel *# LDAP Integration*

composer require vladimir-yuldashev/laravel-queue-rabbitmq *# RabbitMQ Queue*

composer require microsoft/microsoft-graph *# Microsoft Teams API*

composer require swiftmailer/swiftmailer *# Email Integration*

composer require tymon/jwt-auth *# JWT Authentication*

composer require spatie/laravel-permission *# Role & Permission*

composer require spatie/laravel-activitylog *# Activity Logging*

composer require maatwebsite/excel *# Excel Export*

composer require barryvdh/laravel-dompdf *# PDF Generation*

composer require predis/predis *# Redis (for caching)*

*# 3. ติดตั้ง development packages*

composer require --dev laravel/telescope *# Laravel Telescope for debugging*

composer require --dev barryvdh/laravel-debugbar *# Debug bar*

*# 4. สร้าง Models, Controllers, และ Migrations*

php artisan make:model User -mcr

php artisan make:model NotificationTemplate -mcr

php artisan make:model NotificationGroup -mcr

php artisan make:model Notification -mcr

php artisan make:model NotificationLog -mcr

php artisan make:model ApiKey -mcr

php artisan make:model UserPreference -mcr

*# 5. สร้าง Jobs สำหรับ Queue Processing*

php artisan make:job SendTeamsNotification

php artisan make:job SendEmailNotification

php artisan make:job SyncLdapUsers

php artisan make:job GenerateReports

*# 6. สร้าง Services*

php artisan make:provider NotificationServiceProvider

php artisan make:provider LdapServiceProvider

*# 7. สร้าง Middleware*

php artisan make:middleware ApiKeyMiddleware

php artisan make:middleware RateLimitMiddleware

*# 8. สร้าง Commands*

php artisan make:command SyncLdapUsersCommand

php artisan make:command ProcessNotificationQueue

*# 9. Publish และ setup packages*

php artisan vendor:publish --provider="Adldap\Laravel\AdldapServiceProvider"

php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"

php artisan vendor:publish --provider="Spatie\Activitylog\ActivitylogServiceProvider"

*# 10. Generate JWT secret*

php artisan jwt:secret

*# 11. สร้าง database และรัน migrations*

php artisan migrate

php artisan db:seed

*# 12. สร้าง storage links*

php artisan storage:link

*# 13. Clear และ optimize*

php artisan config:clear

php artisan cache:clear

php artisan route:clear

php artisan view:clear

php artisan optimize