Test Strategy

Source File: generated-documents\quality-assurance\test-strategy.md
Generated: 16/07/2025 at 13:58:58
Generated by: Requirements Gathering Agent - PDF Converter

Test Strategy

Generated by adpa-enterprise-framework-automation v3.2.0
Category: quality-assurance
Generated: 2025-07-14T20:57:57.583Z
Description: Comprehensive testing strategy and approach

```
# Test Strategy Document
**Project:** ADPA – Advanced Document Processing & Automation Framewor
**Version:** 3.2.0
**Date:** July 2025
**Owner:** QA & Test Management Team

---

## 1. Testing Objectives and Goals

### 1.1 Objectives
- **Validate functional correctness** of ADPA’s modular automation, en
- **Ensure enterprise-grade security and regulatory compliance** (GDPR
- **Verify robust AI-provider integration** (OpenAI, Google AI, GitHub
- **Guarantee high-quality document generation** across CLI, REST API,
- **Confirm performance, scalability, and reliability** for enterprise
- **Validate integrations** with Confluence, SharePoint, Adobe Documen
```

```
### 1.2 Quality Criteria & Acceptance Thresholds
- **Functional:** 100% of critical business requirements tested; no hi
- **Security:** 0 critical or high-severity vulnerabilities in authent
- **Performance:** System meets or exceeds NFRs (e.g., API latency < 5
- **Compliance:** All documentation and workflows align with specified
- **Usability:** Admin portal and CLI meet usability benchmarks (SUS s
- **Reliability:** >99.5% uptime in staging and production-like test e

### 1.3 Success Metrics & KPIs
- **Test Coverage:** ≥ 90% code coverage on business logic and APIs.
- **Defect Density:** ≤ 0.5 per KLOC (post-stabilization).
- **Defect Leakage:** ≤ 2% to production.
- **Mean Time to Detect (MTTD):** < 2 hours (critical defects in regre
- **Automation Rate:** ≥ 70% of regression and integration tests autom
- **Performance Benchmarks:** API response time, throughput, and resou

---

## 2. Test Scope and Approach

### 2.1 In-Scope
- Core ADPA modules (AI Processing, Document Generator, REST API, CLI,
- AI integrations (OpenAI, Google AI, GitHub Copilot, Ollama, Azure Op
- Framework templates (BABOK, PMBOK, DMBOK).
- Confluence, SharePoint, Adobe Document Services integrations.
- Authentication/Authorization flows (API Key, JWT, OAuth2).
- API endpoints, CLI commands, Admin UI workflows.
- Security, compliance, performance, and usability.

### 2.2 Out-of-Scope
- Legacy or deprecated features.
- Custom client-side scripting outside the core admin portal.
- Third-party service bugs (unless impacting ADPA core flows).
- Experimental, unreleased roadmap features (e.g., mobile app, advance

### 2.3 Test Levels
- **Unit Testing:** All core logic, helper functions, and business rul
- **Integration Testing:** Module-to-module and external API integrati
- **System Testing:** End-to-end flows via CLI, REST API, Admin portal
- **User Acceptance Testing (UAT):** Customer- or business-validated s
- **Regression Testing:** Automated and manual regression cycles for e

### 2.4 Testing Types
```

```
- **Functional Testing:** Requirements validation, template generation
- **Non-functional Testing:** Performance, scalability, reliability, f
- **Security Testing:** Penetration, vulnerability scanning, authentic
- **Compatibility Testing:** OS, Node.js versions, browser compatibili
- **Usability Testing:** Admin portal (Next.js), CLI, error messaging,
- **Compliance Testing:** Standards conformance (BABOK, PMBOK, DMBOK,

### 2.5 Risk-Based & Critical Path Testing
- Prioritize core document generation, AI integration failover, and en
- Focus on authentication/authorization, data privacy, and regulatory
- Early and continuous testing of performance and scalability using pr

---

## 3. Test Environment Strategy

### 3.1 Environment Requirements
- **Backend:** Node.js (≥18), TypeScript (≥5.7), Express.js, REST API
- **Frontend:** Next.js 14, React 18, Tailwind CSS for Admin portal.
- **Databases:** JSON-based config, with optional SQL/NoSQL extensions
- **Third-Party APIs:** AI providers (API keys, credentials), Adobe Cr

### 3.2 Test Data Management
- Use anonymized or synthetic datasets for document generation and int
- Maintain separate test data sets for functional, performance, and se
- Ensure compliance with data privacy (GDPR, etc.)—never use live/PII
- Automated data refresh and teardown scripts for repeatable test runs

### 3.3 Environment Dependencies & Integrations
- Mock or sandbox endpoints for AI providers and external APIs when fe
- Secure storage of API credentials and secrets (env files, vaults, CI
- Integration with enterprise document management systems and identity

### 3.4 Setup, Maintenance, and Refresh
- Automated environment provisioning via scripts (npm, Docker, CI/CD p
- Daily refresh of test data and periodic teardown/rebuild of environm
- Monitoring and health checks to detect drift or stale configuration.

---

## 4. Test Organization and Roles

### 4.1 Team Structure & Responsibilities
```

```
- **QA Manager:** Owns the test strategy, planning, and reporting.
- **Test Leads (Functional, Integration, Automation):** Drive specific
- **Automation Engineers:** Develop and maintain test automation suite
- **Manual Testers:** Execute exploratory, usability, and regression s
- **Security Testers:** Conduct vulnerability and compliance testing.
- **DevOps/Environment Support:** Ensure environment stability and dep

### 4.2 Skills and Training
- Proficiency in Node.js, TypeScript, Jest, and REST API testing.
- Experience with enterprise integrations (SharePoint, Confluence, Ado
- Security testing tools and best practices.
- Familiarity with standards (BABOK, PMBOK, DMBOK, GDPR, SOX).
- Training on new frameworks or tools as features are added.

### 4.3 Communication & Reporting
- Daily standups during test cycles; weekly status reports to stakehol
- Use of test management tools (e.g., Jira, Azure DevOps) for defect t
- Slack/Teams channels for real-time escalation.

### 4.4 Escalation & Decision-Making
- Escalate critical risks or blockers to QA Manager and Product Owner.
- Defect triage meetings for prioritization.
- Clear sign-off authorities for test phase entry/exit and release rea

---

## 5. Risk Assessment and Mitigation

### 5.1 Key Risks
- **Integration failures** with AI providers or enterprise platforms (
- **Security vulnerabilities** in authentication, data handling, or th
- **Performance degradation** under scale (document generation, concur
- **Compliance drift** due to evolving standards or regulations.
- **Resource bottlenecks** (test environment contention, lack of skill

### 5.2 Probability & Impact Assessment
- Integration and security risks: High probability, high impact.
- Performance risks: Medium probability, high impact.
- Compliance risks: Low probability, high impact (regulatory).
- Resource risks: Medium probability, medium impact.

### 5.3 Mitigation Strategies
- Early and continuous integration testing using provider sandboxes.
- Automated security scans and periodic penetration tests.
- Performance benchmarking as part of CI/CD.
- Regular compliance reviews and traceability matrices.
- Cross-training, resource pooling, and on-demand contractor support.

### 5.4 Monitoring and Escalation
- Continuous risk review in QA status meetings.
- Rapid escalation path for critical issues (within 4 working hours).
- Update risk register and mitigation actions regularly.

---

## 6. Test Deliverables and Timeline

### 6.1 Test Deliverables
- **Test Plan:** QA Manager (owner), reviewed by stakeholders.
- **Test Cases & Scripts:** Test Leads, Automation Engineers.
- **Test Data Sets:** Testers, DevOps.
- **Defect Reports:** All testers, tracked in Jira/Azure DevOps.
- **Test Summary & Release Reports:** QA Manager.
- **Traceability Matrix:** Test Leads.

### 6.2 Milestones & Schedule
- **Test Planning:** T0 + 1 week
- **Unit Test Completion:** T0 + 2 weeks
- **Integration Test Completion:** T0 + 4 weeks
- **System Test Completion:** T0 + 6 weeks
- **UAT & Regression:** T0 + 7 weeks
- **Release Readiness:** T0 + 8 weeks

### 6.3 Entry & Exit Criteria
- **Entry:** Requirements/docs finalized; environments provisioned; te
- **Exit:** All planned cases executed; no open critical/high defects;

### 6.4 Review & Approval
- Peer review of test cases/scripts.
- Stakeholder approval of test plan and summary reports.
- Formal sign-off on test phase exit.

---

## 7. Tools and Technologies
```

```
### 7.1 Core Testing Tools
- **Unit/Integration Testing:** Jest, ts-jest (TypeScript).
- **API Testing:** Postman, Newman, supertest.
- **Automation:** Custom suites (Jest), Selenium/WebdriverIO for Admin
- **Test Management:** Jira, Azure DevOps, Xray.
- **Defect Tracking:** Jira, Azure DevOps.

### 7.2 Automation Strategy
- Automate all repeatable functional, regression, and integration test
- Prioritize automation for critical path flows and high-volume APIs.
- Maintain automation suites as code (version controlled in GitHub).

### 7.3 Data Management & Generation
- Synthetics generators for test documents and API payloads.
- Use of fixtures and factories for repeatable test data.

### 7.4 Security, Performance, Accessibility
- **Security:** OWASP ZAP, npm audit, custom scripts.
- **Performance:** k6, Artillery, custom Node.js benchmarks.
- **Accessibility:** axe-core, Lighthouse for Admin portal.

---

## 8. Resource Planning and Budget

### 8.1 Effort & Resource Estimates
- QA Team: 1 QA Manager, 2 Test Leads, 3 Automation Engineers, 2 Manua
- 3 Dedicated test environments (dev, integration, pre-prod).
- 2 Shared AI provider sandbox accounts.

### 8.2 Infrastructure & Licensing
- Node.js/TypeScript-compatible test servers (cloud VMs or containers)
- Licenses for Postman, Jira, Selenium Grid, and performance/security
- Budget for paid AI provider APIs (test quotas), Adobe Creative Cloud

### 8.3 Training & External Dependencies
- Training budget for new integrations (e.g., Adobe SDK, DMBOK modules
- Vendor support for SharePoint, Confluence, Adobe, and AI APIs as nee

### 8.4 Budget Allocation
- ~20% of project budget earmarked for testing activities (staff, tool

---
```

```
## 9. Quality Metrics and Reporting

### 9.1 Test Coverage & Measurement
- **Code Coverage:** Measured with Jest/ts-jest, with badge in CI pipe
- **Requirements Traceability:** All requirements mapped to one or mor
- **Feature Coverage:** All critical/major features with direct test c

### 9.2 Defect Metrics & Tracking
- Defect density, severity distribution, and open/closed rates reporte
- Defect leakage tracked post-release.
- Root cause analysis for escaped defects.

### 9.3 Performance & Acceptance Criteria
- API latency, throughput, and error rates monitored via performance t
- Benchmarks compared against SLOs/SLA targets.

### 9.4 Reporting and Communication
- Daily test execution summaries (during cycles).
- Weekly stakeholder reports (progress, risks, blockers).
- Release summary with test coverage, defect metrics, and readiness as

---

## 10. Continuous Improvement

### 10.1 Lessons Learned & Best Practices
- Conduct post-mortem reviews after major releases; document key findi
- Maintain a repository of reusable test assets and patterns.

### 10.2 Feedback & Process Maturity
- Collect feedback from testers, developers, and business users after 
- Adjust test processes based on retrospectives and QA metrics.

### 10.3 Process Maturity Assessment
- Periodic review using TMMi or custom maturity framework; set targets

### 10.4 Knowledge Transfer & Documentation
- Update test documentation regularly (test cases, guides, environment
- Cross-train team members; maintain onboarding guides for new testers

---
```

```
## Appendix

- **Glossary:** Definitions of standards (BABOK, PMBOK, DMBOK), SLO/SL
- **References:** Test case repositories, requirements docs, CI/CD pip

---

**Prepared by:**
ADPA QA & Test Management Team
For questions or clarifications, contact [QA Lead](mailto:qa.lead@adpa
```