PaperExplainAgent: Interactive Visual Explanations for STEM Papers

Abstract

PaperExplainAgent is an interactive reading assistant designed to help researchers and students make sense of dense STEM papers. Given a PDF, the system responds with simple visual or animation-style sketches that could be rendered with tools like Manim. To build this behavior, we build on open-source LLMs with prompting and integration strategies tailored to dense mathematical texts. Our design is motivated by work in cognitive science and math education showing that well-aligned diagrams and dynamic visualizations can reduce cognitive load and improve conceptual understanding, especially for symbol-heavy material.

Research suggests that structured, visually grounded responses are preferred to generic text-only explanations and are perceived as clearer and more helpful for understanding unfamiliar results. We conclude by discussing limitations of our current prototype and outlining how tighter integration between interactive reading tools and visualization backends could further support mathematical and STEM research.

1 Introduction

1.1 Motivation

Reading advanced mathematics and STEM papers is notoriously difficult for humans. The text is densely packed with symbols and layered definitions, assuming a large body of background knowledge. As a result, readers must frequently pause and flip back to earlier definitions or theorems and even consult external references to follow along andrewhead.info. Studies on reading mathematical notation show that readers constantly shift attention between formulas and the surrounding prose andrewhead.info, and comprehension suffers when they have to juggle complex symbols in working memory andrewhead.info. In standard large language model (LLM) chat interfaces, the assistance is decoupled from the actual paper. Users copy text into a chat and get a lengthy reply that is often unstructured and not directly tied to the PDF. This disconnect makes it hard to align the explanation with the precise notation and context in the paper. There is a clear need for more context-sensitive support that can attach explanations directly to the paper text, helping readers without disrupting their workflow.

1.2 Our Approach: PaperExplainAgent

We propose **PaperExplainAgent**, an interactive reading assistant that provides on-demand, contextual explanations for passages in STEM papers. The envisioned usage is: a user opens a PDF in our interface, the system then returns a **layered explanation** of that highlighted passage. The answer is structured into sections – for example, an *Intuition* overview, a list of *Key Assumptions* or definitions needed, a *Detailed Explanation* of the formal content, and even a *Visual Sketch or Animation Plan* suggesting how one might illustrate the concept. Crucially, these explanations are grounded in the selected passage and remain brief. The goal is to provide *short, targeted help* that illuminates the text while keeping the original paper as the primary focus, rather than generating a long free-standing lecture.

1.3 Contributions

Our work makes the following contributions:

1. **PaperExplainAgent System:** We design an interactive assistant that links PDF document spans to structured explanations tailored for mathematics and STEM content. To our knowledge, this is the first tool that allows users to *select arbitrary text in a research paper and get a multi-faceted explanation* on the fly.

2. **Layered Explanation Format:** We introduce a prompting strategy and output format that separates an explanation into multiple layers – offering high-level intuition, clarifying assumptions, formal details, and explicit visual guidance. This structured format is inspired by cognitive principles and helps prevent the "wall of text" problem common in LLM outputs.

3. **Preliminary Evaluation:** We conduct a pilot evaluation, including automatic metrics adapted from a recent theorem-explanation benchmark and a small user study. We compare our system's explanations to text-only baseline explanations. The results indicate that explanations with explicit visual structure are preferred by users and improve clarity and perceived understanding.

4. **Design Insights:** Through the development and evaluation, we synthesize prior findings from cognitive science and education on **why visual and multimodal explanations aid learning**. We discuss how those insights shaped PaperExplainAgent's design – for example, how adding diagrams can reduce cognitive load – and reflect on the implications for future human-AI interfaces in mathematical research.

---

2 Why Visual Explanations Matter for STEM Reading

2.1 Cognitive challenges of dense mathematical text

Formal STEM texts (especially mathematics) impose a high cognitive load on readers. A reader often must keep numerous definitions, notational conventions, and prior results in working memory at once while parsing new statements. According to cognitive load theory, the *intrinsic load* of complex material can quickly approach or exceed the limits of working memorydigitallearninginstitute.com. In a dense proof, for instance, one might need to remember the meanings of symbols f, g, h (each with its own definition), the statement of a lemma referenced in passing, and the overall goal – all simultaneously. This is extremely demanding, leading to what Sweller calls *extraneous load* when the format or presentation of information forces unnecessary mental effortscience-gate.com. Mathematical notation, while powerful, is also compact and abstract; unlike an illustrative diagram or an example, a symbolic expression gives little intuitive cue for a novice. Research has likened reading math notation to reading code or a foreign language with its own grammarandrewhead.info. Novice readers tend to process formulas symbol-by-symbol and rely on adjacent text for interpretationandrewhead.info, whereas experts chunk symbols into higher-level concepts – a discrepancy that highlights how notation alone can hinder understanding for less experienced readers. In summary, the standard format of STEM papers (dense symbols, few diagrams, and linear text) can create a high *extraneous cognitive load* on learners, who must mentally "translate" and hold multiple ideas at once. ## 2.2 Benefits of visual and multimodal representations

A rich body of empirical work in math education and cognitive science shows that well-chosen visual representations can significantly ease the comprehension of complex concepts. Diagrams and spatial layouts help learners **perceive structure and relationships** that might be implicit in purely symbolic text. For example, adding a diagram to a geometry theorem or a commutative diagram to an algebraic argument can make the relationships between entities immediately clear, rather than forcing the reader to infer them from algebraic notation.

Research by Schnotz and Kürschner (2007) found that graphs and diagrams clarify relationships between variables, effectively simplifying problem-solving processes for learnersscience-gate.com. Visuals also serve as an external memory: instead of mentally keeping track of an object's properties, a diagram can *offload* that information into a spatial form that is easier to recall and manipulate (a form of distributed cognition). In essence, a picture can chunk information into a coherent whole, reducing the number of separate pieces a reader must juggle.

Visual representations can also make **hidden processes visible**. Many STEM concepts involve dynamic or non-intuitive processes – for instance, the limiting behavior of a sequence, the flow of probability mass in a distribution, or the step-by-step execution of an algorithm. In text, readers often have to *mentally simulate* these processes. By contrast, a visual (like a series of snapshots or an animation) can explicitly show what happens, step by step. This reduces the need for learners to imagine the process entirely in their heads, thereby lowering cognitive burdenalibali.psych.wisc.eduscience-gate.com. The educational psychology principle of *dual coding* (Paivio, 1990) posits that combining verbal and visual information yields two cognitive channels instead of one, leading to better understanding and recallscience-gate.com. Indeed, Mayer's Multimedia Principle states that people learn better from words and pictures together than from words alonedigitallearninginstitute.com. There is concrete evidence for these benefits: a recent meta-analysis of visualization interventions in math education found a medium overall effect ($g \approx 0.50$) of incorporating external visualizations on students' learning outcomesresearchgate.net. In practical terms, adding a simple diagram or intuitive sketch alongside a theorem can provide a mental foothold, helping readers grasp the "story" behind the symbols.

> **Figure 2 (concept):** *Text vs. visual-enhanced presentation.* (Left) A dense theorem statement in text-only form, requiring the reader to parse and imagine the relationships. (Right) The same concept illustrated with a structured diagram, breaking the theorem into labeled parts. The diagram uses arrows and spatial grouping to show how each component of the theorem relates, making the logical structure immediately more apparent. ## 2.3 Dynamic and interactive visualizations

Static diagrams are helpful, but *dynamic* and *interactive* visuals can further enhance understanding in ways static images cannot. **Animated** visuals have a time dimension, allowing information to be revealed sequentially rather than all at once. This is valuable because it enables *staging* of complex explanations: learners can be guided through a concept step by step, focusing on one aspect at a time. For example, an animation of an inductive proof can start with the base case, then visually transform the base case into the $n + 1$ case, highlighting the change. This sequential unveiling aligns with the *segmenting principle* in multimedia learning, which says people learn better when complex information is broken into learner-controlled segmentsdigitallearninginstitute.com. By controlling the pace of an animation (pausing, replaying, scrubbing back and forth), users can adjust the presentation to their needs, which increases engagement and active processingdigitallearninginstitute.com. Studies have shown that giving learners control over

animation playback leads to improved comprehension compared to a one-shot video lecturedigitallearninginstitute.comdigitallearninginstitute.com.

Another advantage of dynamic visuals is that they can illustrate changes and **transitions** that are otherwise hard to convey. Consider the classic ε–N definition of a limit: a static diagram might show δ and ε neighborhoods once, but an animation can *move* the δ-interval or shrink the ε-band to demonstrate how for any ε the δ can be adjusted – a moving picture makes the dependency concrete. Interactive visuals, such as an embeddable graph where a user can drag a slider to change a parameter, further let readers *explore* the concept. This interactivity fosters active learning; the reader is not just passively watching an explanation but actively probing the concept ("What if I increase n? How does the graph change?"). By giving agency, interactive elements can improve motivation and allow learners to test their understanding in real time. Overall, dynamic and interactive visualizations align with principles of learner-controlled pacing and feedback, which are known to enhance learning outcomesdigitallearninginstitute.comdigitallearninginstitute.com. While our system currently focuses on generating *plans* for visuals rather than fully rendered animations, the intention is to pave the way for such dynamic content to be integrated seamlessly into the reading experience.

2.4 Prior systems for theorem and paper visualization

Our work builds on and differs from several threads of prior research on explaining mathematical content with the aid of visuals. One closely related effort is **TheoremExplainAgent (TEA)** by Ku et al. (2025), which introduced an *agentic pipeline* for turning formal theorems into long-form explanatory videostiger-ai-lab.github.io. TEA uses one LLM-based agent to plan an explanation (including a storyboard and narration) and another agent to generate Manim animation code, producing narrated videos over 5 minutes longtiger-ai-lab.github.ioarxiv.org. They also created a benchmark called TheoremExplainBench with 240 theorems across multiple disciplines to evaluate such multimodal explanationstiger-ai-lab.github.io. While TEA demonstrates that it's possible to generate detailed animated lectures for theorems, it is geared toward *standalone* explanations (much like a YouTube lesson) rather than on-demand help during reading. The system requires significant computation to produce each video and is not interactive for a user in real time. Our PaperExplainAgent draws inspiration from TEA's goal of multimodal explanationsarxiv.org but targets a different use-case: providing *short, on-the-spot explanations* for arbitrary snippets of a paper. This requires a lighter-weight approach; instead of elaborate 5-minute videos, we focus on concise explanations with optional sketches that can be delivered with low latency.

Another relevant line of work is the design of *augmented reading interfaces* for math and science papers. For example, *ScholarPhi* (Head et al., 2019) allows readers to hover over symbols in a PDF to see their definitions in a tooltipandrewhead.info. This kind of tool addresses the problem of flipping between pages by directly linking mentions of a concept to its definitionandrewhead.info. However, such interfaces typically provide only brief factual annotations (like definitions or acronym expansions) and do not generate new explanatory content. More recent HCI research has looked at **math augmentation**, where authors manually add visual cues to formulas to aid readersandrewhead.info. Head et al. (2022)

documented how authors of blog posts and textbooks use color highlights, annotations, and custom diagrams to make formulas more readableandrewhead.info. Their findings show the value of visual context: for instance, color-coding parts of an equation to match a diagram can significantly help readers map notation to meaningandrewhead.info. This informs our approach of tightly coupling explanations with visuals (even if ours are automatically generated). It also highlights a limitation of prior approaches: they required **manual effort** from authors or editors to create those visual augmentations.

There have also been efforts in the AI community to create datasets and benchmarks that pair math or science content with visual explanations. For example, *MATH-Vision* (Wang et al., 2024) is a dataset of 3,040 math problems from competitions, each accompanied by a diagram or visual contextarxiv.org. It was developed to evaluate multimodal math reasoning by LLMs. While not directly about research papers, such datasets underscore the growing interest in combining text with visuals for mathematical problem solving. Similarly, some prior systems focus on specific domains, like visualizing physics concepts or algorithms, often resulting in fixed videos or interactive demos. These are usually one-off presentations rather than general tools – for instance, an AI might generate a visual explanation for Newton's laws or an algorithm like Dijkstra's, but the pipeline is not easily applied to arbitrary new inputs without manual setup.

In summary, prior work demonstrates the promise of visual and multimodal explanations in STEM. But existing solutions tend to either produce **long, static explanations** (as in videos or tutorials) or require significant **manual authoring** of visuals. They are not optimized for a scenario where a reader can choose *any snippet from any paper* and get a quick, context-aware explanation in the moment. This gap motivates our design of PaperExplainAgent as an on-demand, interactive assistant grounded in the user's current reading context. ## 2.5 Design principles for PaperExplainAgent

From the above insights, we distill several design principles that guide PaperExplainAgent:

- **Reduce Cognitive Load with Targeted Visuals:** Any visual or multimodal aid we add must serve a clear purpose in reducing the reader's cognitive effort. We adhere to the *coherence principle*digitallearninginstitute.com by avoiding superfluous decorations – every diagram or sketch should highlight a key relationship or make an abstract concept more concrete. The visual plan is there to scaffold understanding, not to distract or overwhelm.
- **Layered and Local Explanations:** Instead of dumping a full mini-tutorial, the system provides explanations in **layered chunks**. By separating intuition from formal details, we give readers the choice to consume just the high-level idea or dig into the nitty-gritty as needed. This layered format also aligns with the idea of "pre-training" the reader on key ideas before detailsdigitallearninginstitute.com. Crucially, the explanation stays *local* to the selected text – it doesn't drift off into unrelated tangents, and it's concise enough to be read alongside the paper.
- **Explicit Visual Planning:** We treat visual reasoning as a first-class part of the explanation. Even if the system doesn't render an image on the spot, the answer

explicitly includes a **Visual Sketch/Animation Plan** when appropriate, describing what kind of figure or animation would illustrate the text. By doing so, we encourage both the AI and the user to think in terms of diagrams and dynamic processes, not just algebra. The plan is written in a way that could be handed to a visualization tool (like Manim) or easily sketched by the user. This also serves educational value: it teaches the user *how to visualize* the concept.

- **Quick Interaction and Iteration:** The tool is meant to be used in an iterative reading session. Thus, we prioritize low latency and simplicity. Getting an explanation for a passage should be almost as easy as turning to a friend next to you and asking a question. The UI is kept minimal so that users can highlight another passage and ask a follow-up question if needed. This principle influenced our choice of model and infrastructure – we favor models that are light enough to run on a single GPU in a few seconds per query, enabling a smooth back-and-forth during reading.

By following these principles, PaperExplainAgent aims to integrate into the reading process as a helpful guide, providing the right amount of support at the right time, and doing so in a way that leverages both text and visuals for maximal understanding.

---

3 Design Goals

3.1 Task definition

We formalize the core task that PaperExplainAgent tackles as follows: Given a document D (a PDF research paper), a user-selected text span $s \subset D$ (for example, a theorem statement, a definition, a paragraph, or even a figure caption within the PDF), and an optional natural-language question q from the user about that span, the system must produce a structured explanation E that addresses q in the context of s. The explanation E is composed of multiple sections, typically

$$E = E_{\text{intuition}}, E_{\text{assumptions}}, E_{\text{details}}, E_{\text{visual plan}}$$

corresponding to an intuition or high-level summary, key assumptions or definitions, a detailed technical explanation, and a visual sketch/animation plan, respectively. Not every query will require all sections (for instance, a simple clarification question might only yield an intuition), but the *format* is standardized to encourage completeness and structure.

This task is distinct from a generic question-answering or summarization task because the input includes the *exact text from the paper* that the explanation should focus on. The assistant must remain grounded in that text span s, using it as the primary context. If s is a theorem, for example, E should explain that theorem (what it means, why it's interesting, how to understand its proof, etc.) rather than wander into unrelated topics. The natural language question q further specifies the user's need – it could be something like "What's

the intuition behind this theorem?" or "Why is the condition $n > 5$ needed here?" or even "Can you give an example of this definition in practice?". The system should tailor E to answer q (if provided) or give a general explanation of s if no specific question is asked.

Formally, we can think of the model's input as (D, s, q) and the output as a structured text E in a predefined schema. The challenge is to ensure E is **correct**, **helpful**, and **contextualized**. It should not introduce inaccuracies about the paper, and it should be framed in a way that a reader of this particular paper will find immediately useful (e.g., using the same notation as the paper and not re-deriving facts the paper already stated clearly). This necessitates that the system understand not just general math, but the local context in D around s. In our implementation, we include a bit of that local context (like the section title or preceding sentences) in the prompt to the model to aid disambiguation.

3.2 Target users and usage scenarios

We envision PaperExplainAgent being useful to a range of users in the scientific community, especially those working with mathematically dense literature. Primary target users include:

- **Graduate students and early-career researchers in mathematics, theoretical computer science, physics, and related fields.** These readers often encounter new concepts and tough proofs in papers and would benefit from on-demand clarifications and intuitions. For example, a PhD student reading a topology paper might highlight a particularly abstruse lemma and ask, "What is the intuition behind this lemma? Why might it be true?" to get a quick sanity-check explanation before diving into the formal proof.
- **Advanced undergraduates in proof-heavy courses or reading groups.** Students who are still building their mathematical maturity can use the tool as a study aid. If they get stuck on a definition in a textbook or a step in a proof, they can query an explanation targeted to that snippet.
- **Researchers crossing into a new subfield.** A computer scientist reading a physics paper or vice versa might not be fluent in the paper's idioms. They could highlight a paragraph and ask, "Can you rephrase this in simpler terms?" or "What background should I recall to understand this passage?" to quickly get up to speed. ## 3.3 Design constraints

In developing PaperExplainAgent, we had to navigate several design constraints, both conceptual and practical:

- **Relevance to Math Researchers:** The tool must produce explanations that are genuinely useful to mathematicians and scientists. That means the content of the explanations should be mathematically sound (no hallucinated theorems or false claims) and phrased in a way that respects the formality of the domain. It also means the system should handle notation correctly and be robust to LaTeX symbols, which are common in papers.

- **Rapid Prototyping and Leveraging Existing Tools:** Given the time constraints of development, we chose to build on existing open-source components where possible. For example, we utilized a PDF parsing library to extract text for the model and a lightweight web framework for the UI. The novelty lies in the integration and the prompting strategy, not in reinventing low-level components. This approach allowed us to assemble a working system quickly, focusing our efforts on the core explanation generation behavior.

From these constraints, we defined our main design goals:

- **G1: Layered, structured output.** The system should never just dump a single long paragraph. Every answer must be organized into clear sections (even if some sections are brief or omitted when not needed). This makes it easier for users to scan and find the particular type of information they need (be it a quick intuition or a detailed step).
- **G2: Visual reasoning as a first-class citizen.** Unlike standard QA bots, PaperExplainAgent should always consider if a visual or spatial explanation would help, and if so, include it explicitly. The *Visual Plan* is not an afterthought; it's part of the expected answer format. Our hypothesis (from Section 2) is that this leads to more engaging and comprehensible explanations.
- **G3: Context-anchoring.** Explanations must stick to the highlighted content. This means the model's prompt includes the exact text from the PDF selection and possibly the title of the paper and section, to ground the response. The output should reference ideas from that text (e.g., using the same notation f, U, ϵ as in the passage) rather than giving a generic encyclopedia answer. Essentially, the *paper remains in the loop* at all times.

With these goals in mind, we proceeded to design the architecture and components of PaperExplainAgent, as described in the next section.

---

4 System Design: PaperExplainAgent

4.1 High-level architecture

PaperExplainAgent's system architecture follows a client–server design that integrates an interactive front-end with a powerful AI back-end. The design is inspired by recent agent-based explanation frameworks like TheoremExplainAgent (TEA)arxiv.org and targets the need for multimodal, understandable explanations in STEM research (as highlighted by benchmarks such as TheoremExplainBench's 240-theorem dataset)huggingface.co. In broad strokes, the front-end React application collects user input (API credentials and a PDF document) and sends requests to a FastAPI server. The back-end then orchestrates a **generation pipeline**

that produces a structured explanation for the highlighted text span, optionally accompanied by a visualization (e.g. an animation). The architecture decouples the user interface from the heavy LLM computations, ensuring a smooth user experience even during lengthy explanation generation.

The overall workflow proceeds as follows:

1. **User Input:** The researcher opens the web-based front-end and provides their OpenAI API key along with uploading the PDF of the research paper. The front-end loads the PDF and lets the user highlight a specific passage (such as a difficult theorem, an equation, or a paragraph) that they want explained. The user then submits a question about that highlighted span through a simple UI prompt.

2. **Request to Back-end:** The front-end packages the query (including the exact highlighted text, the question, and the API key or token) and sends it to the back-end via an HTTP request (e.g. a POST to an `/explain` endpoint). This request serves as a job initiation, containing all information needed to generate an explanation.

3. **Job Handling:** Upon receiving the request, the FastAPI **backend** creates a new explanation job (assigning it a unique ID) and immediately responds to the front-end to acknowledge receipt. The heavy lifting is then done asynchronously: the job is handed off to a worker routine so that the main API thread remains responsive. As the job progresses, the backend keeps track of its **status** (e.g. "planning explanation", "generating visuals", "rendering video").

4. **LLM-driven Explanation Generation:** The backend invokes the *TheoremExplainAgent* pipeline (integrated as a Python module or via a subprocess call to `generate_video.py`) to actually produce the explanationarxiv.org. Under the hood, this pipeline uses two coordinated Large Language Model (LLM) agentsarxiv.org: a **planner agent** and a **coding agent**. The planner agent first interprets the highlighted text and the user's question, then formulates a high-level explanation plan comprised of multiple steps or "scenes." Each scene corresponds to a key aspect of the explanation (for example, outlining the intuition, explaining each part of a formula, or providing a concrete example). The planner refines these scene descriptions and may generate a narrated script for each part. Next, the coding agent takes each scene description and generates Python code (using the Manim library) to create accompanying visuals or animationsarxiv.orgarxiv.org. This step leverages prior work showing that agentic pipelines can successfully produce complex visualizations from textual instructionsarxiv.org. If a scene calls for a diagram or animation, the coding agent writes the code to draw it; if the scene is purely conceptual, it may not produce a visual and rely on text explanation only. Throughout this process, the LLM agents use the user-provided API key to access a model (e.g. GPT-4 or a similar powerful LLM) for generating the plan and code.

5. **Visualization Rendering:** Once the coding agent has produced code for all scenes, the back-end executes these code snippets to render the visuals. This is done using

> Manim, a Python toolkit for programmatic mathematical animationsarxiv.org. In parallel, the system uses a text-to-speech module to convert the narrated script into audio for the video's voiceover (using a model like Kokoro or an equivalent, as in TEA's setup). The output of this stage is a **compiled video** file (typically an `.mp4`) that animates the explanation with synchronized narration. Not every query will result in a long video – the system decides the appropriate modality based on the question. For simpler explanations, it might generate a static diagram or just structured text; for complex mathematical content, it will produce a step-by-step animated proof or illustrationarxiv.org. ## 4.2 Frontend and user interface

The front-end of PaperExplainAgent is a web application built with **React** (using the Vite toolchain for fast bundling) in **TypeScript**. The choice of React provides a responsive, dynamic UI, while TypeScript ensures type-safe interactions, reducing bugs in managing complex state (like PDF data and streaming job updates). Styling is done with plain **CSS**, enabling a clean and custom interface without heavy frameworks. The resulting UI is minimalist and focused, presenting the user with a two-pane layout that mirrors the structure of the task: on the left, the original paper; on the right, the explanation. This side-by-side design keeps the source material in constant view alongside the AI-generated insights, which is crucial for understanding and trustsodevelopment.medium.com. By always showing the relevant PDF segment next to the explanation, the system makes it easy for the researcher to verify claims and follow along – an approach known to bridge the gap between an LLM's answer and verifiable reasoning in the source textsodevelopment.medium.com. In other words, the interface itself provides a form of "grounding," assuring users that the explanation is anchored to the document they provided.

Layout and Interaction: The UI is divided into a PDF display area on the left and an explanation display area on the right. The left panel uses a PDF rendering component (e.g. Mozilla's PDF.js) to show the pages of the paper. Users can scroll through the paper and use their cursor to select (highlight) any span of text that they find challenging or want explained. Once a highlight is made, the interface prompts the user to enter a question or simply confirm that they want an explanation of that highlighted passage. For example, a user might highlight an equation or a paragraph and ask, "What does this result mean intuitively?" or "Can you explain how this step follows?" The act of highlighting effectively pins the context – the system knows exactly which text in the PDF the question refers to. This design ensures precision in the queries; it prevents ambiguous questions by tying them to a specific snippet of text.

After the user submits the query, the right panel becomes active. Initially, it might show a placeholder message or spinner along with status updates (fed by the back-end). For instance, as the back-end streams "planning explanation" or "rendering visuals," these messages are displayed in the explanation panel in real time. This feedback keeps the user engaged and informed during the potentially long generation process. Once the back-end finishes and returns the results, the explanation panel dynamically populates with the content.

Explanation Presentation: The explanation is presented in a **layered, structured format** for readability. Rather than a single monolithic block of text, the system breaks the explanation into logical sections and steps. Key portions of the answer are denoted with

clear headings or bolded lead-ins – for example, a "High-Level Idea" section might summarize the highlighted passage, followed by sub-sections like "Step 1: Background," "Step 2: Application of Theorem," and "Conclusion." These headings are generated based on the planner agent's scene breakdown, providing a natural hierarchy to the information. Adequate whitespace and **spacing** is used between paragraphs and sections so that the text does not appear dense or overwhelming. Bullet points or numbered lists are employed when enumerating a sequence of reasoning steps or factors, which makes it easier to scan and digest the explanation. This formatting echoes Mayer's **segmenting principle**, which states that people understand complex material better when it is split into smaller, coherent chunksdigitallearninginstitute.com. By segmenting the explanation into parts, the UI lets readers absorb one idea at a time, improving comprehension. We also apply a subtle form of **progressive disclosure**: the most essential explanation (e.g. the intuitive summary) is shown first, with additional technical details and optional visual content appearing subsequently or upon user interaction. Such a layered reveal of information helps manage cognitive load for users – they see the big picture before delving into details, in line with UX best practices for complex contentinteraction-design.org.

Crucially, the original PDF text remains visible in the left panel while the explanation is read on the right. This adheres to the **spatial contiguity principle** from multimedia learning theory: placing related text and graphics close together helps learners make connections without having to mentally bridge a gapdigitallearninginstitute.com. In our context, the "graphic" is the PDF itself (with the highlighted segment), and the explanatory text is next to it. A user can easily glance back and forth between the paper and the explanation. For instance, if the explanation refers to "the equation above" or "that assumption," the user can see that reference in the PDF immediately. This side-by-side arrangement not only boosts understanding but also trust, since the user can verify each part of the explanation against the source materialsodevelopment.medium.com. Prior research in HCI and explainable AI has noted that users feel more confident in AI-generated answers when they can trace those answers back to the original evidencesodevelopment.medium.com. Our interface capitalizes on this insight by literally highlighting the evidence (in the PDF) as the AI explains it.

Support for Visual Explanations: To complement the textual explanation, the UI integrates visual outputs when available. If the back-end generated an **animation or diagram** (for example, a Manim video illustrating a geometric argument or a plot of a function mentioned in the text), the front-end will display it in the explanation panel. Typically, the video is embedded at an appropriate point in the explanation – for instance, after the textual description of a concept, a small video player might appear allowing the user to play/pause the illustrative animation. Users can thus read the explanation and simultaneously watch the visualization, or do either independently, according to their preference. This multi-modal presentation aligns with the **multimedia principle** – the idea that combining words and pictures (or in this case, narrated animations) can significantly improve learning outcomes compared to text alonedigitallearninginstitute.com. Especially for STEM content, visuals can make abstract concepts more concrete and reveal relationships that are hard to grasp from text alone. As noted by prior work, many mathematical or scientific concepts are best understood through diagrams or dynamic representationsarxiv.org. By providing an optional video, we cater to visual learners and also reinforce the explanation: the text de-

scribes "what and why," while the video shows "how" through motion and imagery. Early user feedback (informal) suggests that having both modalities readily accessible deepens understanding and keeps users more engaged. Moreover, the video includes the **voiceover narration**, effectively turning the explanation into a mini-lecture – the user can listen and watch, which engages dual channels of cognition (auditory and visual), potentially boosting retentionresearchgate.net. It's important to note that the visual element is an enhancement: the explanation text itself is sufficient to answer the question, and the video is there for clarification and additional insight. Users short on time might skip the video, whereas users who want to **explore the concept in depth** will find it valuable. This optionality is another form of layered user experience: advanced content is available on demand, without cluttering the basic answerinteraction-design.org. ## 4.3 Backend explanation engine

The "brain" of PaperExplainAgent is the backend explanation engine. It is responsible for taking the user's query and the selected text and producing the structured answer.

Prompt Construction: We craft the prompt to the LLM carefully to induce the desired output format. The prompt template (in pseudocode) looks roughly like:

```
You are PaperExplainAgent, an assistant for explaining math papers.
Here is an excerpt from a paper:
"[the selected text s]"
Question: "[the user's question q]"
Provide a structured explanation with the following sections:
Intuition - (a high-level intuitive explanation)
Key Assumptions - (list any definitions or assumptions needed)
Detailed Explanation - (detailed, technical explanation)
Visual Sketch/Animation Plan - (a description of a visual illustration)
```

We include a few examples in the prompt (in a few-shot manner) during fine-tuning to solidify this format. At runtime, for each query, we insert the actual selected text and question. We also sometimes append the title of the paper or the section header as additional context like "(This excerpt is from a paper titled 'On the Stability of Frobnications', Section 3: Main Results)", to let the model know the general topic. This helps in case the excerpt is ambiguous or uses notation that might be defined elsewhere in the paper.

Model Inference: The prompt is fed to our fine-tuned LLM, which generates an output. The model was trained to produce answers in markdown format, with section headings like "**Intuition:** …" etc., or a JSON with fields (we experimented with both approaches). We found that instructing the model to produce a markdown with clear headings made it easier to display directly. During inference, we also set some constraints: e.g., a max token limit so it doesn't ramble too long (we aimed for answers usually under 300 tokens, though we allowed up to ~500 if needed for very complex queries).

We also added a few safety/correctness instructions in the prompt, such as: "If the question asks for a proof, provide a sketch but *do not* fabricate full proofs from scratch. If you are unsure about the passage, say you are unsure." This is to mitigate hallucinations. The model, being fine-tuned on real theorem explanations, typically had a good prior to stick to

the content of s, but we wanted extra caution especially for factual accuracy. For instance, if s is a theorem statement, we don't want the model to accidentally state something contrary to the theorem.

Output Parsing: Once the model returns text, the backend parses it into the structured fields for the front-end. Because the model output is in a predictable format (markdown headings), parsing is straightforward: we look for known section keywords (we made them consistent and capitalized to be easy to detect). If a section is missing, we flag it. In the UI, we might omit a section header if the content was empty or say "*(No visual sketch provided.)*" in the visual section if the model didn't output one. In practice, our fine-tuned model almost always produces all sections, sometimes with a short placeholder like "None needed" for assumptions if not applicable.

We also ensure any LaTeX in the output (e.g., formulas) is passed through properly so it renders in the front-end (we use a MathJax plugin in the UI for that). This way, if the model restates a formula or gives an example equation, it appears nicely formatted.

The structured format of the output is intentionally aligned with the cognitive principles discussed in Section 2. By separating intuition from details, we help manage the user's cognitive load – they see the gist first, then can delve deeper. By isolating assumptions, we make sure the reader is reminded of definitions up front (akin to Mayer's *pre-training principle* where learners do better if they're introduced to key terms before the main lessondigitallearninginstitute.com). And by explicitly having a visual section, we ensure that modality is part of the conversation. ## 4.4 Visual explanation planning

A unique feature of our system is the *Visual Sketch / Animation Plan* section of the output. Rather than directly generating an image, the LLM describes what visual aid would be helpful. The format for this in the output is usually a short paragraph or a sequence of steps. For example, for a graph theory theorem, the plan might say: "Draw a sample graph with 5 nodes illustrating the property. Highlight the subset of nodes that form the separator. Then show that removing them disconnects the graph into two components." For an analysis concept, it might suggest: "Imagine the function's graph: first plot f(x), then show an ϵ-band around L=0 and a δ range on the x-axis, illustrating that beyond some N, f(x) stays within the band."

We opted for a textual plan for several reasons:

- **Flexibility:** By keeping it textual, the same plan can be interpreted by a human (the reader might sketch it on paper or just visualize it mentally) or by a program. We considered automatically generating a Manim animation from the plan. In fact, our backend is set up such that if the visual plan follows a certain structured pseudocode, we could feed it to a rendering module. Due to time constraints, we did not fully implement auto-rendering, but we designed the plan format with that in mind (e.g., a list of drawing steps).
- **Avoiding Bad Images:** Generating diagrams or plots automatically (via either an AI image model or programmatically) is non-trivial and could produce misleading or cluttered images if not done carefully. Given our focus on not adding extraneous

> cognitive load, we decided it's safer to propose a visualization than to show one that might be suboptimal. A future iteration could refine this by having a library of premade diagrams for common concepts, or by using a vetted generator.

In the current prototype, the visual plan serves as a scaffold. We've observed in user feedback that even the description of a picture can help – it prompts the reader to form a mental image. It's akin to how a textbook might say "(see Figure 2 for a depiction)" even if one initially only reads the caption. It sets the stage for a visual mode of thinking.

For those cases where we did integrate with a visualization backend (in a separate experiment), the pipeline was: the LLM outputs a plan in a semi-structured form (for instance, a JSON with a list of "scenes" or a simple custom DSL for drawings). A small script parses that and uses a tool (Manim, matplotlib, etc.) to generate frames or an animation. This was done offline for a couple of examples to test feasibility. One example was a calculus limit: the model output a plan with steps "draw function curve; mark L on y-axis; draw horizontal band for epsilon; show vertical line for N; highlight portion where curve is within band." We manually mapped those to a Manim script and it produced a nice 15-second animation. While we didn't integrate this fully due to time, it shows the potential for a near-future extension: an interactive version where clicking the "Visual Sketch" section could render the described image or play an animation.

To summarize, the visual explanation planning in PaperExplainAgent is currently *implicit* (textual description) but is built with an eye toward *explicit* visualization generation. Even as text, it makes the explanations richer. Our evaluation will touch on whether users found these descriptions useful, and our discussion will explore how automating this part could further enhance the reading experience.

References

[1] Max Ku, Thomas Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. 2025. **TheoremExplainAgent: Towards Video-based Multimodal Explanations for LLM Theorem Understanding.** *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, ACL, 2025. (arXiv:2502.19400).

[2] Andrew Head, Amber Xie, and Marti A. Hearst. 2022. **Math Augmentation: How Authors Enhance the Readability of Formulas using Novel Visual Design Practices.** *CHI Conference on Human Factors in Computing Systems (CHI '22)*, pages 1–18. ACM, 2022.

[3] Richard E. Mayer. 2009. **Multimedia Learning (2nd ed.).** Cambridge University Press, New York, NY, USA.

[4] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2024. **Measuring Multimodal Mathematical Reasoning with the MATH-Vision Dataset.** *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, 2024.

[5] Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023. **LLaMA 2: Open Foundation**

and Fine-Tuned Language Models. *arXiv preprint* arXiv:2307.09288, 2023.