Acceptance Criteria

Generated by Requirements Gathering Agent v2.1.2
Category: technical-analysis
Generated: 2025-06-10T08:18:25.448Z
Description: Comprehensive acceptance criteria and validation methods

---

Acceptance Criteria for Requirements Gathering Agent

These acceptance criteria are structured using the Given-When-Then format and categorized by feature. They address both functional and non-functional requirements.

I. Core Functionality: PMBOK Document Generation

- **Requirement:** Generate a complete PMBOK document suite (29 documents).
  - **Given** a valid project README.md file and associated project documentation (as defined in "Supported Project Structures"),
  - **When** the `requirements-gathering-agent` command is executed without any optional flags,
  - **Then** 29 PMBOK-compliant documents are generated in the `generated-documents` directory, with each document containing the appropriate sections and content based on the provided project context. The directory structure should match the specified output structure. All generated files must be valid markdown files.
  - **Given** a project with minimal documentation (only a README.md),
  - **When** the `requirements-gathering-agent` command is executed,
  - **Then** the generated documents should contain placeholder content or appropriate messages indicating insufficient information, rather than failing or producing nonsensical output.
- **Requirement:** Generate documents with PMBOK 7.0 validation.
  - **Given** a project README.md and associated documentation,
  - **When** the `requirements-gathering-agent --validate-pmbok` command is executed,
  - **Then** all generated documents are validated against PMBOK 7.0 standards. A validation report (e.g., JSON or text file) is generated indicating compliance status for each document, including specific areas needing improvement. The report should include a summary of overall compliance and a score (0-100).

- **Requirement:** Generate documents with comprehensive validation and quality assessment.
  - **Given** a project README.md and associated documentation,
  - **When** the `requirements-gathering-agent --generate-with-validation` command is executed,
  - **Then** all generated documents undergo PMBOK 7.0 validation and a quality assessment (0-100 score) based on structure, content, and terminology. A detailed report is generated, providing scores and actionable recommendations for improvement for each document.
- **Requirement:** Support selective document generation.
  - **Given** the command-line options `--core-only`, `--management-plans`, `--planning-artifacts`, `--technical-analysis`, `--generate-stakeholder`,
  - **When** these options are used with the `requirements-gathering-agent` command,
  - **Then** only the specified categories of PMBOK documents are generated.

II. Enhanced Project Analysis

- **Requirement:** Discover and analyze all relevant markdown files beyond README.md.
  - **Given** a project directory structure containing markdown files in common documentation locations (as specified in "Supported Project Structures"),
  - **When** the `requirements-gathering-agent` command is executed,
  - **Then** the tool should automatically discover and process all relevant markdown files within the specified directory structure, up to three levels deep, excluding system directories and generated content. A log should be produced detailing the discovered files and their relevance scores.
- **Requirement:** Assign relevance scores (0-100) to discovered files.
  - **Given** a set of discovered markdown files,
  - **When** the project analysis is performed,
  - **Then** each file is assigned a relevance score (0-100) based on content relevance, file location, naming patterns, content structure, and content depth. The scoring methodology should be documented and transparent.
- **Requirement:** Intelligently categorize discovered files.
  - **Given** a set of discovered markdown files with relevance scores,
  - **When** the project analysis is performed,
  - **Then** files are categorized into: Primary, Planning, Development, Documentation, and Other. The categorization should be consistent and reflect the file's content and relevance to the project.

III. Enhanced Context Manager

- **Requirement:** Utilize up to 90% context for large language models (e.g., Gemini 1.5 Pro).
  - **Given** a large language model with a 2M token context window (e.g., Gemini 1.5 Pro) is configured,
  - **When** the `requirements-gathering-agent` command is executed,
  - **Then** the Enhanced Context Manager should utilize up to 90% of the available context window for document generation. A report should detail the context utilization for each document.
- **Requirement:** Adapt context strategy based on AI model capabilities.
  - **Given** different AI models (e.g., GPT-4, Gemini 1.5 Pro, Claude) are configured,
  - **When** the `requirements-gathering-agent` command is executed,
  - **Then** the Enhanced Context Manager should automatically adapt its context strategy (3-phase approach) to optimize context utilization for each model type, maximizing accuracy without exceeding token limits. A report should show the selected strategy and context utilization for each model.
- **Requirement:** Provide detailed context reports and metrics.
  - **Given** the command-line options `--context-report`, `--analyze-context`, `--context-metrics`,
  - **When** these options are used,
  - **Then** the tool provides detailed reports on context utilization, performance metrics, and optimization recommendations.

IV. Error Handling and Robustness

- **Requirement:** Handle API errors and retries.
  - **Given** an API request to an AI provider fails (e.g., due to network issues or rate limits),
  - **When** the `--with-retry` option is used,
  - **Then** the tool should automatically retry the failed request a specified number of times before reporting the error. The retry logic should include appropriate delays and error handling.
- **Requirement:** Provide informative error messages.
  - **Given** any error occurs during execution (e.g., file access errors, API errors, validation errors),
  - **When** the error occurs,
  - **Then** the tool should provide clear and informative error messages to the user, indicating the nature of the error and suggesting possible solutions.

V. Non-Functional Requirements

- **Performance:** The tool should generate all 29 documents within a reasonable timeframe (e.g., under 10 minutes for a typical project) when using a high-performance AI provider. Performance testing should be conducted with varying project sizes and document generation options.
- **Security:** The tool should securely handle API keys and authentication credentials using environment variables and best practices. The code should be reviewed for security vulnerabilities.
- **Usability:** The CLI should be intuitive and easy to use, with clear documentation and helpful error messages. The generated documents should be well-formatted and easy to read.

These acceptance criteria provide a comprehensive framework for testing and validating the Requirements Gathering Agent. Specific test cases can be derived from these criteria to ensure complete coverage. Further criteria may be needed based on detailed user stories and edge case analysis.