AI-Powered Math Solver: Image-to-Equation Recognition with Stepwise Explanations and Visualizations

Team 48: Aritrajit Roy (62), Rudra Narayan Shaw (222), Rwiddhit Chatterjee (225), Swapnomon Murari (304), Vedika Anand Thakur (321)

Mentor: Srijit Mukherjee

Period of Internship: 19th May 2025 - 15th July 2025

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata

AI-Powered Math Solver: Image-to-Equation Recognition with Stepwise Explanations and Visualizations

Aritrajit Roy, Rudra Narayan Shaw, Rwiddhit Chatterjee, Swapnomon Murari, Vedika Anand Thakur

Abstract

In today's rapidly evolving educational landscape, students are increasingly expected to interact with diverse formats of mathematical content. From handwritten classroom notes to printed textbooks and worksheets. However, current digital tools for mathematical problem-solving often fall short in supporting this variety. Most existing equation solvers are limited to typed, well-formatted inputs and fail to accommodate the more common, everyday scenarios students face. Moreover, these tools typically prioritize providing quick answers over fostering deep understanding. They rarely offer pedagogical explanations, stepwise guidance, or visual support, leaving learners with correct results but minimal insight into the underlying problem-solving process. Leveraging cutting-edge optical character recognition (OCR) through Pix2Text, the system can convert complex visual math inputs captured via camera or scan into structured, machine-readable expressions. These expressions are then processed using symbolic computation frameworks such as SymPy, which perform algebraic simplification, factorization, and equation solving. Crucially, the solver is enhanced with large language models (LLMs), which provide natural language explanations and breakdowns of each step in the solution, alongside visual aids like annotated equations and graphs. This combination of OCR, symbolic math engines, and natural language processing ensures that the system is not only powerful in computation but also educational in presentation. Students can capture problems directly from their notebooks or books, understand each transformation in a solution sequence, and build conceptual clarity. To validate the effectiveness of the proposed solver, it was tested on a broad range of mathematical inputs, including hand-drawn equations, printed textbook problems, and scanned worksheets. The evaluation focused on three key metrics: accuracy of interpretation, correctness of computation, and clarity of explanation. Results indicate a high degree of robustness across input types, particularly in solving polynomial equations, systems of linear equations, and basic algebraic manipulations. Furthermore, the generated explanations were rated favorably by both students and educators in terms of helpfulness and clarity. By bridging the gap between input flexibility and pedagogical transparency, this system sets a new standard for educational technology tools in mathematics.

1. Introduction

Mathematical education in the digital age increasingly relies on intelligent tools to support learning and enhance accessibility. Despite these advancements, students and educators continue to encounter substantial difficulties when working with handwritten or image-based mathematical content. The limitations of conventional OCR systems become especially apparent when interpreting symbolic notations such as fractions, integrals, and matrices. Even when successful in recognizing such content, most existing solvers tend to function as black boxes: they provide answers without transparency, skip intermediary steps, and offer minimal instructional value. This creates a disconnect between computation and comprehension, particularly for learners who need guided, step-by-step support to master foundational concepts. The motivation behind this project is rooted in addressing a crucial pedagogical gap. The need for educational tools that not only solve equations but also explain them clearly and intuitively, much like a human tutor would. The envisioned solution emphasizes accessibility, interpretability, and depth of explanation. It is designed to support image-based and handwritten inputs, enabling students to capture problems directly from notebooks, printed worksheets, or whiteboards. From there, it performs structured symbolic analysis, walks the user through each solution step, and reinforces understanding with visualizations, contextual insights, and practice problems where appropriate. The system combines modern OCR techniques, symbolic computation libraries, and natural language processing to provide a well-rounded educational experience. A distinctive aspect of this work lies in its collaborative and exploratory development approach. Rather than converging immediately on a single shared implementation, all of the authors independently developed their own versions of the hybrid-input equation solver. Each version experimented with different components, ranging from variations in OCR pipelines and parsing strategies to alternative explanation formats and user interfaces. This parallel development process allowed the authors to test diverse ideas, compare performance and usability, and ultimately distill the most effective elements into the final system design. It also ensured that multiple perspectives were incorporated into the tool's development, fostering robustness, adaptability, and innovation. This paper presents the culmination of those efforts: a unified, intelligent system that accepts both handwritten and printed mathematical inputs, solves the equations symbolically, and generates intuitive, stepwise explanations with natural language and visual aids. By prioritizing comprehension alongside correctness, the project aims to set a new standard for educational technology in mathematics, one that truly teaches, rather than simply answers.

2. Objectives

The primary objective of this project is to develop a comprehensive, AI-powered educational system capable of interpreting, solving, and explaining mathematical expressions from a variety of input formats. Designed with pedagogical utility and accessibility in mind, the system aims to bridge the gap between problem-solving and conceptual understanding. Specifically, the project sets out to:

1. Support hybrid input modes by accepting both typed text and image-based inputs, including handwritten or printed equations captured via camera or scan. This ensures that learners can interact with the system naturally, without being restricted to strictly formatted digital input.

2. Utilize robust OCR technologies to extract symbolic mathematical content from images accurately. Special emphasis is placed on handling complex mathematical notations such as fractions, exponents, matrices, and Greek symbols—elements that traditionally pose challenges to general-purpose OCR engines.

3. Perform symbolic parsing and resolution using advanced symbolic computation tools, primarily SymPy. The system is capable of simplifying, factorizing, and solving polynomial and linear algebraic equations, while preserving exact symbolic integrity rather than relying on numerical approximations.

4. Deliver natural language explanations that guide the user through each solution step. These explanations are supported by relevant mathematical theorems and reasoning, mirroring the behavior of a human tutor to foster a deeper understanding.

5. Enhance visualization of results through dynamically generated graphs, annotated equations, and step-maps. These visual aids are particularly effective in helping students grasp abstract mathematical concepts such as function behavior, root multiplicity, and geometric interpretation.

6. Generate practice problems structurally similar to the original input, allowing students to reinforce their understanding through repetition and variation. This feature aims to transform passive consumption of answers into active engagement with the material.

In addition to these core functional goals, the project also includes a systematic limitation analysis of the OCR libraries considered and implemented during development. This analysis evaluates the performance of each library—including Pix2Text, MathPix, and other contenders—in terms of accuracy, speed, and reliability across a wide range of equation types and image qualities. By documenting the shortcomings and strengths of each option, the team was able to make informed decisions about the optimal OCR pipeline and highlight areas for future improvement. Taken together, these objectives aim to produce not just a solver, but a truly educational companion—one that prioritizes interpretability, accessibility, and student-centered learning in the digital study of mathematics.

3. Related Work

The integration of artificial intelligence into mathematics education has led to a variety of tools aimed at automating problem-solving. However, most of these systems tend to optimize for answer retrieval rather than educational transparency, and they often fall short when it comes to handling handwritten or image-based inputs. Traditional OCR systems, such as Tesseract, are built primarily for recognizing natural language text. While effective for document digitization and general-purpose scanning, they struggle significantly with symbolic mathematical expressions. Their difficulty lies in the lack of structure-awareness—mathematical notation involves spatial relationships (e.g., superscripts, fractions, matrices) that are not adequately modeled in standard OCR engines. More recently, domain-specific OCR tools like Pix2Tex have been developed to tackle the unique challenges of math recognition. These models use encoder-decoder architectures inspired by image captioning systems, where a visual input is mapped to a LaTeX or symbolic string. Pix2Tex offers substantially higher accuracy in extracting structured mathematical content, especially from printed and typeset sources. However, this precision comes at the cost of heavy computational resource requirements, making real-time or on-device deployment difficult in lower-end educational environments. On the educational front, commercial applications like Photomath and Microsoft Math Solver are popular among students for their ease of use and instant answer delivery. While these tools provide a convenient means of solving standard problems, they typically focus on the final solution and offer limited pedagogical value. Explanations, if present, are often shallow and templated, lacking customization based on student input or learning progress. Moreover, these platforms rarely support practice generation, a feature critical to reinforcing understanding through repetition and variation. In contrast, our work aims to combine the strengths of existing solutions while addressing their limitations. We build upon reliable symbolic mathematics engines like SymPy and integrate them with Pix2Text for high-accuracy equation extraction. To bridge the explanation gap, we incorporate large language models (LLMs) that generate rich, context-sensitive, step-by-step explanations in natural language, simulating a tutor-like experience. Furthermore, the system introduces visualization modules for graphing and highlighting critical mathematical properties, as well as practice problem generators that adapt to the structure of the original input. Importantly, our project also includes a comparative limitation analysis of various OCR libraries used during development. This investigation informed the design of a modular pipeline that balances accuracy, computational efficiency, and scalability, ensuring that the final tool is both powerful and practical in real-world educational settings. Through this synthesis of modern OCR, symbolic computation, natural language processing, and educational design, our approach advances the current state of intelligent tutoring systems for mathematics.

4. Methodology

The methodology of this project is centered around a modular, multi-stage pipeline that transforms user input into interpretable mathematical solutions enriched with visual and linguistic explanations. Each stage employs specialized technologies for symbolic processing, visual rendering, and pedagogical communication. A crucial component of the project also includes a comparative analysis of OCR technologies to identify their strengths and limitations when dealing with mathematical notation. The following sections describe each stage of the pipeline in detail:

4.1 Input Handling

The system accepts user input in two primary modes. Text Input: Users may directly type equations using standard mathematical syntax. Before parsing, inputs undergo regex-based sanitization to normalize symbols and correct common user formatting errors. The cleaned string is then converted into symbolic form using SymPy. Image Input: For handwritten or printed content, the input image is passed through the Pix2Text OCR pipeline, which converts the visual content into LaTeX-like expressions. Post-processing includes custom regex corrections to ensure compatibility with symbolic computation tools (e.g., replacing ambiguous characters, resolving spacing issues, and disambiguating operators).

4.2 Symbolic Computation

Once an expression is in symbolic form, it is processed using SymPy, an open-source library for symbolic mathematics. Polynomials are interpreted using Poly for structured algebraic manipulation. The system applies simplification, factorization, and root-finding algorithms to return exact symbolic results. Linear systems are parsed using sympify and solved using solve, which can output solutions in closed form or numerical approximations as required. This stage ensures that all computations maintain mathematical rigor, making them suitable for both exact and pedagogical purposes.

4.3 LLM-Based Explanation

The LLM-based explanation module leverages the API of Mistral AI to generate detailed, natural-language explanations of mathematical solutions. To guide the model's reasoning, a custom YAML database was created, containing a collection of key theorems related to polynomial solving—such as the Factor Theorem, Remainder Theorem, and Rational Root Theorem. During execution, the system constructs structured prompts that include LaTeX-formatted solution steps alongside references to relevant entries from this theorem database. These prompts are then sent to Mistral's hosted model via API, which returns step-by-step explanations designed to mirror the clarity and reasoning style of a human tutor. This integration ensures the output is not only correct but also educationally meaningful, linking each computational step to its underlying mathematical principle.

4.4 Visualization

To support visual intuition, the system generates graphical representations of mathematical solutions: Polynomials are plotted over a defined real interval using Matplotlib and NumPy. Real roots are highlighted, and inflection points are marked to demonstrate the function's behavior. Linear systems are visualized as intersecting lines, with solutions annotated directly on the graph. Axes are labeled, and color schemes are applied to differentiate variables and equations. These plots are dynamically generated and embedded alongside explanations, helping learners connect symbolic operations with geometric interpretations.

4.5 OCR Limitation Analysis

Given the centrality of OCR to the hybrid-input design, a thorough limitation analysis was conducted across multiple OCR systems, including Pix2Text, MathPix, and Tesseract. The evaluation criteria included: Accuracy of symbol recognition across diverse inputs (e.g., clean print, noisy scans, cursive handwriting). Structural parsing ability, how well the tool recognized spatial arrangements like superscripts, fractions, and multi-line equations. Speed and computational load, particularly on lower-end hardware. Error patterns, such as misreading variable names, conflating operators (e.g., - vs. –), or misinterpreting Greek letters.

5. System Architecture

The system is designed using a modular architecture, allowing each component to function independently while seamlessly interacting with others. This modular design enhances flexibility, scalability, and maintainability, enabling developers to update or replace components without disrupting the overall pipeline. The primary goal of the architecture is to support a user-friendly, hybrid-input equation-solving system that not only computes results but also teaches the reasoning behind them. The core components of the system include:

5.1 OCR & Parsing Module

Technologies Used: Pix2Text, Regular Expressions (Regex)

Role: This module handles the image input mode, where users upload handwritten or printed equations. The image is processed through Pix2Text, a transformer-based OCR system trained specifically on mathematical notation. The raw LaTeX-style output is then cleaned and normalized using custom regex patterns to remove noise, correct formatting inconsistencies, and prepare it for symbolic parsing.

Output: A cleaned symbolic expression in a format compatible with SymPy.

5.2 Symbolic Computation Engine

Technology Used: SymPy

Role: Once an expression is parsed, it is handed over to SymPy for symbolic manipulation. For polynomials, the engine uses the Poly object to structure expressions, perform simplification, factorization, and root-finding. For linear systems, it uses sympify and solve to compute solutions symbolically or numerically, depending on the context.

Output: A structured solution consisting of ordered steps in symbolic format.

5.3 LLM-Based Explanation Module

Technology Used: Hosted Large Language Model (LLM), Theorem-Aware Prompt Generator

Role: To transform the raw symbolic steps into readable and instructive content, this module interfaces with a large language model. A specially crafted prompting strategy incorporates LaTeX-formatted equations and references to relevant theorems or strategies (e.g., factor theorem, substitution method). The LLM generates a step-by-step, natural-language explanation that aligns with educational practices.

Output: Clear and pedagogically sound explanations mimicking the tone and structure of a human tutor.

5.4 Graphing and Visualization Module

Technologies Used: Matplotlib, NumPy

Role: Visual representation enhances conceptual understanding. This module takes symbolic results and creates dynamic visualizations: For polynomials, it generates real-valued plots with critical points (e.g., roots, turning points) annotated. For linear systems, it displays graphs of equations as lines and highlights intersection points.

Output: Intuitive, annotated visualizations presented alongside symbolic and textual results.

5.5 Gradio-Based Frontend Interface

Technology Used: Gradio

Role: The frontend provides a clean and interactive web-based UI. It includes tabbed sections for: Text Input, Image Upload, Stepwise Solution View, Graph View, Explanation Console. The interface allows users to input problems, see solutions evolve in real time, and interact with different outputs through a seamless experience.

Output: Real-time interaction and immediate feedback via browser.

5.6 OCR Limitation Analysis Module

Role: During development, a dedicated diagnostic module was implemented to analyze and benchmark multiple OCR systems including Pix2Text, MathPix, and Tesseract. This module evaluated: Recognition Accuracy on various input types (handwritten, printed, noisy). Parsing Fidelity—correct detection of spatial structure, brackets, and layout. Performance Overhead on standard and low-resource devices. Common Error Patterns, which informed preprocessing strategies.

Outcome: Pix2Text was chosen for deployment due to its superior performance in structure-aware equation recognition, despite its higher computational demand.

5.7 System Workflow Overview

1. User provides an equation via typed text or uploaded image.

2. Image input is processed through OCR & regex.

3. Parsed symbolic expression is passed to the SymPy engine.

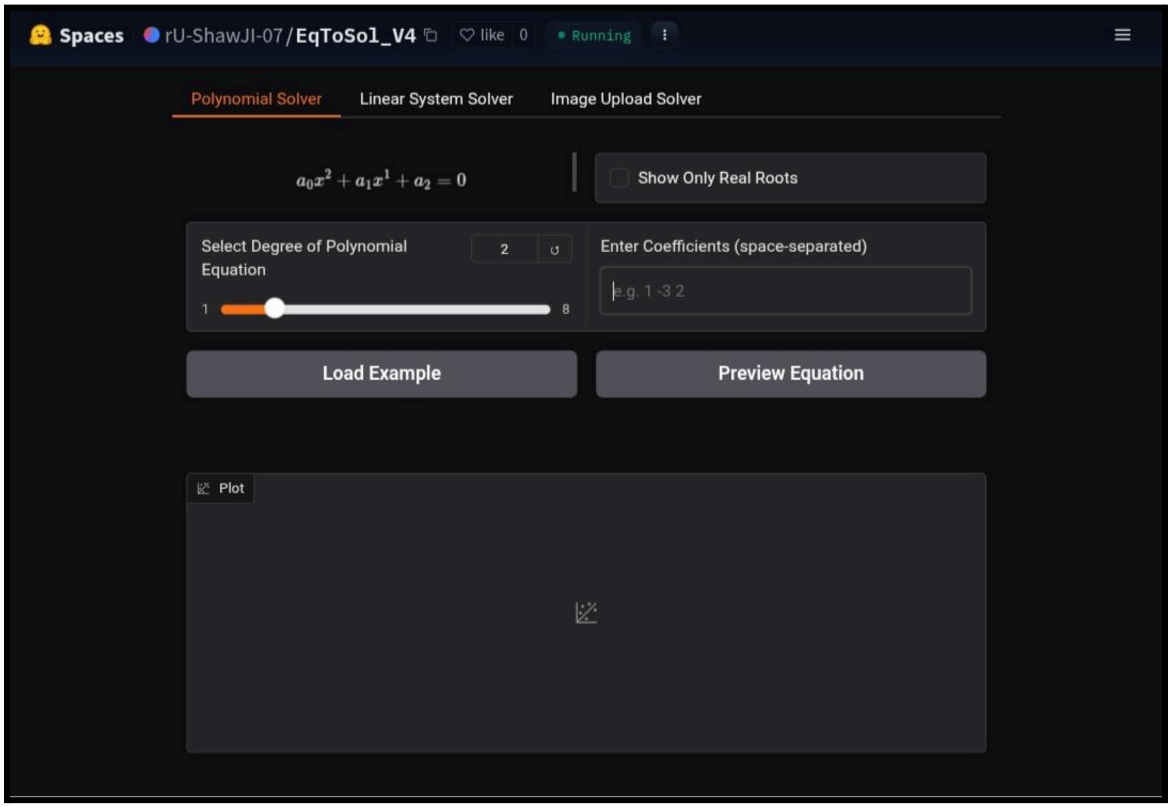
4. Resulting steps are formatted and sent to the LLM module for explanation.

5. Simultaneously, the Graphing module generates relevant plots.

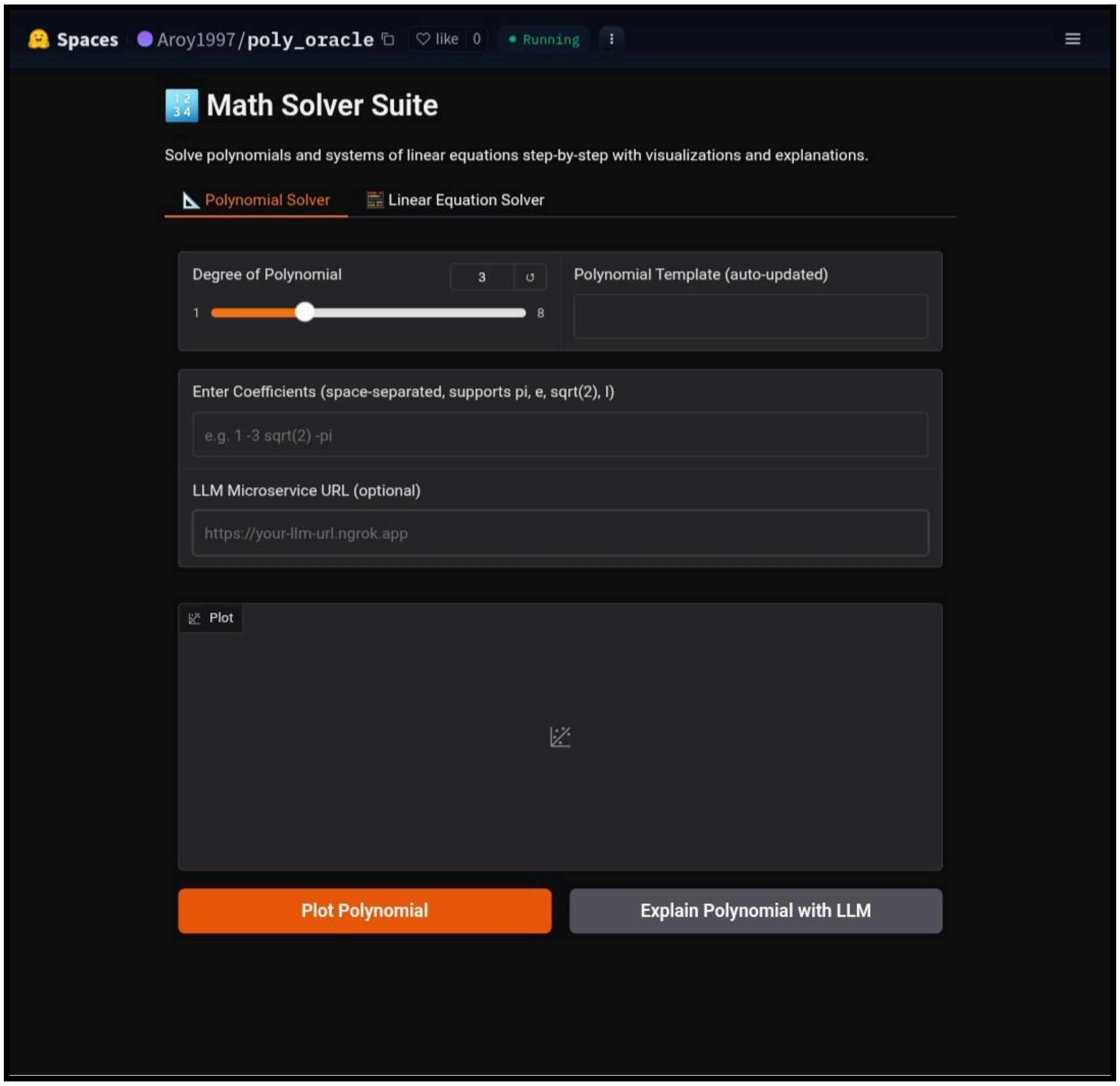
6. All outputs are displayed via the Gradio interface.

This modular architecture allows the system to function as more than a mere equation solver—it becomes an interactive learning platform, merging symbolic computation with explainable AI and visualization. Following are the pictures of all the versions made by the team.

Rudra's version.



Aritrajit's version.



🔢 Math Solver Suite

Solve polynomials and systems of linear equations step-by-step with visualizations and explanations.

Degree of Polynomial: 3 (1–8)

Polynomial Template (auto-updated)

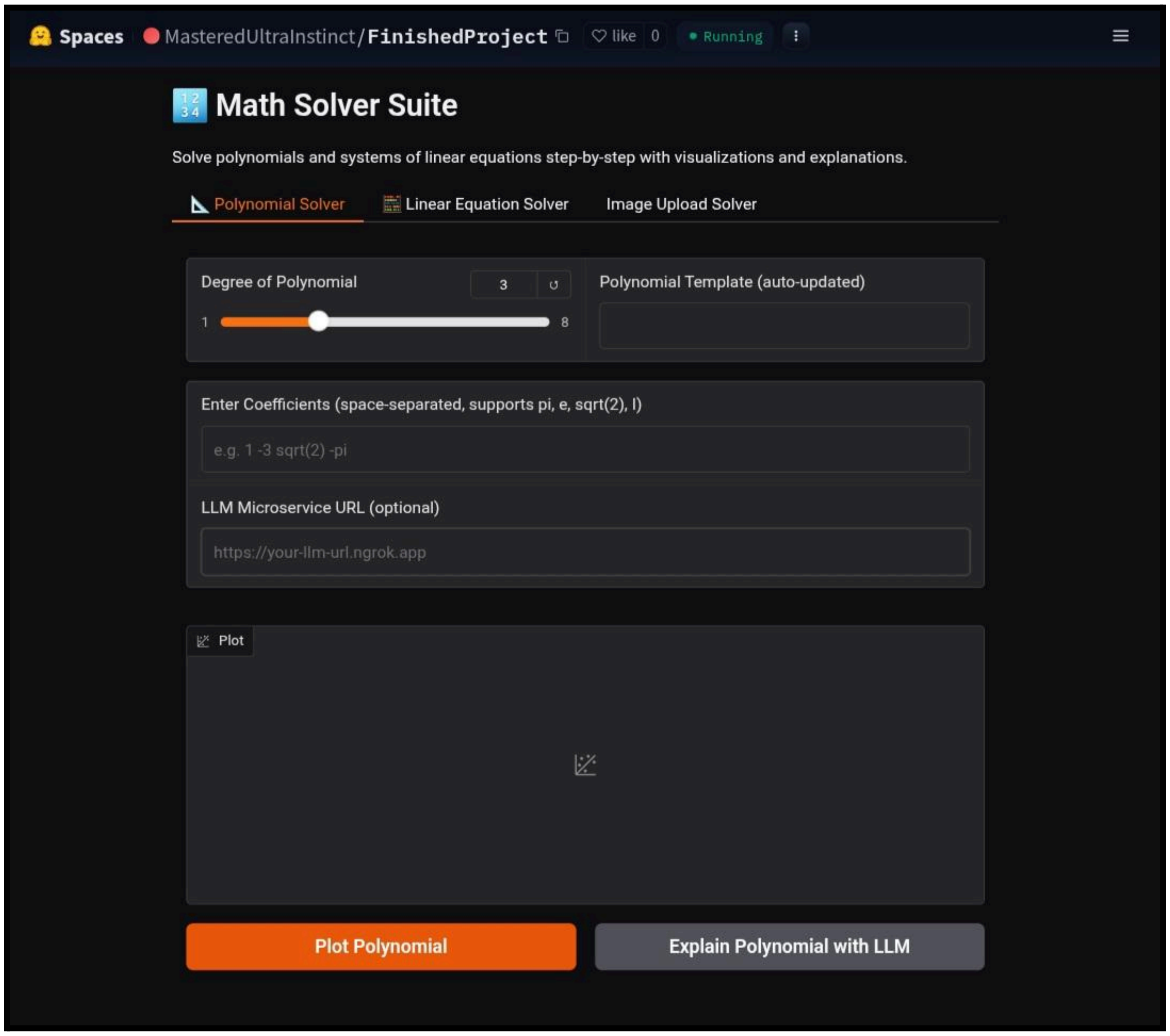
Enter Coefficients (space-separated, supports pi, e, sqrt(2), I)

e.g. 1 -3 sqrt(2) -pi

LLM Microservice URL (optional)

https://your-llm-url.ngrok.app

Swapnomon's version



Math Solver Suite

Solve polynomials and systems of linear equations step-by-step with visualizations and explanations.

Degree of Polynomial: 3 (1–8)

Polynomial Template (auto-updated)

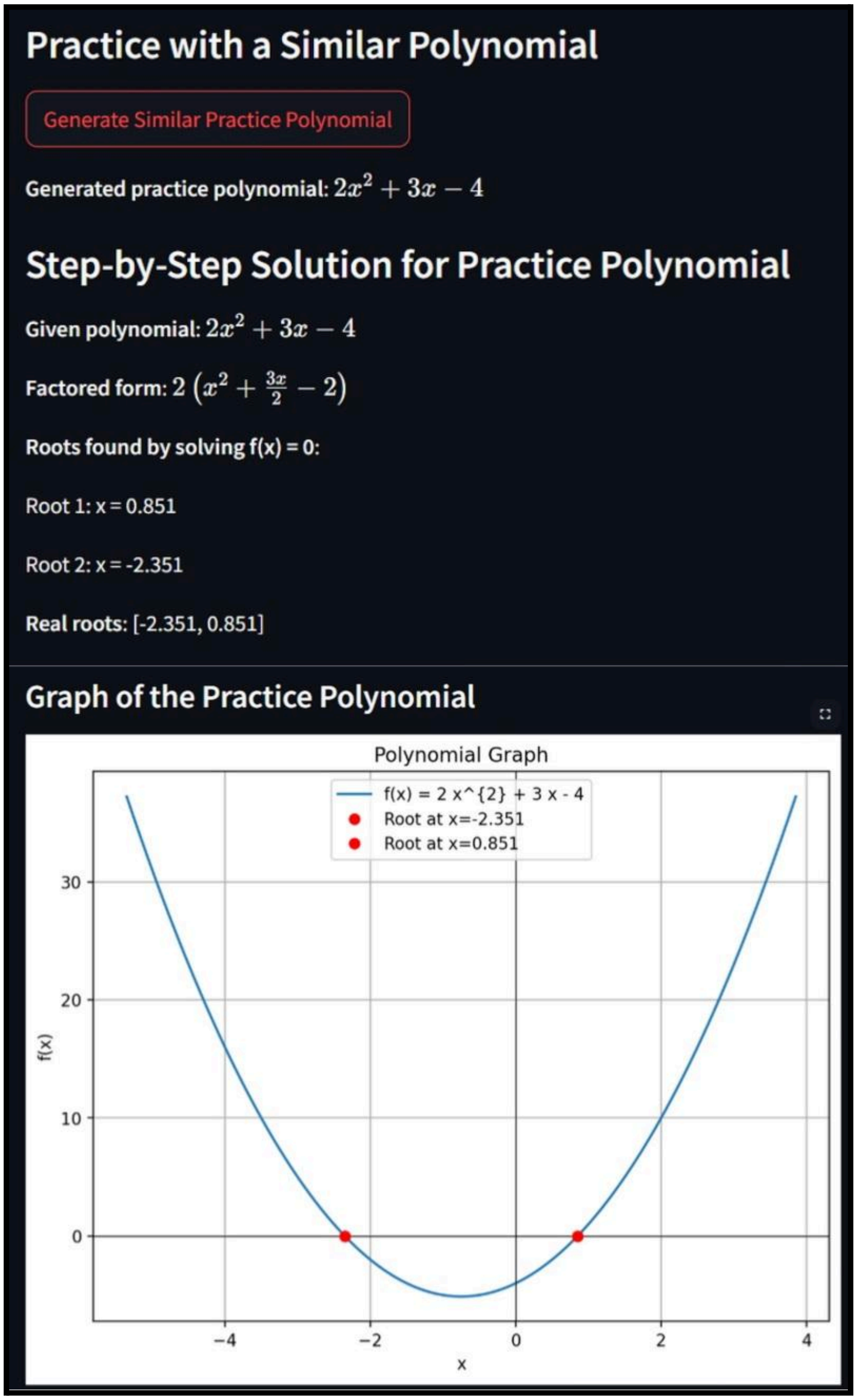
Enter Coefficients (space-separated, supports pi, e, sqrt(2), I)

e.g. 1 -3 sqrt(2) -pi

LLM Microservice URL (optional)

https://your-llm-url.ngrok.app

Vedika's version



Practice with a Similar Polynomial

Generated practice polynomial: $2x^2 + 3x - 4$

Step-by-Step Solution for Practice Polynomial

Given polynomial: $2x^2 + 3x - 4$

Factored form: $2\left(x^2 + \frac{3x}{2} - 2\right)$

Roots found by solving f(x) = 0:

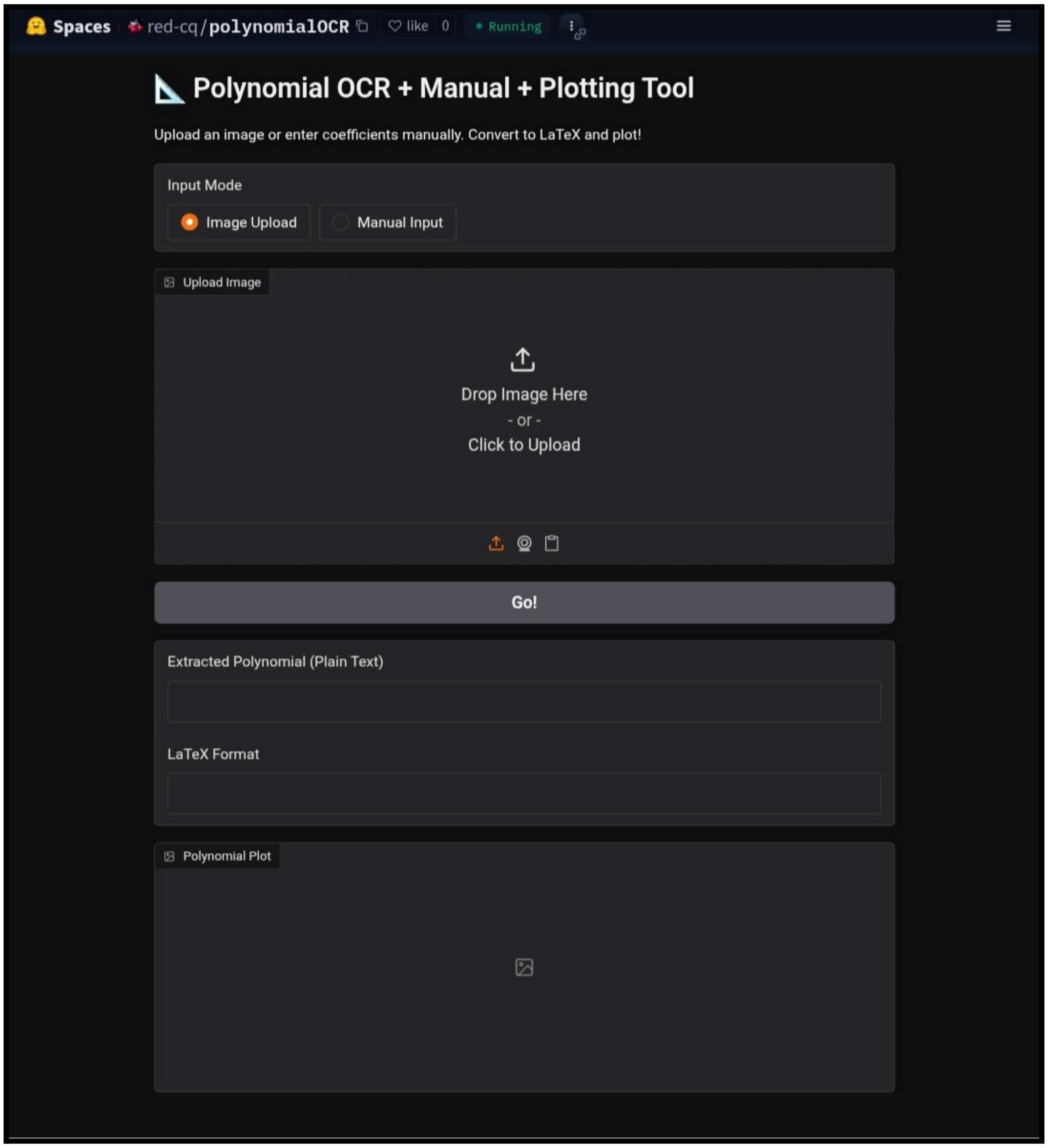
Root 1: x = 0.851

Root 2: x = -2.351

Real roots: [-2.351, 0.851]

Graph of the Practice Polynomial

Rwiddhit’s version



Spaces red-cq/polynomialOCR like 0 Running

📐 Polynomial OCR + Manual + Plotting Tool

Upload an image or enter coefficients manually. Convert to LaTeX and plot!

Input Mode

[✓ Image Upload] Manual Input

Extracted Polynomial (Plain Text)

LaTeX Format

6. Experimental Setup and Evaluation

To test system robustness, we prepared a dataset of 100 equations:

- 50 handwritten
- 30 typed
- 20 scanned

Equations include polynomial expressions (degree 1–5) and linear systems (2×2). Each component—OCR, solver, LLM, and graphing—was tested individually and in an end-to-end setup.

Evaluation Summary

Dimension	Purpose
OCR Accuracy	Test Pix2Text's ability to extract LaTeX from diverse math images
Solver Validity	Ensure SymPy returns mathematically correct, verifiable solutions
LLM Explanation Quality	Assess clarity, relevance, and theorem usage in generated explanations
Graph Accuracy	Match the visual plot to the correct analytical behavior of equations

7. Limitations

Despite its utility, the system has the following constraints:
• OCR accuracy is reduced on low-quality or cursive handwriting.
• The LLM requires hosting; there is no offline fallback mechanism.
• Linear system support is limited to 2×2.
• Visualization is limited to 2D graphs.
• No tracking of student progress or adaptive learning features.
• Gradio frontend is not production-grade (no database, auth, etc.).

Models	Exponents	Variables	Constants	Operators	Coefficients	No extra text	Avg
Tesseract	0.27	0.56	0.63	0.03	0.48	0.67	0.44
LatexOCR	0.72	0.77	0.8	0.69	0.73	0.5	0.70
Pix2Text (Vedika's version)	1	1	1	0.98	0.99	0.97	0.99
Pix2Text (Rudra's version)	1	1	1	1	1	1	1

8. Conclusion

This paper presents an AI-powered tool that interprets and explains algebraic problems from both typed and image input. By integrating OCR, symbolic computation, LLMs, and visualization, the system provides students with an interactive and transparent mathematical assistant. Future work will focus on expanding its capabilities to support multi-variable systems, user analytics, adaptive difficulty, and scalable deployment.

Citations and Links

The following are the links to all the versions created by the authors.

Aritrajit's version: https://huggingface.co/spaces/Aroy1997/poly_oracle

Rudra's version: https://huggingface.co/spaces/rU-ShawJI-07/EqToSol_V4

Vedika's version: https://github.com/VedikaThakur/Solvify/tree/main

Swapnomon's version: https://huggingface.co/spaces/MasteredUltraInstinct/FinishedProject

Rwiddhit's version: https://huggingface.co/spaces/red-cq/polynomialOCR

Luo, J., Zong, Y., Zhang, L., Wu, J., & Liu, Y. (2022). Pix2Text: Screenshot OCR with Transformers. arXiv preprint arXiv:2209.07625. https://arxiv.org/abs/2209.07625

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., ... & Granger, B. E. (2017). SymPy: Symbolic computing in Python. PeerJ Computer Science, 3, e103. https://doi.org/10.7717/peerj-cs.103

Microsoft. (n.d.). Math Solver. Retrieved July 9, 2025, from https://mathsolver.microsoft.com

Mistral AI. (2023). Introducing Mistral 7B. Retrieved July 9, 2025, from https://mistral.ai/news/announcing-mistral-7b

Photomath. (n.d.). Photomath App. Retrieved July 9, 2025, from https://photomath.com

Smith, R. (2007). An overview of the Tesseract OCR engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (Vol. 2, pp. 629–633). IEEE. https://doi.org/10.1109/ICDAR.2007.4376991