Architecture Design

Source File: generated-documents\technical-design\architecture-design.md
Generated: 15/07/2025 at 11:40:49
Generated by: Requirements Gathering Agent - PDF Converter

ArchitectureDesign

Generated by adpa-enterprise-framework-automation v3.2.0
Category: technical-design
Generated: 2025-07-14T20:57:28.718Z
Description:

Architecture Design Document

Project Name: ADPA – Advanced Document Processing & Automation Framework
Version: 1.0
Date: 2025-07-14
Status: Draft

Executive Summary

ADPA (Advanced Document Processing & Automation Framework) is a modular, standards-compliant automation platform designed for enterprise document generation, project management, and business analysis. Leveraging an API-first, microservices-inspired architecture with multi-provider AI orchestration, ADPA accelerates the delivery of professional, standards-based business documentation, supporting frameworks such as BABOK v3, PMBOK 7th Edition, and (in progress) DMBOK 2.0. The system enables seamless integration with enterprise tools (Confluence, SharePoint, Adobe, VCS), enforces robust security and compliance, and is built for extensibility and large-scale deployments.

---

1. System Overview

1.1 Purpose and Scope

- **System Purpose:**
  Automate professional document generation, project management artifacts, and business analysis deliverables using AI, standards-based templates, and unified workflows. ADPA supports enterprise compliance, integration, and scalability.

- **Business Context:**
  ADPA addresses the need for consistent, rapid, and standards-compliant documentation in enterprise environments, particularly for organizations adhering to regulatory frameworks and best practices such as BABOK, PMBOK, and DMBOK.

- **Scope Boundaries:**
  Included:

  - Document generation (CLI, API, web interface)
  - Multi-provider AI orchestration
  - Integration with Confluence, SharePoint, Adobe Document Services
  - Security, compliance, authentication/authorization
  - Analytics and reporting
    Excluded:
  - Real-time collaboration (future phase)

    - Custom workflow engine (future phase)
    - Mobile application (future phase)

1.2 Stakeholders

- **Primary Users:**
    - Business analysts, project managers, data stewards, compliance officers
- **System Administrators:**
    - Responsible for configuration, security, integration setup, and monitoring
- **Developers:**
    - Extending framework, building custom modules, maintaining integration points
- **Business Stakeholders:**
    - Project sponsors, regulatory/compliance managers, executive leadership

---

2. Architectural Principles

2.1 Design Principles

- **Modularity:**
  Encapsulated components (AI, generation, integration) enable independent evolution and testing.
- **Scalability:**
  Service-oriented, stateless API components and horizontal scaling patterns.

- **Reliability:**
  Provider failover, robust error handling, and monitoring.
- **Security:**
  Defense-in-depth: OAuth2/SAML/Active Directory, API keys, encrypted data.
- **Maintainability:**
  TypeScript strict typing, clear module boundaries, code standards (Airbnb, Prettier).
- **Performance:**
  Caching, optimized workflows, and asynchronous processing for I/O-heavy tasks.

2.2 Quality Attributes

- **Availability:** 99.9% uptime target
- **Performance:** <1s response for standard API calls
- **Scalability:** Microservices-ready, supports enterprise load
- **Security:** End-to-end encryption, RBAC, regulatory compliance
- **Usability:** CLI, web, and API interfaces with clear documentation

---

3. System Architecture

3.1 High-Level Architecture

```
 [Presentation Layer]
   |-- Admin Web Portal (Next.js/React)
   |-- CLI Tools (Yargs)
   |-- REST API (Swagger UI)
         ↓
[Application Layer]
   |-- Express.js API Server (TypeSpec/OpenAPI)
         ↓
[Business Logic Layer]
   |-- AI Orchestration Engine
```

```
  |-- Document Generation Engine
  |-- Workflow Automation
        ↓
[Integration Layer]
  |-- Confluence, SharePoint, Adobe, VCS
        ↓
[Data Access Layer]
  |-- Configuration / Template Management
  |-- Analytics & Reporting
        ↓
[Data Storage Layer]
  |-- JSON-based config (extensible to SQL/NoSQL)
  |-- Optional Redis (for caching)
```

3.2 Architectural Patterns

- **Pattern:** Layered and Service-Oriented (API-First Microservices)
- **Rationale:**
  Supports separation of concerns, scalability, and maintainability. API-First enables external integrations and future microservices decomposition.
- **Implementation:**
  Distinct modules for AI, document generation, integrations, and interfaces, all exposed via standardized RESTful APIs.

3.3 System Context Diagram

```
 [External Users / Clients]
   ↕  (CLI, Web, REST)
[ADPA API Server]  <-->  [AI Providers: OpenAI, Google AI, Copilot, Ol
   |         |         |         |
   |         |         |         +--> [Adobe Document Services]
   |         |         +-------------------> [Confluence]
   |         +-----------------------------> [SharePoint]
   |----------------------------------------> [VCS: GitHub, GitLab]
   |
[Enterprise IAM (AD/SAML/OAuth2)]
```

```
    |
[Data Storage (JSON, Redis, SQL/NoSQL future)]
```

---

4. Component Design

4.1 Core Components

4.1.1 AI Processing Engine

- **Purpose:**
  Orchestrates requests to multiple AI providers for document analysis, authoring, and template filling.
- **Responsibilities:**
  Provider selection, failover, context management, API abstraction.
- **Interfaces:**
  Consumed by Document Generation and exposed via REST endpoints and CLI.
- **Dependencies:**
  openai, @google/generative-ai, @azure/openai, Copilot APIs, Ollama.
- **Technology Stack:**
  TypeScript modules, async/await, provider SDKs.

4.1.2 Document Generation Engine

- **Purpose:**
  Produces standards-compliant documents from templates and AI-generated content.
- **Responsibilities:**
  Template parsing, variable injection, output formatting (Markdown, PDF, JSON), workflow automation.
- **Interfaces:**
  Invoked via API, CLI, and Web Admin Portal.
- **Dependencies:**
  @adobe/pdfservices-node-sdk, custom templates, AI Processing

Engine.
- **Technology Stack:**
  TypeScript, template engine, Adobe SDK.

4.1.3 REST API Server

- **Purpose:**
  Serves as external API for all client interactions.
- **Responsibilities:**
  Request routing, authentication, validation, error handling, OpenAPI documentation.
- **Interfaces:**
  Exposes endpoints for document jobs, templates, publishing, analytics.
- **Dependencies:**
  express, express-validator, swagger-ui-express, helmet, cors, TypeSpec-generated routes.
- **Technology Stack:**
  Node.js, TypeScript, Express.js.

4.1.4 CLI Interface

- **Purpose:**
  Enables command-line automation and scripting.
- **Responsibilities:**
  Command parsing, batch operations, provider selection.
- **Interfaces:**
  Yargs-based commands, interacts with API and local modules.
- **Dependencies:**
  yargs, ts-node, node-fetch.
- **Technology Stack:**
  Node.js, TypeScript.

4.1.5 Integration Layer

- **Purpose:**
  Bridges ADPA with enterprise systems (Confluence, SharePoint,

Adobe, VCS).

- **Responsibilities:**
  API interactions, OAuth2 authentication, file transfer, metadata management, versioning.
- **Interfaces:**
  Service modules invoked by API/CLI.
- **Dependencies:**
  @microsoft/microsoft-graph-client, @adobe/pdfservices-node-sdk, OAuth libraries, REST APIs.
- **Technology Stack:**
  TypeScript, SDKs, OAuth2.

4.1.6 Admin Interface

- **Purpose:**
  Web-based configuration, monitoring, and workflow management.
- **Responsibilities:**
  User management, document/job dashboards, template editors, usage analytics.
- **Interfaces:**
  Next.js frontend, REST API backend.
- **Dependencies:**
  Next.js, React, Tailwind CSS.
- **Technology Stack:**
  Node.js, React 18, Next.js 14.

4.1.7 Analytics & Reporting

- **Purpose:**
  Collects usage metrics and system health for admins and business users.
- **Responsibilities:**
  API metrics, job tracking, compliance reports, error monitoring.
- **Interfaces:**
  Exposed via web dashboard and API endpoints.
- **Dependencies:**
  winston, express-winston, analytics DB (future: SQL/NoSQL).

- **Technology Stack:**
  TypeScript, logging libraries.

4.2 Component Interaction

- **Communication Patterns:**
  - Synchronous (REST API, CLI commands)
  - Asynchronous (workflow jobs, integration callbacks)
- **Data Flow:**
  - User/client submits request → API routes to business logic → AI processing (if needed) → document generated → output stored or published via integration layer.
- **Error Handling:**
  - Centralized error middleware in Express, retry/failover logic in AI engine, user feedback via CLI/API, logging for auditing.

---

5. Data Architecture

5.1 Data Model

- **Conceptual Model:**
  - Users, Roles, Templates, Document Jobs, Providers, Integrations, Analytics Events
- **Logical Model:**
  - JSON-based entities (users, templates, jobs).
  - Relationships: Users ↔ Jobs; Jobs ↔ Templates; Jobs ↔ Outputs; Integrations ↔ Jobs.
- **Physical Model:**
  - Filesystem (JSON), extensible to SQL/NoSQL (e.g., MongoDB, PostgreSQL).

5.2 Data Storage Strategy

- **Primary Database:**

    - JSON files for configuration and outputs (v3.x), migration-ready for SQL/NoSQL.
- **Caching Strategy:**
    - Redis (planned/optional) for workflow state, job queues, and session management.
- **Data Backup:**
    - Regular filesystem/database backups, versioned outputs, cloud storage options.
- **Data Migration:**
    - Schema versioning in JSON, migration scripts for future DB transition, rollbacks via backup snapshots.

5.3 Data Security

- **Encryption:**
    - At-rest: OS-level encryption, planned DB encryption; In-transit: HTTPS/TLS.
- **Access Control:**
    - RBAC, per-user API tokens, OAuth2/SAML for integrations.
- **Data Privacy:**
    - PII minimization, GDPR-aligned data policies, user-level data segregation.
- **Audit Logging:**
    - All access and changes logged, integrated with analytics.

---

6. Integration Architecture

6.1 External Integrations

- **Third-Party APIs:**
    - OpenAI, Google AI, Azure OpenAI, GitHub Copilot, Ollama, Confluence, SharePoint, Adobe Document Services, VCS (GitHub/GitLab/Azure DevOps).
- **Authentication:**

    - OAuth2 (Confluence, SharePoint, Adobe), API keys (AI providers), SAML (enterprise SSO).
- **Data Exchange:**
    - JSON (REST), Markdown, PDF, Office formats; HTTP(S), WebSockets (planned).
- **Error Handling:**
    - Retry logic, provider failover, circuit breakers for integrations, fallback to local generation where possible.

6.2 Internal Integrations

- **Service Communication:**
    - Module-to-module via TypeScript imports, REST API endpoints for admin portal.
- **Event Handling:**
    - Promise-based async flows, job queues (future: pub/sub for workflows).
- **Workflow Management:**
    - Automated pipelines (document generation → publishing → analytics).

---

7. Technology Stack

7.1 Development Stack

- **Frontend:**
    - Next.js 14, React 18, Tailwind CSS
- **Backend:**
    - Node.js 18+, TypeScript 5.7+, Express.js, Yargs
- **Database:**
    - JSON (current), Redis (optional), extensible to SQL/NoSQL
- **Caching:**
    - Redis (planned)
- **Message Queue:**

    - Not currently implemented; future: RabbitMQ, Azure Service Bus

7.2 Infrastructure

- **Cloud Platform:**
    - Azure (primary), deployable on AWS/GCP
- **Containerization:**
    - Docker (Q2 2025), Kubernetes (Q2-Q3 2025)
- **CI/CD:**
    - GitHub Actions, Azure DevOps pipelines
- **Monitoring:**
    - Winston logging, Express health checks, (future: Prometheus/Grafana)
- **Security:**
    - OAuth2, SAML, API keys, SSL/TLS

7.3 Development Tools

- **Version Control:**
    - Git (GitHub, GitLab, Azure DevOps), standard branching
- **Project Management:**
    - Jira, Azure DevOps, GitHub Projects
- **Documentation:**
    - Markdown, Confluence, GitHub Wiki
- **Testing:**
    - Jest, ts-jest, integration with CI

---

8. Security Architecture

8.1 Security Requirements

- **Authentication:**
    - Multi-factor, enterprise SSO, API keys/JWT for API/CLI
- **Authorization:**

    - Role-based access, fine-grained permissions (planned)
- **Data Protection:**
    - AES/GCM encryption standards, secure secret management (dotenv, Azure Key Vault)
- **Network Security:**
    - HTTPS everywhere, CORS configuration, firewall rules

8.2 Threat Model

- **Assets:**
    - Enterprise data, user credentials, document outputs, configuration secrets
- **Threats:**
    - Unauthorized access, data breaches, API abuse, provider compromise
- **Vulnerabilities:**
    - Dependency vulnerabilities, misconfigurations, insufficient rate limiting
- **Risk Assessment:**
    - Regular security reviews, dependency scanning, audit logging

8.3 Security Controls

- **Preventive Controls:**
    - Helmet, input validation, rate limiting, RBAC, API key rotation
- **Detective Controls:**
    - Winston/Express logging, health endpoints, alerting (planned)
- **Corrective Controls:**
    - Incident response runbooks, backup/restore, rapid patching

---

9. Performance and Scalability

9.1 Performance Requirements

- **Response Time:**

    - <1s for API metadata/endpoints, <5s for complex document generation
- **Throughput:**
    - 100+ concurrent users/jobs (scalable)
- **Resource Utilization:**
    - Auto-tuned Node.js process pools, container resource limits

9.2 Scalability Strategy

- **Horizontal Scaling:**
    - Stateless API, container orchestration, load balancers
- **Vertical Scaling:**
    - Resource allocation per workload, containerized limits
- **Database Scaling:**
    - Planned: migrate to cloud DB with read replicas/sharding

9.3 Performance Optimization

- **Caching Strategy:**
    - Redis for workflow state and provider responses
- **Database Optimization:**
    - N/A (JSON); planned: indexing, query optimization in SQL/NoSQL
- **Code Optimization:**
    - TypeScript profiling, async/await for non-blocking I/O

---

10. Deployment Architecture

10.1 Environment Strategy

- **Development:**
    - Local Docker Compose, .env configs, test data
- **Testing:**
    - CI/CD pipeline, automated Jest/integration tests
- **Staging:**

    - Mirror of production, external integrations in sandbox mode
- **Production:**
    - Hardened containers, managed secrets, external integrations, auto-scaling

10.2 Deployment Process

- **CI/CD Pipeline:**
    - GitHub Actions: build → test → deploy; Azure DevOps for enterprise deployments
- **Blue-Green Deployment:**
    - Planned for zero-downtime upgrades
- **Rollback Strategy:**
    - Versioned releases, rapid rollback via container tags

10.3 Infrastructure as Code

- **Configuration Management:**
    - Terraform/ARM templates (planned)
- **Container Orchestration:**
    - Kubernetes (Q2-Q3 2025 roadmap)
- **Monitoring and Alerting:**
    - Health endpoints, logging, (future: alerting integration)

---

11. Risk Management

11.1 Technical Risks

- **Single Points of Failure:**
    - Mitigated via stateless services, container orchestration
- **Technology Dependencies:**
    - Monitored for vendor changes, open-source alternatives available
- **Performance Bottlenecks:**
    - Load testing, async processing, provider failover

11.2 Mitigation Strategies

- **Redundancy:**
  - Multi-provider AI, HA deployments, backup integrations
- **Monitoring:**
  - Express health endpoints, logs, (future: monitoring stack)
- **Documentation:**
  - Code and user documentation, runbooks for support

---

12. Implementation Roadmap

12.1 Phase 1: Foundation

- **Duration:** Q1 2025
- **Deliverables:**
  - Core API/CLI
  - AI orchestration
  - BABOK/PMBOK compliance
- **Success Criteria:**
  - End-to-end document generation, publish to Confluence/SharePoint

12.2 Phase 2: Enhancement

- **Duration:** Q2 2025
- **Deliverables:**
  - DMBOK 2.0 support
  - Docker/Kubernetes deployment
  - Analytics dashboard
- **Success Criteria:**
  - Cloud-native deployments, advanced reporting, Adobe premium output

12.3 Phase 3: Optimization

- **Duration:** Q3 2025
- **Deliverables:**
  - Enterprise SSO
  - Real-time collaboration
  - Mobile/web enhancements
- **Success Criteria:**
  - SSO working, collaboration prototypes, mobile POC

---

13. Appendices

Appendix A: Glossary

- **ADPA:** Advanced Document Processing & Automation Framework
- **BABOK:** Business Analysis Body of Knowledge
- **PMBOK:** Project Management Body of Knowledge
- **DMBOK:** Data Management Body of Knowledge
- **API-First:** Design strategy prioritizing public API contracts

Appendix B: References

- [BABOK v3, PMBOK 7th Edition, DMBOK 2.0]
- TypeSpec Documentation
- OpenAPI 3.0 Specification

Appendix C: Assumptions and Constraints

- **Assumptions:**
  - Enterprise IAM integration possible via OAuth2/SAML
  - Users have access to required third-party APIs (OpenAI, Confluence, etc.)
- **Constraints:**
  - Some advanced features (real-time collaboration, mobile) are roadmap items

    - Initial storage is file-based; migration to DB required for high scale

---

Document Control:

- **Created:** 2025-07-14
- **Last Updated:** 2025-07-14
- **Next Review:** 2025-10-01
- **Approved By:** [Approval authority]
- **Version History:**
    - v1.0 - Initial draft

For further information, refer to GitHub Wiki and internal technical documentation.
