# Product Requirement Document: Agentic AI Academic Review System (AARIS)

| **Attribute** | **Detail** |
| --- | --- |
| **Product Name** | Agentic Academic Review System (AARIS) |
| **Version** | 1.0 (Initial Release) |
| **Target Audience** | Academic Journal Editors, Publishers, Reviewers |
| **Status** | Draft |
| **Target Release** | Q4 2025 |

## 1. Introduction & Goals

### 1.1 Objective

The primary objective of AARIS is to leverage a Multi-Agent System (MAS) architecture, powered by Large Language Models (LLMs) and advanced tooling, to automate and enhance the initial screening, quality assessment, and structured feedback generation for academic manuscripts, significantly reducing the burden on human editors and reviewers while improving review quality and turnaround time.

### 1.2 Key Results (KRs)

1. **Reduce Review Time:** Decrease the average time from manuscript submission to initial assessment (Agent Report Generation) by 90% (e.g., from 48 hours to less than 5 hours).
2. **Improve Quality Assessment Consistency:** Achieve an Inter-Agent Reliability score (Krippendorff's Alpha or similar metric comparing agent assessments on standardized test data) of over 0.85 for "Overall Scientific Quality" and "Novelty" scores.
3. **Increase User Satisfaction:** Achieve a human Editor satisfaction score of 4.5/5 on the utility and clarity of the agent-generated review report.

## 2. Problem Statement

The traditional academic peer review process is plagued by three core issues:

1. **Time Lag:** Manual reviewer selection and the time required for human review often lead to months-long delays, slowing down the dissemination of critical research.
2. **Reviewer Fatigue:** Skilled reviewers are often overwhelmed, leading to declining response rates and lower-quality, rushed feedback.
3. **Inconsistent Quality:** Reviews can be subjective, incomplete, or lack a deep quantitative check (e.g., complex statistical validation).

AARIS solves this by providing a consistent, detailed, and lightning-fast pre-review assessment, allowing human editors to focus on nuanced judgment and decision-making rather than repetitive screening tasks.

## 3. Stakeholders & User Personas

| **Persona** | **Role in System** | **Key Pain Points Solved by AARIS** |
| --- | --- | --- |
| **Journal Editor** | System Administrator, Decision Maker | Excessive time spent on initial screening; difficulty getting timely reviews. |
| **Reviewer** | End User (Receives Agent Summary) | Reviewing papers from scratch is time-consuming; need for pre-vetted material. |
| **Author** | Indirect User | Long wait times for a decision; receiving vague or inconsistent feedback. |

## 4. Solution Overview: The Multi-Agent System (MAS)

AARIS will utilize a Multi-Agent System (MAS) architecture, where specialized, goal-driven AI agents collaborate under an Orchestrator. This modular approach ensures robustness, specialization of tasks, and the ability to integrate reflection loops.

### 4.1 Agentic Design Patterns

The system will employ the following core agentic design patterns:

1. **Tool Use (Acting):** Agents will be provisioned with specialized tools (e.g., Google Search for literature checks, Plagiarism APIs, static code analysis tools).
2. **Planning (Reasoning):** The Orchestrator will dynamically generate and decompose the review task into sub-tasks for specialized agents.
3. **Reflection:** Agents will be instructed to critique their own output before reporting to the Orchestrator, improving accuracy (e.g., "Review the plagiarism report and justify the severity score").

## 5. Detailed Functional Requirements (Agent Roles & Features)

The system will feature a minimum of five specialized agents working in parallel.

| **Agent Role** | **Primary Function** | **Key Features & Output** |
| --- | --- | --- |
| **1. Orchestrator Agent (The Editor)** | Workflow management, task delegation, result integration, final report generation. | Manages the sequence of agent execution, handles inter-agent communication, and structures the final output report. |
| **2. Methodology & Statistics Agent** | Assesses the rigor of the research design and data analysis. | **Output:** Score (1-5) and critique on: Sampling Methods, Control Groups, Statistical Methods, Data Sufficiency, and Reproducibility. |
| **3. Literature & Novelty Agent** | Contextualizes the work against existing research. | **Tool Use:** Google Search (or other academic databases). **Output:** Score and critique on: Originality, Timeliness, Relevance of Citations (identifies outdated or missing seminal works), and placement of the paper within the field. |
| **4. Clarity & Presentation Agent** | Evaluates readability and adherence to journal standards. | **Output:** Score and critique on: Language Quality (grammar, clarity, ambiguity), Manuscript Structure (logical flow), and Figure/Table Appropriateness (labels, clarity, data representation). |
| **5. Integrity & Ethics Agent** | Performs essential compliance and integrity checks. | **Tool Use:** Plagiarism API, Conflict-of-Interest (CoI) pattern matching. **Output:** Report on: Plagiarism Score (similarity %), CoI indicators (e.g., self-citation density, common author/reviewer names), and Ethical Considerations (if declared or implied). |
| **6. Synthesis Agent** | Creates the final, unified human-readable output. | **Output:** A structured PDF/Markdown report containing: Summary of the Paper, Strengths, Major Issues, Minor Issues, and an aggregated Recommendation (Accept/Revise/Reject). |

## 6. Non-Functional Requirements

| **Requirement Type** | **Description** |
| --- | --- |
| **Performance** | Agent-generated review reports must be completed within **4 hours** of submission. All agents must operate concurrently. |
| **Scalability** | Architecture must be able to handle simultaneous processing of **50 manuscripts** without performance degradation. |
| **Security & Privacy** | All manuscript data must be encrypted in transit and at rest (Firestore). LLM API calls must use secure, non-logging endpoints. |
| **Explainability (XAI)** | The final Synthesis Report must include a section justifying each major finding by referencing the specific text in the manuscript and the action taken by the responsible agent. |
| **Data Persistence** | All submissions, agent intermediate states, and final reports must be stored securely using Firebase Firestore. |

## 7. Proposed Agentic AI Architecture

The AARIS architecture is a classic Multi-Agent System (MAS) structure based on the **Orchestration/Delegation Pattern** .

### 7.1 Core Architectural Components

| **Component** | **Description** | **Technologies / Role** |
| --- | --- | --- |
| **User Interface (UI)** | A simple web application (HTML/React/Angular) for manuscript upload and viewing the final report. | Presentation Layer. Must handle secure file upload. |
| **API Gateway & Ingestion** | Handles manuscript file (PDF/DOCX) upload, parses the document into structured text/JSON, and triggers the Orchestrator. | Document Parsers, Secure Endpoint. |
| **Agent Orchestrator (The Planner)** | The brain of the system. Receives the parsed manuscript, generates a dynamic multi-step plan, delegates tasks to the specialist agents, and monitors progress. | LLM-based Planning Engine (ReAct/Reflexion loops), Workflow Manager. |
| **Specialist Agents (5x)** | Autonomous LLM instances pre-prompted with specific roles and access to specialized tools. They execute the tasks assigned by the Orchestrator. | LLM (gemini-2.5-flash-preview-09-2025), Individual Prompting, Tool-Use Logic. |
| **External Tool Layer (The Environment)** | A set of controlled, external APIs and databases the agents can call (Actuators). | **Tools:** Plagiarism API, Citation Database (Google Search), Code Analysis Sandbox (if applicable). |
| **Memory & State (Firestore)** | Centralized database for long-term and short-term memory, coordination, and state management. | **Collections:** /artifacts/{appId}/public/data/submissions (for paper and report), /artifacts/{appId}/public/data/agent\_state (for coordination). |

### 7.2 Data Flow and Coordination

1. **Submission:** Editor uploads the manuscript via the UI.
2. **Ingestion:** The document is parsed into text/JSON format and saved as a document in the submissions collection in Firestore.
3. **Orchestration:** The **Orchestrator Agent** is activated, loads the document content, and generates a plan: "Task 1: Run Methodology Agent. Task 2: Run Novelty Agent...".
4. **Parallel Execution:** The Orchestrator pushes specialized tasks and relevant context snippets to the respective **Specialist Agents** via a shared state in the agent\_state collection.
5. **Tool Use & Reflection:** Each Specialist Agent executes its task. For example, the **Novelty Agent** uses its Google Search tool to check recent literature, runs a reflection loop on its findings, and saves its structured critique back to the agent\_state document.
6. **Synthesis & Consolidation:** Once all Specialist Agents report Task\_Completed status, the **Orchestrator** activates the **Synthesis Agent**.
7. **Final Report:** The Synthesis Agent compiles all critiques, scores, and recommendations into the final, cohesive **Review Report** document (Markdown/PDF format) and saves it back to the submissions collection.
8. **Notification:** The UI is updated, notifying the Editor that the Agent Review is ready for human decision.

### 7.3 Data Persistence Requirements (Firebase Firestore)

**Mandatory Firestore Initialization:**

const appId = typeof \_\_app\_id !== 'undefined' ? \_\_app\_id : 'default-app-id';  
const firebaseConfig = JSON.parse(typeof \_\_firebase\_config !== 'undefined' ? \_\_firebase\_config : '{}');  
  
// Note: Initialization and Auth will be handled in the client application (UI).  
// The user's UID will define which documents they can access.  
// Example Collection Path (Public/Shared data for a single submission):  
// `/artifacts/{appId}/public/data/submissions/{submissionId}`

## 8. Success Metrics (KPIs)

| **Metric Category** | **Key Performance Indicator (KPI)** | **Target** |
| --- | --- | --- |
| **Efficiency** | **Time-to-Review Report (TTRR)**: Time from submission to final Synthesis Agent report availability. | < 5 hours |
| **Quality** | **Inter-Agent Reliability (IAR)**: Agreement score between agents on a sample set of papers. | > 85% |
| **Utility** | **Editor Time Saved:** Average time reduction for the editor in the initial screening process. | 50% reduction |
| **Adoption** | **Agent Usage Rate:** Percentage of submitted papers processed by AARIS before human review. | 100% |