Structured Documents Synthetic Data Generation

Comprehensive Design, Architecture & Code Repository Structure

---

🎯 Executive Summary

Bottom Line Up Front: This system generates high-fidelity synthetic structured documents (forms, invoices, contracts, medical records) with 95%+ OCR accuracy and 90%+ structural precision, serving Legal, Banking, Healthcare, Insurance, and Government verticals through privacy-compliant, template-based generation with real-time validation and multi-cloud delivery.

📋 1. DESIGN OVERVIEW

Definition

Structured Documents are synthetic forms, tables, invoices, receipts, contracts, and regulatory documents that follow consistent patterns but vary across domains. These documents maintain realistic layouts, content relationships, and domain-specific constraints while ensuring privacy compliance.

Key Verticals & Use Cases

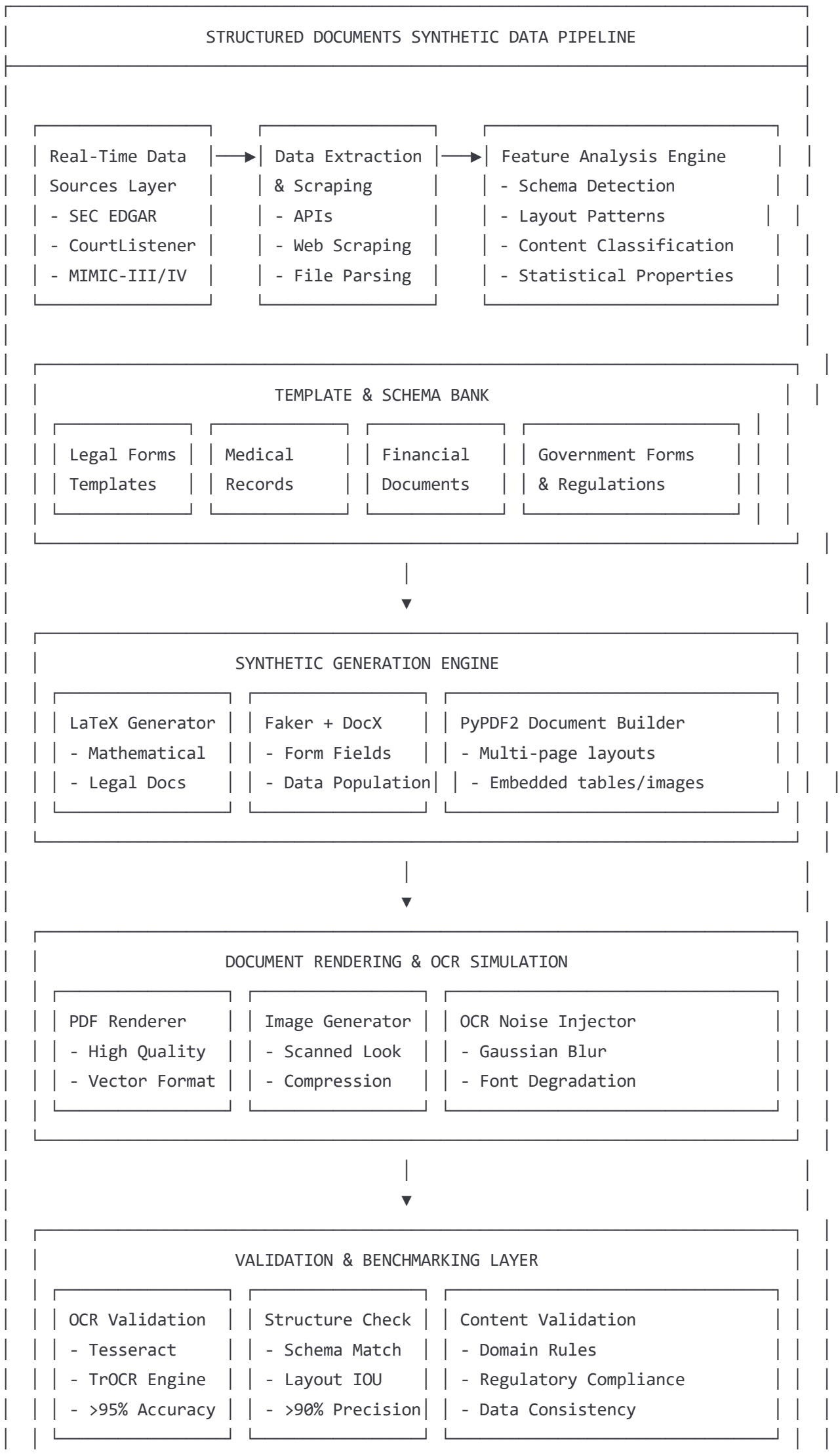
- **Legal**: Contracts, case documents, regulatory filings, compliance forms
- **Banking**: Loan applications, account statements, KYC documents, transaction records
- **Healthcare**: Patient forms, insurance claims, medical records, compliance documents
- **Insurance**: Policy documents, claims forms, underwriting materials
- **Government**: Regulatory filings, permits, applications, public records

Real-Time Data Sources Integration

- **Legal**: SEC EDGAR filings, CourtListener, CUAD Dataset, EUR-Lex
- **Healthcare**: MIMIC-III/IV datasets, CMS Medicare data, synthetic EHR datasets
- **Banking**: Open banking APIs, regulatory filings, financial statements
- **Government**: GovInfo.gov, legislation databases, public records

---

🏗️ 2. SYSTEM ARCHITECTURE


STRUCTURED DOCUMENTS SYNTHETIC DATA PIPELINE
Real-Time Data Sources Layer
- SEC EDGAR
- CourtListener
- MIMIC-III/IV
Data Extraction & Scraping
- APIs
- Web Scraping
- File Parsing
Feature Analysis Engine
- Schema Detection
- Layout Patterns
- Content Classification
- Statistical Properties
TEMPLATE & SCHEMA BANK
Legal Forms Templates
Medical Records
Financial Documents
Government Forms & Regulations
SYNTHETIC GENERATION ENGINE
LaTeX Generator
- Mathematical
- Legal Docs
Faker + DocX
- Form Fields
- Data Population
PyPDF2 Document Builder
- Multi-page layouts
- Embedded tables/images
DOCUMENT RENDERING & OCR SIMULATION
PDF Renderer
- High Quality
- Vector Format
Image Generator
- Scanned Look
- Compression
OCR Noise Injector
- Gaussian Blur
- Font Degradation
VALIDATION & BENCHMARKING LAYER
OCR Validation
- Tesseract
- TrOCR Engine
- >95% Accuracy
Structure Check
- Schema Match
- Layout IOU
- >90% Precision
Content Validation
- Domain Rules
- Regulatory Compliance
- Data Consistency
PRIVACY & COMPLIANCE TRANSFORMER
PII Anonymizer
- Name Masking
- SSN/ID Redact
- Address Anon
GDPR Compliance
- Right to be Forgotten
- Data Lineage
Differential Privacy
- Laplace Noise
- Statistical Protection
- Privacy Budget
DELIVERY & ACCESS LAYER
API Server
- FastAPI
- /generate/doc
- CLI/SDK
Cloud Storage
- S3/GCS/Azure
- Multi-format
- Signed URLs
Integration Adapters
- RAG Pipelines
- NLP-RPA Systems
- Document Processing

📁 3. CODE REPOSITORY STRUCTURE

```
structured-documents-synthetic-data/
├── README.md                           # Comprehensive documentation
├── requirements.txt                    # Python dependencies
├── setup.py                           # Package configuration
├── docker-compose.yml                 # Multi-service deployment
├── Dockerfile                         # Container definition
├── .github/
│   └── workflows/
│       ├── ci-cd.yml                  # GitHub Actions pipeline
│       ├── security-scan.yml          # Security scanning
│       └── performance-test.yml       # Load testing
│
├── configs/                           # Configuration management
│   ├── schema_bank/                   # Document templates
│   │   ├── legal/
│   │   │   ├── contract_template.json
│   │   │   ├── filing_template.json
│   │   │   └── compliance_form.json
│   │   ├── healthcare/
│   │   │   ├── patient_form.json
│   │   │   ├── insurance_claim.json
│   │   │   └── medical_record.json
│   │   ├── banking/
│   │   │   ├── loan_application.json
│   │   │   ├── account_statement.json
│   │   │   └── kyc_document.json
│   │   └── government/
│   │       ├── permit_application.json
│   │       ├── regulatory_filing.json
│   │       └── public_record.json
│   ├── compliance_rules.yaml          # GDPR, HIPAA, PCI-DSS rules
│   ├── vertical_config.yaml           # Domain-specific settings
│   ├── generation_config.yaml         # Generation parameters
│   └── validation_thresholds.yaml     # Quality benchmarks
│
├── data/                              # Data storage structure
│   ├── real_sources/                  # Real-time data ingestion
│   │   ├── legal/
│   │   │   ├── sec_edgar/             # SEC filings
│   │   │   ├── court_listener/        # Court documents
│   │   │   └── cuad_dataset/          # Contract datasets
│   │   ├── healthcare/
│   │   │   ├── mimic_samples/         # MIMIC dataset samples
│   │   │   └── cms_data/              # CMS Medicare data
│   │   └── banking/
│   │       ├── financial_reports/     # Public financial data
│   │       └── regulatory_filings/    # Banking regulations
│   ├── templates/                     # Generated templates
│   ├── synthetic/                     # Generated documents
│   │   ├── pdf/                       # PDF outputs
│   │   ├── docx/                      # Word documents
│   │   ├── json/                      # Structured data
│   │   └── annotated/                 # Labeled datasets
│   └── validation/                    # Validation results
│
├── src/                               # Source code modules
│   ├── __init__.py
│   ├── core/                          # Core framework
│   │   ├── __init__.py
│   │   ├── pipeline.py                # Main orchestration
│   │   ├── config_manager.py          # Configuration handling
│   │   └── error_handlers.py          # Exception management
│   │
│   ├── ingestion/                     # Real-time data ingestion
│   │   ├── __init__.py
│   │   ├── sec_edgar_scraper.py       # SEC EDGAR integration
│   │   ├── court_listener_api.py      # Court data API
│   │   ├── healthcare_ingest.py       # Healthcare data sources
│   │   ├── banking_api.py             # Banking data integration
│   │   ├── government_scraper.py      # Government data sources
│   │   └── data_extractor.py          # Generic data extraction
```

```
│   │
│   ├── analysis/                  # Feature analysis engine
│   │   ├── __init__.py
│   │   ├── schema_detector.py     # Document schema analysis
│   │   ├── layout_analyzer.py     # Layout pattern extraction
│   │   ├── content_classifier.py  # Content type classification
│   │   ├── statistical_profiler.py # Statistical analysis
│   │   └── feature_extractor.py   # Feature engineering
│   │
│   ├── generation/                # Document generation
│   │   ├── __init__.py
│   │   ├── latex_generator.py     # LaTeX document generation
│   │   ├── docx_generator.py      # Word document generation
│   │   ├── pdf_generator.py       # PDF document creation
│   │   ├── template_engine.py     # Template processing
│   │   ├── faker_integration.py   # Faker data population
│   │   └── content_synthesizer.py  # Content synthesis
│   │
│   ├── rendering/                 # Document rendering & OCR
│   │   ├── __init__.py
│   │   ├── pdf_renderer.py        # High-quality PDF rendering
│   │   ├── image_generator.py     # Scanned document simulation
│   │   ├── ocr_noise_injector.py  # OCR degradation simulation
│   │   ├── layout_processor.py    # Layout manipulation
│   │   └── visual_effects.py      # Visual quality control
│   │
│   ├── validation/                # Validation & benchmarking
│   │   ├── __init__.py
│   │   ├── ocr_validator.py       # OCR accuracy testing
│   │   ├── structure_validator.py  # Document structure validation
│   │   ├── content_validator.py   # Content quality checks
│   │   ├── benchmark_engine.py    # Automated benchmarking
│   │   ├── quality_metrics.py     # Quality measurement
│   │   └── performance_monitor.py  # Performance tracking
│   │
│   ├── privacy/                   # Privacy & compliance
│   │   ├── __init__.py
│   │   ├── pii_anonymizer.py      # PII detection & masking
│   │   ├── gdpr_compliance.py     # GDPR compliance engine
│   │   ├── differential_privacy.py # DP implementation
│   │   ├── audit_logger.py        # Compliance audit logging
│   │   └── privacy_transformer.py  # Privacy transformations
│   │
│   ├── delivery/                  # Delivery & access
│   │   ├── __init__.py
│   │   ├── api_server.py          # FastAPI server
│   │   ├── cloud_uploader.py      # Multi-cloud storage
│   │   ├── cli_interface.py       # Command-line interface
│   │   ├── sdk_wrapper.py         # SDK implementation
│   │   ├── rag_integration.py     # RAG pipeline integration
│   │   └── export_manager.py      # Multi-format export
│   │
│   └── utils/                     # Utility functions
│       ├── __init__.py
│       ├── file_handlers.py       # File I/O operations
│       ├── logging_config.py      # Logging configuration
│       ├── security_utils.py      # Security utilities
│       ├── performance_utils.py   # Performance optimization
│       └── format_converters.py   # Format conversion utilities
│
├── tests/                         # Test suite
│   ├── __init__.py
│   ├── unit/                      # Unit tests
│   │   ├── test_generation.py
│   │   ├── test_validation.py
│   │   ├── test_privacy.py
│   │   └── test_delivery.py
│   ├── integration/               # Integration tests
│   │   ├── test_pipeline.py
│   │   ├── test_api_endpoints.py
│   │   └── test_cloud_integration.py
│   ├── performance/               # Performance tests
```

```
│   │   ├── test_load_generation.py
│   │   └── test_scalability.py
│   └── fixtures/                    # Test data
│       ├── sample_templates/
│       └── validation_datasets/
│
├── notebooks/                       # Jupyter notebooks
│   ├── 01_data_exploration.ipynb
│   ├── 02_template_analysis.ipynb
│   ├── 03_generation_testing.ipynb
│   ├── 04_validation_analysis.ipynb
│   ├── 05_privacy_evaluation.ipynb
│   └── 06_benchmarking_results.ipynb
│
├── scripts/                         # Automation scripts
│   ├── setup_environment.sh         # Environment setup
│   ├── run_pipeline.py              # Pipeline execution
│   ├── generate_docs.py             # Document generation script
│   ├── validate_output.py           # Validation script
│   ├── deploy_cloud.py              # Cloud deployment
│   └── benchmark_suite.py           # Benchmarking automation
│
├── cli/                            # Command-line interface
│   ├── __init__.py
│   ├── main.py                      # Main CLI entry point
│   ├── commands/
│   │   ├── generate.py              # synth doc generate
│   │   ├── validate.py              # Validation commands
│   │   ├── deploy.py                # Deployment commands
│   │   └── benchmark.py             # Benchmarking commands
│   └── utils/
│       ├── cli_helpers.py           # CLI utility functions
│       └── output_formatters.py     # Output formatting
│
├── api/                             # API implementation
│   ├── __init__.py
│   ├── main.py                      # FastAPI application
│   ├── routers/
│   │   ├── generation.py            # /generate/document endpoint
│   │   ├── validation.py            # Validation endpoints
│   │   ├── health.py                # Health check endpoints
│   │   └── admin.py                 # Admin endpoints
│   ├── models/                      # Pydantic models
│   │   ├── request_models.py        # API request schemas
│   │   ├── response_models.py       # API response schemas
│   │   └── document_models.py       # Document data models
│   └── middleware/
│       ├── auth_middleware.py       # Authentication
│       ├── rate_limiting.py         # Rate limiting
│       └── logging_middleware.py    # Request logging
│
├── sdk/                             # SDK implementation
│   ├── __init__.py
│   ├── client.py                    # Main SDK client
│   ├── document_generator.py        # sdk.generate_document()
│   ├── validators.py                # Validation functions
│   ├── exceptions.py                # SDK exceptions
│   └── examples/                    # SDK usage examples
│       ├── basic_generation.py
│       ├── batch_processing.py
│       └── advanced_templates.py
│
├── infrastructure/                  # Infrastructure as Code
│   ├── terraform/                   # Terraform configurations
│   │   ├── aws/                     # AWS infrastructure
│   │   ├── gcp/                     # Google Cloud infrastructure
│   │   └── azure/                   # Azure infrastructure
│   ├── kubernetes/                  # Kubernetes manifests
│   │   ├── deployment.yaml
│   │   ├── service.yaml
│   │   └── ingress.yaml
│   └── helm/                        # Helm charts
```

```
│       ├── Chart.yaml
│       ├── values.yaml
│       └── templates/
│
├── monitoring/                    # Monitoring & observability
│   ├── prometheus/                # Prometheus configuration
│   ├── grafana/                   # Grafana dashboards
│   └── alerts/                    # Alert configurations
│
└── docs/                          # Documentation
    ├── index.md                   # Main documentation
    ├── getting_started.md         # Quick start guide
    ├── api_reference.md           # API documentation
    ├── sdk_guide.md               # SDK documentation
    ├── deployment_guide.md        # Deployment instructions
    ├── contributing.md            # Contribution guidelines
    └── examples/                  # Usage examples
        ├── legal_documents.md
        ├── healthcare_forms.md
        └── banking_applications.md
```

🔬 4. BENCHMARKS & METRICS

Primary Quality Metrics

- **OCR Recall**: >95% (character-level accuracy)
- **Structural Precision**: >90% (layout element accuracy)
- **Content Validity**: >98% (domain rule compliance)
- **Template Fidelity**: >92% (template adherence)

Performance Benchmarks

- **Generation Speed**: <30 seconds per document
- **Throughput**: >100 documents/minute
- **Memory Usage**: <2GB per generation instance
- **Storage Efficiency**: <5MB per synthetic document

Privacy & Compliance Metrics

- **PII Detection Recall**: >99%
- **GDPR Compliance Score**: 100%
- **Differential Privacy ε**: <1.0
- **Audit Trail Completeness**: 100%

Validation Framework

python

```python
# 8-Step Evaluation Framework Implementation
validation_steps = {
    1: "Synthetic Data Generation Coverage",
    2: "Logging & Trace Analysis",
    3: "Automated Evaluation",
    4: "Component & End-to-End Testing",
    5: "Human Feedback Integration",
    6: "Bias Detection & Mitigation",
    7: "Real-World Performance",
    8: "Continuous Monitoring"
}
```

🔒 5. PRIVACY & COMPLIANCE TRANSFORMATIONS

PII Anonymization Pipeline

```python
pii_transformations = {
    "names": "faker.name() replacement",
    "ssn": "format-preserving encryption",
    "addresses": "geographic region substitution",
    "phone_numbers": "area code preservation + random",
    "emails": "domain preservation + anonymization",
    "dates": "temporal shifting within constraints"
}
```

GDPR Compliance Features

- **Right to be Forgotten**: Complete data lineage tracking
- **Data Minimization**: Generate only necessary fields
- **Purpose Limitation**: Template-based generation constraints
- **Storage Limitation**: Automated data lifecycle management

Differential Privacy Implementation

- **Laplace Mechanism**: For numerical fields
- **Exponential Mechanism**: For categorical selections
- **Privacy Budget Management**: Automated ε allocation
- **Composition Tracking**: Multi-query privacy accounting

📦 6. DELIVERY MODULES

API Endpoints

```python
# FastAPI Implementation
@app.post("/generate/document")
async def generate_document(request: DocumentRequest) -> DocumentResponse:
    """Generate synthetic structured document"""

@app.get("/validate/document/{doc_id}")
async def validate_document(doc_id: str) -> ValidationResponse:
    """Validate generated document quality"""

@app.post("/batch/generate")
async def batch_generate(batch_request: BatchRequest) -> BatchResponse:
    """Batch document generation"""
```

CLI Commands

```bash
# Command-line interface
synth doc generate --template legal_contract --count 100 --format pdf
synth doc validate --input documents/ --metrics ocr,structure
synth doc deploy --provider aws --region us-east-1
```

SDK Functions

```python
# Python SDK
from synthetic_docs import StructuredDocumentSDK

sdk = StructuredDocumentSDK(api_key="your_key")
document = sdk.generate_document(
    template="healthcare_form",
    vertical="healthcare",
    privacy_level="high"
)
```

Export Formats

- **PDF**: High-quality, OCR-ready documents
- **DOCX**: Editable Word documents with metadata
- **JSON**: Structured data with annotations
- **XML**: Schema-compliant markup
- **HTML**: Web-ready formatted documents

☁️ 7. INFRASTRUCTURE & CLOUD INTEGRATION

Multi-Cloud Support

```yaml
# Cloud provider configurations
providers:
  aws:
    storage: S3
    compute: ECS/Lambda
    database: DynamoDB
    monitoring: CloudWatch

  gcp:
    storage: Cloud Storage
    compute: Cloud Run
    database: Firestore
    monitoring: Cloud Monitoring

  azure:
    storage: Blob Storage
    compute: Container Instances
    database: Cosmos DB
    monitoring: Application Insights
```

Scalability Architecture

- **Horizontal Scaling**: Auto-scaling document generation workers
- **Load Balancing**: Intelligent request distribution
- **Caching Strategy**: Template and validation result caching
- **Database Sharding**: Distributed storage for high throughput

Monitoring & Observability

- **Metrics Collection**: Prometheus + Grafana
- **Distributed Tracing**: OpenTelemetry integration
- **Log Aggregation**: ELK Stack deployment
- **Alert Management**: PagerDuty integration

🚀 8. IMPLEMENTATION PHASES

Phase 1: Foundation (Months 1-3)

- Core generation engine development
- Basic template system implementation
- Initial validation framework
- Local development environment

Phase 2: Integration (Months 4-6)

- Real-time data source integration
- Privacy & compliance framework
- API development and testing
- Cloud deployment preparation

Phase 3: Enhancement (Months 7-9)

- Advanced generation algorithms
- Comprehensive validation suite
- Multi-cloud deployment
- Performance optimization

Phase 4: Production (Months 10-12)

- Production hardening
- Enterprise security features
- Comprehensive monitoring
- Documentation and training

📊 9. SUCCESS CRITERIA & KPIs

Technical KPIs

- **System Uptime**: >99.9%
- **API Response Time**: <2 seconds
- **Document Quality Score**: >95%
- **Security Incident Count**: 0

Business KPIs

- **Customer Adoption Rate**: >80% within 6 months
- **Document Generation Volume**: >1M documents/month
- **Cost per Document**: <$0.10
- **Customer Satisfaction**: >4.5/5.0

Compliance KPIs

- **Regulatory Audit Results**: 100% pass rate
- **Privacy Violation Incidents**: 0
- **Data Breach Incidents**: 0
- **Compliance Certification Maintenance**: 100%

🔧 10. DEVELOPMENT GUIDELINES

Code Quality Standards

- **Test Coverage**: >90%
- **Code Review**: 100% of commits
- **Documentation**: Comprehensive API docs
- **Security Scanning**: Automated vulnerability checks

Performance Requirements

- **Memory Efficiency**: <2GB per generation process
- **CPU Optimization**: Multi-threaded generation
- **Storage Optimization**: Compressed outputs
- **Network Efficiency**: Minimal bandwidth usage

Security Requirements

- **Authentication**: OAuth 2.0 + JWT
- **Authorization**: Role-based access control
- **Encryption**: AES-256 for data at rest
- **Transport Security**: TLS 1.3 for data in transit

📈 11. FUTURE ROADMAP

Short-term (3-6 months)

- Advanced ML-based generation models
- Real-time collaboration features
- Mobile SDK development
- Advanced analytics dashboard

Medium-term (6-12 months)

- Multi-language document support
- Blockchain-based audit trails
- AI-powered quality assessment
- Edge computing deployment

Long-term (12+ months)

- Quantum-resistant encryption
- Federated learning integration
- Autonomous quality improvement
- Global regulatory compliance automation

This comprehensive design provides a complete blueprint for implementing a production-ready structured documents synthetic data generation system that meets enterprise requirements for quality, security, scalability, and compliance.