PrivacyLayer - Guardrails and Requirements (Business Perspective)

📋 Table of Contents

🎯 Business Concept & Vision

Vision

"Every company should be able to process their data securely and GDPR-compliant without losing efficiency."

Mission

We provide a complete solution for automated anonymization of personally identifiable data that helps companies minimize compliance risks and optimize their data processing workflows.

Business Value

- **Cost Savings**: 80-90% reduction in manual work
- **Compliance Costs**: 60-70% reduction through automatic documentation
- **Error Costs**: 95% reduction in compliance violations
- **Risk Minimization**: GDPR compliance out-of-the-box

🔧 Core Requirements

1. Intelligent Anonymization

The system must automatically detect various types of personally identifiable data:

Supported Data Types

- **Names**: Personal and company names
- **Email Addresses**: Automatic detection and replacement
- **Phone Numbers**: German and international formats
- **Addresses**: Complete address data
- **IBAN**: Bank account information
- **Credit Card Numbers**: PCI-DSS compliant handling
- **Social Security Numbers**: SSN detection

Anonymization Format

```
Original: "Hello, I am Max Mustermann. My email is max@example.com."
Anonymized: "Hello, I am {{Name_1}}. My email is {{Email_1}}."
```

2. Re-Identification

Original data must be safely restorable:

Approval Process

- Each re-identification requires a legitimate reason
- Complete logging of all access
- Time limitation: Automatic deletion after defined time

3. Multi-Tenant Architecture

Each customer has their own isolated environment:

Data Isolation

- No mixing of tenant data
- Individual configuration per customer
- Separate audit trails per customer

4. Configurable Filters

Customers can create their own anonymization rules:

Filter Types

- **Regex Patterns**: For complex detection patterns
- **String Matches**: For exact text search

- **String Lists**: For multiple specific terms
- **Priorities**: Define order of application

5. Advanced Features

Dynamic Detection with Local LLM

A local language model analyzes requests and identifies potentially critical strings:

- **Context-Aware Analysis**: LLM understands context and identifies sensitive information
- **Local Processing**: All analysis happens locally for data privacy
- **Real-Time Detection**: Dynamic identification of new patterns
- **Confidence Scoring**: LLM provides confidence levels for detected patterns

String List Injected Filter

Dynamic filtering with request-specific string lists for flexible anonymization:

- **Request-Level Filtering**: Inject specific strings for each request
- **Dynamic Context**: Adapt to specific use cases and requirements
- **Temporary Rules**: Apply filters only for specific requests
- **Flexible Configuration**: No need to pre-configure all possible strings

---

🔒 Compliance & Legal Requirements

GDPR Compliance

1. Data Minimization

- Only necessary data is processed
- Automatic deletion after retention policy
- Anonymization as standard

2. Data Portability

- Customers can export their data
- Anonymized data is safely transferable
- Compliance during system changes

3. Audit Trail

- Complete logging of all data access
- WORM-compliant logs (Write Once Read Many)
- Traceability for compliance audits

🛡️ Security Requirements

Encryption

1. Data Encryption

- All sensitive data is encrypted stored
- AES-256-GCM encryption
- Secure key management

2. Multi-Tenant Isolation

- Complete data isolation between customers
- No mixing of customer data
- Separate configurations per customer

3. Access Control

- API key-based authentication
- Audit logging of all access

Security Measures

API Security

- JWT token authentication
- API key authentication
- Input validation and sanitization

Database Security

- Encrypted connections (SSL/TLS)
- Secure password hashing (bcrypt)

⚙️ Technical Guardrails

Architecture Requirements

1. Multi-Tenant Design

- Complete isolation between tenants
- Tenant-specific API keys
- Separate audit logs per tenant
- Tenant-specific configurations

2. Scalable Architecture

- Connection pooling for database

- Caching strategies implemented

3. Fault Tolerance

- Graceful degradation on errors
- Automatic recovery
- Circuit breaker pattern
- Health checks and monitoring

Database Design (Example from PoC)

Table Structure

```sql
-- Tenants (Customers)
CREATE TABLE tenants (
    id UUID PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT NOW()
);

-- Filter Definitions
CREATE TABLE filter_definitions (
    id UUID PRIMARY KEY,
    tenant_id UUID REFERENCES tenants(id),
    name VARCHAR(255) NOT NULL,
    category VARCHAR(100) NOT NULL,
    regex_pattern TEXT,
    string_match TEXT,
    priority INTEGER DEFAULT 1,
    is_active BOOLEAN DEFAULT true
);

-- Transformations (Anonymization processes)
CREATE TABLE transformations (
    id UUID PRIMARY KEY,
    tenant_id UUID REFERENCES tenants(id),
    configuration_id UUID,
    content_size INTEGER,
    processing_time_ms INTEGER,
    status VARCHAR(50),
    context TEXT,
    created_at TIMESTAMP DEFAULT NOW(),
    expires_at TIMESTAMP
);

-- Mapping Entries (encrypted original values)
CREATE TABLE mapping_entries (
    id UUID PRIMARY KEY,
    transformation_id UUID REFERENCES transformations(id),
    placeholder VARCHAR(100) NOT NULL,
    encrypted_value TEXT NOT NULL,
    category VARCHAR(100),
    confidence DECIMAL(3,2)
);

```

```sql
-- Audit Logs (WORM-compliant)
CREATE TABLE audit_logs (
    id UUID PRIMARY KEY,
    tenant_id UUID REFERENCES tenants(id),
    user_id UUID,
    event_type VARCHAR(100) NOT NULL,
    timestamp TIMESTAMP DEFAULT NOW(),
    metadata JSONB,
    severity VARCHAR(20) DEFAULT 'INFO'
);
```

API Design (Example from PoC)

RESTful Endpoints

```
POST /api/v1/anonymize          # Single anonymization
POST /api/v1/anonymize/bulk     # Bulk anonymization
POST /api/v1/deanonymize        # Re-identification
GET  /api/v1/config/filters     # Manage filters
GET  /api/v1/audit/trail/:id    # Retrieve audit trail
```

Response Format

```json
{
  "success": true,
  "data": {
    "anonymizedContent": "Hello {{Name_1}}",
    "transformationId": "uuid-here",
    "mappingsCount": 1,
    "processingTimeMs": 150
  }
}
```

📊 Quality Requirements (if possible)

Performance Metrics

Anonymization

- **Processing Time**: ≤ 300ms for 10KB text
- **Throughput**: ≥ 1000 requests/minute
- **Error Rate**: < 0.1%

De-anonymization

- **Processing Time**: ≤ 100ms
- **Accuracy**: 100% correct restoration
- **Security**: No unauthorized access

Filter Lookup

- **Processing Time**: ≤ 100ms

- **Scalability**: Support for 500+ filters per tenant

Quality Assurance

Testing Requirements

- **Unit Tests**: 99% code coverage
- **Integration Tests**: All API endpoints
- **Performance Tests**: Load testing with realistic data
- **Security Tests**: Penetration testing and vulnerability scans

Code Quality

- **Linting**: ESLint with Airbnb standards
- **Code Review**: Mandatory for all changes
- **Documentation**: Complete API documentation
- **Monitoring**: Real-time performance monitoring

📊 Monitoring & Audit

Real-Time Monitoring

- **Database Performance**: Query times and connection pool

Application Metrics

- **Response Times**: API performance
- **Error Rates**: 4xx/5xx HTTP status
- **Throughput**: Requests per second
- **Availability**: Uptime and downtime

Audit & Compliance

Audit Trail

- **Complete Logging**: All data access
- **WORM Compliance**: Write Once Read Many
- **Immutability**: Audit logs cannot be modified

Compliance Reporting

- **Automatic Reports**: Daily/weekly/monthly
- **GDPR Compliance**: Automatic assessment
- **Export Functions**: JSON/CSV export
- **Dashboard**: Real-time compliance overview

Alerting & Notifications

Critical Alerts

- **System Down**: Immediate notification
- **Security Breaches**: Unauthorized access
- **Performance Issues**: High response times
- **Compliance Violations**: Audit trail gaps

Notification Channels

- **Email**: Detailed reports
- **Slack/Teams**: Real-time alerts
- **SMS**: Critical system failures
- **Webhook**: Integration into existing systems