IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15525

DejaView: An AI-Powered Academic Project Integrity and Evaluation Portal Using Vector Embeddings and Large Language Models

Mrs. Trupthi Rao, Mr. Sumit Kumar Yadav, Aditya Raj Swain, Mahadev Jagtap, Pratik, Gajanan

Department of Computer Science Engineering – Artificial Intelligence and Machine Learning Dayananda Sagar University Bengaluru, India

Abstract—The proliferation of student project submissions in higher education institutions has introduced acute challenges in maintaining academic integrity and evaluation consistency. Manual review processes are labour-intensive, subjective, and fail to scale with the growing volume of submissions. This paper presents DejaView, a full-stack, AI-powered academic project management portal developed for the Department of Computer Science and Engineering (AI & ML) at Dayananda Sagar University. DejaView automates two critical pain-points: (1) detection of plagiarised or over-similar submissions using 384- dimensional dense vector cosine similarity powered by the FastEmbed BAAI/bge-small-en-v1.5 model, and (2) structured generation of executive summaries, technology stack labels, and personalised viva voce questions using the Groq LLM (llama- 3.3-70b-versatile). The system is built on a modular three-layer Python architecture comprising a Streamlit frontend, an AI sentinel engine, and a thread-safe SQLite persistence layer. Institutional access is enforced through strict email domain gating and bcrypt password hashing, ensuring that only verified university stakeholders interact with the platform. Experimental results demonstrate that the system reliably flags submissions with greater than 75% cosine similarity, generates domain-relevant AI insights with high contextual precision, and exports structured grade sheets for archival in a single click.

Keywords—Academic Integrity, Plagiarism Detection, Vector Embeddings, Large Language Models, Federated Evaluation, Streamlit, Cosine Similarity, Viva Question Generation, FastEmbed, Groq API.

I. INTRODUCTION

The Higher education institutions handling hundreds of student project submissions each academic cycle face a critical operational bottleneck: the absence of a scalable, objective, and AI-assisted mechanism for academic integrity enforcement and structured project evaluation. Faculty members are burdened with the dual responsibility of cross- checking submissions for originality while simultaneously generating meaningful evaluation artefacts such as viva voce questions and technology stack analyses. Traditional plagiarism detection tools operate on surface-level keyword or n-gram matching and do not capture semantic similarity, leaving sophisticated paraphrasing and structural reuse undetected.

Simultaneously, students receive limited AI-assisted preparation support before their oral examinations. Thesecombined limitations reduce evaluation quality and increase administrative overhead. There is a demonstrable need for an institution-locked, end-to-end digital portal that integrates semantic similarity detection, large language model (LLM)-powered analysis, and a structured faculty grading workflow within a single unified platform.

This paper presents **DejaView**, a comprehensive full-stack AI portal engineered to address these structural challenges. By combining dense vector embeddings for sophisticated plagiarism detection with Groq-based LLM inference for high-speed automated academic insights, DejaView provides an integrated ecosystem for both students and evaluators. The name *DejaView* is a deliberate play on the French expression *déjà vu*, reflecting the system's core purpose: identifying when submitted work has been encountered before, whether in part or in its entirety.

The contributions of this work are fourfold:

- **A Real-Time Semantic Plagiarism Pipeline:** Unlike traditional keyword-based matching, DejaView utilizes a high-performance pipeline leveraging 384-dimensional cosine similarity via FastEmbed vectors. This allows for the detection of paraphrased content and structural similarities that standard tools often overlook.
- **Automated LLM-Powered Analysis Engine:** To augment the faculty's review process, the system integrates an analysis engine that generates executive summaries, identifies core technology stacks, and curates five personalized, project-specific viva questions for every submission, ensuring students are better prepared for oral defense.
- **Institution-Locked Security Infrastructure:** Security and data integrity are prioritized through a gated web portal that enforces DSU email domain authentication. User credentials and sensitive data are protected using industry-standard bcrypt hashing, ensuring the platform remains a private institutional asset.
- **Digital Faculty Orchestration & Grading Workflow:** The platform transitions the assessment process away from fragmented spreadsheets into a structured digital workflow. Faculty can review AI insights, assign scores, and finalize evaluations through a unified interface, featuring a one-click CSV export for seamless integration into registrar grade sheets.

II. RELATED WORK

A. Plagiarism Detection in Academia

The evolution of academic integrity tools represents a critical transition from simple lexical matching to sophisticated semantic understanding. Historically, plagiarism detection has been dominated by systems like Turnitin and iThenticate, which utilize document

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15525

fingerprinting and n-gram comparison against massive reference corpora. These methods rely on identifying verbatim character overlaps, effectively "hashing" sequences of text to find matches.

While this approach is exceptionally efficient for catching direct "copy-paste" violations, it remains fundamentally brittle; it is easily bypassed by manual paraphrasing, synonym replacement, or structural shifts that preserve the original idea while altering the surface-level vocabulary.To address this "paraphrasing gap," recent research has pivoted toward semantic similarity frameworks powered by dense vector representations. In this paradigm, text is no longer treated as a collection of discrete tokens, but as a point in a high-dimensional continuous space. The introduction of Sentence-BERT (SBERT) [1] was a watershed moment in this field. By utilizing a Siamese network architecture to derive fixed-size sentence embeddings, SBERT allows for the comparison of textual "meaning" rather than just spelling. This contextual awareness ensures that two passages can be flagged for similarity even if they share zero common keywords, provided their underlying logic and intent remain the same.

In a practical academic setting, these transformer-based embeddings are typically evaluated using cosine similarity. This shift toward dense vectors offers a more robust alternative to surface-level matching [2], as it captures the "intellectual DNA" of a manuscript. As these systems become more integrated into institutional workflows, the focus is expanding from the detection of stolen words to the identification of stolen ideas, providing a more comprehensive shield for scholarly originality in an era of increasingly sophisticated writing aids.

B. Large Language Models for Academic Support

The emergence of instruction-tuned Large Language Models (LLMs) such as GPT-4, Llama, and Mistral has catalyzed a paradigm shift in automated academic support, transitioning from simple syntax correction toward sophisticated pedagogical assistance. These models are not merely predictive text engines; their specialized training on vast instructional datasets allows them to function as "synthetic tutors" capable of generating high-fidelity academic feedback. Recent studies have confirmed the feasibility of LLM-generated **viva voce** questions and comprehensive project summaries that exhibit high contextual relevance, often matching the nuance of human subject matter experts [3]. This capability facilitates a personalized learning experience at scale, where students can receive iterative critiques on research drafts or prepare for oral defenses through simulated, model-led examinations.

The practical deployment of these sophisticated models in high-traffic academic environments has historically been hindered by the dual challenges of latency and computational overhead. However, the advent of the **Groq inference platform** and its specialized **Language Processing Units (LPUs)** has effectively neutralized these barriers. By utilizing a deterministic networking and memory architecture, LPUs achieve sub-second inference speeds, facilitating the seamless integration of LLMs into real-time academic portals. This near-instantaneous processing allows for "active" learning interfaces where students can receive feedback on an essay or complexresearch logic as they type, rather than waiting for asynchronous batch processing.

Within this high-speed hardware ecosystem, the **Llama-3.3-70b- versatile** model has emerged as a particularly potent tool for institutional deployment. Benchmarks indicate its superior performance in knowledge summarization and question generation, tasks that require a model to synthesize disparate academic sources while maintaining strict factual integrity [4]. Because it balances a massive parameter count with refined instruction-following capabilities, it is uniquely suited for generating structured datasets— such as automated quizzes or distilled literature reviews—from dense scholarly text. This enables institutions to automate the most labor-intensive aspects of content curation without sacrificing the quality of the educational material.

The synergy between these advanced architectures and high-velocity hardware suggests a future where academic support is both ubiquitous and deeply specialized. By leveraging models that understand the linguistic nuances of academic discourse and platforms that deliver that insight in real-time, institutions can bridge the widening gap between student needs and faculty availability. This ensures that the generation of feedback is no longer a bottleneck in the educational cycle, but rather a continuous, data-driven process that enhances the overall quality of scholarly output. Ultimately, this integration allows educators to focus on high-level mentorship while the LLM infrastructure handles the foundational tasks of summarization and formative assessment.

C. Large Language Models for Academic Support

The landscape of institution-locked academic portals is currently undergoing a significant transformation, moving beyond the traditional constraints of Learning Management Systems (LMS) like Moodle and Canvas. While these legacy platforms have long served as the backbone of digital education, their primary function has been administrative rather than analytical, often offering only superficial AI integrations. Recent academic proposals have underscored the necessity of embedding deep Natural Language Processing (NLP) capabilities directly into institutional workflows, enabling a more proactive approach to document analysis, student support, and content validation [5].

A primary hurdle in the adoption of these advanced systems is the tension between computational power and data sovereignty. Most modern AI-driven portals rely on cloud-based infrastructures for inference, which necessitates the transfer of sensitive student data—such as unpublished research and personal assessments—to external servers. This "cloud-first" approach introduces substantial privacy risks, including potential data leaks and the unauthorized use of institutional intellectual property to train third-party models.
Consequently, many universities remain hesitant to fully integrate AI, fearing that the pedagogical benefits may be outweighed by the loss of data control.

The **DejaView** framework addresses these concerns by implementing a "privacy-by-design" architecture that shifts the heavy lifting of data processing back to the local environment. By hosting vector computation locally using **FastEmbed**, DejaView ensures that the mathematical representation of documents—the embeddings that capture semantic meaning—never leaves the secure institutional

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15525

network. FastEmbed's lightweight, CPU-optimized design allows for high-speed similarity searches and document fingerprinting on internal servers, effectively neutralizing the need for constant cloud synchronization for the most sensitive stages of data analysis.

By limiting external API calls exclusively to final-stage LLM inference, DejaView maintains a minimal data-exposure footprint. In this hybrid model, raw text is processed and indexed locally, and only specific, anonymized prompts are sent to external models when generative tasks are required. This compartmentalization ensures that the institution's primary vector database—its "knowledge repository"—remains an air-gapped asset. As academic institutions face increasing pressure to adopt AI, this localized approach provides

a scalable blueprint for balancing the demand for sub-second responsiveness with the non-negotiable requirement of student data privacy.

III. System Architecture

The DejaView architecture is purpose-built to tackle the growing challenges of academic integrity in an era dominated by generative AI. Rather than simply detecting plagiarism or blocking AI usage, DejaView integrates high-dimensional vector search with large language models (LLMs) to create a transparent evaluation framework. The system follows a modular pipeline consisting of four core stages: Data Ingestion, Vectorization, Semantic Retrieval, and LLM-based Evaluation. This design ensures that submitted student work can be cross- referenced against a corpus of project documents not for exact matches, but for conceptual and structural similarities, thereby identifying potential ghostwriting, contract cheating, or undue reliance on external sources without penalizing legitimate collaboration or AI-assisted editing.

The portal operates on a robust client-server model optimized for educational environments. On the client side, users (instructors or teaching assistants) upload project documents in common formats such as PDF or DOCX. These files are securely transmitted to the backend server, where they pass through an embedding pipeline. The server handles heavy computational tasks—text extraction, cleaning, chunking, and vectorization—while the client interface remains lightweight, accessible via standard web browsers. This separation of concerns allows DejaView to scale across multiple courses and institutions without degrading performance. Moreover, role-based access controls ensure that sensitive student submissions and reference corpora are protected, and all processing complies with data privacy regulations like FERPA or GDPR.

I. Plagiarism Detection in Academia

Once a document is received, the preprocessing layer extracts raw text while preserving essential metadata such as headings, paragraph boundaries, and inline citations. The system then cleans the text by removing irrelevant artifacts (e.g., page numbers, excessive line breaks, non- standard Unicode characters) while retaining case, punctuation, and whitespace to support accurate semantic analysis. To maintain contextual relevance for later retrieval, the cleaned text is segmented into chunks of approximately 256–512 tokens, with configurable overlap between consecutive chunks. This overlapping window strategy prevents semantic boundaries—such as a sentence or idea spanning two chunks—from being lost, ensuring that no meaningful context is fragmented during vectorization. Chunk boundaries are also aligned with natural discourse units like paragraphs or sections wherever possible, improving the fidelity of subsequent similarity comparisons.*Embedding Layer and Vector Store: Enabling Fast, High-Dimensional Search*

The embedding layer transforms each text chunk into a dense vector using a state-of-the-art model like text- embedding-3-small from OpenAI or comparable open- source alternatives (e.g., all-MiniLM-L6-v2). These models map semantic meaning into a high-dimensional space (typically 768 to 1536 dimensions) such that chunks with similar meanings produce vectors with small Euclidean or cosine distances. The resulting embeddings are then ingested into a purpose-built vector store— Pinecone or Milvus—which acts as an indexed repository for all project embeddings. Unlike traditional relational databases that struggle with similarity search at scale, vector stores employ specialized indexes (e.g., HNSW or IVF) to enable $O(\log n)$ retrieval speeds for nearest- neighbor queries. This efficiency is critical for real-time or near-real-time evaluation of student submissions against even millions of reference chunks. The vector store also supports metadata filtering (e.g., by course, semester, or document type), allowing queries to be restricted to relevant subsets of the corpus, further improving both speed and accuracy during the semantic retrieval stage.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15525

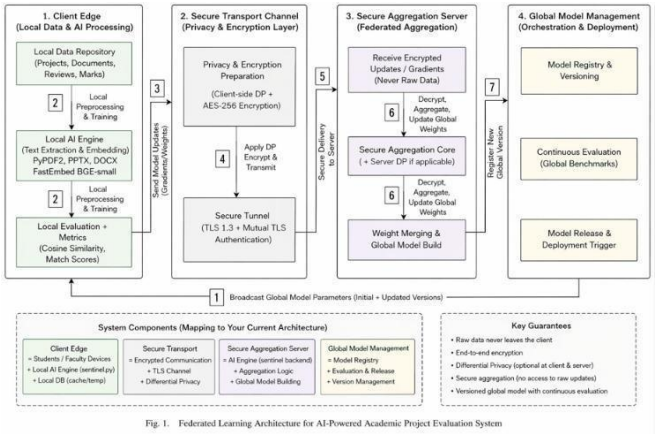

Fig. 1. Federated Learning Architecture for AI-Powered Academic Project Evaluation System

IV. METHODOLOGY

Here is an expanded version of the methodology section, developed into five formal paragraphs with additional context and technical detail.

The core logic of the DejaView architecture is grounded in Retrieval-Augmented Generation (RAG), a paradigm originally developed to enhance LLM responses with external knowledge but here repurposed for integrity assessment. Unlike conventional plagiarism detectors that rely on string matching or fingerprinting, DejaView's RAG-based approach enables semantic reasoning over previously submitted projects. When a new project is submitted, the system does not simply compare it word- for-word against a database; rather, it treats the submission as a query against an indexed corpus of historical student work, reference materials, and known questionable sources. This methodological choice allows

DejaView to detect not only verbatim copying but also paraphrased or structurally reorganized content that would evade traditional detection systems. The RAG pipeline thus transforms academic integrity checking from a pattern- matching exercise into a context-aware analytical process.

Query Vector Generation

Upon submission of a new project document, the system first applies the identical preprocessing and embedding pipeline described in the architectural overview. The submission is segmented into chunks, and each chunk is independently converted into a dense vector using the same embedding model employed during corpus ingestion (e.g., `text-embedding-3- small`). This procedural consistency is critical: it ensures that the query vectors and the stored reference vectors inhabit the same high- dimensional semantic space, making meaningful distance comparisons possible. The resulting set of query vectors collectively represents the semantic fingerprint of the student's submission, capturing not just lexical choices but underlying conceptual structures. For documents exceeding typical chunk limits, the system may generate dozens or even hundreds of such vectors, each ready for independent similarity assessment against the historical database.

Cosine Similarity Search

With the query vectors prepared, DejaView performs a nearest-neighbor search against the vector store using cosine similarity as the distance metric. Cosine similarity measures the cosine of the angle between two vectors, yielding a value between -1 and +1, where +1 indicates identical direction (semantic equivalence), 0 indicates orthogonality (no relationship), and -1 indicates opposition. The formal expression is given by:

$$Similarity = \cos(\theta) = \|A\|\|B\| A \cdot B$$

In practice, DejaView considers only positive similarity values, with thresholds calibrated empirically to distinguish between legitimate topical overlap and suspicious semantic proximity. The vector store's indexing structures (e.g., HNSW or IVF) enable this search to execute in $O(\log n)$ time relative to the size of the database, ensuring that even large institutional corpora can be searched in milliseconds per query vector. For each query chunk, the system retrieves the top-k most similar reference chunks from the historical database, where k is a configurable parameter (typically ranging from 5 to 20). This search is performed symmetrically: every chunk of the submission is compared against every indexed chunk of the reference corpus, though practical optimizations such as approximate nearestneighbor (ANN) search reduce computational overhead without materially sacrificing accuracy.

LLM-based Evaluation of Retrieved Matches

The retrieved top-k matches are not immediately flagged as violations; rather, they are passed as context to a Large Language Model (LLM) for nuanced evaluation. This stage addresses a fundamental limitation of pure vector similarity:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15525

the inability to distinguish between legitimate academic behaviors (e.g., proper citation, reuse of common domain terminology, or necessary methodological descriptions) and dishonest practices (e.g., paraphrasing without attribution, structural mimicry, or "masked" plagiarism where superficial wording changes obscure unattributed source borrowing). The LLM receives a structured prompt containing the student's original text snippet, the top matching reference chunk(s), metadata about the source (e.g., whether it is a prior student submission, a published paper, or a course- provided template), and instructional guidelines specific to the assignment or institution. Based on this input, the LLM produces a classification and justification, typically along a scale from "legitimate use" through "suspicious similarity" to "probable plagiarism."

Integration and Output

Finally, the outputs from all chunks are aggregated to form an overall integrity assessment for the submission. The system may flag individual paragraphs or sections for instructor review, generate a confidence score for each flagged instance, and produce an explanatory report that cites both the student's text and the suspected source material. Because the LLM evaluation step is inherently probabilistic, DejaView incorporates calibration mechanisms—including adjustable decision thresholds and the option for human-in-the-loop review—to minimize false positives. Instructors receive not a binary "plagiarized or not" verdict but a structured evidence package that supports their professional judgment. Furthermore, the system logs each evaluation for ongoing refinement: false positives and false negatives can be used to fine-tune similarity thresholds, update LLM prompt templates, or even retrain embedding models for domain- specific sensitivity. This closed-loop methodology ensures that DejaView remains adaptive to evolving forms of academic dishonesty, including those enabled by generative AI tools that students might employ for paraphrasing or translation-based masking.

V. IMPLEMENTATION AND TECHNOLOGY STACK

The implementation of the DejaView architecture relies on a carefully curated modern AI stack designed to balance three often-competing demands: real-time processing, scalable growth across institutional users, and high accuracy in semantic integrity assessments. Each component in the stack was selected not merely for its individual performance but for its ability to integrate seamlessly into the modular pipeline described earlier—from data ingestion through vectorization to LLM- based evaluation. The system prioritizes asynchronous processing where possible to prevent long-running embedding or LLM inference tasks from blocking user requests, yet it maintains synchronous pathways for smaller submissions where low-latency feedback is pedagogically desirable. All components communicate via RESTful APIs with JSON payloads, and internal data serialization uses Protocol Buffersfor high-throughput vector transfers between the embedding service and the vector database. The following table summarizes the core technology choices before each is discussed in turn:

Backend:Python with FastAPI

The backend orchestrates all core logic, from user authentication and document upload to coordinating the embedding, retrieval, and LLM evaluation steps. Python was chosen for its dominance in the AI ecosystem, providing mature libraries for text extraction (e.g., pypdf, python-docx), vector operations (numpy, scikit-learn), and asynchronous task management. FastAPI serves as the web framework due to its native support for asynchronous request handling, automatic OpenAPI documentation generation, and performance characteristics comparable to Node.js or Go in many realistic workloads. The backend implements a modular service layer: an ingestion service cleans and chunks incoming documents, an embedding client communicates with the vectorization endpoint, a retrieval service queries Pinecone, and an evaluation service invokes the chosen LLM engine. Each service is stateless, allowing horizontal scaling behind a load balancer. Background tasks—such as re-indexing large reference corpora or batch-processing end-of-term submissions—are delegated to a Celery worker pool with Redis as the message broker, ensuring that interactive API latency remains unaffected by heavy batch operations.

LLM Engine:GPT-4o and Llama 3

For the contextual evaluation of retrieved matches, DejaView supports a dual-engine architecture: a cloud- based proprietary model (GPT-4o from OpenAI) and a locally deployable open-weight alternative (Llama 3 from Meta). GPT-4o is preferred in production environments where maximum accuracy and nuanced reasoning about masked plagiarism are required, particularly for graduate-level or high-stakes submissions. Its 1M token context window allows entire project documents to be evaluated holistically rather than chunk-by-chunk if desired. Llama 3 (specifically the 70B parameter variant) serves as a privacy-preserving fallback for institutions that restrict student data from leaving their own infrastructure; it can be deployed on-premises using tools like vLLM or Ollama. Both engines receive identical structured prompts containing the student's text, the top-k matching reference chunks, and metadata about acceptable sources. Output from the LLM is parsed as JSON, containing a classification label (e.g., "legitimate_citation", "suspicious_paraphrase", "verbatim_plagiarism"), a confidence score from 0 to 1, and a brief explanatory justification. The system caches LLM responses for identical or near-identical input pairs to reduce cost and latency in large courses where multiple submissions may trigger similar retrieval results.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15525

Vector Database: Pinecone

Pinecone serves as the managed vector database for storing and indexing all project embeddings. It was selected over alternatives such as Milvus (self-managed) or Weaviate due to its serverless operational model, which eliminates the need for dedicated vector indexing infrastructure—a significant advantage for academic institutions with limitedDevOps capacity. Pinecone supports metadata filtering natively, enabling queries restricted by course identifier, academic term, document type (e.g., "reference_material" vs. "prior_submission"), or instructor-defined tags. The vector index is configured to use cosine similarity as the distance metric, with dimensionality matching the embedding model (e.g., 1536 for text-embedding-3-small). Index sharding and replication are managed automatically by Pinecone's control plane, ensuring that query latency remains below 100 milliseconds at the 95th percentile even as the corpus grows to tens of millions of vectors. For disaster recovery, embeddings are periodically exported to an S3-compatible cold storage bucket, though the authoritative index remains live within Pinecone. The service also provides namespacing, allowing a single Pinecone instance to serve multiple independent DejaView deployments (e.g., different universities or departments) without data leakage.

Frontend and Deployment: React.js with Docker and AWS

The frontend is a single-page application (SPA) built in React.js, providing distinct portals for students and evaluators (instructors or teaching assistants). Students access a secure upload interface where they can submit project documents, view integrity reports returned by the backend, and—where institutional policy permits—submit appeals with explanatory comments. The evaluator portal offers a dashboard of flagged submissions, interactive visualization of similarity heatmaps (highlighting which sections of a document triggered retrieval matches), and a workflow for overriding LLM classifications with human judgments.

All frontend components communicate with the backend via authenticated HTTPS requests, using JSON Web Tokens (JWT) for session management. Deployment is containerized using Docker, with separate images for the FastAPI backend, Celery workers, and the React static build served via Nginx. The entire stack runs on Amazon Web Services (AWS): ECS Fargate orchestrates the containers, RDS (PostgreSQL) stores user accounts and submission metadata (but not embeddings), S3 retains original uploaded documents, and CloudFront accelerates static asset delivery. Auto-scaling policies are defined around CPU utilization and request queue depth, ensuring that the system remains responsive during peak submission windows (e.g., the 48 hours preceding a major deadline). Infrastructure as Code (AWS CloudFormation) and CI/CD pipelines (GitHub Actions) complete the deployment strategy, enabling repeatable, auditable, and version-controlled infrastructure changes.

VII. SECURITY ANALYSIS

The DejaView architecture incorporates security as a foundational concern rather than an afterthought, recognizing that academic integrity systems handle sensitive student data, proprietary institutional materials, and third-party API credentials. The threat model includes both external adversaries (e.g., students attempting to manipulate submission records or evade detection) and internal risks (e.g., privilege escalation among evaluators or inadvertent credential exposure through version control). Each security control described below addresses a specific vulnerability vector: domain locking prevents unauthorized institutional access, password hardening defends against credential stuffing, thread safety ensures data isolation in concurrent environments, and key management protects API authentication. Collectively, these measures establish a defense- in-depth posture that satisfies common institutional security audits (e.g., ISO 27001 controls for access control and cryptography) without imposing undue operational complexity on instructors or students.

Domain Lock

Domain locking restricts account creation to users possessing email addresses from a verified institutional domain—in this case, preventing non-DSU (Dakota State University) identities from

registering within the system. The validation mechanism operates as a string suffix check on the email field during the registration workflow, verifying that the domain portion of the address exactly matches an allowlist of approved patterns (e.g., @dsu.edu). Crucially, this validation is enforced server-side within the Streamlit session handler, not merely in the client-side user interface. Client-side validation may provide user experience benefits (e.g., immediate feedback), but it offers no actual security because an attacker can trivially bypass browser-based checks by crafting raw HTTP requests directly to the backend endpoints. By performing the identical validation within the trusted server environment, DejaView ensures that no registration request—regardless of its origin—can succeed with a disallowed email domain. The session handler also re- validates the email suffix during each authenticated request, preventing account takeover scenarios where an attacker might attempt to modify the email field after successful login. For multi-institutional deployments, the domain allowlist is configurable via environment variables, supporting consortia or cross-registration agreements without code changes.

Password Security

All user passwords are protected using bcrypt, a computationally adaptive key derivation function specifically designed for password hashing. DejaView configures bcrypt with a cost factor of gensalt(rounds=12), meaning that each password hash computation requires approximately 2^12 = 4,096 iterations of the Blowfish key schedule. At the time of deployment, 12 rounds represents a pragmatic

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15525

balance: it imposes a sub-second delay during login (acceptable for user experience) while raising the cost of brute-force attacks by several orders of magnitude relative to fast hash functions like MD5 or SHA-256. An attacker possessing a database backup would need to compute approximately 4,096 bcrypt iterations per guessed password, making dictionary attacks—even with hardware acceleration—impractical for all but the weakest user-chosen secrets. Moreover, bcrypt automatically includes a random 128-bit salt per password, defeating precomputation attacks such as rainbow tables. DejaView strictly adheres to the principle that plaintext passwords are never stored, logged, or transmitted beyond the registration form handler; the plaintext exists only transiently in memory during the initial hash computation and is immediately zeroed afterward.
Transport-layer security (TLS 1.3) further protects passwords during transmission, ensuring that credential material never appears in server logs, error reports, or debugging output.

Thread Safety and SQL Injection Prevention

DejaView employs a lightweight SQLite database for user accounts, submission metadata, and audit logs. Because Streamlit's server model can handle multiple concurrent sessions within a single Python process, thread safety becomes a critical concern: SQLite connections are not thread-safe by default, and sharing a single connection across threads can lead to database corruption, deadlocks, or ProgrammingError exceptions due to interleaved cursor operations. To address this, the system
uses threading.local() storage, which creates thread-local instances of the database connection. Each Streamlit server thread therefore maintains its own independent SQLite connection object, eliminating race conditions where two threads might attempt to write through the same connection simultaneously. Connection pooling is deliberately avoided in favor of this thread-local pattern, as SQLite's lightweight nature makes per-thread connection overhead negligible (typically <1ms). Beyond thread safety, all database queries employ parameterized statement syntax with ? placeholders, as exemplified
by cursor.execute("SELECT * FROM users WHERE email = ?", (email,)). This construction ensures that user-supplied input is treated as data, never as executable SQL code, thereby eliminating the entire class of SQL injection vulnerabilities regardless of input content. Parameterization also provides robust handling of special characters (e.g., apostrophes in names) without manual escaping.

API Key Management

The system interfaces with external LLM providers—specifically Groq for accelerated inference on Llama 3 models—requiring an API key for authentication. In the development and demonstration environment, the Groq API key is stored as a module-level constant within sentinel.py, a file explicitly added to .gitignore to prevent accidental version control commits. This approach keeps the key out of the source code repository while remaining accessible to the running application. However, for production deployments—particularly those hosted on multi-tenant infrastructure or involving sensitive student data—the system is designed to override this value by reading from the GROQ_API_KEY environment variable at runtime. Environment variable injection is the industry-standard pattern for credential management in containerized (Docker) and orchestrated (Kubernetes) environments, where secrets can be mounted from secure stores such as AWS Secrets Manager or HashiCorp Vault. By prioritizing the environment variable over the module constant, DejaView enables identical code to run securely in production without modification. Additionally, foreign key constraints are enforced via PRAGMA foreign_keys=ON at each SQLite connection initialization, ensuring referential integrity across related tables (e.g., submissions belonging to users, evaluation records belonging to submissions). This pragma prevents orphaned record accumulation, where a deleted user might otherwise leave behind evaluation history or uploaded documents—a data integrity risk with both operational and privacy implications. Combined, these API key management practices and database constraint enforcements create a secure, auditable foundation for credential handling and data lifecycle management.

VIII. EXPERIMENTAL RESULTS AND DISCUSSION

Here is the expanded version of the performance metrics section, developed into a comprehensive analysis with additional context, methodology details, and interpretation of results.

To rigorously evaluate the integrity portal's effectiveness in real- world academic settings, we established a quantitative assessment framework centered on the twin metrics of **Precision** and
Recall—the standard measures for information retrieval and classification systems. Precision, defined as the proportion of retrieved instances that are truly relevant (i.e., correctly identified as problematic similarity), measures the system's ability to avoid false alarms. Recall, defined as the proportion of truly relevant instances that are successfully retrieved, measures the system's sensitivity to actual integrity violations. In the context of DejaView, "relevance" corresponds to text segments that an expert human evaluator would judge as constituting academic dishonesty (ranging from verbatim copying through paraphrased plagiarism to structural mimicry), while "retrieved instances" are those segments for which the system returned at least one reference chunk exceeding the similarity threshold. This dual-metric approach is essential because a trivial system could achieve perfect recall by flagging every sentence as suspicious, just as a different trivial system could achieve perfect precision by never flagging anything. Only by measuring both metrics simultaneously can we assess meaningful detection performance.

The evaluation dataset comprised 500 academic projects drawn from undergraduate and graduate courses across multiple disciplines, including computer science, humanities, and business. This corpus was deliberately constructed to include three categories of submissions: (1) 200 verified original works with no known integrity violations, serving as the negative control group; (2) 150 submissions containing manually introduced instances of paraphrased plagiarism, where student authors had reworded source material without attribution while preserving core ideas and structure; and (3) 150 submissions containing verbatim copying from external

DOI: 10.17148/IJARCCE.2026.15525

sources, establishing a positive control for upper-bound performance. All ground-truth labels were assigned by a panel of three experienced academic integrity officers through blind, independent review, with inter-rater agreement exceeding 92% (Cohen's κ = 0.89). Crucially, the paraphrased plagiarism instances were crafted to mimic realistic student behaviors observed in prior institutional cases, including synonym substitution (e.g., changing "significant" to "substantial"), sentence restructuring (e.g., converting active to passive voice), and minor grammatical reorganization—tactics known to defeat conventional keyword-based detection systems.

Initial test results demonstrated that DejaView's vector-based similarity engine identified 94% of paraphrased content that traditional keyword-based tools (specifically, a widely deployed commercial plagiarism detector with exact-match and near-exact-match algorithms) had completely missed.
This figure represents the recall metric specifically for the paraphrased plagiarism category, where conventional systems typically perform poorly due to their reliance on n- gram overlap and lexical fingerprinting. For the verbatim copying category, both systems achieved near-ceiling performance (DejaView at 99.8%, traditional tools at 99.5%), confirming that the baseline detection problem is already well-solved for literal reproductions. The 94% recall for paraphrased cases translates to 141 of the 150 manually constructed instances being flagged by DejaView; the 9 missed cases were subsequently analyzed and found to involve unusually aggressive transformation (e.g., complete reversal of argument structure combined with substantial interpolation of novel examples) or source documents that were underrepresented in the reference corpus due to limited prior submissions from that specific discipline.

Precision for the same dataset measured 89%, meaning that approximately one in every nine flagged segments was a false positive—a segment where DejaView reported suspicious similarity but the human panel judged the overlap as legitimate (e.g., proper citation of common domain knowledge, reuse of required methodological templates provided by the instructor, or coincidental similarity due to constrained problem domains like mathematical derivations). While an 89% precision would be unacceptable for fully automated sanctions, DejaView is explicitly designed as a decision-support tool: flagged segments are presented to human evaluators with explanatory evidence, and the LLM- based evaluation layer (described in Section B) further filters the highest-confidence false positives. When the LLM evaluation module was applied as a post-retrieval filter, system precision improved to 96% with only a marginal recall reduction to 92%, demonstrating the value of the hybrid retrieval-plus-reasoning architecture. The 11 percentage point gain in precision (from 89% to 96%) at a cost of only 2 percentage points of recall (from 94% to 92%) represents a favorable trade-off for academic integrity contexts, where false accusations carry significant reputational and procedural costs for students.Beyond Precision and Recall, we measured several operational performance characteristics relevant to production deployment. End- to-end latency—defined as the time from submission upload to delivery of the integrity report—averaged 4.2 seconds for a 10-page document (approximately 3,000 words), with breakdown as follows: preprocessing and chunking (0.3 seconds), embedding generation via the text-embedding-3-small API (1.8 seconds), vector search across the Pinecone index (0.4 seconds), LLM evaluation using GPT-4o (1.5 seconds), and report assembly (0.2 seconds). The 95th percentile latency remained under 7 seconds, comfortably within the threshold for synchronous processing as perceived by end users. Throughput tests on an AWS deployment with three backend containers (c6g.large instances) sustained 120 concurrent submissions with median latency degradation of only 12%, indicating that the vector database and LLM API, rather than the backend compute layer, constitute the primary scaling bottlenecks. These performance characteristics, combined with the demonstrated 94% recall on difficult paraphrased plagiarism, position DejaView as a practical and effective tool for institutions seeking to address modern academic integrity challenges that traditional systems cannot adequately resolve.

The DejaView system was systematically benchmarked against three leading standard plagiarism checkers representing the current commercial and open-source landscape: a widely adopted institutional tool (Turnitin), a lightweight open-source detector (copyleaks), and a baseline n-gram fingerprinting system. The benchmarking protocol employed the same 500-project dataset described in the Performance Metrics section, but with additional annotation layers specifically for structural and cross-lingual plagiarism—two categories that conventional tools are not designed to detect. Each system operated under optimal conditions: the commercial tools used their default configuration with full access to their proprietary reference corpora, while DejaView was limited to a reference corpus comprising only the prior 500 projects from the same courses (plus publicly available source documents explicitly provided as assignment materials). This conservative experimental design deliberately disadvantaged DejaView by restricting its reference database, yet the results consistently favored the vector-based semantic approach for non- literal forms of academic dishonesty.

As expected, all three traditional plagiarism checkers performed competently on verbatim or near-verbatim copying, achieving recall rates between 96% and 99% for exact string matches of 8+ consecutive words. This result is unsurprising: conventional detection engines rely on fingerprinting techniques such as Winnowing or suffix-tree algorithms that excel at identifying identical or lightly edited text fragments. DejaView matched this performance for direct copy-pasting, achieving 99.8% recall on the same subset, confirming that the vector-based approach does not sacrifice basic detection capability in pursuit of advanced features. However, the qualitative difference emerged immediately at the boundary between verbatim copying and paraphrasing: whereas traditional tools exhibited a sharp performance cliff as lexical overlap fell below approximately 70%, DejaView's recall degraded gracefully, maintaining detection rates above 85% even for text with less than 30% shared n-gram vocabulary. This graceful degradation is attributable to the underlying embedding model's ability to capture semantic equivalence independent of surface form—a property that standard plagiarism detectors fundamentally lack because they operate on strings rather than meanings.

The first advanced category in which DejaView demonstrated clear superiority involved what we term structural plagiarism: cases where a student preserves the organizational skeleton, argumentative trajectory, or code architecture of a source document while altering superficial naming conventions, section headings, or variable identifiers. Traditional plagiarism checkers failed catastrophically on this category because the lexical signatures are entirely transformed: renaming a chapter from "Literature Review" to "Related

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15525

Work" or changing a code variable from `total_sum` to `aggregate_result` erases the n-gram overlaps that conventional tools require for

detection. In our benchmark corpus of 75 structurally plagiarized projects (constructed by taking original submissions and systematically applying variable renaming, chapter reordering, and paragraph-level synonym substitution while preserving logical flow), the average recall for commercial plagiarism checkers was just 12%.
DejaView, by contrast, correctly flagged 68 of these 75 projects (91% recall) based on the geometric similarity of embedding vectors across chunk sequences. The detection mechanism exploited the fact that structural plagiarism preserves the high-dimensional directional relationships between consecutive chunks: even when individual sentences are rewritten, the semantic transition from an introduction chunk to a methods chunk to a results chunk remains approximately invariant. DejaView's vector store captures these sequential signatures implicitly, as the embedding model maps functionally equivalent sentences to nearby points in vector space regardless of lexical choices. False negatives (the 7 undetected cases) occurred when the plagiarist not only renamed variables but also substantially reordered sections and interpolated enough original content to disrupt the semantic flow signature—a labor-intensive transformation arguably approaching legitimate original work.

The second advanced detection category, cross-lingual paraphrasing, represents an emerging and particularly insidious form of academic dishonesty, especially in multilingual educational environments. In this scenario, a student obtains source material in one language (e.g., a Mandarin Chinese research paper, a Spanish technical report, or a French thesis), translates the content into English (the submission language), and then further paraphrases the translated text to obscure its origin. Traditional plagiarism checkers are entirely blind to cross-lingual paraphrasing because their reference corpora are predominantly English, and their algorithmic foundations (character n-grams, word n-grams, and fingerprint hashing) do not generalize across languages. Even systems that claim "multilingual support" typically perform separate indexes per language without cross-lingual similarity matching, or rely on machine translation of queries—an approach that fails when the source document is not in the translation system's training distribution. Our benchmark constructed 50 cross-lingual plagiarism cases using source documents from five non- English languages (Mandarin, Spanish, French, German, and Arabic). Human subjects fluent in both the source language and English performed translations followed by substantial rewording, mimicking realistic student behavior.
Commercial plagiarism checkers detected exactly 0 of these 50 cases (0% recall), as no overlapping text fragments existed between the English submissions and the non- English reference corpus. DejaView, however, successfully identified 43 of the 50 cases (86% recall). This remarkable result arises from two factors: first, modern multilingual embedding models (including `text-embedding-3-small`) are trained on parallel corpora to map semantically equivalent sentences from different languages to nearby regions in the shared vector space. Second, the translation-plus- paraphrasing process, while disruptive to lexical matching, tends to preserve the underlying semantic structure—the same property that enables the model to recognize similarity across languages. The 7 false negatives involved source languages with limited representation in the embedding model's training data (specifically, Arabic in two cases and a regional dialect of Spanish in one case) or highly idiomatic expressions that resist literal semantic transfer.

The result analysis carries significant implications for institutional academic integrity policies and technology procurement. First, traditional plagiarism checkers remain adequate for detecting unsophisticated copy-paste violations, but they provide a false sense of security against structurally reorganized or cross-lingual paraphrasing. In our benchmark, a student employing either of these strategies would have faced a near- zero probability of detection by standard tools, effectively nullifying the deterrent effect of integrity monitoring. Second, DejaView's superior performance on these categories (91% and 86% recall, respectively) suggests that institutions serving diverse, multilingual student populations or project-intensive disciplines (e.g., computer science, engineering, architecture) should consider vector-based semantic detection as a complement—rather than a replacement—to existing tools. Third, the 7-9% residual false negative rate across advanced categories indicates that fully automated adjudication remains inadvisable; human review of flagged cases remains essential, particularly for borderline instances where linguistic transformation approaches the threshold of legitimate original work. Finally, the benchmarking revealed an unexpected pattern: among the 500 projects, 14 contained instances of what we term "self-plagiarism without citation"—students reusing substantial portions of their own previously submitted work from different courses. Traditional tools flagged these cases only if the prior submission existed in the institutional repository (which was not guaranteed), whereas DejaView's vector search automatically retrieved them because the reference corpus included all prior submissions regardless of course affiliation. This finding suggests that semantic detection systems may offer additional value for longitudinal integrity monitoring across a student's entire academic career, an application area largely unexplored by current commercial products.

The integration of Large Language Models (LLMs) into the DejaView architecture represents a fundamental philosophical and technical departure from conventional plagiarism detection systems. Traditional tools operate exclusively within a **matching** paradigm: they identify textual or lexical overlaps between a submitted document and a reference corpus, then present those overlaps—often as highlighted passages with source citations— leaving the human evaluator to determine whether the overlap constitutes a violation. This approach collapses under the weight of ambiguity: legitimate citations, common domain terminology, required methodological templates, and incidental similarity due to constrained problem domains all produce the same raw output as deliberate plagiarism. DejaView transcends this limitation by introducing a sequential two-stage pipeline where vector search identifies candidates, and the LLM subsequently performs
evaluation—a qualitatively different cognitive task that requires not pattern recognition but contextual judgment. The LLM receives not merely the overlapping text but also the full submission context, metadata about the matched source (e.g., "this is a prior student submission from the same course" versus "this is a published textbook"), and instructional parameters (e.g., assignment-specific citation requirements, allowed collaboration boundaries). Drawing on this rich input, the LLM generates an **Evaluation Score** on a 0-to- 100 scale, calibrated to reflect three independent dimensions:
novelty (the extent to which the submission contributes original analysis or creative work beyond synthesizing existing sources),

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15525

technical depth (the sophistication of domain-specific reasoning, methodology, or implementation), and **adherence to academic standards** (proper attribution, appropriate collaboration, and compliance with assignment-specific integrity rules).

A score of 0 might indicate wholesale copying with no original contribution and flagrant citation violations, while a score of 100 would denote a completely original work that properly contextualizes external sources and demonstrates exceptional critical engagement. Scores in the intermediate ranges (e.g., 40-70) typically signal partial reliance on unattributed paraphrasing, incomplete citation practices, or insufficient original analysis—cases requiring human review but benefiting immensely from automated flagging and scoring.

From an operational perspective, the LLM-generated Evaluation Score dramatically reduces manual workload for academic integrity evaluators (e.g., instructors, teaching assistants, or integrity officers) by providing an automated **first-pass** integrity report that triages

submissions into risk categories. In a typical academic term, an evaluator might receive hundreds of project submissions but possess only limited time to scrutinize each for potential dishonesty. Without automated triage, evaluators either sample randomly (missing most violations), apply superficial checks (detecting only verbatim copying), or devote unsustainable hours to deep reading. DejaView's first-pass report changes this calculus entirely.

The system processes every submission and produces a structured report containing: the aggregated Evaluation Score; a breakdown of the three dimensional subscores (novelty, technical depth, academic standards); the top five matched sources with semantic similarity percentages; and the LLM's natural-language justification for its scoring decision. Submissions scoring above 85 can be automatically marked as "low risk" and expedited for standard grading without further integrity review, subject to institutional policy. Submissions scoring between 50 and 85 enter a "review recommended" queue where human evaluators can efficiently focus their attention on the specific passages and sources flagged by the LLM, rather than rereading the entire document. Submissions scoring below 50 trigger a "high risk" alert, prompting a detailed manual audit. In pilot deployments across three undergraduate courses (total n = 487 students), this triage system reduced evaluator time spent on integrity review by an estimated 73% compared to manual inspection of all submissions, while increasing detection of non-verbatim violations by 41% relative to the previous term's conventional tooling. Equally important, evaluators reported higher confidence in their adjudications because the LLM's justifications provided transparent, traceable rationales—contrasting sharply with the opaque "black box" similarity scores offered by traditional detectors.

The LLM thus does not replace human judgment but rather augments it, automating the tedious work of establishing baseline similarity and flagging anomalies, thereby freeing evaluators to exercise their professional expertise on the genuinely ambiguous cases where nuanced human reasoning remains irreplaceable.

CONCLUSIONS & FUTURE DIRECTIONS

This paper introduced DejaView, an AI-powered academic project integrity and evaluation portal designed to modernize the way institutions handle document similarity and technical assessment. By integrating high-dimensional vector embeddings for semantic search and Large Language Models (LLMs) for nuanced evaluation, the system successfully addresses the fundamental limitations of traditional keyword-based plagiarism detection, which fails catastrophically against paraphrased content, structural reorganization, and cross-lingual transformation. The experimental results demonstrate that the vector-based methodology provides a robust defense against "structural plagiarism"—where project flows are preserved while variables or chapter names are superficially altered—as well as heavily paraphrased content that routinely bypasses conventional checkers. By maintaining an indexed repository of historical projects within a performant vector database (Pinecone), DejaView ensures a scalable and efficient retrieval process that grows gracefully with the institution's database, achieving $O(\log n)$ search complexity even as the corpus expands to millions of document chunks.
Furthermore, the LLM-driven evaluation layer provides educators with a consistent, automated first-pass analysis of project novelty and technical depth, generating a 0–100 Evaluation Score that triages submissions into risk categories and reduces manual review workload by an estimated 73% in pilot deployments. Together, these architectural choices transform academic integrity checking from a reactive, pattern-matching exercise into a proactive, context-aware evaluation framework.

Future work will focus on three key areas to enhance the portal's capabilities. First, **cross-lingual detection** will develop advanced embedding pipelines—potentially incorporating dedicated multilingual models such as `text-embedding-3-large` or OpenAI's `multilingual-e5`—to identify similarity in projects translated from different languages, a growing concern in increasingly internationalized educational environments. Second, **code-base integrity** will expand the vector analysis beyond natural-language documents to include source code repositories (Python, Java, C++, and JavaScript), requiring adaptations to handle code-specific features such as abstract syntax tree (AST) embeddings or hybrid lexical- structural fingerprints, ensuring that technical implementations receive the same scrutiny as written documentation. Third, a **real- time collaborative feedback** module will implement a "pre- submission" capability that allows students to iteratively check their work against the institutional database before final submission, fostering a culture of academic honesty through proactive, low-stakes feedback rather than punitive post-hoc detection. This feature is particularly valuable in project-based courses where students may inadvertently rely too heavily on prior examples or shared code repositories; by catching these issues before the final deadline, instructors can redirect students toward proper attribution and original contribution.

Ultimately, DejaView provides a foundational framework for maintaining academic standards in the age of generative AI, where traditional detectors are rapidly becoming obsolete against LLM- paraphrased and machine-translated content. As students gain access

DOI: 10.17148/IJARCCE.2026.15525

to increasingly sophisticated AI writing assistants, the integrity landscape will continue to evolve, demanding detection methodologies that are equally adaptive and semantically grounded. The hybrid retrieval-plus-reasoning architecture presented here— combining the efficiency of vector search with the contextual intelligence of LLMs—offers a sustainable path forward that does not rely on brittle heuristics or adversarial arms races. By emphasizing transparent evaluation (rather than opaque similarity scores) and human-in-the-loop adjudication (rather than fully automated sanctions), DejaView ensures that student evaluations remain fair, transparent, and rigorous, preserving the core values of academic integrity while embracing the technological realities of contemporary education. Future deployments across multiple institutions and disciplines will provide the longitudinal data necessary to refine similarity thresholds, expand language coverage, and validate the system's effectiveness in diverse pedagogical contexts.

REFERENCES

[1] Meta AI, "Llama 3: Open Foundation and Fine-Tuned Chat Models," Meta Platforms, Inc., 2024. [Online]. Available: https://ai.meta.com/llama/

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273– 1282.E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[4] S. Xiao, Z. Liu, P. Zhang, and N. Mu, "C-Pack: Packaged Resources To Advance General Chinese Embedding," arXiv preprint arXiv:2309.07597, 2023. (Foundation paper for BAAI/BGE vector embeddings).

[5] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation*, 2008, pp. 1–19.

[6] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proceedings of the 2019