# Software Requirements Specification (SRS)

## Artificial Assessment Intelligence for Educators (AAIE)

**Module:** Model (LLM) Development
**Version:** 0.1
**Date:** August 4, 2025

# 1. Introduction

### 1.1 Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the custom LLM-based academic integrity detection and feedback system, developed under the AAIE (Artificial Assessment Intelligence for Educators) project. In the current scope, this tool aims to classify student-written submissions as AI-generated, human-written, or hybrid using large language models and prompt engineering techniques. It is also capable of providing rubric-aligned feedback generation to assist educators. The project is in its experimental phase with a focus on prompting, evaluation, and model performance benchmarking.

### 1.2 Scope

This system allows users (educators only) to:

- Upload and validate student submissions in supported formats (DOCX, PDF).
- Upload only essay-based or short-answer questions.
- Detect AI-generated, human-written, or hybrid content.
- Align feedback with predefined academic rubrics.
- Export detection and feedback results in structured formats (JSON, CSV, PDF) for educators.
- *(Future) Provide percentage-based AI content estimation with confidence scores.*
- *(Future) Generate inline or section-based annotations highlighting AI-detected text.*
  *(Future) Accept student-declared AI usage reports (meta-submissions) and verify alignment with actual content*
- *(Future) Provide controlled feedback visibility or AI usage analysis to students under defined policies*
- *(Future) Enable downloadable feedback reports for student review*
- *(Future) Integration with LMS (Learning Management Systems) via APIs.*
- *(Future) Support for multilingual and multi-subject submissions.*

### 1.3 Definitions, Acronyms, and Abbreviations

- **AAIE:** Artificial Assessment Intelligence for Educators
- **LLM:** Large Language Model
- **RAG:** Retrieval-Augmented Generation
- **FSL:** Few-Shot Learning
- **ZSL:** Zero-Shot Learning
- **CoT:** Chain of Thought
- **AI**: Artificial Intelligence
- **BPE**: Byte Pair Encoding
- **GPT**: Generative Pre-trained Transformer
- **BOW**: Bag of Words
- **DOCX**: Microsoft Word Document
- **PDF**: Portable Document Format

# 2. Overall Description

## 2.1 Product Perspective

The AAIE LLM subsystem is a backend component of a broader academic integrity pipeline. It integrates prompt engineering, inference through APIs or local models, and evaluation pipelines into a unified backend engine. Current experiments focus on prompt templates with multiple LLMs and prototype interfaces for educators.

## 2.2 User Needs

- Instructors want to verify if AI tools were used in assignment submissions.
- Developers require an experimental sandbox to test prompt strategies and models.
- University admins seek reliable reporting on AI-assisted academic dishonesty.
- Students require transparency in how AI use is detected and evaluated.

## 2.3 Current Constraints

- Submissions must be in text or convertible format (DOCX, PDF).
- Prompts must operate within the token limits of the selected LLM.
- Evaluation assumes access to labeled validation sets.
- Real-time inference is limited by model latency or API rate limits.

## 2.4 Assumptions and Dependencies

- The input data is student-written academic content (essays, short responses).
- The system will evolve from API-based LLM use to self-hosted open-source models.
- Evaluation methods assume ground truth labels (Human, AI, Hybrid).
- All text will be pre-processed (e.g., punctuation stripping, lowercasing) before model input.

# 3. System Features

## 3.1 Upload Submission

**Description:**
Allows users (educators or admins) to upload student submission data in .docx, or .pdf format. Text is extracted and cleaned for classification.

**User Story:**
As an instructor, I want to upload a student's essay so it can be checked for potential AI use.

**Acceptance Criteria:**

- Accepts .txt, .docx, and .pdf formats.
- Extracts clean text with consistent formatting.
- Displays a preview and word count before analysis.
- Submissions are logged with metadata (file name, date, course ID).

## 3.2 Classify Submission

**Description:**
Analyzes the submission text to classify it as AI-generated, human-written, or hybrid, and calculates a percentage estimate of AI involvement.

**User Story:**
As an instructor, I want the system to detect AI usage in student work so I can ensure academic integrity.

**Acceptance Criteria:**

- Uses prompt-engineered LLM queries for classification.
- Returns a categorical label: "AI-generated," "Human-written," or "Hybrid."
- Stores detection results in the system for future review.
- Works on entire documents, not just sections.

## 3.3 Feedback Generation

**Description:**
Generates general feedback for a submission aligned to a preloaded academic rubric.

**User Story:**
As an instructor, I want to receive structured feedback mapped to grading criteria so I can speed up assessment.

**Acceptance Criteria:**

- Accepts rubrics in .csv or .json format.
- Generates feedback per rubric criterion.
- Summarizes strengths and areas for improvement.
- Feedback output is editable before saving.

### 3.4 Export Results

**Description:**
Exports detection and feedback results for use outside the platform.

**User Story:**
As an instructor, I want to save reports so I can share them with students or academic review committees.

**Acceptance Criteria:**

- Exports in , CSV, and JSON formats.
- Includes metadata (student name/ID, submission date, course ID).
- Preserves rubric feedback in all formats.
- Supports both manual download and API export

## 4. Future Work

### 4.1 Percentage-Based AI Detection with Confidence Scores

**Description:**
Adds the ability to estimate the proportion of a document likely generated by AI, along with a confidence score for the classification. This provides more granular insight than categorical results.

**User Story:**
As an academic integrity officer, I want to know the approximate percentage of AI-generated content in a submission so I can make more informed decisions.

**Acceptance Criteria:**

- Outputs a numerical percentage estimate (0–100%) for AI-generated content.
- Includes a confidence score (0–1 or 0–100%) representing model certainty.
- Supports whole-document analysis (not section-based in initial version).
- Results are stored alongside the standard classification.

## 4.2 Highlighting AI-Detected Text

**Description:**
Provides visual annotations within the submission to indicate AI-detected segments, either sentence-by-sentence or by section.

**User Story:**
As an instructor, I want to see exactly which sections of a student's work may be AI-generated so I can review them in context.

**Acceptance Criteria:**

- Color-codes text segments based on classification (AI-generated, human-written, uncertain).
- Allows toggling annotations on/off.
- Supports both full-document and section-based modes.
- Includes a legend explaining highlight colours.

## 4.3 Accept Student-Declared AI Usage Reports and Verify Alignment

**Description:**
Allows students to submit a declaration of how and where they used AI in their work, and compares it to the system's AI detection results for verification.

**User Story:**
As an instructor, I want to see if a student's declared AI usage matches the system's findings so I can assess honesty and policy compliance.

**Acceptance Criteria:**

- Accepts a secondary text submission describing AI use.
- Aligns declared AI use sections with detected sections.
- Flags discrepancies between declared and detected usage.
- Generates a summary report of matches and mismatches.

## 4.4 Controlled Feedback Visibility or AI Usage Analysis to Students

**Description:**
Allows institutions to define rules for what feedback or AI usage information is visible to students, depending on policy.

**User Story:**
As a university admin, I want to control whether students see full AI usage details, summary results, or no details, to align with institutional policy.

**Acceptance Criteria:**

- Supports configurable visibility settings (e.g., full, summary, none).
- Applies settings consistently to all student-facing views.
- Logs when and how feedback is shared.
- Ensures restricted data is hidden from unauthorized access.

## 4.5 Downloadable Feedback Reports for Student Review

**Description:**
Allows students to download a formatted feedback report containing rubric-based feedback and AI detection outcomes.

**User Story:**
As a student, I want to download a feedback report so I can review and address areas for improvement.

**Acceptance Criteria:**

- Generates PDF reports with institution branding.
- Includes feedback text, rubric scores, and optional AI usage details.
- Accessible from the student portal once released by instructor.
- Supports download logging for audit purposes.

## 4.6 Integration with LMS (Learning Management Systems) via APIs

**Description:**
Enables automatic exchange of submission data, detection results, and feedback between the AAIE platform and supported LMS platforms, including Deakin's OnTrack and DeakinSync.

**User Story:**
As an instructor, I want results to sync automatically to my LMS (including OnTrack and DeakinSync) so I don't have to manually transfer files or scores.

**Acceptance Criteria:**

- Provides secure REST or GraphQL API endpoints.
- Supports push and pull operations for submissions, results, and feedback.
- Fully compatible with OnTrack, DeakinSync, and major LMS platforms (Moodle, Canvas, Blackboard).
- Allows scheduled and real-time syncing options.
- Logs all data exchanges for audit and compliance purposes.

### 4.7 Support for Multilingual and Multi-Subject Submissions

**Description:**
Extends AI detection and feedback capabilities to handle multiple languages and discipline-specific writing styles.

**User Story:**
As an instructor, I want the system to detect AI-generated content in multiple languages and subject areas so I can use it across my courses.

**Acceptance Criteria:**

- Detects AI-generated content in at least 5 major languages.
- Supports subject-specific detection models for different disciplines.
- Allows instructors to select language and subject during upload.
- Maintains performance within agreed response times for multilingual models.

## 5. External Interface Requirements

### 5.1 User Interface

The LLM Module will be accessed via the AAIE web platform dashboard, offering a clean and accessible layout for educators and admins.

**UI Components:**

- **Upload Panel:** Drag-and-drop or file picker for .docx, .pdf, .txt submissions.
- **Submission Preview:** Displays extracted text preview, word count, and file metadata before analysis.
- **Detection Results View:** Shows categorical classification ("AI-generated," "Human-written," or "Hybrid") and generated feedback aligned to rubric criteria.
- **Model Comparison View:** Side-by-side display of outputs from different LLM backends.
- **Rubric Feedback Panel:** Structured feedback view organized by rubric criteria.
- **Export Buttons:** One-click export to .pdf, .csv, .json.
- **Admin Controls:** Manage rubrics, configure LMS integration, and adjust feature availability.

### 5.2 Software Interfaces

- **Backend APIs:** REST and GraphQL endpoints for submission processing, results retrieval, and rubric management.
- **Model Interfaces:** Integration with multiple LLM APIs (e.g., OpenAI, Anthropic) and/or local deployment of fine-tuned models using Hugging Face transformers.
- **Data Processing Libraries:** Python libraries such as pandas, nltk, and langchain for text processing and orchestration.

- **Export Services:** PDF generation using reportlab, JSON export via standard serialization, and CSV via pandas.
- **LMS Integration:** API connectors for onTrack, DeakinSync, Moodle, Canvas, and Blackboard.

### 5.3 Hardware Interfaces

- No specialized hardware required for users; accessed via web browser.
- Backend requires cloud or on-premise compute environment with GPU acceleration for LLM inference (e.g., NVIDIA A100, RTX 6000).

## 6. Non-Functional Requirements

### 6.1 Performance

- Must process and return classification + rubric feedback for documents ≤5000 words within 10 seconds under normal load.
- Model comparison processing must complete within 15 seconds.
- Export operations should complete in ≤3 seconds for supported formats.

### 6.2 Security

- All uploaded documents transmitted via HTTPS with TLS 1.2 or higher.
- Documents stored securely with access controls and automatic deletion after policy-defined retention period.
- Compliance with GDPR, FERPA, and Deakin University's data protection policies.
- LMS integrations must use secure, token-based authentication (OAuth 2.0).

### 6.3 Reliability

- System must handle unsupported or corrupted files gracefully, providing clear error messages.
- Backend services must have 99.5% uptime excluding scheduled maintenance.
- Automatic retry for failed model inference requests up to 3 attempts.
- Comprehensive logging for all processing steps for troubleshooting and auditing.

### 6.4 Usability

- Interface must be intuitive and require minimal training for instructors.
- Support for screen readers and WCAG 2.1 AA accessibility compliance.
- All actions should be achievable within 3 clicks from the dashboard home.

#### 6.5 Maintainability

- Modular backend architecture to allow independent updates to model integration, export services, and UI components.
- All code documented according to project standards with inline comments and API documentation.

# 7. Appendices

## 7.1 Glossary

- **LLM:** an AI system trained on extensive datasets for language understanding and generation.
- **AI-generated content:** Text produced primarily by an AI language model.
- **Hybrid content:** Submission containing both AI-generated and human-written text.
- **Prompt engineering:** The practice of designing inputs (prompts) to elicit optimal outputs from an AI model.
- **Few-shot learning:** Technique where a model is given a few labeled examples to guide its responses.
- **LMS:** Learning Management System; software for managing educational courses and assessments.
- **OnTrack / DeakinSync:** Deakin University's official academic management and student portal systems.
- **REST API:** Representational State Transfer Application Programming Interface; allows software systems to communicate via HTTP requests.
- **GraphQL:** Query language for APIs enabling clients to request specific data.
- **LSTM**: Long Short-Term Memory, a deep learning architecture for learning from sequential data.
- **WCAG:** Web Content Accessibility Guidelines; standards for making web content accessible to people with disabilities.

# 8. References

- OpenAI API Documentation – https://platform.openai.com/docs
- Anthropic Claude API Documentation – https://docs.anthropic.com/
- Hugging Face Transformers – https://huggingface.co/docs/transformers
- LangChain Documentation – https://python.langchain.com/
- Pandas Documentation – https://pandas.pydata.org/
- spaCy Documentation – https://spacy.io/usage
- NLTK Documentation – https://www.nltk.org/
- ReportLab User Guide – https://www.reportlab.com/documentation/
- JSON Official Standard – https://www.json.org/json-en.html
- CSV File Format Guide – https://tools.ietf.org/html/rfc4180
- Moodle Developer Docs – https://moodledev.io/docs
- Canvas LMS API Documentation – https://canvas.instructure.com/doc/api/
- Blackboard Developer Portal – https://developer.blackboard.com/
- Deakin University Data Governance and Privacy – https://www.deakin.edu.au/about-deakin/governance-and-leadership/policies
- Family Educational Rights and Privacy Act (FERPA) – https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html
- WCAG 2.1 Accessibility Guidelines – https://www.w3.org/TR/WCAG21/