Acceptance Criteria

> **Source File:** generated-documents\technical-analysis\acceptance-criteria.md
> **Generated:** 30/07/2025 at 07:01:35
> **Generated by:** Requirements Gathering Agent - PDF Converter

Acceptance Criteria

Generated by adpa-enterprise-framework-automation v3.2.0
Category: technical-analysis
Generated: 2025-07-15T18:09:05.129Z
Description: Comprehensive acceptance criteria and validation methods

---

Acceptancecriteria

Project: ADPA - Advanced Document Processing & Automation Framework
Version: 3.2.0
Document Date: 2025-07-08
Prepared by: Requirements Documentation Consultant

---

1. General Product Acceptance

1.1 **Modularity and Standards Compliance**

- The framework must support modular integration of document processing components for business analysis, project management, and data management.
- All generated documentation must comply with at least one of: BABOK v3, PMBOK 7th Edition, or DMBOK 2.0 standards.
- Multi-framework integration must enable unified reporting and cross-referencing across BABOK, PMBOK, and DMBOK artifacts.

1.2 **Technology and Platform Requirements**

- The system must be built using Node.js (>=18.0.0) and TypeScript (>=5.7.2).
- CLI tools, REST API, and admin web interface must be available and installable via npm.
- The system must provide detailed installation and configuration documentation for each interface (CLI, API, Admin Web).

---

2. Core Functional Acceptance

2.1 **Document Generation**

- The system must generate professional, standards-compliant documents for:
  - BABOK v3 (requirements elicitation, stakeholder analysis, planning, validation, enterprise analysis)
  - PMBOK 7th Edition (project charter, scope, risk, resource, schedule, and cost management)
  - DMBOK 2.0 (data governance, architecture, quality, security, privacy) — where marked production ready or in progress.
- Document generation must support multiple output formats: Markdown, PDF, JSON.
- Document templates must be configurable and extensible by users.

2.2 **AI-Powered Automation**

- Users must be able to select and configure AI providers from: OpenAI, Google AI, GitHub Copilot, Ollama, Azure OpenAI.

- The system must support automatic failover between AI providers.
- Intelligent context management must be implemented, ensuring relevant context is injected for all document generation tasks.

2.3 **Automated Workflows**

- End-to-end document generation pipelines must be available, automating multi-step processes (input analysis, template selection, AI processing, output formatting).
- Batch processing support for CLI and API must be implemented.

2.4 **Enterprise Context Integration (Palantir Foundry)**

- 2.4.1 Contextual Enrichment: The system must be able to query a configured Palantir Foundry instance to retrieve project-specific context (e.g., risks, milestones, resources, budget data) based on the user's query or document request.
- 2.4.2 Ontology Alignment: The integration must correctly map ADPA's data needs to the corresponding objects and properties within the Foundry Ontology, ensuring semantic accuracy.
- 2.4.3 Prompt Injection: The retrieved Foundry context must be intelligently and automatically injected into the prompt sent to the selected AI provider, enriching the context beyond the user's initial input.

---

3. Interface Acceptance

3.1 **CLI Interface**

- A command-line tool must be provided, supporting:
  - Document generation (`adpa generate`)
  - Integration setup (Confluence, SharePoint, Adobe)
  - Provider selection and configuration
- CLI help and usage documentation must be accessible via `--help`.

3.2 **REST API**

- API must conform to OpenAPI 3.0 specification, auto-generated via TypeSpec.
- API endpoints must include:
  - `/api/v1/generate` – Document generation
  - `/api/v1/templates` – Template management (CRUD)
  - `/api/v1/confluence/publish` – Publish to Confluence
  - `/api/v1/sharepoint/upload` – Upload to SharePoint
  - `/api/v1/frameworks` – Supported frameworks listing
- API must include health, readiness, and metrics endpoints.

3.3 **Admin Web Interface**

- Next.js-based admin portal must allow for management of users, templates, documents, and integrations.
- Portal must be accessible at the configured port (default: 3001) after installation.

---

4. Integration Acceptance

4.1 **Confluence Integration**

- System must support OAuth2 authentication with Atlassian Confluence.
- Users must be able to publish generated documents directly to Confluence spaces and pages.
- Document metadata and versioning must be preserved.

4.2 **SharePoint Integration**

- System must support OAuth2/Azure AD authentication for SharePoint Online.
- Users must be able to upload documents to specified SharePoint folders/libraries.
- Metadata tagging and version control must be supported.

4.3 **Adobe Document Services**

- System must support professional PDF generation using Adobe APIs.
- Where Adobe Creative Suite is enabled, advanced layout (InDesign), visualization (Illustrator), and image enhancement (Photoshop) must be available per the phase 2 architecture.
- OAuth2 authentication and credential management for Adobe APIs must be implemented.

4.4 **Palantir Foundry Integration**

- 4.4.1 Authentication: The system must support secure, user-delegated authentication with Palantir Foundry, preferably using OAuth2, to ensure requests are made on behalf of the user.
- 4.4.2 Data Retrieval: The integration must successfully execute queries against the Foundry API to fetch data from specified objects and datasets.
- 4.4.3 Configuration: The Admin Web Interface and/or a configuration file must provide a secure mechanism for system administrators to configure the Foundry instance URL and necessary API credentials.

---

5. Compliance, Security, and Enterprise Readiness

5.1 **Authentication and Authorization**

- All interfaces (CLI, API, Admin) must support secure authentication methods (API key, JWT, OAuth2).
- Role-based access control must be enforced for multi-user/admin scenarios.
- 5.1.3 Inherited Access Control (Foundry): When integrated with Palantir Foundry, the system must honor the user's existing permissions within Foundry. The integration must not allow a user to see or use context from Foundry that they are not already authorized to access directly in the platform.

5.2 **Regulatory Compliance**

- The system must demonstrate controls supporting regulatory requirements: GDPR, SOX, PCI DSS, Basel III, MiFID II, FINRA, HIPAA, FedRAMP, where applicable.
- Audit trails must log user actions (creation, modification, publishing, deletion).
- 5.2.3 Governed Data Usage: All data retrieved from Palantir Foundry for contextual enrichment must be handled in accordance with the data governance classifications and markings (e.g., sensitivity, purpose) defined within Foundry. The audit trail must log which Foundry objects were accessed for a given document generation task.

5.3 **Security Best Practices**

- Security headers (Helmet), CORS, rate limiting, and input validation must be enabled by default.
- Sensitive configuration (API keys, secrets) must never be exposed in logs or error messages.

---

6. Performance, Scalability, and Reliability

6.1 **Scalability**

- The system must support deployment in horizontally scalable environments (microservices, Docker/Kubernetes).
- Load balancing and Redis-based caching must be available for high-throughput scenarios.

6.2 **Reliability**

- Health and readiness endpoints must report accurate system status.
- AI provider failover must be automatic and transparent to the user.
- All critical operations (document generation, publishing, uploads) must include proper error handling and retry logic.

- 6.2.4 Foundry API Error Handling: The integration must include robust error handling for scenarios where the Foundry API is unavailable, returns an error, or a query times out. In such cases, the system should gracefully degrade, notifying the user that enterprise context is unavailable, rather than failing the entire operation.

6.3 **Testing and Quality**

- Comprehensive unit, integration, and performance test suites must be provided (Jest-based).
- Test coverage reports must be available.
- All major features (document generation, integrations, API endpoints) must pass end-to-end tests.

---

7. Usability and Documentation

7.1 **User Guidance**

- Quick start guides and detailed usage examples must be included for CLI, API, and admin interface.
- Interactive menus and prompts (e.g., for AI provider selection) must be intuitive, with validation and error feedback.

7.2 **Developer Documentation**

- API documentation (Swagger UI/Redoc) must be accessible from the running API server.
- Codebase must include clear inline comments and architectural overviews.

7.3 **Support & Community**

- Links to GitHub Issues, Discussions, and enterprise support must be visible in documentation.
- Contribution guidelines and code standards must be published and enforced.

---

8. Roadmap and Extensibility

8.1 **Extensible Architecture**

- The framework must allow for the addition of new document types, templates, AI providers, and integrations with minimal code changes.

8.2 **Planned Features**

- Progress on DMBOK 2.0 support, Docker/Kubernetes deployment, analytics dashboard, SSO integration, workflow automation, real-time collaboration, and mobile support must be tracked per roadmap.

---

9. Acceptance Verification

- All acceptance criteria must be traceable to user stories, requirements, and/or referenced industry standards.
- Demonstrable evidence (e.g., successful test results, generated documentation samples, UI walkthroughs) must be provided for each acceptance item.
- Stakeholder sign-off is required prior to production release.

End of Acceptance Criteria