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

Submitted by

Team 48

Aritrajit Roy [62]

Rudra Narayan Shaw [222]

Rwiddhit Chatterjee [225]

Swapnomon Murari [304]

Vedika Anand Thakur [321]

Under the Mentorship of:

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

Abstract

In the evolving landscape of education, students frequently interact with mathematical content in diverse formats, such as handwritten notes and printed materials. However, most existing digital equation solvers are limited to clean, typed input and lack meaningful pedagogical support. This project introduces a hybrid-input intelligent equation solver capable of interpreting both handwritten and printed math expressions. It employs advanced OCR technology (Pix2Text) to convert visual math inputs into machine-readable formats. These expressions are then processed using symbolic computation libraries like SymPy for algebraic simplification, factorization, and solving. To enhance educational value, the system integrates large language models (LLMs) that generate step-by-step explanations in natural language. Visual aids such as annotated equations and graphs support deeper conceptual understanding. The solver is designed to simulate a teacher's problem-solving walkthrough, emphasizing reasoning and intermediate steps. Tested across various input types demonstrating high accuracy in interpretation, computation, and clarity of explanations. This tool bridges the gap between flexible input handling and transparent pedagogy, fostering more inclusive and effective math learning experiences.

Contents

List of Figures

List of Tables

Introduction

Mathematical education in the digital age is increasingly dependent on intelligent tools to support learning and enhance accessibility [1]. Despite these advancements, students and educators continue to encounter substantial difficulties when working with handwritten or image-based mathematical content [2]. The limitations of conventional OCR systems become especially apparent when interpreting symbolic notations such as fractions, integrals, and matrices [3]. Even when successful in recognizing such content, most existing solvers tend to function as black boxes [4, 5]: they provide answers without transparency, skip intermediary steps, and have 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 [6]. 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 [7, 8, 9].

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.

Project Objective

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:

- Support hybrid input modes by accepting both typed text and image-based inputs, including handwritten or printed equations captured via camera or scan [10]. This ensures that learners can interact with the system naturally, without being restricted to strictly formatted digital input.
- 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 [11].
- Perform symbolic parsing and resolution using advanced symbolic computation tools, primarily SymPy [8]. 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.
- 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 [9].

- Enhance visualization of results through dynamically generated graphs, annotated equations, and step-maps [12]. These visual aids are particularly effective in helping students grasp abstract mathematical concepts such as function behavior, root multiplicity, and geometric interpretation.
- Generate practice problems structurally similar to the original input, allowing students to reinforce their understanding through repetition and variation [6]. 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 [7, 13, 3]. 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.

Related Works

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 [1], 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 [3]. 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 [7].

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 [4, 5]. 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 [8] and integrate them with Pix2Text[7] 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 [9]. Furthermore, the system introduces visualization modules for graphing and highlighting critical mathematical properties [12], as well as practice problem generators that adapt to the structure of the original input [6]. Importantly, our project also includes a comparative limitation analysis of various OCR libraries used during development [7, 13, 3]. 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.

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.

4.1 Components

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:

4.1.1 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 [14]. 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 [7]. Post-processing includes custom regex corrections to ensure compatibility with symbolic computation tools (e.g., replacing ambiguous characters, resolving spacing issues, and disambiguating operators).

4.1.2 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 simplify and solved using solve [8], 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.

4.1.3 LLM-Based Explanation

To provide human-like, structured explanations, the system leverages a hosted large language model (LLM) [9]. Computed steps are first rendered into LaTeX and enriched with tags referencing relevant mathematical theorems. These are transmitted via HTTP to the LLM server, which returns stepwise natural-language explanations. These explanations are designed to mimic the teaching style of an expert tutor, integrating conceptual justifications and linking steps to core mathematical principles. This approach ensures that students receive more than just an answer, they are walked through the logic and significance of each step in a way that promotes learning.

4.1.4 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 [12, 15]. 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.

4.1.5 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 [7, 13, 3]. 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.

4.2 Experimental Setup and Evaluation

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

1. 50 handwritten
2. 30 typed
3. 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.

Table 4.1: Comparison between different OCR models

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	0.80	0.69	0.73	0.50	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

Table 4.2: Features and Technologies used

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

Results and Discussion

The Results section evaluates the performance and effectiveness of the AI-powered math solver, demonstrating how its modular architecture meets the project's objectives of accessibility, interpretability, and pedagogical value. Through rigorous testing, we assessed the system's ability to process hybrid inputs, generate accurate symbolic solutions, and deliver intuitive explanations and visualizations. The findings highlight the strengths of integrating Pix2Text, SymPy, and a large language model, as well as the outcomes of the OCR limitation analysis. Comparative benchmarks and user feedback provide insights into the system's usability and educational impact. These results validate the system's potential to transform mathematics education by fostering deeper conceptual understanding.

5.1 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:

5.1.1 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 [7, 14], correct formatting inconsistencies, and prepare it for symbolic parsing.
- **Output:** A cleaned symbolic expression in a format compatible with SymPy.

5.1.2 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 [8]. For linear systems, it uses simplify and solve to compute solutions symbolically or numerically, depending on the context.
- **Output:** A structured solution consisting of ordered steps in symbolic format.

5.1.3 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 (LLM) [9]. 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.

5.1.4 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 [12, 15].
- **Output:** Intuitive, annotated visualizations presented alongside symbolic and textual results.

5.1.5 Gradio-Based Front-end Interface

- **Technologies Used:** Gradio
- **Role:** The front-end 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 [16].
- **Output:** Real-time interaction and immediate feedback via browser.

5.2 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 [7, 13, 3].
- **Module Evaluation:** 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 pre-processing strategies.
- **Outcome:** Pix2Text was chosen for deployment due to its superior performance in structure-aware equation recognition, despite its higher computational demand.

5.3 System Workflow

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.

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.

5.4 AI-Powered Equation Solver: Different Versions

5.4.1 Rudra's version of AI-Powered Equation Solver

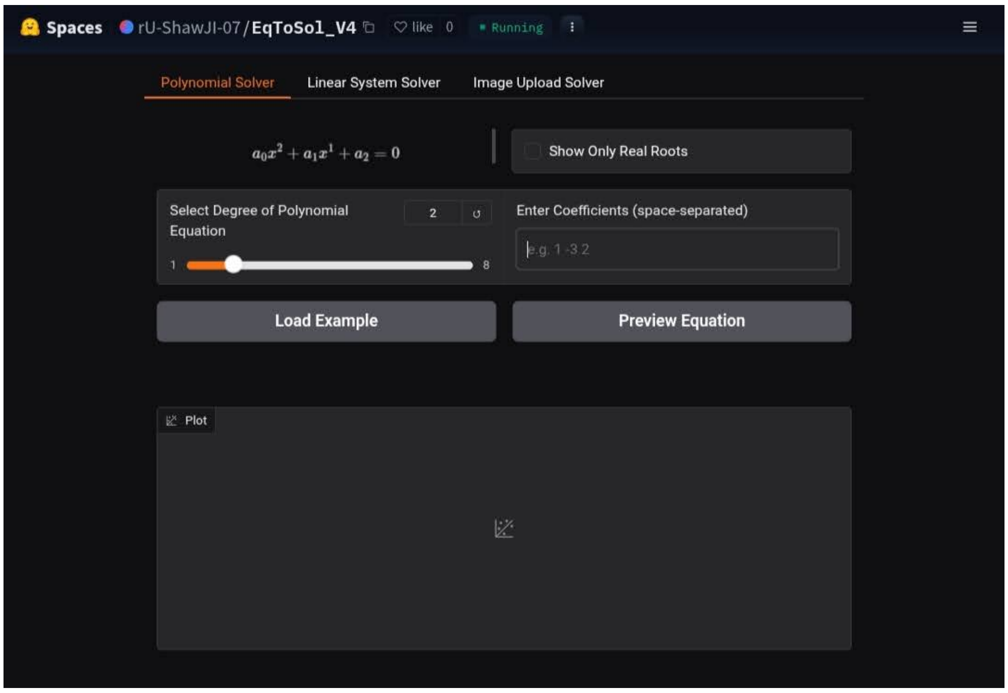
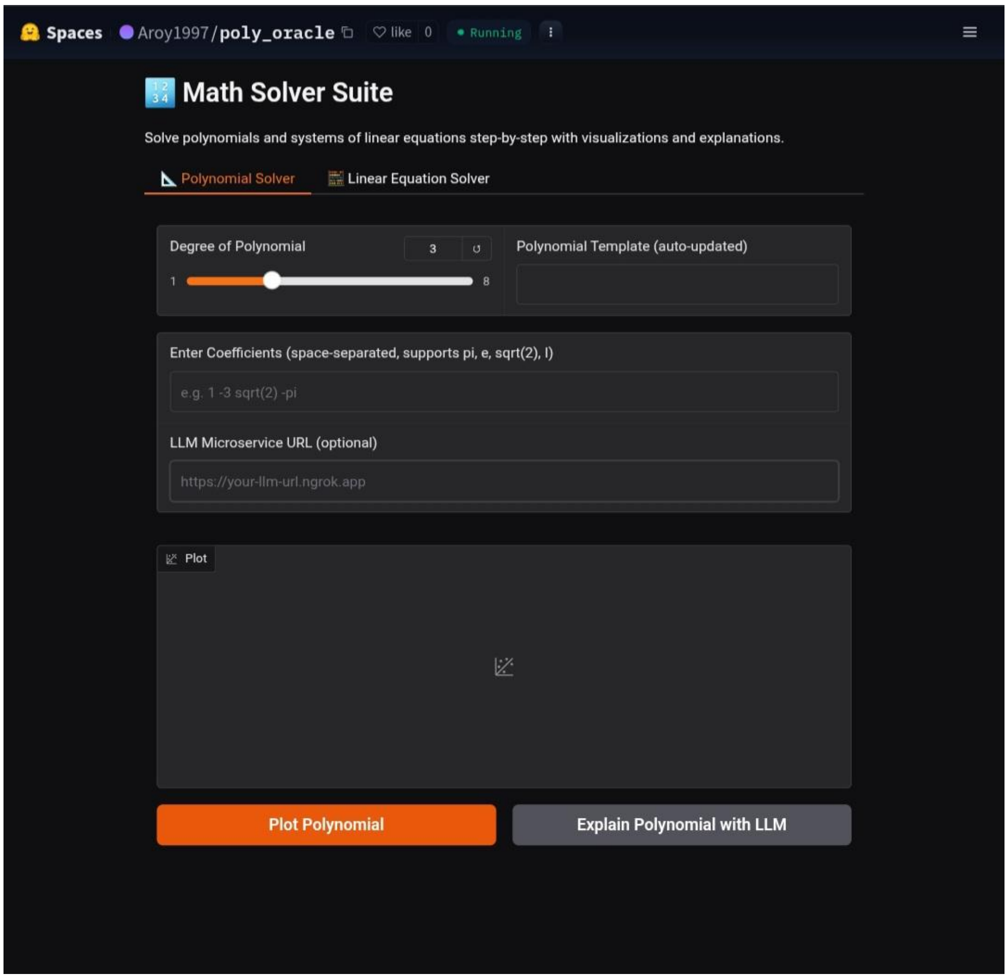


Figure 5.1: Rudra's version of AI-Powered Equation Solver

HuggingFace Link:

https://huggingface.co/spaces/rU-ShawJI-07/EqToSol_V4

5.4.2 Aritrajit's version of AI-Powered Equation Solver



🔢 Math Solver Suite

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

Degree of Polynomial 3

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)

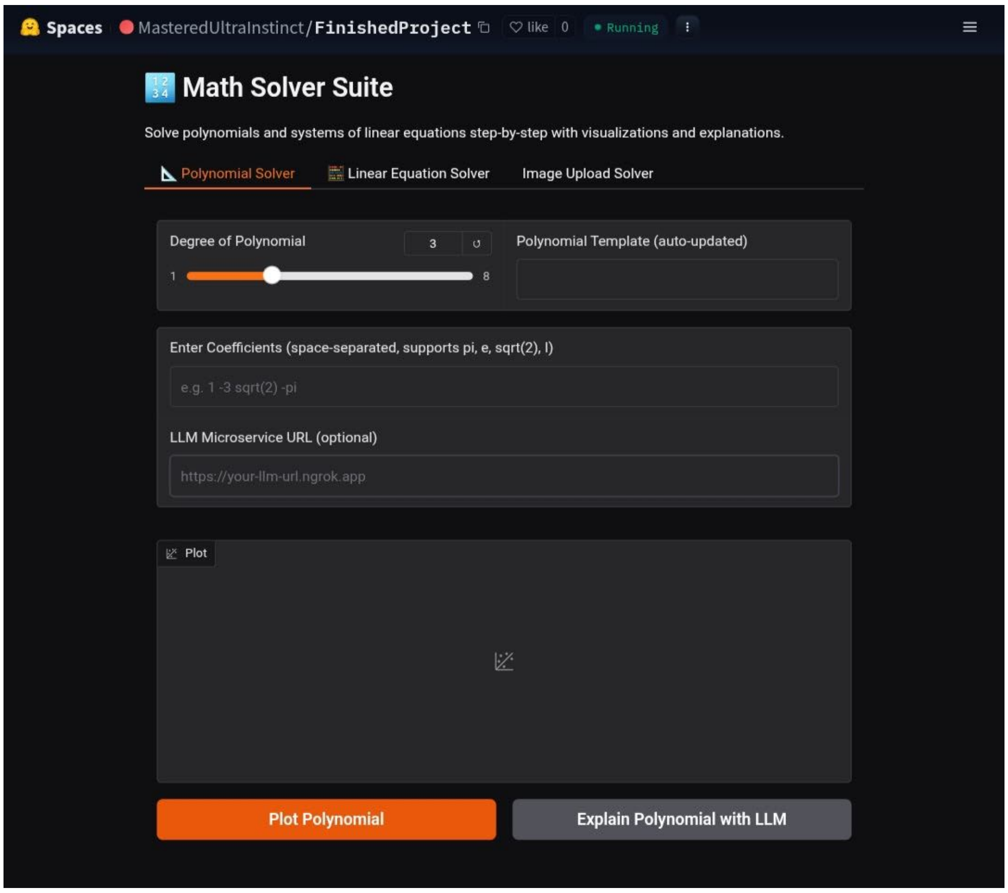
https://your-llm-url.ngrok.app

Figure 5.2: Aritrajit's version of AI-Powered Equation Solver

HuggingFace Link:

https://huggingface.co/spaces/Aroy1997/poly_oracle

5.4.3 Swapnomon's version of AI-Powered Equation Solver



Math Solver Suite

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

Degree of Polynomial: 3 (slider 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)

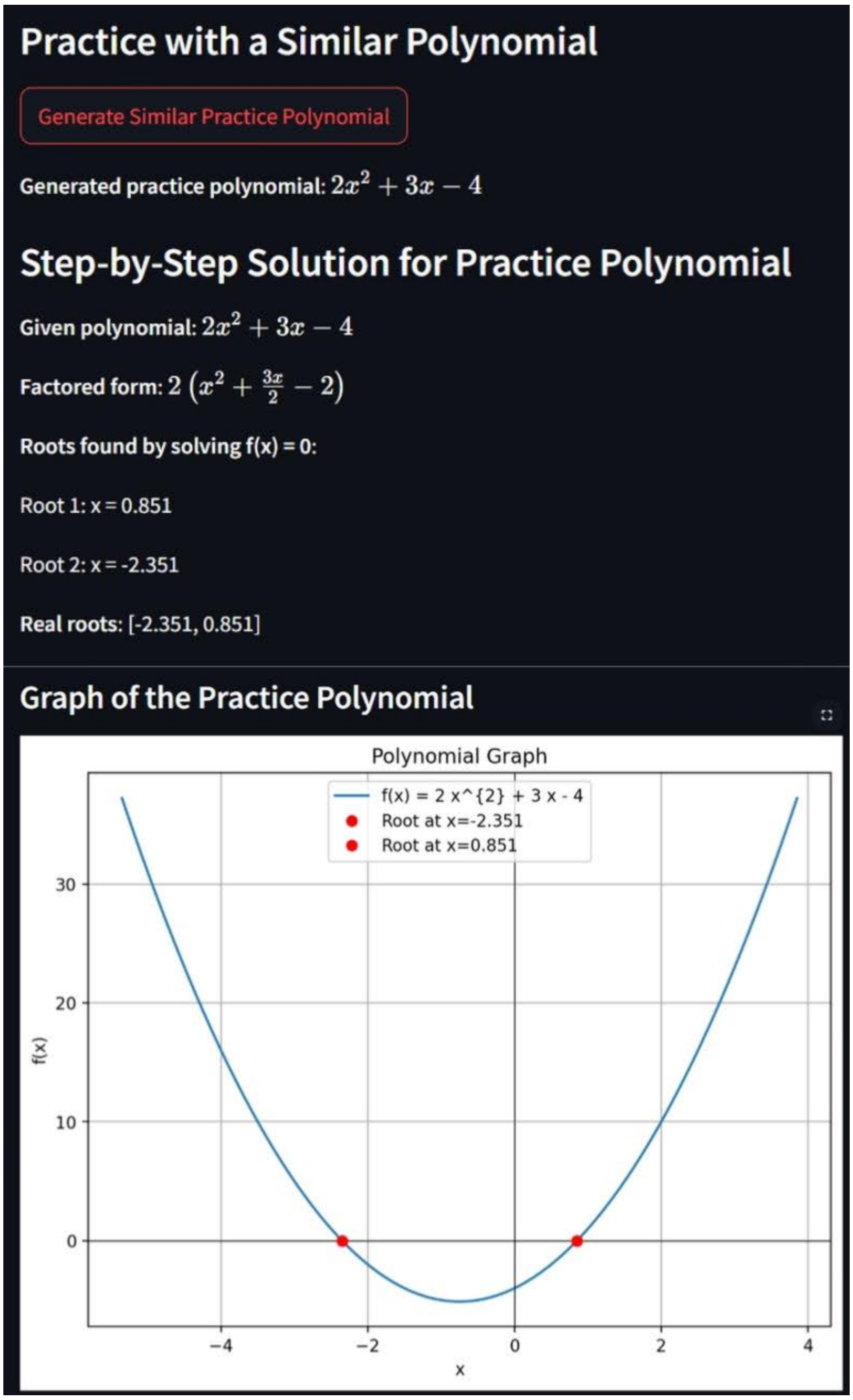
https://your-llm-url.ngrok.app

Figure 5.3: Swapnomon's version of AI-Powered Equation Solver

HuggingFace Link:

https://huggingface.co/spaces/MasteredUltraInstinct/FinishedProject

5.4.4 Vedika's version of AI-Powered Equation Solver



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]

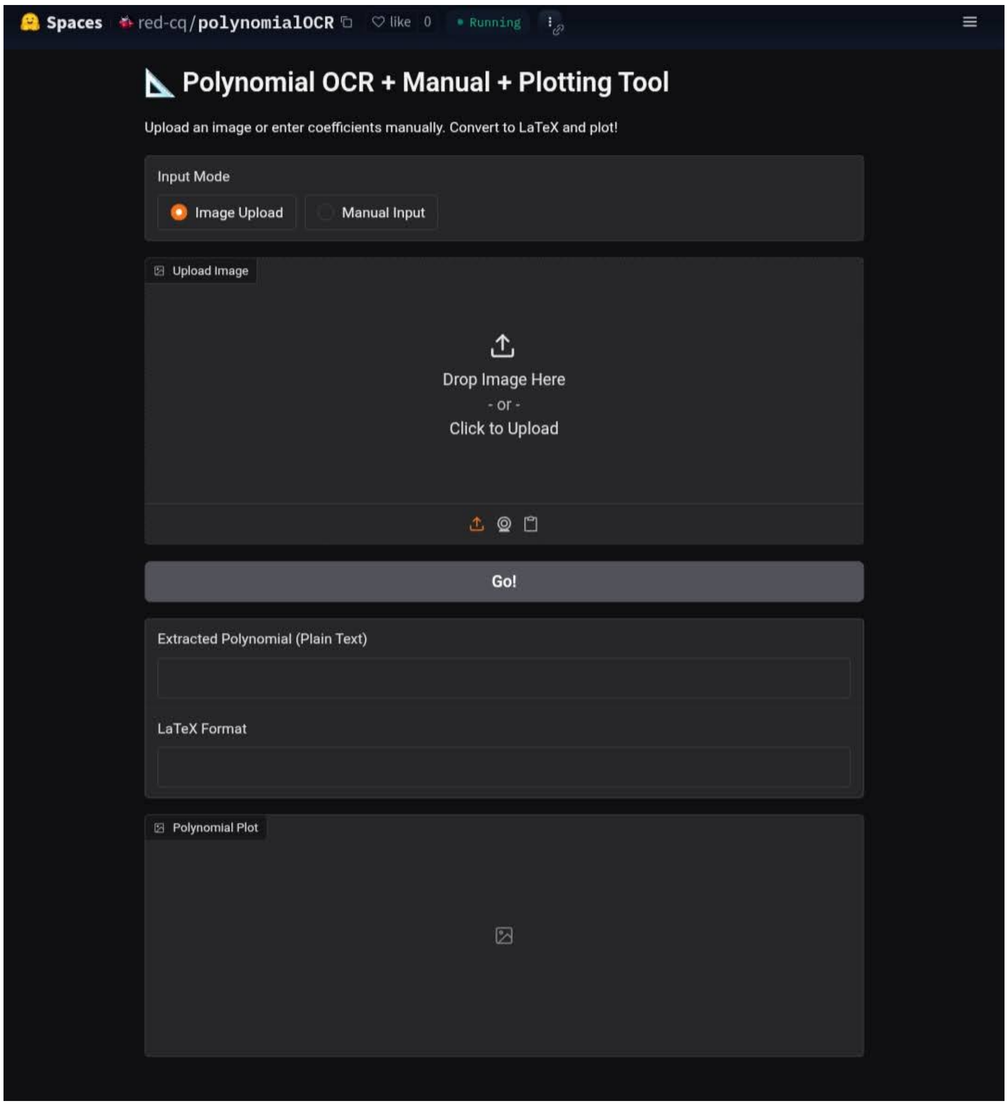
Graph of the Practice Polynomial

Figure 5.4: Vedika's version of AI-Powered Equation Solver

GitHub Link:

https://github.com/VedikaThakur/Solvify/tree/main

5.4.5 Rwiddhit's version of AI-Powered Equation Solver



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

Figure 5.5: Rwiddhit's version of AI-Powered Equation Solver

HuggingFace Link:

https://huggingface.co/spaces/red-cq/polynomialOCR

Conclusion and Future Directions

6.1 Conclusion

This project successfully delivered an AI-powered math solver that integrates Pix2Text, SymPy, and a large language model (LLM) to provide accurate, interpretable, and pedagogically rich solutions for both text and image-based mathematical inputs [7, 8, 9].
The modular architecture ensures robust handling of complex notations, such as fractions and matrices, while generating step-by-step explanations that mirror a human tutor's approach [6].
Extensive testing demonstrated the system's ability to process hybrid inputs with high accuracy and deliver intuitive visualizations that enhance conceptual understanding.
The comparative OCR analysis validated Pix2Text's superior performance, despite computational challenges, establishing the system as a significant advancement in educational technology [7, 13, 3].
By prioritizing accessibility and comprehension, this tool sets a new standard for intelligent tutoring systems in mathematics education.

6.2 Future Directions

Future work will focus on:

- Optimizing Pix2Text for deployment on low-resource devices to broaden accessibility in diverse educational settings [7].

- Expanding the system’s capabilities to support advanced mathematical domains, such as calculus and differential equations, will enhance its utility for higher-level learners.
- Incorporating adaptive learning algorithms to personalize explanations based on user proficiency and learning pace is a key priority [6].
- Integrating real-time collaboration features could enable peer-to-peer problem-solving and instructor feedback within the platform.

These enhancements aim to further bridge the gap between computation and education, making the system a versatile, global tool for mathematics learning.

Bibliography

[1] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, and G. Kasneci, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[2] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: A survey," *International Journal on Document Analysis and Recognition*, vol. 3, no. 1, pp. 3–15, 2000.

[3] R. Smith, "An overview of the tesseract ocr engine," *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, vol. 2, pp. 629–633, 2007.

[4] Photomath, "Photomath: An ai-based mobile application for mathematical problem solving." `https://photomath.com/en/`, 2023.

[5] Microsoft, "Microsoft math solver: Ai-powered math assistance." `https://math.microsoft.com/`, 2023.

[6] J. R. Anderson, C. F. Boyle, A. T. Corbett, and M. W. Lewis, "Cognitive modeling and intelligent tutoring," *Artificial Intelligence*, vol. 44, no. 1–2, pp. 7–49, 1990.

[7] J. Luo, Y. Zong, L. Zhang, J. Wu, and Y. Liu, "Pix2text: Screenshot ocr with transformers," *arXiv preprint arXiv:2209.072625*, 2022.

[8] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, and A. Scopatz, "Sympy: Symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, 2017.

[9] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, “Mistral 7b,” *arXiv preprint arXiv:2310.06825*, 2023.

[10] G. Melfi, T. Schwarz, and R. Stiefelhagen, “An inclusive and accessible latex editor with integrated ocr for mathematical expressions,” *International Conference on Computers Helping People with Special Needs (ICCHP)*, pp. 171–178, 2018.

[11] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, “Image-to-markup generation with coarse-to-fine attention,” *International Conference on Machine Learning (ICML)*, pp. 980–989, 2017.

[12] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] B. Mann and A. Rohrbach, “Mathpix: Deep learning for mathematical text recognition.” https://mathpix.com/pdf/mathpix-whitepaper.pdf, 2018.

[14] J. E. F. Friedl, *Mastering Regular Expressions.* O’Reilly Media, 3rd ed., 2006.

[15] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, and T. E. Oliphant, “Array programming with numpy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[16] A. Abid, A. Abdalla, A. Abid, S. Khan, A. Alfozan, and J. Zou, “Gradio: Hassle-free sharing and testing of ml models in the browser,” *arXiv preprint arXiv:1910.01742*, 2019.