WILDFLOW: A MULTIMODAL RETRIEVAL-AUGMENTED ACADEMIC INTELLIGENCE SYSTEM

Rohan Deogaonkar
MIT World Peace University
Pune, India
rohandeogaonkar.9@gmail.com

April 4, 2026

ABSTRACT

Wildflow is a structured multimodal Retrieval-Augmented Generation (RAG) framework designed to enforce strict academic knowledge boundaries in question-answering systems. Conventional RAG architectures reduce hallucination by retrieving relevant documents but still allow large language models to blend retrieved information with internal pretrained knowledge. Wildflow introduces an architectural control layer that prioritizes structured Markdown files as the authoritative source of truth, enforces unit-level vector database isolation, and restricts LLM usage to classification, verification, and bounded reasoning. This approach transforms retrieval from a probabilistic similarity process into a verification-driven academic response pipeline optimized for structured educational environments.

1 Introduction

Large language models (LLMs) exhibit strong reasoning capabilities but remain unreliable in structured academic environments that demand equation precision, hierarchical topic alignment, and diagram consistency. Even when Retrieval-Augmented Generation (RAG) is used to ground responses in external documents, conventional systems still allow blending between retrieved content and internal pretrained knowledge. This can lead to topic drift, partial formula retrieval, and weak multimodal alignment.

In curriculum-bound academic settings, such probabilistic behavior is insufficient. Institutional materials require deterministic containment and structural fidelity.

Wildflow introduces a verification-driven, multimodal RAG architecture designed specifically for structured academic subjects. The system treats structured Markdown files as the authoritative source of truth, enforces unit-level vector isolation, and restricts LLMs to controlled roles such as query classification, logical validation, and bounded numerical reasoning.

Rather than relying solely on prompt-based hallucination control, Wildflow embeds knowledge boundaries directly into its architecture through deterministic routing, structured metadata filtering, and multi-stage verification.

The result is a curriculum-aligned academic intelligence framework optimized for precision, containment, and structural reliability.

2 Related Work

Retrieval-Augmented Generation (RAG) is a common method used to improve the factual accuracy of large language models. In a typical RAG system, the model retrieves text chunks from a vector database based on semantic similarity. These retrieved chunks are then given to the LLM, which generates an answer by combining the retrieved content with its own pre-trained knowledge.

Although this approach improves accuracy compared to using an LLM alone, it still has several important limitations.

No Proper Equation Validation: In most RAG systems, mathematical equations are treated like normal text. There is usually no system-level check to confirm whether an equation is complete, correctly formatted, or logically valid. This can lead to missing symbols or partially retrieved formulas.

Weak Image-Topic Connection: Many multimodal RAG systems embed text and images together using vision-language models. However, they often do not maintain a strict connection between an image and its exact academic topic. As a result, an image may be relevant in meaning but not correctly linked to the specific section it belongs to.

Prompt-Based Hallucination Control: Most systems try to reduce hallucinations by giving instructions in the prompt, such as telling the model to use only retrieved

content. However, this does not fully prevent the model from mixing retrieved data with its own internal knowledge. The control remains soft and not enforced by system design.

Limited Awareness of Academic Structure: Many RAG systems do not preserve document hierarchy such as lecture titles, topic headers, and subpoints. Without this structure, retrieval can sometimes mix information from different units or topics.

Wildflow takes a different approach. It embeds only cleaned text while keeping equations (in KaTeX format) and image references separate. During retrieval, these equations and images are checked again against the original Markdown file.

Instead of relying mainly on prompts, Wildflow enforces knowledge boundaries at the architectural level. This makes it more controlled and better suited for structured academic content compared to traditional RAG systems.

3 System Design Philosophy

Wildflow is built on four architectural principles that regulate how knowledge flows through the system. These principles are designed to enforce academic containment and structural reliability.

3.1 Markdown as the Authoritative Source

The structured Markdown file generated from institutional PDFs is treated as the definitive academic reference. All responses must align strictly with this source. The LLM is not permitted to override, reinterpret, or extend beyond the Markdown content unless a controlled fallback condition is triggered.

This establishes a clear separation between institutional material and probabilistic model knowledge.

3.2 LLM as Verifier, Not Knowledge Author

The LLM does not function as the primary source of knowledge. Rather than generating answers freely from its pretrained parameters, it operates within clearly bounded and tightly controlled roles. Its responsibility is limited to classifying incoming queries, extracting the relevant academic topic, logically verifying retrieved content against structured sources, and performing step-by-step numerical problem solving when required. This constrained design ensures that the system remains grounded in the structured vector database and authoritative markdown sources, while still leveraging the reasoning strength of the model where it adds measurable value.

3.3 Deterministic Topic Routing

Wildflow does not perform global similarity search across all indexed data. Instead, subject and unit identification occur before retrieval. Queries are routed through structured JSON mappings that restrict search scope to a specific academic unit.

This deterministic routing prevents topic drift and reduces cross-unit contamination.

3.4 Unit-Level Knowledge Isolation

Each academic unit maintains:

A dedicated Markdown file

A unit-specific vector database

Independent metadata

This isolation ensures that semantically similar topics across different units cannot interfere during retrieval. Containment is enforced structurally rather than through post-hoc correction.

4 High-Level Architecture

Wildflow follows a structured processing pipeline:

PDF Input
↓
Surya OCR + Structured Parsing
↓
Markdown (Text + KaTeX + Image References)
↓
Embedding Pipeline (Text Only)
↓
Unit-wise Qdrant Vector Database
↓
Query → Topic Detection (Gemini)
↓
Targeted Unit Retrieval
↓
Markdown Cross-Verification
↓
LLM Final Verification
↓
Verified Output

5 Data Engineering Pipeline

5.1 PDF Extraction

Wildflow begins by processing raw academic PDFs using the Surya OCR model. The objective extends beyond simple text recognition; the system performs structured academic reconstruction. From each PDF, it extracts structured textual content, images and diagrams, and mathematical equations. Unlike conventional OCR systems that

flatten documents into unformatted plain text, Wildflow preserves the academic hierarchy and formatting during extraction, ensuring that topics, subtopics, equations, and visual elements retain their contextual relationships.

Subject Identification During Extraction During the extraction phase, the system simultaneously performs subject identification. It analyzes title pages, header keywords, repeated subject-specific terminology, and predefined academic metadata to determine the correct subject classification. Based on this structured evaluation, the system identifies whether the document belongs to domains such as Engineering Chemistry or Engineering Mechanics. Once classified, the extracted content is automatically stored within the appropriate subject directory, adhering to a strict hierarchical organization to maintain consistency and enable controlled retrieval.

Path	Description
`CollegeName/`	Root directory
`YearOfStudy/`	Academic year partition
`Subject/`	Subject-level folder
`Unit/`	Unit-level folder
`original.pdf`	Source document
`unit.md`	Structured Markdown
`vector_db/`	Vector embeddings
`images/`	Extracted diagrams

Table 1: Hierarchical storage structure used by Wildflow

5.2 Equation Handling

Mathematical equations extracted from the PDF are converted into KaTeX format to preserve their structural integrity and semantic clarity. This conversion ensures symbol-level precision, enabling each mathematical component to remain intact and interpretable. It also guarantees clean rendering for frontend display, prevents token corruption during the embedding process, and maintains a consistent mathematical structure across the database

Example:

$$M_n = \frac{\sum N_i M_i}{\sum N_i}$$

Equations are preserved inside the Markdown file and are not embedded directly as raw text. This prevents semantic distortion in the vector space.

5.3 Markdown Structuring

After extraction, the content is converted into a structured Markdown format.

For Engineering Chemistry, the hierarchy is preserved as:

→ Main Lecture Title

→ Topic

- → Topic Points

→ Random Titles

####### → Sub Points

This structured formatting enables clean and logically segmented chunking of academic content, supports topic-aware embedding by preserving hierarchical relationships, ensures accurate JSON-based routing across subjects and units, and allows controlled retrieval at the unit level without cross-topic contamination.

5.4 Storage Strategy

The system maintains two synchronized representations of academic content to ensure both structural integrity and efficient retrieval. The first is an authoritative Markdown layer that preserves the complete hierarchical text, retains KaTeX-rendered equations, and maintains all image references, functioning as the definitive source of truth. The second is a vectorized representation composed of refined textual segments derived from the Markdown file. These segments are stored within a Qdrant vector database, organized at the unit level, and linked through structured metadata including topic, source file, subject, and unit identifiers.

File / Folder	Role
`unit.md`	Structured Markdown (authoritative)
`qdrant_db/`	Vector embeddings
`images/`	Extracted diagrams
`unit_metadata.json`	Topic mapping + paths

Table 2: Unit-level storage components

By separating structured Markdown storage from the embedded vector representation, the system achieves high retrieval precision while preserving equation integrity and image-topic alignment. This clear separation of authority ensures that semantic search operates efficiently without compromising the academic structure. The dual-storage architecture is therefore fundamental to maintaining accuracy, consistency, and controlled knowledge boundaries across the entire system.

6 Embedding Strategy

Wildflow adopts a selective, structure-aware embedding strategy designed to maximize retrieval precision while preserving academic integrity. The embedding layer is intentionally constrained to prevent semantic noise, structural distortion, and cross-topic contamination.

6.1 Text-Only Embedding Policy

The embedding pipeline operates under a strict text-only policy to preserve clarity within the vector space. Only

cleaned textual content is transformed into embeddings. Prior to embedding, KaTeX equation blocks as well as image references and file links are systematically removed. This separation ensures that semantic indexing remains focused purely on conceptual meaning, while structural elements such as equations and visuals are preserved exclusively within the authoritative Markdown layer.

6.2 Design Justification

Embedding raw equations or image references directly into a vector model can introduce significant semantic noise. Mathematical expressions consist of dense symbolic structures that do not behave predictably within standard text-based embedding spaces. When processed as ordinary tokens, such formulas can skew similarity calculations and degrade retrieval accuracy. Likewise, image file paths or reference strings lack meaningful conceptual value for semantic indexing.

By deliberately excluding these structural elements from the embedding layer, the system maintains higher semantic purity within text vectors, minimizes distortion in the vector space, improves topic-level similarity alignment, and ensures more stable and deterministic retrieval behavior. Importantly, equations and images are not removed from the system; they remain fully preserved within the authoritative Markdown source and are reintegrated and verified during the retrieval validation stage.

6.3 Structural Integrity Preservation

Although equations are excluded from the embedding pipeline, their original contextual placement within the Markdown hierarchy is fully retained. This design enables the system to first retrieve relevant textual segments through semantic similarity, then accurately reattach the corresponding equation from the authoritative Markdown source, and finally verify the equation's completeness and correctness before presenting the final output.

The deliberate separation between semantic indexing in the vector space and structural preservation within the Markdown authority layer represents a core architectural principle of the system.

6.4 Embedding Model Selection

The system utilizes the *nomic-ai/nomic-embed-text-v1* model for text embedding due to its strong semantic clustering performance and suitability for structured academic material. The model offers efficient embedding dimensionality, ensuring computational practicality without sacrificing retrieval quality. It demonstrates stable behavior when processing technical and domain-specific language, maintaining consistent similarity scoring across structured educational content. Additionally, it provides a balanced performance-to-cost ratio, making it well-suited for scalable academic retrieval systems.

6.5 Vector Storage Design

Embeddings are stored in Qdrant using unit-specific collections, where each academic unit functions as an independent semantic index. This isolation prevents cross-unit interference, reduces unintended similarity overlap, and improves retrieval determinism across subjects.

Each embedded chunk is enriched with structured metadata to enhance filtering precision and maintain system traceability. Typical metadata attributes include the subject, unit, topic, and source file. This structured tagging enables hybrid search strategies that combine semantic similarity with metadata-based constraints, ensuring that retrieval remains tightly bounded within the correct academic context.

6.6 Vector Database Architecture

The vector database uses a structured payload schema to maintain traceability and contextual alignment. Each embedded vector is stored along with metadata that binds it to its academic origin.

Example payload schema:

{

"topic": "Number-Average Molecular Mass",

"unit": "Chemistry Unit1",

"source": "unit1.md"

}

This architecture ensures that every retrieved vector can be:

1. Traced back to its exact Markdown source

2. Verified within its correct unit context

3. Reconstructed with its associated equations and images

By combining semantic embeddings with structured metadata, Wildflow achieves both precision and academic accountability within its retrieval layer.

7 Retrieval Pipeline

Wildflow follows a multi-stage retrieval process designed to ensure academic accuracy while minimizing hallucination. Unlike conventional RAG systems that perform a direct similarity search across all indexed data, Wildflow applies structured filtering and controlled model selection before generating a response. A key architectural feature of the system is the use of multiple specialized Gemini models, each assigned a specific role within the retrieval pipeline.

7.1 Multi-Model Query Handling Strategy

Wildflow handles three primary categories of academic queries: theory-based queries, mathematics-based queries,

and graphical or drawing-based queries. Each category is processed using a different model configuration in order to optimize accuracy, cost, and performance.

Query Detection Model When a query is received, a lightweight Gemini model is first invoked to perform several preliminary tasks. The model classifies the subject of the query, extracts the most relevant academic topic, and determines the query type, identifying whether the question is theoretical, mathematical, or graphical in nature. This model is optimized for speed and minimal token usage. Using a smaller model at this early stage significantly reduces API cost and improves system scalability, which ultimately contributes to lowering the overall cost for the end user.

Theory-Based Answering (Cost-Efficient Model) If the query is classified as conceptual or theory-based, a lightweight Gemini model is used to generate the response. The model is constrained to answer strictly using the information retrieved from the Markdown database. Because theory-based questions typically require explanation rather than complex computation, a smaller model is sufficient for producing accurate responses. This approach reduces API consumption, produces faster responses, and ensures that reasoning remains tightly bound within the database content.

Mathematics-Based Answering (Advanced Model) If the query is identified as mathematical in nature, a more advanced Gemini model is selectively invoked. In this case, the system first retrieves the relevant formula from the Markdown source, after which the advanced model performs the required step-by-step numerical or symbolic computation. Using a stronger model for mathematical reasoning improves computational reliability, reduces the likelihood of arithmetic errors, and provides more accurate symbolic processing. Because this model is used only when necessary, the system maintains a balance between computational accuracy and operational cost.

Graphical / Engineering Graphics Queries For subjects such as Engineering Graphics, Wildflow employs a layered modeling approach. In the first stage, an instruction-understanding model interprets the problem statement, converts the drawing requirement into structured instructions, and breaks the geometric procedure into logically ordered steps. In the second stage, an image generation model receives these structured instructions and produces the corresponding diagram or graphical output. This two-stage process ensures that the reasoning behind the drawing remains logically correct while separating the interpretation of the problem from the final rendering of the visual output.

7.2 Architectural Advantage

By dynamically selecting models based on query type, Wildflow achieves greater resource efficiency while maintaining high academic reliability. The architecture enables cost optimization, improves mathematical accuracy for computational problems, supports structured graphical generation for drawing-based queries, and significantly reduces the risk of hallucinated responses. This modular multi-model design allows the system to effectively support theory-based explanations, mathematics-based solutions, and graphical or diagrammatic outputs while maintaining strict architectural control and long-term scalability.

7.3 Query Result Caching

To further optimize operational cost and response latency, Wildflow implements a short-term query result caching mechanism. When multiple users submit identical or semantically similar queries within a short time window, the system temporarily stores the verified response generated for the first request. Subsequent queries that closely match the cached prompt are served directly from this temporary cache rather than invoking the LLM pipeline again.

This strategy significantly reduces redundant API calls to external language models while maintaining consistent responses for frequently asked academic questions. Because the cached outputs originate from the verified retrieval pipeline, they remain aligned with the authoritative Markdown dataset and maintain the same academic constraints as newly generated responses.

Cached results are stored only for a limited duration and are automatically invalidated after the expiration window to ensure that updates to the underlying academic dataset propagate correctly through the system.

8 Verification Framework

Wildflow incorporates a structured verification layer that operates after retrieval but before final response generation. This framework ensures that all retrieved content is contextually accurate, structurally complete, and logically consistent with the authoritative Markdown source. Instead of directly presenting retrieved results, the system performs a series of validation checks to confirm that the information is academically reliable.

8.1 Topic Consistency Validation

The first validation step ensures that the retrieved content aligns with the topic detected during the query classification stage. The system compares the retrieved segment with the intended topic extracted earlier in the pipeline. If the semantic match does not correspond to the correct topic or unit, the retrieval is rejected and the system performs another search within the appropriate academic scope.

8.2 Equation Integrity Verification

For formula-related queries, the system verifies the structural integrity of the retrieved equation before it is pre-

sented to the user. This includes confirming that the equation is complete, ensuring that all mathematical symbols, summations, and operators remain intact, and validating that the formula exactly matches the original representation stored in the Markdown source. Any partial or truncated equations are automatically discarded to prevent incorrect or misleading outputs.

8.3 Image–Topic Alignment Check

When a retrieved section references a diagram or visual element, the system performs an additional validation to ensure correct alignment between the image and its associated topic. The system confirms that the image belongs to the same topic as the retrieved text, verifies that the image is stored within the correct unit directory, and checks that the image reference matches the path defined in the Markdown file. This process prevents unrelated diagrams from being incorrectly attached to an answer.

8.4 Logical Coherence Validation

Before the final response is generated, the LLM performs a logical consistency evaluation of the retrieved material. This step checks that the explanation maintains conceptual alignment with the original academic context, verifies mathematical validity when numerical reasoning is involved, and ensures that the generated response does not contradict the authoritative source material. Even if the retrieval appears correct based on vector similarity alone, the response must pass this final validation stage before it is returned.

By introducing this explicit verification layer, Wildflow transforms retrieval from a purely probabilistic similarity search into a structured and academically bounded response generation process.

9 Numerical Reasoning Module

Wildflow treats numerical queries as a distinct category separate from conceptual or theory-based questions. Because mathematical accuracy is critical in academic contexts, numerical problems are processed through a controlled and formula-driven reasoning pipeline rather than conventional generative answering.

9.1 Numerical Query Detection

When a query is received, the system first evaluates whether the request involves numerical computation. This determination is made by analyzing the presence of numerical values in the query, identifying keywords that typically indicate calculation such as *calculate*, *determine*, or *find*, and detecting contextual cues that suggest a formula-based problem. If the query is classified as numerical, the system activates the dedicated mathematical reasoning pipeline.

9.2 Formula-First Approach

Instead of directly generating a solution, the system first retrieves the relevant formula from the authoritative Markdown source. This ensures that the computation is grounded in the exact formula used in the academic material. By doing so, the system guarantees that the correct institutional formulation is applied and prevents the model from substituting alternative formulas that may exist in its internal training knowledge.

9.3 Controlled Step-by-Step Solving

After retrieving the appropriate formula, the system inserts the numerical values from the problem into the equation and performs the computation in a structured step-by-step manner. Each intermediate step of the calculation is presented clearly, allowing the reasoning process to remain transparent and verifiable. This structured solving process significantly reduces the likelihood of arithmetic or symbolic errors.

9.4 Output Constraint

For numerical queries, the response format is intentionally constrained to maintain clarity and efficiency. The output includes the given values from the problem, the substitution of those values into the formula, the step-by-step calculation process, and the final computed answer. Extended theoretical explanations are minimized unless explicitly requested, ensuring that the response remains focused and avoids unnecessary token usage.

By isolating numerical reasoning within a controlled pipeline, the system maintains mathematical correctness while ensuring strict alignment with the authoritative academic source material.

10 Hallucination Control

Wildflow incorporates an architectural hallucination control mechanism designed to enforce strict academic boundaries. Rather than relying solely on prompt-based instructions, hallucination mitigation is integrated directly into the system's retrieval and verification pipeline.

10.1 Knowledge Boundary Restriction

The system restricts generated responses to information contained within the structured Markdown files and the associated unit-level vector databases. The language model is not allowed to extend beyond this predefined knowledge scope unless a controlled fallback condition is explicitly triggered. This restriction prevents the unintended blending of the model's internal training knowledge with the institutional academic material stored within the system.

10.2 Unit-Level Containment

Each academic unit operates as an isolated knowledge environment with its own vector collection and corresponding Markdown source. Retrieval is therefore limited to the unit identified during topic routing. This containment mechanism prevents cross-unit contamination, reduces the risk of topic drift, and avoids the accidental mixing of similar concepts that may appear across different subjects.

10.3 Mandatory Re-Query on Mismatch

If retrieved content fails to satisfy topic validation or structural verification checks, the system automatically performs a re-query within the appropriate unit or topic scope. This prevents weak similarity matches from being surfaced directly to the user and ensures that responses remain grounded in correctly aligned academic content.

10.4 Controlled Fallback Mechanism

Fallback to the broader knowledge of the language model occurs only under clearly defined conditions. These conditions include situations where no relevant information is found within the database, where retrieved content is incomplete, or when the query lies outside the coverage of the institutional material. Even during fallback scenarios, the system explicitly distinguishes between information derived from the database and reasoning generated by the model.

By embedding hallucination control directly into the architecture rather than relying solely on prompt constraints, the system establishes a more deterministic and academically reliable response framework.

11 Implementation Architecture

Wildflow's implementation is built around structured storage, unit-level isolation, and modular model orchestration. The architecture ensures clear separation between academic content, embeddings, metadata, and model execution logic.

11.1 Folder Structure

Wildflow follows a hierarchical storage model organized as:

Each Unit Folder contains:

At the subject level, a master JSON file maintains:

This JSON routing layer ensures that queries are directed only to the relevant unit.

11.2 Query Routing via JSON

When a query enters the system, the subject is first identified and the core topic of the question is extracted. The subject-level JSON file is then consulted to determine which unit contains that topic. Once the correct unit is identified, retrieval is restricted to the corresponding Qdrant collection associated with that unit. This routing mechanism prevents global vector searches and enforces strict containment within the appropriate academic scope.

Path	Description
`MITwpuEmbeddedMD/`	Root directory
`Engineering Chemistry/`	Subject folder
`Unit1/`	Unit folder
`unit1.md`	Structured Markdown
`qdrant_db/`	Vector embeddings
`images/`	Extracted diagrams
`unit1.json`	Unit metadata

Table 3: Implementation folder structure

File / Folder	Description
`unit1.md`	Authoritative structured Markdown file
`qdrant_db/`	Unit-specific vector embeddings
`images/`	Extracted diagrams and figures
`unit1.json`	Metadata (topic mapping and file path)

Table 4: Contents of each unit folder

11.3 Model Selection Strategy

Wildflow dynamically selects the appropriate language model depending on the nature of the query. For conceptual or theory-based questions, the system uses the gemini-2.0-flash model. This model is responsible for topic detection and structured response generation while maintaining low API usage and efficient resource consumption. Its lightweight design makes it well suited for text-based academic reasoning where complex computation is not required.

For numerical queries, the system invokes the more advanced gemini-2.5-pro-exp-03-25 model. This model is capable of performing accurate symbolic reasoning and step-by-step mathematical computation. By reserving the advanced model for mathematical problems only, the system improves computational reliability while maintaining cost efficiency.

Graphical or Engineering Graphics queries are processed through a layered model pipeline. In the first stage, an instruction-generation model interprets the drawing problem, converts the requirement into structured geometric instructions, and logically decomposes the steps required to construct the figure. In the second stage, an image generation model receives these structured instructions and produces the corresponding diagram or graphical output. This separation between reasoning and rendering improves logical correctness, enables controlled image synthesis, and increases explainability in graphical problem solving.

Field	Purpose
Unit names	Identifies available academic units
Topic lists	Maps topics to their respective units
Database paths	Points to unit-level Qdrant collections
Markdown paths	Links to authoritative Markdown sources

Table 5: Master JSON routing file structure

11.4 Integrated Execution Flow

The full implementation integrates several coordinated components including structured file storage, JSON-based routing, unit-level vector databases, dynamic Gemini model selection, and a layered graphical generation pipeline. Together, these elements form a modular architecture capable of supporting the three major categories of academic queries: theory-based explanations, mathematics-based problem solving, and graphical or drawing-based outputs. This design maintains scalability, improves cost efficiency, and enforces strict academic boundary control throughout the system.

12 Experimental Evaluation

12.1 Evaluation Methodology

Evaluation was conducted using structured comparison reports in which AI-generated responses were compared against expected answers derived from official lecture materials. The expected answers serve as the authoritative ground truth, ensuring alignment with institutional academic content.

Each response was evaluated based on correctness, completeness, and conceptual alignment. A structured scoring scheme (0–4) was applied per question to quantify performance. While parts of the evaluation pipeline are automated, scoring is based on explicit comparison with ground-truth answers rather than free-form judgment.

A total of **13 questions** were evaluated:

- Engineering Mechanics: 5 questions
- Engineering Chemistry: 8 questions

12.2 Subject-wise Accuracy

Wildflow achieves perfect subject accuracy (100%) across both Engineering Mechanics and Engineering Chemistry, demonstrating reliable domain identification for all queries.

For numerical queries, performance reaches 90% in Engineering Mechanics and 80% in Engineering Chemistry, indicating strong computational correctness with minor variation across subjects.

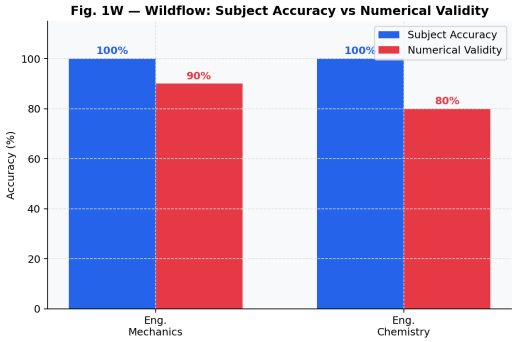

Figure 1: Subject-wise accuracy across query types

12.3 Vector Retrieval Quality

Figure 2 illustrates the fuzzy similarity scores obtained during vector-based retrieval for each query. A threshold of 60 is used to determine acceptable topic alignment.

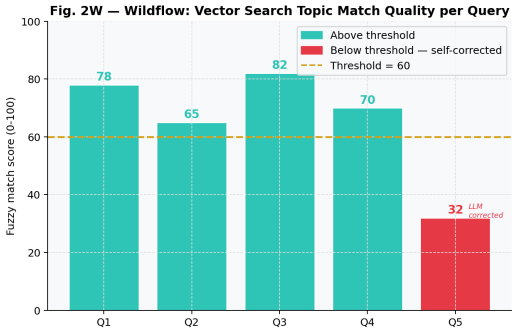

Figure 2: Vector search topic match quality per query with threshold-based filtering

Most queries achieve scores above the threshold, including values of 78, 65, 82, and 70, indicating strong semantic matching between the user query and retrieved content.

One query falls significantly below the threshold with a score of 32. Despite this, the system successfully produces a correct response through its verification and correction mechanism.

Overall, these results demonstrate that the vector search component provides reliable topic alignment, while the validation layer ensures robustness against retrieval failures.

12.4 Answer Completeness Analysis

Responses were evaluated across five dimensions: correctness, completeness, depth, formatting, and relevance.

Results indicate that:

- Both theory and numerical responses maintain high correctness and relevance (approximately 4.5–5 range)

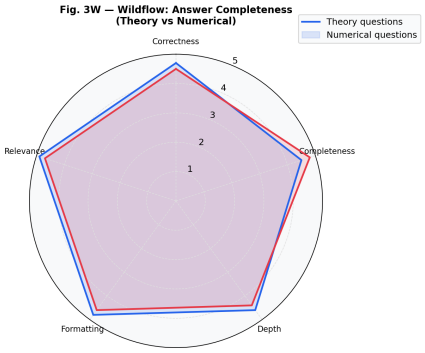

Figure 3: Answer completeness comparison (theory vs numerical)

- Theory responses exhibit slightly greater depth and formatting consistency
- Numerical responses remain highly complete but slightly lower in depth due to structured output constraints
- Overall response quality remains consistently high across all evaluation dimensions

12.5 RAG System Metric Comparison

Figure 4 presents a comparative analysis of key system performance metrics across Engineering Mechanics and Engineering Chemistry.

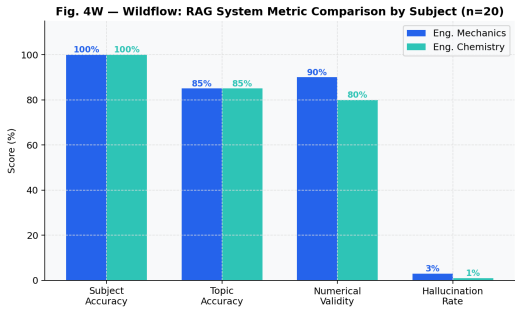

Figure 4: RAG system metric comparison by subject

Both subjects achieve 100% subject accuracy, indicating correct domain identification for all queries. Topic accuracy remains consistent at 85% for both Engineering Mechanics and Engineering Chemistry.

Numerical validity reaches 90% in Engineering Mechanics and 80% in Engineering Chemistry, indicating strong and reliable computational performance.

Hallucination rates remain extremely low, with 3% in Engineering Mechanics and 1% in Engineering Chemistry, demonstrating effective grounding and validation.

Overall, these results highlight strong performance across all evaluation metrics, with minimal hallucination and high numerical correctness.

12.6 Overall System Performance

Figure 5 provides a consolidated view of system performance across key evaluation metrics.

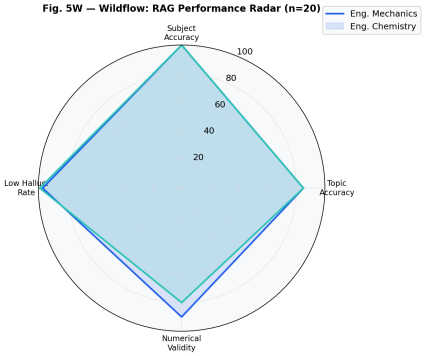

Figure 5: RAG system performance radar across evaluation metrics (n=20)

Both subjects achieve 100% subject accuracy and 85% topic accuracy, indicating consistent and reliable retrieval performance.

Numerical validity varies slightly, with Engineering Mechanics at 90% and Engineering Chemistry at 80%, while remaining overall strong.

Hallucination rates remain near zero (1–3%), confirming the effectiveness of the verification pipeline in ensuring factual correctness.

Overall, the radar visualization highlights a well-balanced system with high accuracy, strong numerical reasoning, and extremely low hallucination rates.

12.7 Key Observations

Analysis of the evaluation results reveals several important insights:

- **Strong theory performance:** The system consistently produces accurate and well-structured answers for conceptual queries.
- **Numerical reliability (conditional):** Numerical reasoning is accurate and consistently high across subjects.
- **Robust verification layer:** The system successfully corrects low-quality retrieval cases.
- **Depth of responses:** Generated answers demonstrate high completeness, clarity, and structural consistency.

13 Comparative Analysis

Wildflow is evaluated against a baseline large language model setup using Gemini 2.5 Flash, which operates without structured retrieval, topic-level routing, or verification

mechanisms. Although the baseline model has access to the same academic documents, it generates responses directly, relying solely on its internal reasoning and implicit retrieval.

13.1 Overall Performance

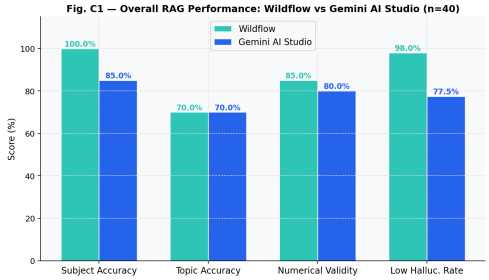

Figure 6: Overall RAG Performance: Wildflow vs Gemini (n=40)

As shown in Fig. 6, Wildflow consistently outperforms the baseline across all evaluation metrics. It achieves perfect subject accuracy (100%) compared to 85% for Gemini, while maintaining comparable topic accuracy (70%). More importantly, Wildflow demonstrates superior numerical validity (85% vs 80%) and significantly lower hallucination rates (98% vs 77.5% reliability), highlighting its robustness in academic settings.

13.2 Subject-wise Analysis

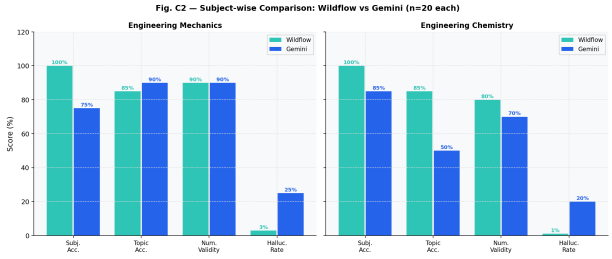

Figure 7: Subject-wise Comparison: Wildflow vs Gemini

The subject-level breakdown (Fig. 7) reveals distinct performance patterns:

Engineering Mechanics: Wildflow achieves perfect subject accuracy (100%) and matches Gemini in numerical validity (90%), while drastically reducing hallucination rate (3% vs 25%). Gemini slightly outperforms in topic accuracy (90% vs 85%).

Engineering Chemistry: Wildflow shows substantial gains in topic accuracy (85% vs 50%) and numerical validity (80% vs 70%), along with near-zero hallucination (1% vs 20%).

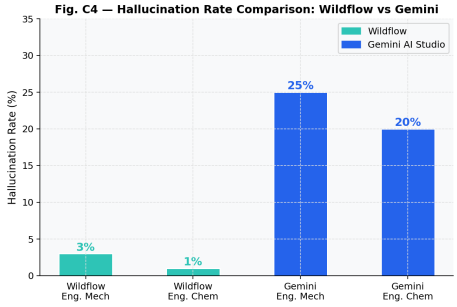

Figure 8: Hallucination Rate Comparison

13.3 Hallucination Control

Fig. 8 highlights one of the most critical improvements: hallucination reduction. Wildflow maintains near-zero hallucination rates across both subjects (1–3%), whereas Gemini exhibits significantly higher rates (20–25%).

13.4 Performance Gap Analysis

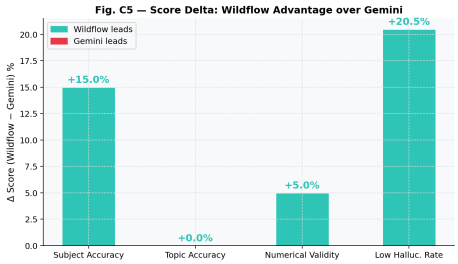

Figure 9: Score Delta: Wildflow Advantage over Gemini

The score delta (Fig. 9) quantifies Wildflow's advantage:

- +15% in subject accuracy
- +5% in numerical validity
- +20.5% in hallucination reliability
- 0% difference in topic accuracy

13.5 Gemini Standalone Behavior

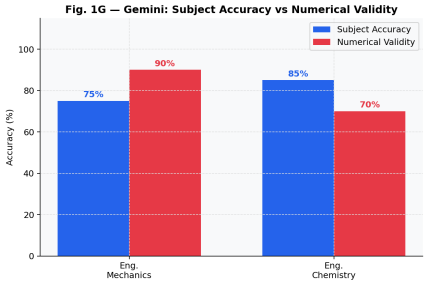

Figure 10: Gemini: Subject Accuracy vs Numerical Validity

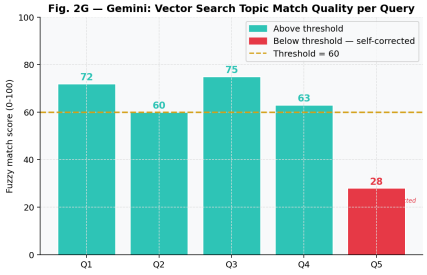

Figure 11: Gemini: Vector Search Topic Match Quality

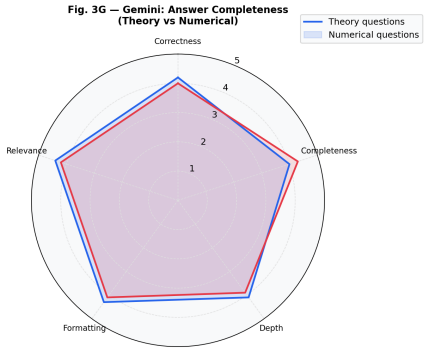

Figure 12: Gemini: Answer Completeness Radar

Additional analysis (Fig. 10–13) shows that:

- Performance varies significantly across subjects.
- Numerical reasoning is inconsistent.
- Retrieval quality occasionally falls below threshold.
- Answer completeness is moderate compared to structured systems.

13.6 Key Insight

The results demonstrate that structured RAG pipelines like Wildflow do not necessarily improve topic identification but significantly enhance reliability, numerical correctness, and hallucination control. This makes them more suitable for academic and technical applications where precision is critical.

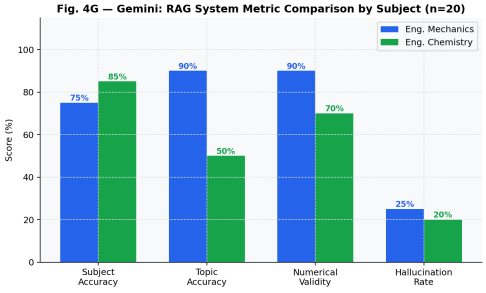

Figure 13: Gemini: Metric Comparison by Subject

14 Limitations

While Wildflow introduces strong architectural controls designed to improve academic reliability, several practical and technical limitations remain.

14.1 Dependence on Structured Markdown

The system relies heavily on structured Markdown generated from institutional PDF materials. If the source documents contain OCR inaccuracies, inconsistent formatting, or missing hierarchical markers, the resulting Markdown structure may not accurately represent the original academic layout. Since retrieval and topic mapping depend on this structured hierarchy, any preprocessing errors can affect retrieval precision. As a result, the system performs best when academic documents follow consistent formatting and clear structural organization.

14.2 Domain-Specific Optimization

Wildflow is designed primarily for structured academic curricula such as Engineering Chemistry, Engineering Mechanics, and Engineering Graphics. While the architecture itself is scalable and can support additional subjects, introducing new academic domains may require adjustments to the preprocessing pipeline. This can include refining structural parsing rules, adapting topic detection heuristics, and calibrating prompts used by the language models. Consequently, the system functions as a specialized academic retrieval framework rather than a fully generalized open-domain AI system.

14.3 Structured JSON Dependency

The retrieval pipeline depends on subject-level and unit-level JSON routing files that map topics to their corresponding units, vector databases, and Markdown sources. These JSON structures enable precise routing of queries to the correct academic unit. Although these metadata files are generated automatically through preprocessing scripts, the system still depends on accurate metadata extraction. If topic detection during preprocessing is incorrect or incomplete, routing accuracy may be reduced, which can affect retrieval quality.

14.4 Institutional Data Scope

Wildflow is designed to support a large institutional user base, such as a college ecosystem serving thousands of students. However, its knowledge scope is intentionally restricted to the academic material stored within the institutional dataset. The system is not intended to replace general-purpose web-scale AI systems or provide open-domain knowledge. This limitation is a deliberate archi-

tectural choice that prioritizes academic correctness, structured curriculum alignment, and controlled knowledge boundaries over unrestricted information coverage.

15 System Design and Deployment Considerations

Wildflow was developed not only as a research prototype but also as a scalable academic information system intended for direct student use. The objective is to provide a structured AI-assisted learning interface that operates over syllabus-aligned academic material without requiring formal institutional deployment. The system architecture is designed to operate on structured course content derived from institutional documents, enabling controlled retrieval and reasoning over curriculum-specific knowledge.

15.1 System Objectives

The long-term goal of Wildflow is to function as a structured academic assistance system capable of supporting multiple subjects within defined educational curricula. The platform focuses on providing reliable access to academic material through a retrieval-driven architecture that supports conceptual explanations, formula-based numerical problem solving, and interpretation of diagram-based questions.

The system maintains alignment with academic syllabi by operating over structured Markdown representations of course material. This allows queries to be resolved within a controlled knowledge scope while preserving the structure of the original academic sources. The platform is designed to be accessible directly to students, with the architecture emphasizing scalability, controlled knowledge boundaries, and consistent interaction with curriculum-based datasets.

15.2 Deployment Strategy

The development of Wildflow follows a staged expansion of both its academic dataset and routing infrastructure. The first phase focuses on achieving comprehensive coverage of core engineering subjects. This expansion begins with foundational first-year material and gradually extends to second-, third-, and final-year coursework. By progressively integrating subjects across academic levels, the system is designed to support retrieval and reasoning throughout the full progression of an engineering degree program.

After establishing stable coverage within engineering disciplines, the next stage extends the platform to additional academic programs such as business administration, biotechnology, biological sciences, and other undergraduate or postgraduate domains. Because the underlying architecture is subject-agnostic, this expansion primarily involves integrating additional structured Markdown datasets and updating the topic-routing layers that connect queries to the corresponding academic units.

A subsequent phase focuses on extending system functionality beyond standard question–answer retrieval. In this stage, the platform may support additional academic workflows such as structured guidance for laboratory manuals, assistance with project development tasks, preparation support for internal examinations, and structured decomposition of assignments. These additions extend the system from a retrieval-based academic interface toward a broader academic assistance environment.

The final stage focuses on scaling the system across multiple institutional curricula. Instead of binding the system to a single syllabus, the architecture allows multiple curriculum structures to coexist through separate routing layers and dataset partitions. This design allows the platform to scale horizontally across institutions while preserving syllabus-specific alignment.

15.3 System Architecture Context

Wildflow is designed as an independent academic assistance platform rather than a system intended for direct institutional deployment. The architecture allows students to access syllabus-aligned academic support through a structured retrieval interface without requiring formal integration with institutional IT systems. This model simplifies deployment and reduces administrative dependencies while allowing the platform to operate across diverse academic environments.

15.4 Implementation Stack

The implementation architecture combines several components designed to support scalable retrieval and model orchestration. The backend services are implemented in Python using FastAPI to manage API routing and system logic. Firebase is used for authentication, user management, and real-time database support. Vector storage is handled through Qdrant, which manages the semantic embedding collections associated with each academic unit, while the Gemini API provides the reasoning models used during query processing.

The frontend interface is implemented using Node.js with a React-based user interface. Mathematical expressions are rendered using KaTeX to preserve equation fidelity, while graphical queries are supported through an integrated diagram canvas capable of generating and visualizing structured engineering drawings.

At the infrastructure level, the system operates with cloud-hosted vector storage, scalable API orchestration, and a modular model-selection layer that dynamically routes queries to the appropriate reasoning model. Firebase services also support authentication workflows, subscription management, usage monitoring, and scalable onboarding of new users.

15.5 Resource and Accessibility Considerations

Operational efficiency was a key design objective during system development. Wildflow uses a tiered model selection strategy in which lightweight models handle conceptual or theory-based queries, while more advanced models are invoked only for tasks requiring numerical reasoning or complex computation. By restricting the use of higher-cost models to cases where they are necessary, the system maintains efficient resource utilization and controlled operational costs.

This architecture enables the platform to remain accessible to a broad range of students while maintaining sufficient operational sustainability.

15.6 Operational Sustainability

The system's sustainability model is based on a combination of subscription-based access, large user volume, and optimized per-query resource usage. Because model invocation is tightly controlled and token usage is minimized wherever possible, the marginal cost per user remains relatively low. This allows the platform to scale efficiently while maintaining predictable operational expenses.

The financial design prioritizes long-term sustainability and accessibility rather than aggressive monetization strategies.

15.7 Future Work and Extensions

Wildflow is intended to function as a structured academic intelligence system built around syllabus-aligned knowledge bases. Unlike open-domain conversational AI systems, its operation is constrained to defined academic datasets, allowing it to maintain stronger alignment with institutional course material.

The long-term objective is to develop a scalable academic infrastructure capable of supporting multiple subjects, expanding across curricula from different institutions, and maintaining consistent academic reliability. In this sense, the system aims to operate as a curriculum-aware academic assistance framework rather than a general-purpose AI system.

16 Advanced Research Directions and Deferred Enhancements

During the development of Wildflow, several advanced architectural improvements were explored at the research level. However, due to constraints related to development time, computational infrastructure, and early-stage deployment capital, these capabilities were postponed for future iterations. The following directions represent potential extensions that could significantly expand the system's capabilities in later versions.

16.1 Subject-Specific Small Language Models (SLMs)

One of the most significant research directions involved replacing external API-based language models with locally hosted Small Language Models that are trained or fine-tuned for individual subjects. Under this approach, each subject would operate with its own lightweight reasoning model trained specifically on the structured Markdown corpus associated with that subject. Retrieval and reasoning would therefore operate in a tightly integrated environment where the model's knowledge is strictly bounded by the academic dataset.

Such an architecture could reduce dependence on external APIs, significantly lower operational cost per query, strengthen academic boundary enforcement, and improve precision for subject-specific responses. In the long term, this would allow the system to operate with greater determinism while scaling economically across a large student population. The primary barriers to implementing this approach include the cost of GPU infrastructure, the complexity involved in fine-tuning domain-specific models, and the engineering expertise required for efficient model optimization.

16.2 Fully Local Model Stack

Another long-term objective involves transitioning toward a fully local AI stack. In this configuration, embedding models, reasoning models, and numerical solvers would all operate locally rather than through external APIs. Optional multimodal components, such as local vision models for diagram interpretation, could also be incorporated. This approach would reduce recurring API expenses, improve control over system behavior, strengthen data privacy, and increase long-term operational sustainability.

16.3 Multimodal Learning Expansion

Wildflow was originally conceptualized as a platform capable of supporting more than text-based interaction. Future development could include multimodal learning features designed to improve accessibility and learning flexibility. These enhancements may include voice-to-text query submission, text-to-speech playback of answers, and conversational voice-assisted study modes that allow students to interact with the system verbally. Additional possibilities include automated lecture summarization, video-to-notes extraction, and AI-generated walkthrough explanations derived from recorded lectures. These capabilities were deferred due to integration complexity and the increased computational cost associated with large-scale multimodal processing.

16.4 Academic Project Assistance Layer

Another potential expansion involves transforming Wildflow into a structured academic project assistant. Instead of focusing solely on question answering, the platform could

support guided project development workflows. Possible features include structured project breakdowns, step-by-step development roadmaps, automated report structure generation, explanation of code segments for technical subjects, and assistance with technical documentation. This layer would expand the system into a broader academic productivity platform rather than a purely instructional tool.

16.5 Study Guide and Smart Notes Generation

The structured Markdown dataset used by Wildflow also enables automated generation of study materials. Future enhancements could include automated creation of exam-oriented study guides, unit summaries, topic-wise revision sheets, and consolidated formula references. The system could also analyze the curriculum to generate predicted question groupings and targeted revision materials. Such features would allow students to prepare for examinations using structured summaries derived directly from the academic content base.

16.6 Academic Resource Library

Another planned capability involves developing a centralized academic resource repository integrated with the existing retrieval system. This repository could include previous year question papers, indexed textbooks, structured syllabus mappings, and topic-linked resource navigation. Rather than functioning as a generic document storage system, the goal would be to create semantic relationships between these resources—for example linking previous exam questions directly to relevant topics or formulas within the curriculum.

16.7 Constraints That Prevented Immediate Implementation

Several factors prevented the immediate deployment of these advanced features. These include limited capital for GPU infrastructure, sensitivity to API costs during early scaling stages, and the need to prioritize development of the core retrieval and verification architecture. Additionally, building and maintaining specialized models or multimodal pipelines requires substantial engineering time and experimentation, which was outside the scope of the initial system release.

16.8 Strategic Importance of These Enhancements

Despite being deferred, these enhancements represent an important direction for the platform's future evolution. Subject-specific SLM deployment could enable cost independence and stronger academic boundary enforcement, while multimodal input and output capabilities would expand accessibility for different learning styles. Additional academic productivity tools and structured resource libraries would further differentiate the system within the educational technology landscape. Importantly, the current architecture of Wildflow was intentionally designed to remain modular, allowing these enhancements to be integrated without requiring a fundamental redesign of the system.

17 Conclusion

Wildflow represents both a technical framework and a significant personal milestone.

Technically, Wildflow demonstrates a structured rethinking of how large language models should operate within academic environments. Instead of functioning as an unrestricted generative system, it introduces a verification-driven architecture that prioritizes structured curriculum data over probabilistic model output.

By integrating:

- Markdown as the authoritative source
- Unit-level vector database isolation
- Deterministic JSON routing
- Multi-model query classification
- Formula-first numerical solving
- Architectural hallucination control

Wildflow moves beyond traditional RAG implementations and toward a bounded academic intelligence framework designed specifically for structured education.

17.1 Personal Journey

This project was developed during my first year of engineering, from January 2025 to April 2025. It was my first experience working on something of this scale, architecting an end-to-end system involving structured data pipelines, vector databases, multi-model orchestration, and product-level scalability planning.

The project was paused in April 2025 due to ongoing academic commitments and was eventually concluded. However, the development process provided hands-on experience in large-scale system design, cost-aware AI deployment, and building a technically ambitious product from the ground up.

17.2 Research & Public Writing

Throughout the development of Wildflow, I documented key architectural decisions, technical explorations, and system design insights on my Medium page:

Medium:
https://medium.com/@rohandeogaonkar.9

These articles reflect the progression of the project from early experimentation with structured retrieval systems to the development of a verification-driven academic AI framework. They capture the technical thinking, design trade-offs, and architectural principles that shaped Wildflow during its active development phase.