9. Acceptance Criteria

9.1 Core Functionality

- All document editors must be fully functional with save/load capabilities
- GitHub integration must successfully connect and perform all operations
- AI features must provide relevant and contextual suggestions
- RTM must accurately display all traceability relationships
- Repository structure must be automatically created and maintained

9.2 Performance

- Application must load within 3 seconds on standard broadband
- All user interactions must respond within 1 second
- System must handle 1000+ artifacts without performance degradation
- Badge generation must complete within 2 seconds

9.3 Deployment

- Application must deploy successfully to at least 3 different hosting platforms
- CI/CD pipeline must complete without errors
- Production deployments must be secure and scalable
- Badges must be accessible with 99.9% uptime

9.4 Repository Integration

- Must successfully create standardized folder structures
- All generated files must be Git-friendly and diff-compatible
- Repository templates must work across different project types

- Badge integration must work seamlessly with existing README files

10. Badge System Specifications

10.1 Standard Badge Categories

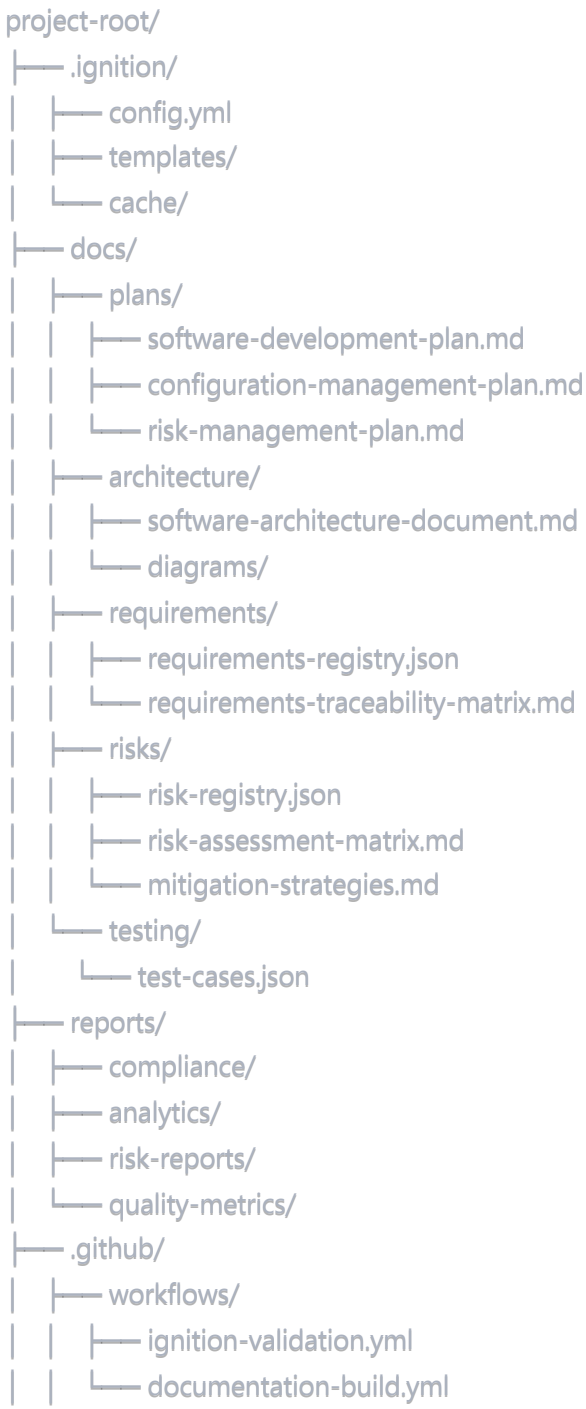
1. **Project Status Badges**
   - Project Health Score (0-100)
   - Documentation Completeness (%)
   - Requirements Coverage (%)
   - Test Coverage (%)
   - CMMI Compliance Level (1-5)
2. **Achievement Badges**
   - "Ignition Powered" (Basic usage)
   - "Process Champion" (Full CMMI compliance)
   - "Documentation Master" (100% document completion)
   - "Traceability Expert" (100% RTM coverage)
   - "Collaboration Leader" (Multi-contributor projects)
3. **Time-Based Badges**
   - "Consistent Maintainer" (6+ months active)
   - "Long-term Project" (1+ year maintained)
   - "Rapid Delivery" (Project completed in <3 months)
4. **Methodology Badges**
   - "Agile Ready" (Agile-compatible setup)
   - "DevOps Integrated" (CI/CD pipelines configured)
   - "Enterprise Grade" (Full compliance suite)

10.2 Badge Technical Specifications

- Format: SVG with dynamic content
- Hosting: CDN-distributed for fast loading
- Verification: Cryptographic signatures for authenticity
- Caching: Smart cache invalidation for real-time updates
- Integration: One-click addition to README files

11. Repository Structure Template

11.1 Recommended Directory Structure


project-root/
.ignition/
config.yml
templates/
cache/
docs/
plans/
software-development-plan.md
configuration-management-plan.md
risk-management-plan.md
architecture/
software-architecture-document.md
diagrams/
requirements/
requirements-registry.json
requirements-traceability-matrix.md
risks/
risk-registry.json
risk-assessment-matrix.md
mitigation-strategies.md
testing/
test-cases.json
reports/
compliance/
analytics/
risk-reports/
quality-metrics/
.github/
workflows/
ignition-validation.yml
documentation-build.yml

```
│   └── PULL_REQUEST_TEMPLATE.md
└── README.md (with Ignition badges)
```

12. Additional Enhancement Ideas

12.1 Integration Ecosystem

- **Slack/Teams Integration**: Real-time notifications for risk escalations and document updates
- **JIRA/Azure DevOps Integration**: Bi-directional sync of requirements and risks
- **Email Integration**: Automated stakeholder notifications and report distribution
- **Calendar Integration**: Risk review scheduling and milestone tracking

12.2 Advanced Analytics

- **Predictive Analytics**: AI-driven project success probability based on historical data
- **Benchmarking**: Compare project metrics against industry standards
- **Trend Analysis**: Long-term organizational process improvement insights
- **Resource Optimization**: AI recommendations for resource allocation based on risk profiles

12.3 Compliance and Standards

- **Multi-Standard Support**: ISO 27001, SOC 2, HIPAA, FDA 21 CFR Part 11
- **Regulatory Mapping**: Automatic mapping of requirements to regulatory standards
- **Audit Trail**: Complete audit logging for compliance purposes
- **Digital Signatures**: Support for document signing and approval workflows

12.4 Mobile and Accessibility

- **Mobile App**: Native mobile app for on-the-go project monitoring
- **Progressive Web App**: Offline capability for field work

- **Accessibility**: Full WCAG 2.1 AA compliance
- **Multi-language Support**: Internationalization for global teams

12.5 Enterprise Features

- **Single Sign-On (SSO)**: Integration with corporate identity providers
- **Multi-tenant Architecture**: Support for multiple organizations
- **Custom Branding**: White-label options for enterprise customers
- **API Gateway**: Enterprise-grade API management and security

12.6 Advanced Risk Management

- **Monte Carlo Simulation**: Probabilistic risk analysis for project outcomes
- **Risk Correlation Analysis**: Identify cascading risk effects
- **Scenario Planning**: "What-if" analysis for risk mitigation strategies
- **Risk Heat Maps**: Visual representation of risk landscapes
- **Automated Risk Monitoring**: Real-time risk indicators from integrated tools

12.7 Community and Marketplace

- **Template Marketplace**: User-contributed templates and methodologies
- **Plugin System**: Third-party extensions and integrations
- **Community Hub**: Knowledge sharing and best practices
- **Certification Program**: Ignition methodology certification for professionals

12.8 Advanced AI Features

- **Natural Language Processing**: Extract requirements from unstructured documents
- **Intelligent Document Generation**: AI-written first drafts of planning documents

- **Anomaly Detection**: Identify unusual patterns in project data
- **Smart Recommendations**: Context-aware suggestions for process improvements

12.9 DevOps and Automation

- **Infrastructure as Code**: Terraform/CloudFormation templates for deployment
- **GitOps Integration**: Automated deployment through Git workflows
- **Container Orchestration**: Kubernetes operators for enterprise deployment
- **Microservices Architecture**: Scalable, modular system design

12.10 Data and Analytics

- **Data Lake Integration**: Connect to enterprise data warehouses
- **Real-time Dashboards**: Live updating project metrics
- **Custom Metrics**: User-defined KPIs and measurements
- **Machine Learning Models**: Predictive models for project success factors# Ignition Project Requirements Document

1. Project Overview

1.1 Project Description

Ignition is a comprehensive bootstrapping tool designed to accelerate software development project initiation by providing structured documentation, methodology alignment, and momentum-building capabilities from day one.

1.2 Project Goals

- Streamline project setup and documentation processes
- Align development practices with proven methodologies (CMMI v1.3)
- Enable collaborative development through Git-based workflows

- Provide AI-assisted content generation and project insights
- Ensure traceability and compliance throughout the development lifecycle

2. Functional Requirements

2.1 Core Platform Requirements

2.1.1 Data Persistence and Collaboration

- **REQ-001**: The system SHALL store the entire project state in a single `ignition-project.json` file
- **REQ-002**: The system SHALL integrate with GitHub repositories for version control
- **REQ-003**: The system SHALL support collaborative workflows via Git pull/push operations
- **REQ-004**: The system SHALL maintain data integrity and provide rollback capabilities
- **REQ-005**: The system SHALL support project import/export via JSON files

2.1.2 User Interface and Navigation

- **REQ-006**: The system SHALL organize views into logical groups (Project, Artifacts, Documents, etc.)
- **REQ-007**: The system SHALL provide scalable dropdown navigation in the header
- **REQ-008**: The system SHALL maintain responsive design for various screen sizes
- **REQ-009**: The system SHALL provide comprehensive in-app help documentation

2.2 GitHub Integration Requirements

2.2.1 Repository Management

- **REQ-010**: The system SHALL connect to GitHub repositories for project storage
- **REQ-011**: The system SHALL browse repository files and directory structures
- **REQ-012**: The system SHALL link Configuration Items (CIs) to repository artifacts

- **REQ-013**: The system SHALL perform repository audits to verify artifact existence
- **REQ-014**: The system SHALL generate GitHub issues from requirements and test cases

2.2.2 Repository Structure Management

- **REQ-015**: The system SHALL create standardized folder structures for project artifacts
- **REQ-016**: The system SHALL generate a `docs/` folder containing all planning documents
- **REQ-017**: The system SHALL create subdirectories for each document type (e.g., `docs/plans/`, `docs/architecture/`, `docs/requirements/`)
- **REQ-018**: The system SHALL generate a `.ignition/` folder for tool-specific metadata and configurations
- **REQ-019**: The system SHALL create a `templates/` folder for reusable document templates and CI templates
- **REQ-020**: The system SHALL maintain a `reports/` folder for generated analytics and compliance reports
- **REQ-021**: The system SHALL auto-generate appropriate `.gitignore` entries for Ignition-specific files
- **REQ-022**: The system SHALL support customizable folder structure templates based on project type
- **REQ-023**: The system SHALL automatically organize exported documents into appropriate repository folders
- **REQ-024**: The system SHALL maintain folder structure consistency across project updates

2.3 Document Management Requirements

2.3.1 Structured Document Editors

- **REQ-025**: The system SHALL provide a structured editor for Software Development Plan (SDP)

- **REQ-026**: The system SHALL provide a structured editor for Configuration Management (CM) Plan
- **REQ-027**: The system SHALL provide a structured editor for Software Architecture Document (SAD)
- **REQ-028**: The system SHALL provide a structured editor for Interface Control Document (ICD)
- **REQ-029**: The system SHALL support PDF export for all document types
- **REQ-030**: The system SHALL automatically save documents to appropriate repository folders
- **REQ-031**: The system SHALL generate documents in both Markdown and PDF formats for GitHub-friendly viewing
- **REQ-032**: The system SHALL maintain document version history through Git commits
- **REQ-033**: The system SHALL support document templates that can be customized per project type

2.3.2 Architecture Diagramming

- **REQ-034**: The system SHALL generate Mermaid.js diagrams within the SAD editor
- **REQ-035**: The system SHALL support natural language prompts for diagram generation
- **REQ-036**: The system SHALL allow direct editing of generated diagrams
- **REQ-037**: The system SHALL save diagrams as both Mermaid source files and rendered images in the repository

2.4 Badging and Recognition System

2.4.1 GitHub Badge Generation

- **REQ-038**: The system SHALL generate README badges showing Ignition usage
- **REQ-039**: The system SHALL provide badges for CMMI compliance levels achieved

- **REQ-040**: The system SHALL create badges showing project health metrics (requirements coverage, test coverage, etc.)
- **REQ-041**: The system SHALL generate badges for document completion status
- **REQ-042**: The system SHALL provide customizable badge styles and colors
- **REQ-043**: The system SHALL generate shield.io compatible badge URLs for dynamic updating
- **REQ-044**: The system SHALL create a badge showcase page within the generated documentation

2.4.2 Achievement and Certification Badges

- **REQ-045**: The system SHALL award badges for completing specific milestones (e.g., "First SDP Complete", "100% Requirements Coverage")
- **REQ-046**: The system SHALL provide badges for methodology compliance (e.g., "CMMI Level 3 Aligned")
- **REQ-047**: The system SHALL create time-based badges (e.g., "Project Maintained for 6 Months")
- **REQ-048**: The system SHALL generate badges for collaboration metrics (e.g., "Multi-Contributor Project")
- **REQ-049**: The system SHALL support custom organizational badges for internal recognition
- **REQ-050**: The system SHALL provide badge verification through cryptographic signatures

2.4.3 Social and Portfolio Integration

- **REQ-051**: The system SHALL generate LinkedIn-compatible project summaries with badges
- **REQ-052**: The system SHALL create portfolio-ready project cards with achievement highlights
- **REQ-053**: The system SHALL provide social media sharing templates for project milestones
- **REQ-054**: The system SHALL generate developer profile badges for GitHub profile READMEs

- **REQ-055**: The system SHALL support team badges for collaborative achievements

2.5 Artifact Management Requirements

2.5.1 Requirements Management

- **REQ-056**: The system SHALL provide dashboards for creating and managing requirements
- **REQ-057**: The system SHALL track requirement status and lifecycle
- **REQ-058**: The system SHALL support requirement categorization and prioritization
- **REQ-059**: The system SHALL export requirements to repository files in structured formats (JSON, YAML, CSV)

2.5.2 Configuration Item Management

- **REQ-060**: The system SHALL manage Configuration Items (CIs) with full lifecycle tracking
- **REQ-061**: The system SHALL link CIs to repository artifacts
- **REQ-062**: The system SHALL provide CI status dashboards
- **REQ-063**: The system SHALL generate CI registry files for repository storage

2.5.3 Test Case Management

- **REQ-064**: The system SHALL create and manage test cases
- **REQ-065**: The system SHALL track test case execution status
- **REQ-066**: The system SHALL link test cases to requirements
- **REQ-067**: The system SHALL export test cases to repository-friendly formats

2.5.4 Risk Management as Code

- **REQ-068**: The system SHALL provide structured risk identification and assessment capabilities
- **REQ-069**: The system SHALL manage risk registry as version-controlled JSON/YAML files

- **REQ-070**: The system SHALL support risk categorization (Technical, Schedule, Budget, Resource, External)
- **REQ-071**: The system SHALL track risk probability, impact, and mitigation strategies
- **REQ-072**: The system SHALL provide risk matrix visualization and reporting
- **REQ-073**: The system SHALL link risks to requirements, CIs, and test cases for impact analysis
- **REQ-074**: The system SHALL generate automated risk reports and dashboards
- **REQ-075**: The system SHALL support risk workflow management (identified, assessed, mitigated, closed)
- **REQ-076**: The system SHALL provide risk trend analysis and historical tracking
- **REQ-077**: The system SHALL generate risk-based testing recommendations
- **REQ-078**: The system SHALL support risk-based project scheduling insights
- **REQ-079**: The system SHALL export risk data to project management tools

2.6 Traceability Requirements

2.6.1 Requirements Traceability Matrix (RTM)

- **REQ-080**: The system SHALL provide a comprehensive RTM visualization
- **REQ-081**: The system SHALL show bidirectional links between requirements, CIs, and test cases
- **REQ-082**: The system SHALL identify traceability gaps and coverage issues
- **REQ-083**: The system SHALL export RTM data to repository files for version control
- **REQ-084**: The system SHALL include risk traceability in RTM (requirements to risks, risks to mitigations)

2.7 CMMI Framework Requirements

2.7.1 Process Area Management

- **REQ-085**: The system SHALL provide an interactive CMMI v1.3 Process Areas dashboard
- **REQ-086**: The system SHALL map planning documents to CMMI Process Areas
- **REQ-087**: The system SHALL visualize CMMI compliance status
- **REQ-088**: The system SHALL generate CMMI compliance reports for repository storage
- **REQ-089**: The system SHALL map risk management processes to CMMI Risk Management (RSKM) Process Area

2.8 Repository Integration and File Management

2.8.1 Code-as-Documentation Philosophy

- **REQ-076**: The system SHALL treat all project artifacts as code within the repository
- **REQ-077**: The system SHALL generate human-readable documentation from structured data files
- **REQ-078**: The system SHALL maintain consistency between UI edits and repository file changes
- **REQ-079**: The system SHALL support Git-based workflows for all project artifacts
- **REQ-080**: The system SHALL provide diff-friendly file formats for better change tracking

2.8.2 Repository Template System

- **REQ-081**: The system SHALL provide GitHub repository templates for different project types
- **REQ-082**: The system SHALL generate GitHub Actions workflows for document automation
- **REQ-083**: The system SHALL create repository-specific configuration files (.ignition-config.yml)
- **REQ-084**: The system SHALL support organization-wide template repositories
- **REQ-085**: The system SHALL auto-generate appropriate repository structure during initialization

2.8.3 Continuous Integration for Documentation

- **REQ-086**: The system SHALL provide GitHub Actions for automated document generation
- **REQ-087**: The system SHALL validate document completeness through CI/CD pipelines
- **REQ-088**: The system SHALL auto-update badges based on CI/CD results
- **REQ-089**: The system SHALL generate automated compliance reports on every commit
- **REQ-090**: The system SHALL provide PR templates for document changes
- **REQ-091**: The system SHALL include automated risk assessment validation in CI/CD pipelines

2.9 Branding and Visual Identity Requirements

2.9.1 Mandatory Logo Usage

- **REQ-092**: The system SHALL use the official Ignition logo in all user interfaces
- **REQ-093**: The system SHALL display the "Powered by Castle Bravo" branding consistently
- **REQ-094**: The system SHALL include the official logo in all generated documents and exports
- **REQ-095**: The system SHALL use the official logo in all badges and external representations
- **REQ-096**: The system SHALL maintain logo quality and proportions across all display sizes
- **REQ-097**: The system SHALL provide logo usage guidelines for users and integrators

2.9.2 Brand Consistency

- **REQ-098**: The system SHALL use consistent color scheme derived from the official logo
- **REQ-099**: The system SHALL maintain consistent typography and visual elements
- **REQ-100**: The system SHALL provide branded templates for all document types
- **REQ-101**: The system SHALL include branding in splash screens and loading states

2.10 AI Integration Requirements

2.10.1 Content Generation

- **REQ-102**: The system SHALL integrate with Google Gemini API for AI capabilities
- **REQ-103**: The system SHALL provide AI-assisted content generation for planning documents
- **REQ-104**: The system SHALL generate intelligent suggestions for CIs, requirements, and test cases
- **REQ-105**: The system SHALL provide a chat interface for CMMI and development questions
- **REQ-106**: The system SHALL provide AI-assisted risk identification and assessment
- **REQ-107**: The system SHALL generate risk mitigation strategy suggestions

2.10.2 Enhanced AI Context (Future Enhancement)

- **REQ-108**: The system SHALL compile comprehensive project context for AI analysis
- **REQ-109**: The system SHALL provide holistic project insights and recommendations
- **REQ-110**: The system SHALL provide AI-driven risk correlation analysis

2.11 Analytics and Reporting Requirements

2.11.1 Project Health Dashboards

- **REQ-111**: The system SHALL provide visual insights into artifact status
- **REQ-112**: The system SHALL display coverage metrics and analytics
- **REQ-113**: The system SHALL show GitHub audit results and compliance status
- **REQ-114**: The system SHALL generate repository-stored analytics reports
- **REQ-115**: The system SHALL provide risk dashboard with heat maps and trend analysis
- **REQ-116**: The system SHALL generate executive-level risk summary reports

2.12 Quality Assurance and Validation Requirements

2.12.1 Automated Quality Checks

- **REQ-117**: The system SHALL validate document completeness and consistency
- **REQ-118**: The system SHALL perform automated traceability validation
- **REQ-119**: The system SHALL validate CMMI compliance requirements
- **REQ-120**: The system SHALL check for orphaned requirements, risks, and test cases
- **REQ-121**: The system SHALL validate risk assessment completeness and logic
- **REQ-122**: The system SHALL provide quality gates for document approval workflows

2.12.2 Compliance Validation

- **REQ-123**: The system SHALL validate against industry standards (ISO 9001, FDA, etc.)
- **REQ-124**: The system SHALL provide compliance checklists and validation reports
- **REQ-125**: The system SHALL support custom validation rules and checks
- **REQ-126**: The system SHALL generate audit trails for all quality activities

2.13 Collaboration and Communication Requirements

2.13.1 Team Collaboration Features

- **REQ-127**: The system SHALL support real-time collaborative editing of documents
- **REQ-128**: The system SHALL provide commenting and annotation capabilities
- **REQ-129**: The system SHALL support review and approval workflows
- **REQ-130**: The system SHALL provide activity feeds and notification systems
- **REQ-131**: The system SHALL support team role assignments and permissions

2.13.2 Stakeholder Communication

- **REQ-132**: The system SHALL generate stakeholder-specific reports and dashboards
- **REQ-133**: The system SHALL provide executive summary views with key metrics

- **REQ-134**: The system SHALL support automated report distribution and scheduling
- **REQ-135**: The system SHALL provide presentation-ready exports for stakeholder meetings

3. Non-Functional Requirements

3.1 Performance Requirements

- **REQ-136**: The system SHALL load the main interface within 3 seconds
- **REQ-137**: The system SHALL handle projects with up to 1000 requirements efficiently
- **REQ-138**: The system SHALL provide responsive user interactions (< 1 second response)
- **REQ-139**: The system SHALL efficiently sync large repository structures
- **REQ-140**: The system SHALL handle complex risk correlation analysis within 5 seconds

3.2 Security Requirements

- **REQ-105**: The system SHALL securely store and transmit API keys
- **REQ-106**: The system SHALL implement proper authentication with GitHub
- **REQ-107**: The system SHALL protect against common web vulnerabilities
- **REQ-108**: The system SHALL not commit sensitive data to version control
- **REQ-109**: The system SHALL provide secure badge verification mechanisms

3.3 Compatibility Requirements

- **REQ-110**: The system SHALL support modern web browsers (Chrome, Firefox, Safari, Edge)
- **REQ-111**: The system SHALL be compatible with GitHub API v3/v4
- **REQ-112**: The system SHALL support responsive design for mobile devices
- **REQ-113**: The system SHALL generate files compatible with standard Git workflows

3.4 Reliability Requirements

- **REQ-114**: The system SHALL provide 99.9% uptime when properly deployed
- **REQ-115**: The system SHALL handle network failures gracefully
- **REQ-116**: The system SHALL provide data backup and recovery mechanisms
- **REQ-117**: The system SHALL maintain data integrity during repository synchronization

4. Technical Requirements

4.1 Technology Stack

- **REQ-118**: The system SHALL be built using React and TypeScript
- **REQ-119**: The system SHALL use Mermaid.js for diagram generation
- **REQ-120**: The system SHALL integrate with Google Gemini API
- **REQ-121**: The system SHALL use GitHub API for repository operations
- **REQ-122**: The system SHALL use Shield.io API for badge generation

4.2 Development Environment

- **REQ-123**: The system SHALL support local development with hot reloading
- **REQ-124**: The system SHALL provide environment variable configuration
- **REQ-125**: The system SHALL include comprehensive development documentation
- **REQ-126**: The system SHALL provide repository structure validation tools

5. Deployment Requirements

5.1 Enhanced Deployment Options

5.1.1 Static Site Hosting

- **REQ-127**: The system SHALL support deployment to Netlify with automatic builds
- **REQ-128**: The system SHALL support deployment to Vercel with zero configuration

- **REQ-129**: The system SHALL support deployment to GitHub Pages
- **REQ-130**: The system SHALL support deployment to AWS S3 + CloudFront

5.1.2 Container Deployment

- **REQ-131**: The system SHALL provide Docker containerization
- **REQ-132**: The system SHALL support Docker Compose for local development
- **REQ-133**: The system SHALL support Kubernetes deployment manifests

5.1.3 CI/CD Pipeline

- **REQ-134**: The system SHALL provide GitHub Actions workflows for automated deployment
- **REQ-135**: The system SHALL include automated testing in CI/CD pipeline
- **REQ-136**: The system SHALL support environment-specific deployments (dev, staging, prod)
- **REQ-137**: The system SHALL provide automated badge updates through CI/CD

5.1.4 Environment Management

- **REQ-138**: The system SHALL support environment variable injection for different deployment targets
- **REQ-139**: The system SHALL provide secure API key management for production deployments
- **REQ-140**: The system SHALL support feature flags for gradual rollouts
- **REQ-141**: The system SHALL provide deployment-specific repository configuration

5.2 Build and Distribution

- **REQ-142**: The system SHALL provide optimized production builds
- **REQ-143**: The system SHALL support CDN distribution for static assets
- **REQ-144**: The system SHALL implement proper caching strategies

- **REQ-145**: The system SHALL provide bundle analysis and optimization
- **REQ-146**: The system SHALL generate repository-ready deployment packages

6. Infrastructure Requirements

6.1 Hosting Requirements

- **REQ-147**: The system SHALL support horizontal scaling for high availability
- **REQ-148**: The system SHALL implement proper load balancing
- **REQ-149**: The system SHALL provide SSL/TLS encryption for all communications
- **REQ-150**: The system SHALL support custom domain configuration
- **REQ-151**: The system SHALL provide badge endpoint hosting with high availability

6.2 Monitoring and Observability

- **REQ-152**: The system SHALL provide application performance monitoring
- **REQ-153**: The system SHALL implement error tracking and alerting
- **REQ-154**: The system SHALL provide usage analytics and metrics
- **REQ-155**: The system SHALL support log aggregation and analysis
- **REQ-156**: The system SHALL monitor badge generation and delivery success rates

7. Maintenance and Support Requirements

7.1 Updates and Versioning

- **REQ-157**: The system SHALL support semantic versioning
- **REQ-158**: The system SHALL provide automated dependency updates
- **REQ-159**: The system SHALL maintain backward compatibility for project files
- **REQ-160**: The system SHALL provide migration tools for breaking changes

- **REQ-161**: The system SHALL support repository structure migrations

7.2 Documentation and Training

- **REQ-162**: The system SHALL provide comprehensive user documentation
- **REQ-163**: The system SHALL include video tutorials and walkthroughs
- **REQ-164**: The system SHALL provide API documentation for extensions
- **REQ-165**: The system SHALL maintain a knowledge base for common issues
- **REQ-166**: The system SHALL provide repository setup and best practices guides

8. Future Enhancement Requirements

8.1 Integration Enhancements

- **REQ-167**: The system SHALL support integration with Jira for issue tracking
- **REQ-168**: The system SHALL support integration with Confluence for documentation
- **REQ-169**: The system SHALL support integration with Slack for notifications
- **REQ-170**: The system SHALL provide webhook support for external integrations
- **REQ-171**: The system SHALL support organization-wide badge systems

8.2 Advanced Features

- **REQ-172**: The system SHALL support multi-project workspaces
- **REQ-173**: The system SHALL provide advanced reporting and analytics
- **REQ-174**: The system SHALL support custom templates and methodologies
- **REQ-175**: The system SHALL provide role-based access control
- **REQ-176**: The system SHALL support enterprise badge management and verification

9. Acceptance Criteria

9.1 Core Functionality

- All document editors must be fully functional with save/load capabilities
- GitHub integration must successfully connect and perform all operations
- AI features must provide relevant and contextual suggestions
- RTM must accurately display all traceability relationships

9.2 Performance

- Application must load within 3 seconds on standard broadband
- All user interactions must respond within 1 second
- System must handle 1000+ artifacts without performance degradation

9.3 Deployment

- Application must deploy successfully to at least 3 different hosting platforms
- CI/CD pipeline must complete without errors
- Production deployments must be secure and scalable

10. Risk Assessment

10.1 Technical Risks

- **API Rate Limiting**: Gemini API usage may be throttled for high-volume operations
- **Browser Compatibility**: Advanced features may not work in older browsers
- **Data Migration**: Project file format changes may require migration tools

10.2 Mitigation Strategies

- Implement API request queuing and retry logic
- Provide graceful degradation for unsupported browsers

- Maintain backward compatibility and migration utilities