# **Architectural Decision Record (ADR) for Guardrails in AI Fraud Detection and Short-Answer Grading System**

## **1. Title**

Implementing Guardrails for AI Fraud Detection and Short-Answer Grading System

## **2. Context**

The AI Fraud Detection System processes responses from two databases:

* Aptitude Test Ungraded Database
* Submission Ungraded Database

The system applies AI-based fraud detection techniques, including:

* Answer Collusion Detection using SBERT + FAISS for semantic similarity
* Time-Based Answer Manipulation using LSTM-based anomaly detection
* AI-Based Behavioral Fraud Detection using Isolation Forest for abnormal submission patterns

To ensure data integrity, accuracy, security, and efficiency, we need to apply guardrails before:

* Serving fraud detection results to experts to prevent false positives
* Storing flagged cases in the system to avoid unnecessary reprocessing
* Caching validated summaries to optimize response times

Additionally, the Short-Answer Grading System requires structured input validation and output validation to ensure:

* Candidate responses are formatted correctly before processing
* AI-generated grading results follow a structured format with a valid grade, feedback, and confidence score
* AI hallucinations and inconsistencies are eliminated before sending results to candidates

## **3. Decision**

We will implement guardrails in both the fraud detection and short-answer grading system to:

* Validate AI-generated fraud alerts before presenting them to experts
* Ensure fraud detection outputs follow a structured format (e.g., JSON)
* Apply structured XML-based input prompts for AI in short-answer grading
* Validate AI grading output to ensure a structured response with grade, feedback, and confidence score
* Detect hallucinations such as incorrectly flagged fraudulent cases or invalid grading outputs
* Enforce security measures to prevent unauthorized modifications and prompt injection attacks

### **Guardrails Implementation Strategy**

#### **Input Validation**

* Ensure candidate answers are properly formatted before processing
* Use structured XML-based prompts for LLM grading requests
* Sanitize database queries to prevent SQL injections
* Prevent incorrect input structures from being processed

#### **Output Filtering & Validation**

* AI-generated fraud results must follow a strict JSON schema
* Validate fraud confidence scores and ensure meaningful thresholds
* Validate grading output to ensure it contains:
  + grade (integer between 0-100)
  + feedback (structured text following predefined format)
  + confidence\_score (between 0-100)
* Flag inconsistent fraud classifications if the same answer is flagged differently in separate runs
* Implement response validation before sending AI-generated grades to candidates

#### **Security & Compliance**

* Prevent prompt injection attacks by filtering special characters in inputs
* Restrict access to flagged fraud reports so that only authorized reviewers can see them
* Implement logging and monitoring for AI outputs to detect any anomalies in fraud detection or grading

## **4. Alternatives Considered**

|  |  |
| --- | --- |
| **Alternative** | **Reason for Rejection** |
| No Guardrails (Direct AI Output to Experts and Candidates) | Risks false positives, hallucinations, and biased outputs |
| Rule-Based Filtering Only (No AI) | Fails to detect complex fraud patterns such as semantic similarity and time anomalies |
| Human-Only Review (No AI Pre-Filtering) | Too slow, not scalable for high-volume fraud detection and grading |

Chosen Approach: Hybrid AI + Guardrails System

* AI detects fraud patterns and grades responses, but guardrails refine the output before final review

## **5. Architecture Impact**

* Fraud Processing Module applies AI fraud detection models
* Guardrails Layer filters, validates, and refines AI-generated fraud alerts
* Short-Answer Grading Module applies XML-structured input and validates output
* Validated results are stored in the Fraud Reports Database and Candidate Results Database
* Cached summaries reduce repeated fraud processing

## **6. Risks & Mitigation**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Impact** | **Mitigation Strategy** |
| False fraud alerts | Medium | Expert reviews flagged cases before action |
| Hallucinated AI outputs | High | Use FAISS validation and confidence thresholding |
| Prompt injection attacks | High | Sanitize user inputs before processing |
| High system latency | Medium | Cache validated fraud reports for faster access |
| Incorrect AI grading outputs | High | Validate grading output structure before sending to candidates |

## **7. Acceptance Criteria**

* Fraud alerts must be JSON-validated before reaching experts
* Less than 5 percent false positives in AI fraud detection
* Guardrails must prevent prompt injections and hallucinations
* Caching must reduce fraud processing time by 50 percent
* AI-generated grading results must contain a valid grade, feedback, and confidence score in the correct format

## **8. Implementation Plan**

* Add structured JSON validation to AI fraud detection pipeline
* Integrate AI hallucination detection filters before serving responses
* Cache fraud results after guardrail validation to optimize performance
* Implement XML-based structured prompts for LLM grading input
* Validate AI grading outputs before presenting to candidates
* Monitor system accuracy and fine-tune fraud detection and grading models

## **9. Decision Status**

Approved – Implementation in progress