

Technical Architecture Guide

NeuralFlow AI Platform v2.0

Document Version: 2.3 | Last Updated: December 15, 2024

Classification: Internal - Engineering Team

1. System Overview

The NeuralFlow AI platform is a comprehensive, cloud-native AI automation system designed for enterprise-scale deployments. Our architecture prioritizes scalability, reliability, security, and maintainability while enabling rapid development and deployment of AI-powered solutions.

Architecture Principles:

- Microservices-based for independent scaling and deployment
- Event-driven communication for loose coupling
- Multi-tenant with data isolation
- Cloud-agnostic design with provider abstraction
- API-first approach for all services

2. High-Level Architecture

CLIENT LAYER

Web Application

Mobile Apps

API Clients

Third-Party Integrations



API GATEWAY LAYER

Load Balancer

Authentication

Rate Limiting

Request Routing



APPLICATION LAYER

Document Processing

Conversational AI

Analytics Engine

Workflow Orchestration



AI/ML LAYER

LLM Gateway

Vector Database

Model Registry

Training Pipeline



DATA LAYER

PostgreSQL

Redis Cache

S3 Storage

Event Stream

3. Core Components

3.1 API Gateway

The API Gateway serves as the single entry point for all client requests, handling authentication, rate limiting, request validation, and routing to appropriate microservices.

```
# API Gateway Configuration Example
gateway:
  host: api.neuralflow-ai.com
  port: 443
  ssl: true
  rate_limit: requests_per_minute: 1000 burst: 100
  auth:
    type: jwt
    token_expiry: 3600
  routes:
    - path: /v1/documents/*
      service: document-processor
      methods: [POST, GET]
    - path: /v1/chat/*
      service: conversational-ai
      methods: [POST, GET, DELETE]
```

3.2 Document Processing Service

Handles intelligent document ingestion, OCR, extraction, classification, and analysis. Supports multiple document formats including PDF, DOCX, images, and scanned documents.

Component	Technology	Purpose
Document Parser	PyPDF2, python-docx, Pillow	Extract text and metadata from documents
OCR Engine	Tesseract, AWS Textract	Optical character recognition for images
Entity Extraction	spaCy, Custom NER Models	Identify key entities and relationships
Classification	Fine-tuned BERT, GPT-4	Categorize document types
Data Validation	Custom Rules Engine	Validate extracted data accuracy

3.3 Conversational AI Service

Powers chatbots and virtual assistants with natural language understanding, context management, and multi-turn conversation capabilities.







Important: All conversational AI implementations must include content filtering, PII detection, and conversation logging for compliance purposes.

3.4 RAG (Retrieval-Augmented Generation) System

Our RAG implementation combines vector search with large language models to provide accurate, contextual responses grounded in customer knowledge bases.

```
# RAG Pipeline Architecture
1. Document Ingestion -> Chunking (500-1000 tokens) -> Embedding Generation (text-embedding-ada-002) -> Vector Storage (Pinecone/Weaviate)
2. Query Processing -> Query Embedding -> Semantic Search (k=5-10) -> Reranking (Cohere Rerank) -> Context Assembly
3. Generation -> Prompt Construction -> LLM Inference (GPT-4, Claude) -> Response Validation -> Citation Generation
```

4. Technology Stack

 Backend Python 3.11 FastAPI Celery	 Frontend React 18 TypeScript Next.js 14	 Database PostgreSQL 15 Redis 7 MongoDB
 AI/ML OpenAI GPT-4 Claude 3 LangChain	 Infrastructure AWS/GCP Kubernetes Docker	 Monitoring Datadog Sentry Prometheus

5. Data Flow

Understanding how data flows through our system is critical for debugging, optimization, and feature development.

5.1 Document Processing Flow

Step	Action	Output	Avg Time
1	Document Upload	S3 URL, Job ID	200ms
2	Format Detection	Document Type	50ms
3	Text Extraction	Raw Text, Metadata	2-5s
4	OCR (if needed)	Recognized Text	5-15s
5	Entity Extraction	Structured Data	1-3s

Step	Action	Output	Avg Time
6	Classification	Document Category	500ms
7	Validation	Confidence Scores	300ms
8	Storage	Database Record	100ms

6. Security Architecture

Security is embedded at every layer of our architecture, from network isolation to application-level access controls.

6.1 Security Layers

Layer	Mechanism	Implementation
Network	VPC Isolation	Private subnets, NAT gateways, security groups
Application	Authentication	JWT tokens, OAuth 2.0, SSO integration
Data	Encryption	AES-256 at rest, TLS 1.3 in transit
Access Control	RBAC	Fine-grained permissions, role hierarchies
Monitoring	Audit Logs	Immutable logs, SIEM integration
Compliance	Data Residency	Region-specific deployments, data sovereignty

6.2 API Authentication Flow

```
# Authentication Sequence
1. Client Request POST /v1/auth/login Body: {email, password}
2. Credential Validation |-> Hash password (bcrypt)
```

```
└─> Query user database └─> Validate credentials 3. Token Generation └─>
Create JWT payload └─> Sign with RSA private key └─> Set expiration (1
hour) 4. Response { "access_token": "eyJ0eXAiOiJKV1...",
"refresh_token": "dGhpc2lzY...", "expires_in": 3600 }
```

7. Performance Optimization

We employ multiple strategies to ensure optimal performance at scale:

7.1 Caching Strategy

Cache Type	Use Case	TTL	Invalidation
Redis - Hot Data	Frequent queries, session data	5-60 min	Event-based
CDN - Static Assets	Images, JS, CSS files	24 hours	Version-based
Application Cache	Configuration, feature flags	15 min	Time-based
Database Query Cache	Expensive read queries	5 min	Write invalidation

8. Monitoring & Observability

Comprehensive monitoring ensures we can detect, diagnose, and resolve issues before they impact customers.

Key Metrics Tracked:

- **Golden Signals:** Latency, Traffic, Errors, Saturation
- **Business Metrics:** API usage, model accuracy, processing throughput
- **Infrastructure:** CPU, memory, disk I/O, network bandwidth

- **Cost:** Cloud spend by service, AI API costs

9. Disaster Recovery

9.1 Backup Strategy

Data Type	Backup Frequency	Retention	RTO	RPO
Production Database	Continuous	30 days	< 1 hour	< 5 min
Document Storage	Daily	90 days	< 4 hours	24 hours
Configuration	On change	Indefinite	< 30 min	0
Model Artifacts	On deployment	All versions	< 2 hours	0

10. Deployment Pipeline

```
# CI/CD Pipeline Stages
1. Code Commit (GitHub)  ↳ Trigger webhook
2. Build Stage  ↳ Run linters (flake8, black)  ↳ Run unit tests (pytest)
   ↳ Build Docker image  ↳ Push to container registry
3. Test Stage  ↳ Integration tests  ↳ Security scanning (Snyk)  ↳ Performance tests
4. Staging Deployment  ↳ Deploy to staging cluster  ↳ Run smoke tests  ↳ Manual approval gate
5. Production Deployment  ↳ Canary deployment (5% traffic)  ↳ Monitor metrics (15 min)  ↳ Gradual rollout (25%, 50%, 100%)
   ↳ Automated rollback if errors
```

11. API Endpoints Reference

Endpoint	Method	Purpose	Auth Required
/v1/documents/upload	POST	Upload document for processing	Yes
/v1/documents/{id}	GET	Retrieve document results	Yes
/v1/chat/conversation	POST	Start new conversation	Yes
/v1/chat/message	POST	Send message in conversation	Yes
/v1/analytics/query	POST	Run analytics query	Yes
/v1/health	GET	System health check	No