CREST Gold Submission

Olympiad Problem Generator

A Free Community Platform for AI-Generated Novel Math Olympiad Problems

Vihan Patel

2026

Contents

1 Introduction **2**

2 This document **2**

3 What Olympiad Math really is. **2**

4 Aims and Objectives **4**

5 Wider Purpose **4**

6 Background Research **5**

7 Approaches Considered and Chosen **6**
7.1 AI Role 6
7.2 Access 8
7.3 Mathematical Scope 9

8 How it Works **10**
8.1 A comparison with CogAtom 11
8.2 A walkthrough of the expert end. 11

9 Community Feedback **14**

10 Ethical Considerations **16**

11 Decisions and Their Impact on the Project **17**

12 Timeline **18**

13 Conclusions and Limitations **19**

14 Personal Reflection **20**

References **20**

1 Introduction

In this report, I document the design and prototype development of an Olympiad problem generator. At the core of my system is a hand-curated library of math Olympiad intuitions that I have carefully compiled over two years of intensive involvement in Olympiads. The system uses a well-structured pipeline to construct novel Olympiad-style problems in Algebra, with a special focus on inequalities(my personal favorite). I have developed this not as a standalone piece of software, but as a working core of a free, community-governed platform intended to contribute to the growing Olympiad community.

In this report, I will explain what my system does, why it was designed the way it is, what has been built so far, what is still on paper and not yet implemented, and what the honest limitations are.

Link to the Main Project

2 This document

I created this report using Overleaf. The average high schooler uses Google Docs, Microsoft Word, and PowerPoint. However, I was introduced to LaTeX typesetting early on in my math journey. For the past two and a half years, I have done most of my work in LaTeX. I just love the superior formatting that other average applications can in no way match. And hence, that explains my use of Overleaf for this project.

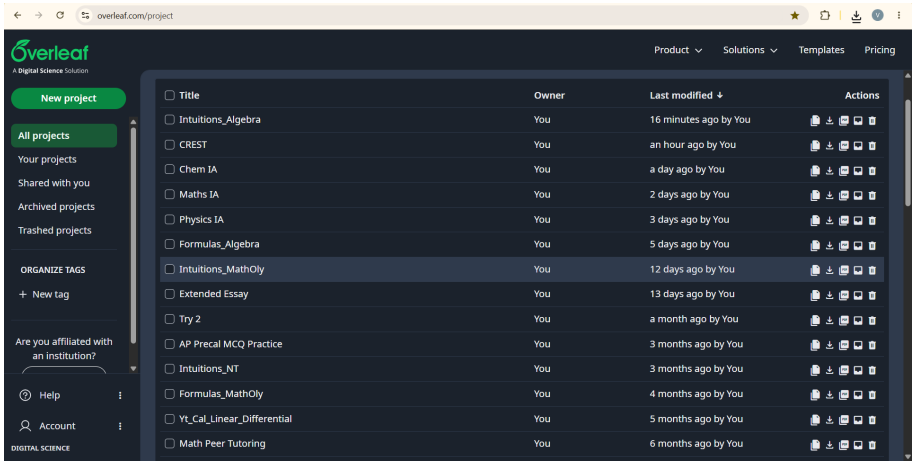


Figure 1: A few of my projects in Overleaf over the years.

3 What Olympiad Math really is.

Instead of assuming that the reader is just as aware of the subject matter as I am, I would take a paragraph here to explain the same. School mathematics, at almost every level, is about teaching a formula and then testing how well students can apply it. For instance, we learn the quadratic formula and follow it up with word problems and questions on how to apply it.

Olympiad math is different from this. Very rarely is a problem about how well a student can apply a taught method. An average problem deliberately hides the technique required inside

a statement that looks completely unrelated to it. The mathematics, to the core, isn't too far beyond high school (since it is a high school thing). However, the mere disguise and deep-seated 'intuition' required to crack a problem is what makes this a completely different league.

Here is an example of an average problem from AIME [1] (American Invitational Mathematics Examination).

Example AIME Problem

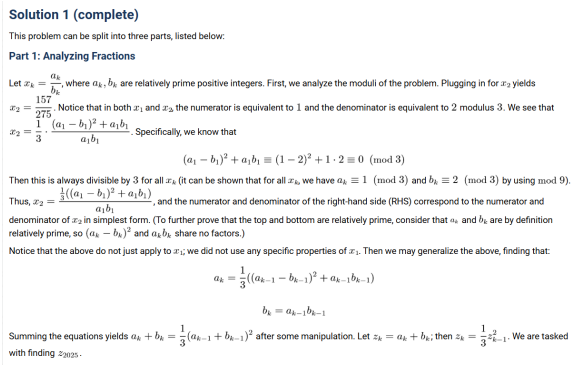
Problem

Let the sequence of rationals $x_1, x_2, \ldots$ be defined such that $x_1 = \frac{25}{11}$ and

$$x_{k+1} = \frac{1}{3}\left(x_k + \frac{1}{x_k} - 1\right).$$

x_{2025} can be expressed as $\frac{m}{n}$ for relatively prime positive integers m and n. Find the remainder when $m+n$ is divided by 1000.

Table 1: Representative AIME problem illustrating Olympiad-style mathematics.



Solution 1 (complete)

This problem can be split into three parts, listed below:

Part 1: Analyzing Fractions

Let $x_k = \frac{a_k}{b_k}$, where a_k, b_k are relatively prime positive integers. First, we analyze the moduli of the problem. Plugging in for x_2 yields $x_2 = \frac{157}{275}$. Notice that in both x_1 and x_2 the numerator is equivalent to 1 and the denominator is equivalent to 2 modulus 3. We see that $x_2 = \frac{1}{3} \cdot \frac{(a_1-b_1)^2 + a_1b_1}{a_1b_1}$. Specifically, we know that

$$(a_1 - b_1)^2 + a_1b_1 \equiv (1-2)^2 + 1 \cdot 2 \equiv 0 \pmod 3$$

Then this is always divisible by 3 for all x_k (it can be shown that for all x_k we have $a_k \equiv 1 \pmod 3$ and $b_k \equiv 2 \pmod 3$ by using $\bmod 9$). Thus, $x_2 = \frac{\frac{1}{3}((a_1-b_1)^2 + a_1b_1)}{a_1b_1}$, and the numerator and denominator of the right-hand side (RHS) correspond to the numerator and denominator of x_2 in simplest form. (To further prove that the top and bottom are relatively prime, consider that a_k and b_k are by definition relatively prime, so $(a_k - b_k)^2$ and a_kb_k share no factors.)

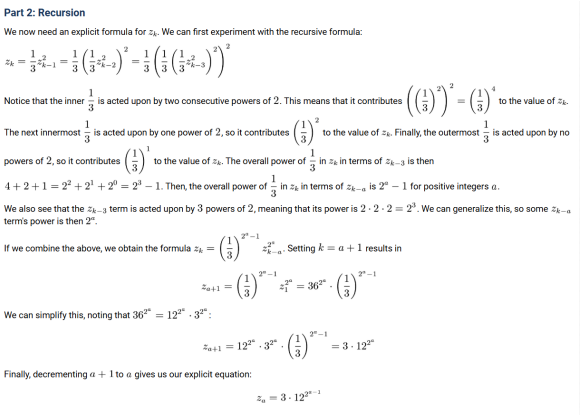
Notice that the above do not just apply to x_1; we did not use any specific properties of x_1. Then we may generalize the above, finding that:

$$a_k = \frac{1}{3}((a_{k-1} - b_{k-1})^2 + a_{k-1}b_{k-1})$$

$$b_k = a_{k-1}b_{k-1}$$

Summing the equations yields $a_k + b_k = \frac{1}{3}(a_{k-1} + b_{k-1})^2$ after some manipulation. Let $z_k = a_k + b_k$; then $z_k = \frac{1}{3}z_{k-1}^2$. We are tasked with finding z_{2025}.

(a) Solution (Part 1)



Part 2: Recursion

We now need an explicit formula for z_k. We can first experiment with the recursive formula:

$$z_k = \frac{1}{3}z_{k-1}^2 = \frac{1}{3}\left(\frac{1}{3}z_{k-2}^2\right)^2 = \frac{1}{3}\left(\frac{1}{3}\left(\frac{1}{3}z_{k-3}^2\right)^2\right)^2$$

Notice that the inner $\frac{1}{3}$ is acted upon by two consecutive powers of 2. This means that it contributes $\left(\left(\frac{1}{3}\right)^2\right)^2 = \left(\frac{1}{3}\right)^4$ to the value of z_k. The next innermost $\frac{1}{3}$ is acted upon by one power of 2, so it contributes $\left(\frac{1}{3}\right)^2$ to the value of z_k. Finally, the outermost $\frac{1}{3}$ is acted upon by no powers of 2, so it contributes $\left(\frac{1}{3}\right)^1$ to the value of z_k. The overall power of $\frac{1}{3}$ in z_k in terms of z_{k-3} is then $4 + 2 + 1 = 2^2 + 2^1 + 2^0 = 2^3 - 1$. Then, the overall power of $\frac{1}{3}$ in z_k in terms of z_{k-a} is $2^a - 1$ for positive integers a.

We also see that the z_{k-3} term is acted upon by 3 powers of 2, meaning that its power is $2 \cdot 2 \cdot 2 = 2^3$. We can generalize this, so some z_{k-a} term's power is then 2^a.

If we combine the above, we obtain the formula $z_k = \left(\frac{1}{3}\right)^{2^a-1} z_{k-a}^{2^a}$. Setting $k = a+1$ results in

$$z_{a+1} = \left(\frac{1}{3}\right)^{2^a-1} z_1^{2^a} = 36^{2^a} \cdot \left(\frac{1}{3}\right)^{2^a-1}$$

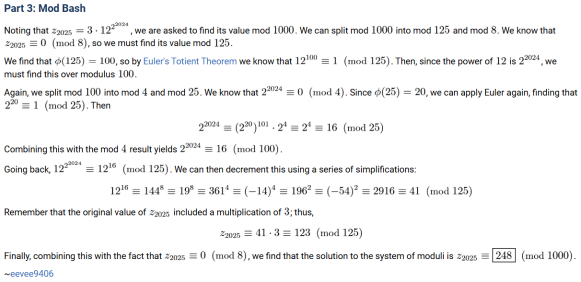
We can simplify this, noting that $36^{2^a} = 12^{2^a} \cdot 3^{2^a}$:

$$z_{a+1} = 12^{2^a} \cdot 3^{2^a} \cdot \left(\frac{1}{3}\right)^{2^a-1} = 3 \cdot 12^{2^a}$$

Finally, decrementing $a+1$ to a gives us our explicit equation:

$$z_a = 3 \cdot 12^{2^{a-1}}$$

(b) Solution (Part 2)



Part 3: Mod Bash

Noting that $z_{2025} = 3 \cdot 12^{2^{2024}}$, we are asked to find its value mod 1000. We can split mod 1000 into mod 125 and mod 8. We know that $z_{2025} \equiv 0 \pmod 8$, so we must find its value mod 125.

We find that $\phi(125) = 100$, so by Euler's Totient Theorem we know that $12^{100} \equiv 1 \pmod{125}$. Then, since the power of 12 is 2^{2024}, we must find this over modulus 100.

Again, we split mod 100 into mod 4 and mod 25. We know that $2^{2024} \equiv 0 \pmod 4$. Since $\phi(25) = 20$, we can apply Euler again, finding that $2^{20} \equiv 1 \pmod{25}$. Then

$$2^{2024} \equiv (2^{20})^{101} \cdot 2^4 \equiv 2^4 \equiv 16 \pmod{25}$$

Combining this with the mod 4 result yields $2^{2024} \equiv 16 \pmod{100}$.

Going back, $12^{2^{2024}} \equiv 12^{16} \pmod{125}$. We can then decrement this using a series of simplifications:

$$12^{16} \equiv 144^8 \equiv 19^8 \equiv 361^4 \equiv (-14)^4 \equiv 196^2 \equiv (-54)^2 \equiv 2916 \equiv 41 \pmod{125}$$

Remember that the original value of z_{2025} included a multiplication of 3; thus,

$$z_{2025} \equiv 41 \cdot 3 \equiv 123 \pmod{125}$$

Finally, combining this with the fact that $z_{2025} \equiv 0 \pmod 8$, we find that the solution to the system of moduli is $z_{2025} \equiv \boxed{248} \pmod{1000}$.

~eevee9406

(c) Solution (Part 3)

Figure 2: Worked solution demonstrating the depth of reasoning required by Olympiad mathematics.

This distinction is the reason the entire project is the way it is. There are plenty of systems that produce mathematically correct statements drawn on ordinary textbook exercises. To create something of the Olympiad level is a completely different game.

4 Aims and Objectives

To put it concisely, my central aim of this project is:
To design and build a working prototype of an AI-assisted system that generates non-trivial, Olympiad competition-appropriate mathematics problems in Algebra as a foundation of a free community platform.
I break it down into five objectives:

Objective 1. Build a structured, machine-readable data set. Throughout my Olympiad journey, I had solved books worth of problems. However, my main objective here was to convert this personal library into a structured format that the machine can use for problem construction. This was one of the most important and time-consuming parts of my project, as the pipeline would need something strong to be built upon.

Objective 2. Design and implement a generation pipeline. Another important objective was to structure the pipeline very thoroughly. A vague and undefined path would have made the model use different approaches every time for problem generation. This would have led to inconsistent results.

Objective 3. Safeguards against duplication and repetition. This is perhaps one of my more difficult objectives. Giving a model a data set and a fixed pipeline is kind of and more likely to give fixed results and repetitive problems. Hence, a self-duplication check is a must for my model to maintain the novelty of the problems.

Objective 4. Wider platform architecture. My aim for this entire project is not to make this a personal use platform. I want to create this for the community as a whole. I would have to design the interface that makes the curator and donor roles (defined later) clear and easy to manage.

Objective 5. Community response. For any project, one must go beyond one's own personal viewpoint and be open to what others have to say. I aim to get a small user base to share what they think about the project.

5 Wider Purpose

I think one would have some idea of my purpose at this point. My purpose is to test whether a small, expert-curated intuition library can be turned into a reusable, inspectable pipeline that everyone can eventually run, extend, or benefit from, without having the need for deep Olympiad expertise or deep AI expertise themselves. And this, when I thought about it, is for a particular reason.

The population that needs more Olympiad-level practice material is much larger than the population capable of writing it well. Creating a genuinely novel problem that has both mathematical depth and a deep-seated intuition is a completely different skill set. I can solve as many problems, but when I sat down to create one myself, I saw the real struggle. In his paper, Chen et al. [2] notes that the pool of human-authored Olympiad-level problems is "static, limited in size, and does not fulfill" the demands of data training. This scarcity has become "a primary obstacle to the rigorous development and evaluation of advanced reasoning systems." School curriculum

problems are nowhere near this level, and hence the average teacher cannot contribute at all.

So I also aim to test whether the current bottlenecks described in AI research can be addressed, at a small scale, in the direction of human learners. If the underlying principle points towards something genuinely useful, then this project can be expanded to a larger scale as well. Also, if I think of it, this can be something all professors and math circles could replicate. Create their own intuition libraries to run a similar pipeline for their own areas without needing the problem 'setter' skillset.

6 Background Research

The most relevant piece of literature I found was CogAtom [2]. It is a framework for synthesizing Olympiad-level mathematics for the purpose of training and evaluating large language models. I would like to briefly describe its methodology in some depth because it is somewhat parallel to this project and can justify some parts of my own design.

The CogAtom [2] pipeline has three broad stages.

First, a large seed set of existing problems drawn from datasets like GSM8K, MATH, and other sources is filtered using an LLM-as-judge protocol. Basically, from what I understand, a GPT scores each candidate problem three times against a five-point rubric measuring reasoning depth, and only problems scoring 3 or higher are retained. From this filtered set, the system extracts "reasoning atoms": individual, fine-grained knowledge or technique units. These atoms then become nodes in a larger graph.

Second, the new problems are constructed not directly from atoms, but from paths through this graph. A biased random walk samples a sequence of atoms, favoring less common, lower-degree nodes that encourage novel combinations rather than obvious ones.

Third, this combination of atoms is handed to a large language model, which synthesizes a full problem statement and a worked solution from it.

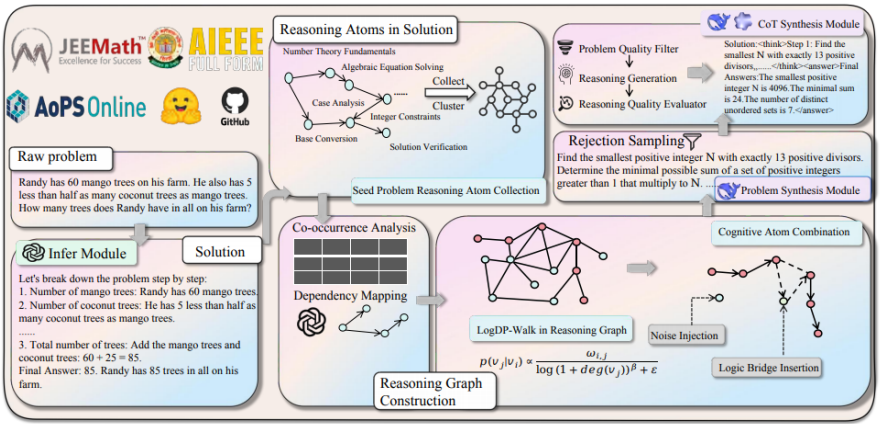


Figure 3: A figure taken directly from the CogAtom [2] paper that illustrates their three step approach.

When I go on to explain the structure of my model, I will draw parallels with this model and evaluate some nuanced differences. So do come back to this. Also, to the best of my knowledge (and I could be wrong), I found no existing public product that generates Olympiad-style problems specifically for human practice and training. Chen et al. [2] paper remains a theoretical model and hasn't been implemented yet.

7 Approaches Considered and Chosen

The Approach.

From how I see it, there were three main decisions about the approaches I took with this project. They were broadly: what the AI's role in problem generation should actually be, how broad should the mathematical scope be, and how open access should the platform be. I discuss each below with my reasoning and the alternative chosen.

7.1 AI Role

If I map this out correctly, I went ahead with three different approaches before settling on the current one.

First, when I began the project, my idea was to simply ask a generative AI model directly for a question using my intuitions that I would feed it. The absence of any kind of structuring meant that the AI model would often hallucinate to create a problem that had nothing to do with my intuitions. Or it would create a problem that had a very surface-level intuition. Basically, it treated my intuitions as 'formulas' that had to be asked. I quickly discarded this entire approach.

Second, as a solution, I thought of having a two-step feed of data. Along with the intuitions, I also entered 'Formulas' for the AI model to get a stronger idea about question generation. So the AI model would pick a formula, and then pick an intuition, and work with it to create a problem. In order to make it more structured, I even added tags such as 'ID, Title, Difficulty' to make it more identifiable for the large language model. However, this turned out to be even worse. What really separates school math from Olympiad math is the absence of structure. But what I just did was create a recipe with perfect structure. This meant that the AI gave me even worse problems than before. I realized at this point the underlying issue. Olympiad problems shouldn't have structure. But the AI model needs structure to work. I had to somehow figure out a balanced approach.

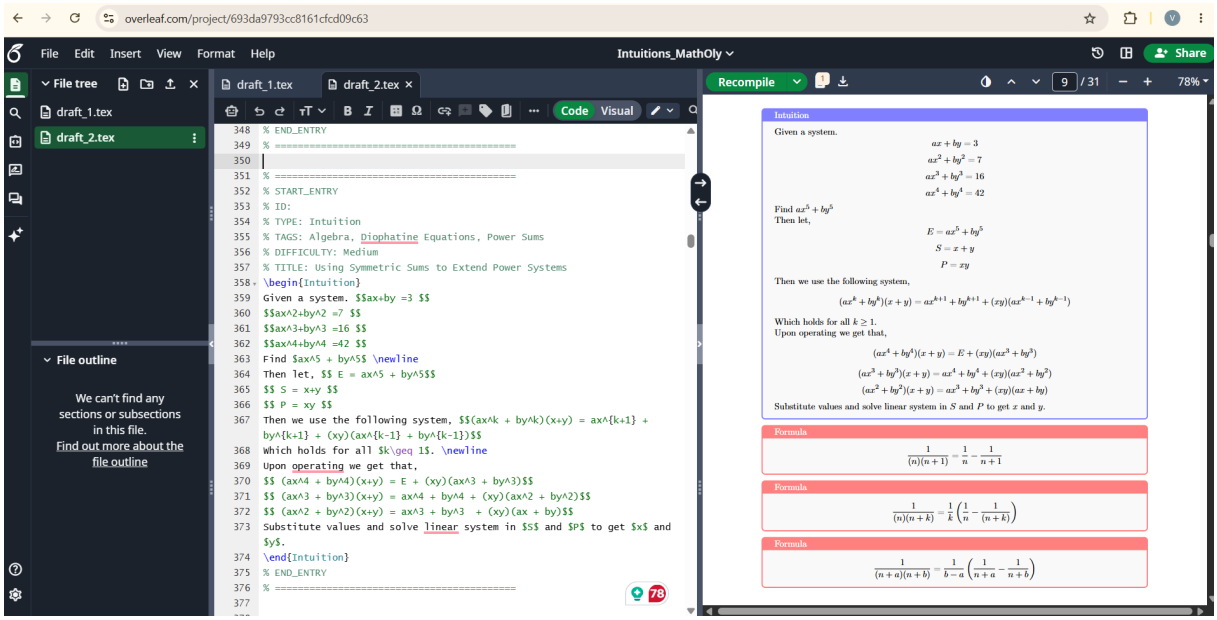
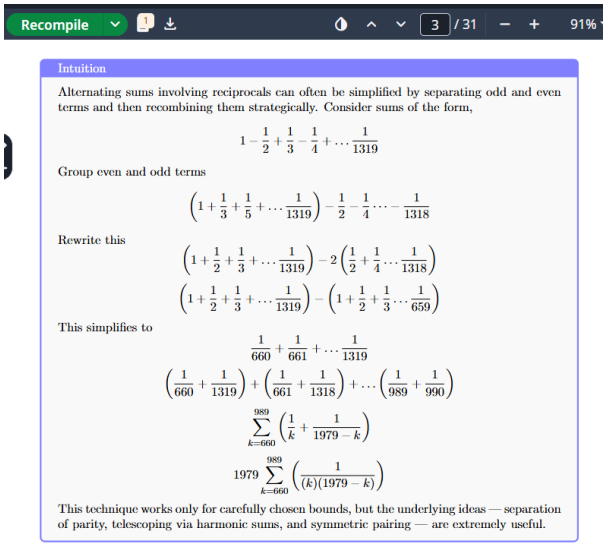
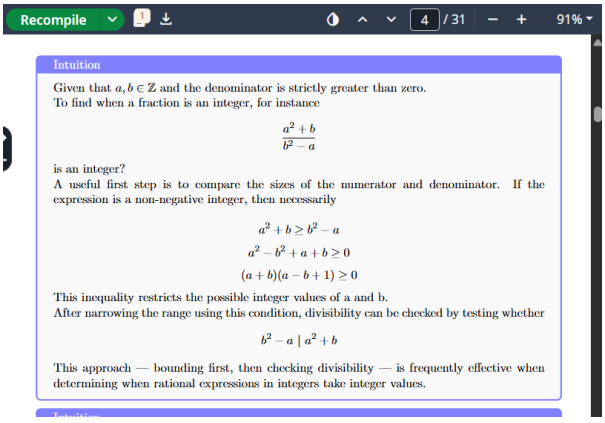


Figure 4: A screenshot of my attempts with the second approach.

Third, and final, I came to what seems for now the best approach. The thing that was wrong with the first approach was that my intuitions were written for a human and not for a machine. There's a difference in how a machine interprets math and how a human does. The AI model didn't need formulas or a very highly structured pathway. The intuitions were enough. Just the way it was written had to be changed. So I directly went to Claude to find the best way for AI models to read such data. After some back and forth, I finally was satisfied with Claude's approach, and it did a complete refinement of the intuitions while keeping the core idea the same. For instance, below is what one intuition looked like before and after the AI refinement.



(a) Intuition 1.



(b) Intuition 2.

Figure 5: Sample of 2 intuitions before refinement.

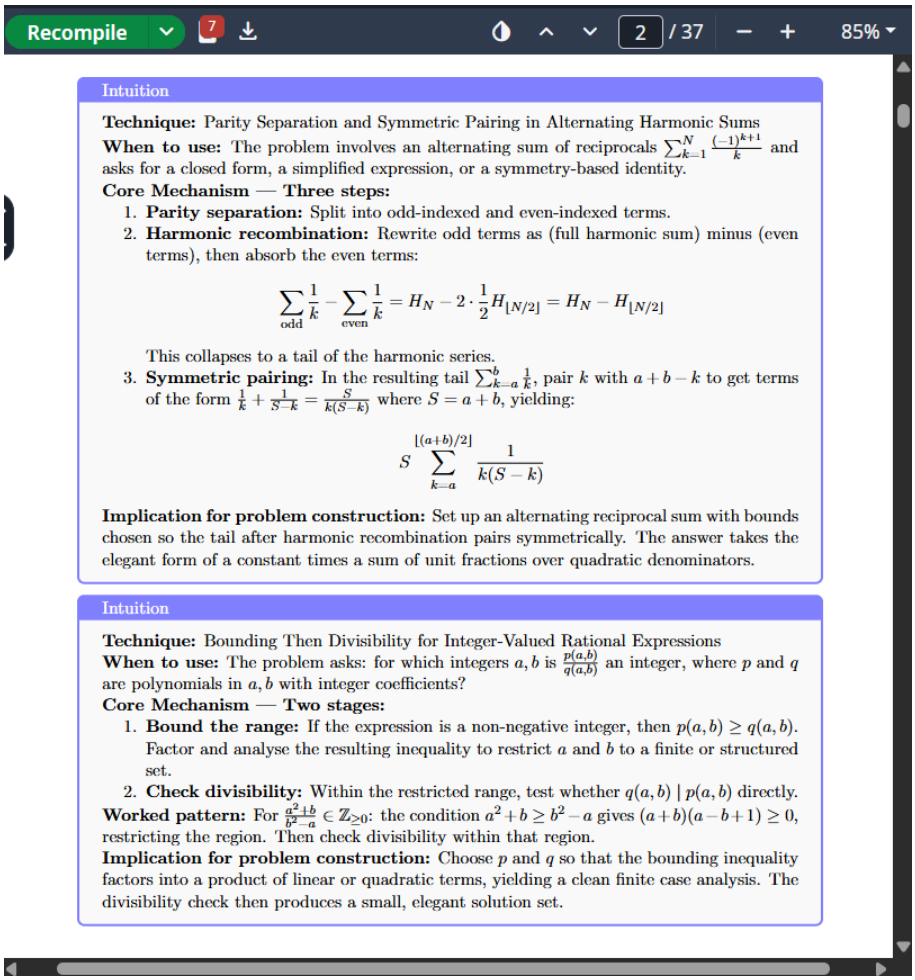


Figure 6: Sample of the same two intuitions after refinement.

The AI model was now giving much better problems. You can see the difference in the before and after cases very clearly. Initially, it seemed like I was feeding it a 'smart' part of a solution. Which is what an intuition basically is. After the refinement, the intuitions read like a general trick or a general 'smart' way to tackle a problem. I clarify the exact pipeline the model uses to generate the problems in a later section. But these are the broad three approaches I took.

7.2 Access

The earlier version I made had the idea that every user would have open access to curate and approve generated problems. I simply thought that more reviewers would mean a genuinely larger platform. However, I quickly reconsidered this when I handed my brother to use it, and he passed a problem that wasn't quite up to the mark. I would definitely need someone qualified to 'pass' a problem. This was crucial to keep the standards of the bank high.

So I came up with the three-role structure. The students would use the question bank directly. Mathematical experts, coaches, professors, and similarly qualified people would have the access to generate problems and review them before adding it to the bank. Donors would fund the ongoing generation cost, with the donation volume determining the generation capacity. This keeps the platform free at the point of use for the group it primarily serves, while giving quality control to people qualified to exercise it, and giving the platform a plausible route to sustaining

itself beyond a single project cycle.

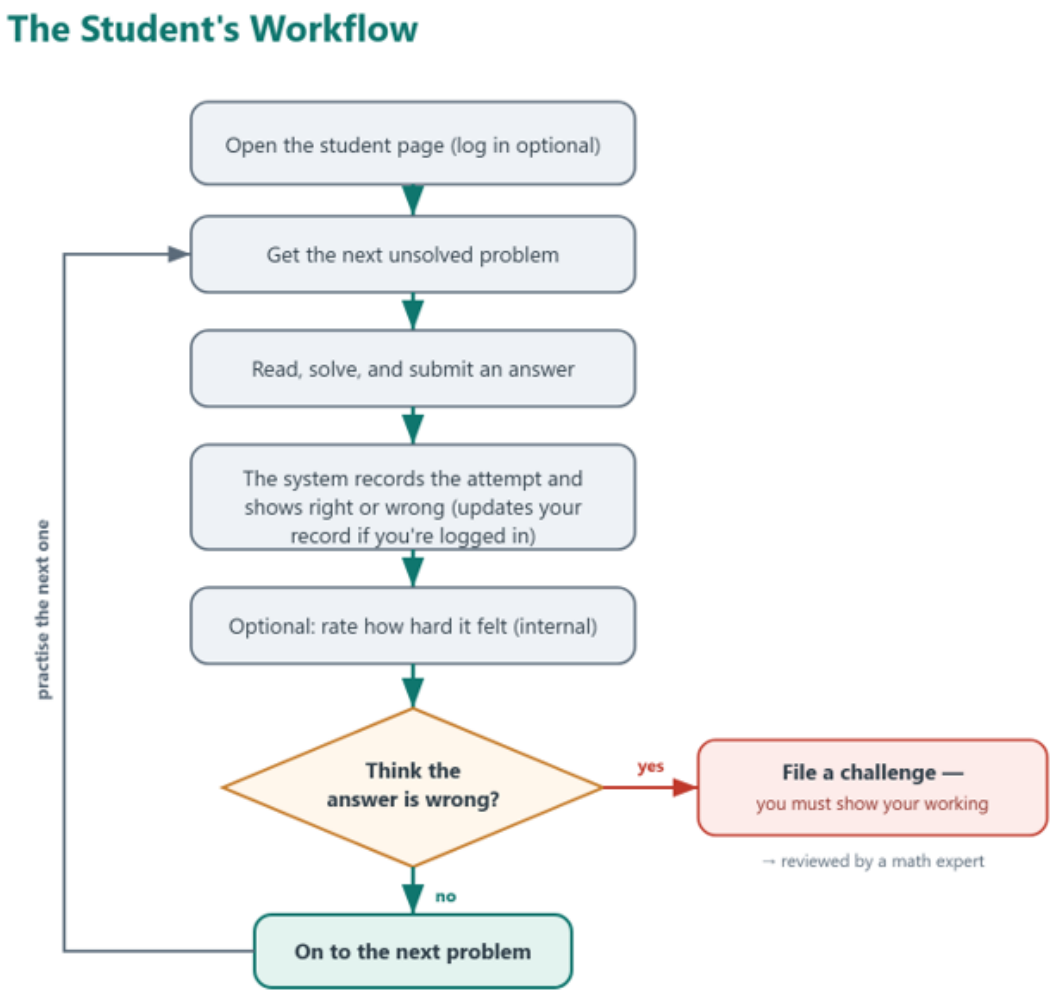


Figure 7: Image generated using AI.

7.3 Mathematical Scope

Every Olympiad aspirant knows that there are four sections: Algebra, Number Theory, Combinatorics, Geometry. Now omitting any of these sections would be unfair to the math community. However, I decided to only go with Algebra, with Number Theory as a part of near expansion. The reasons were simple. The quality of problems generated depends highly on the seeded intuitions. If the intuitions are of low quality, so will the problems be. Being really frank here, Geometry was simply not my cup of tea. I would have to wrap my head around a problem to solve it, let alone create intuitions. So I had to discard this section. Combinatorics was one of my stronger points, but my experience came into play here. Whenever I would come back from an Olympiad, I would hop on to AI to check my answers. And every time, the tougher Combinatorics problems would always be wrong. In fact, different AI models gave different answers. Undercounting and overcounting are one significant reason for this failure. So I discarded this as well. I was decent at Number theory, but I just didn't have enough intuitive depth. As it is always quality over quantity, I stuck to Algebra.

As for the difficulty level, I chose AIME because it is the perfect sweet spot. The questions are

always ‘find the value’ or ‘compute the value of’ kinds, which was perfect for me since I wasn’t thinking about proof-based problems anyway. The difficulty, sitting right between AMC and USAMO, was perfect as well. So I have attempted to mirror the difficulty of AIME questions throughout. Too easy questions at the level of AMC would make no sense. Questions at the level of USAMO were something I couldn’t even imagine being able to create.

8 How it Works

I mentioned the idea of a pipeline numerous times in my report before as well. I explain here what exactly the pipeline is and how it works.

Intuition Selection. A random sample of 8 to 15 intuitions was drawn per request to generate. Critically, the sample deliberately excludes intuitions that were used recently to generate problems. If the model wasn’t forced to take a relatively fresh combination, it may soon easily reach repetition early on, which would go against the entire novelty point.

Backward Construction. The model was coded to select exactly one intuition as the problem’s core from the sample it selected. It then constructed a problem in which that technique gets well-disguised, solves it fully, and reports both the final answer and the intuition used for the purpose. This produces a single candidate problem per generation run. An earlier version generated three independent candidates and selected the best via a separate rating pass. This was deliberately dropped once it became clear that generating three candidates from the same seed sample could itself produce near-duplicate problems sharing the same underlying technique. Something I didn’t want at all.

Banking. Now before the math expert intervention takes, the generated problem is saved to the database just as it is.

Duplication report. A separate, low-cost model call checks whether the problem just generator has any structural resemblance or roots from the 933 real AIME problems spanning 1983 to 2024. This check is doesn’t work as a filtering gate. It never blocks a problem from proceeding, it only flags a similarity for a curator’s own judgment. Once this check completes, the problem’s status moves to checked, and the fully automatic part of the pipeline stops here.

Expert Verification. At this point, the question is with the expert who requested a generation. The expert will have all the relevant information of the problem. If he/she finds it fit, they manually approve it, and it goes to the final step of verification. This last step is the most computationally expensive step. A fresh model call receives only the problem statement and attempts to solve it from scratch with maximum effort. If it matches the answer, then it finally makes it to the student bank, where it lives forever.

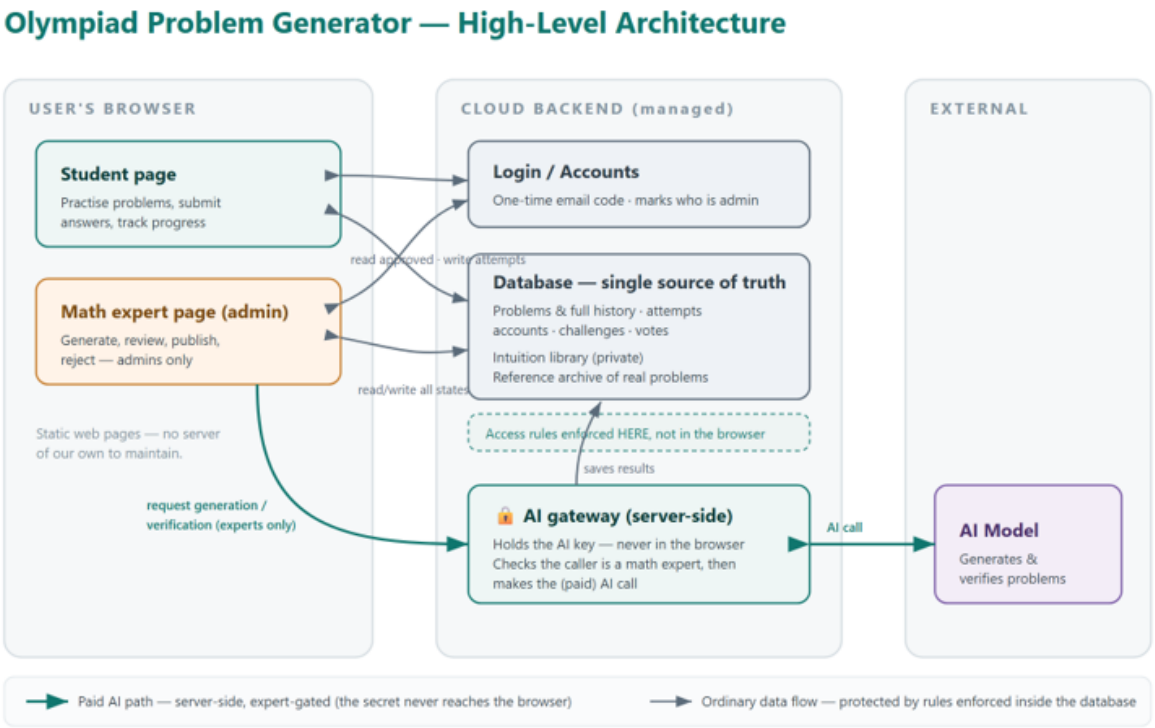


Figure 8: Image generated using AI.

8.1 A comparison with CogAtom

Several of my design choices independently converge with the CogAtom [2] framework discussed earlier. Both systems separate generation and quality control as distinct pipeline stages rather than trusting a single generation call outright. CogAtom's [2] synthesis stage is followed by "comprehensive quality assessment" and rejection sampling before a problem is retained. This is directly analogous to this system's verification step. Both systems also build an explicit mechanism to push generation away from recently used material. CogAtom [2] has a degree-penalized random walk that favors less common atom combinations to avoid predictable and overused conceptual pairings. Since I don't have that much data, I can't replicate the random walk, but I did make sure that the seed selection step explicitly chose fresher sets and not repetitive ones.

I would also like to highlight the differences as well. They're as significant as the parallels. CogAtom's [2] verification and filtering exist to produce large volumes of machine-training data. My system's verification stages exist to give rise to a much smaller number of individual problems that a human student or curator will actually see and use. They are measured by whether a specific problem is mathematically sound and genuinely novel and not by aggregate statistics over thousands of generated items.

8.2 A walkthrough of the expert end.

The reader of this report will only have access to the student's side of the interface. Not the math curator. I will walk through how it looks on the other end.

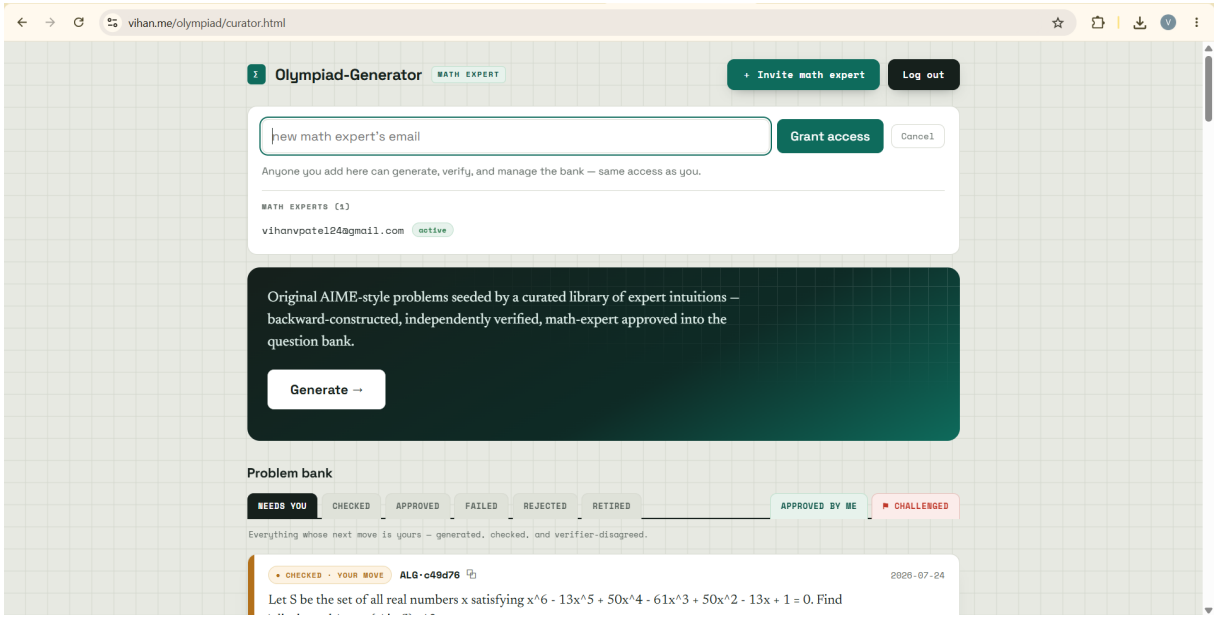


Figure 9: This is how the expert interface looks. A person has access to this only if I enter the email right up there. A big 'Generate' button will begin the pipeline process.

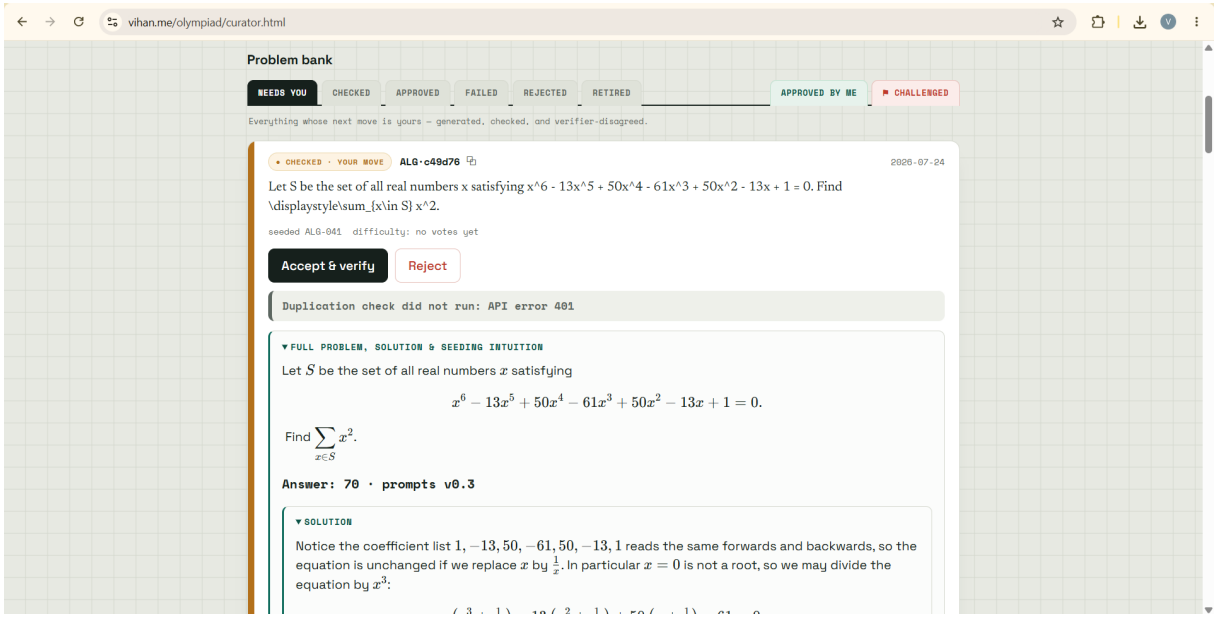


Figure 10: It then generates a problem after 2 to 3 minutes.

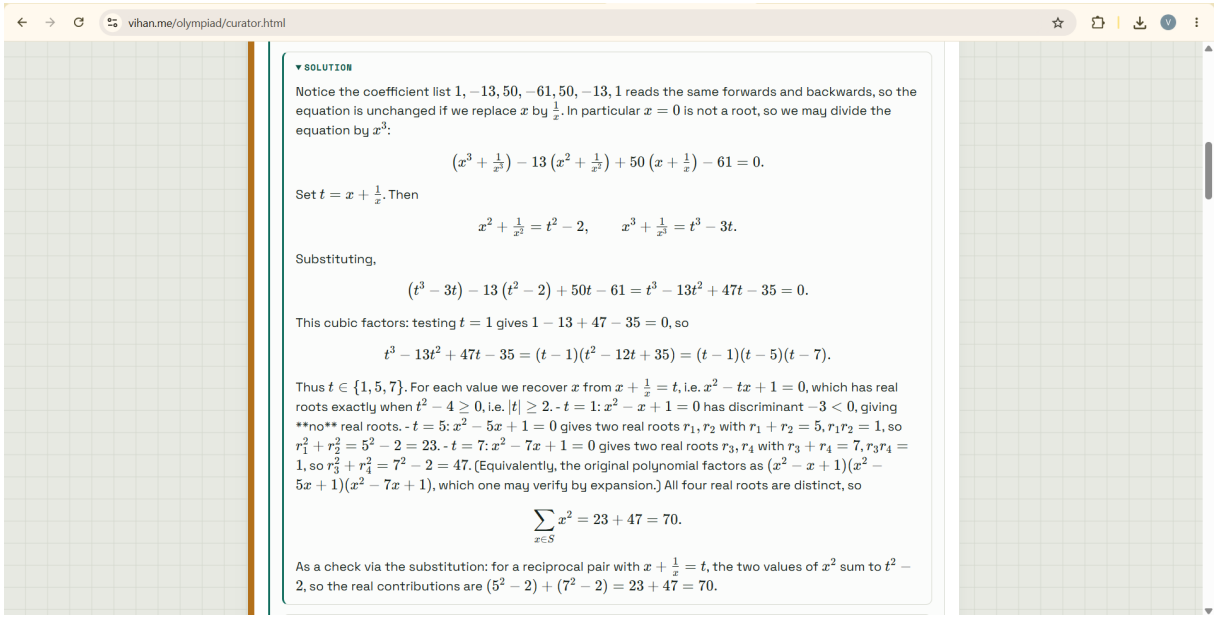


Figure 11: The math expert can independently try and then have a look at the complete step-by-step solution.

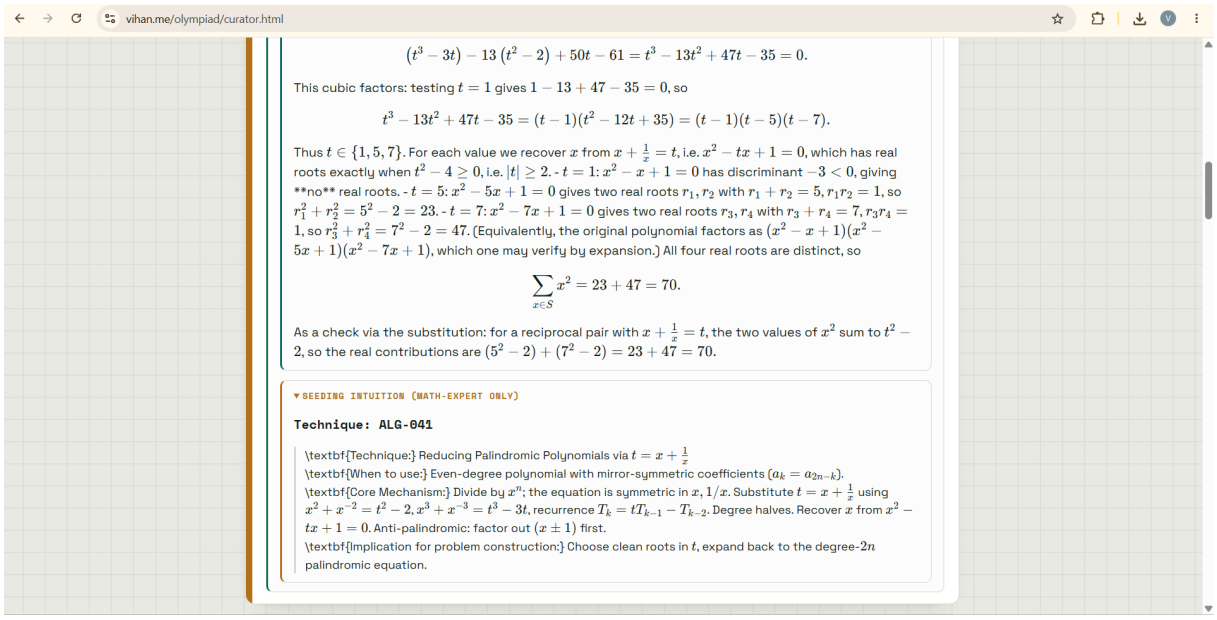


Figure 12: The expert can also have a look at the exact intuition that gave rise to the problem.



Figure 13: The problem is then accepted or rejected. If it gets rejected, it is still saved under a different sub-tab in the interface. Accepted problems then become live and are added to the student question bank.

9 Community Feedback

A community tool must resonate with the community. Hence, I found it valuable to share the resource with people and see what they feel about it. I shared it with three advanced Olympiad friends of mine. I have tried to summarize what each had to say.

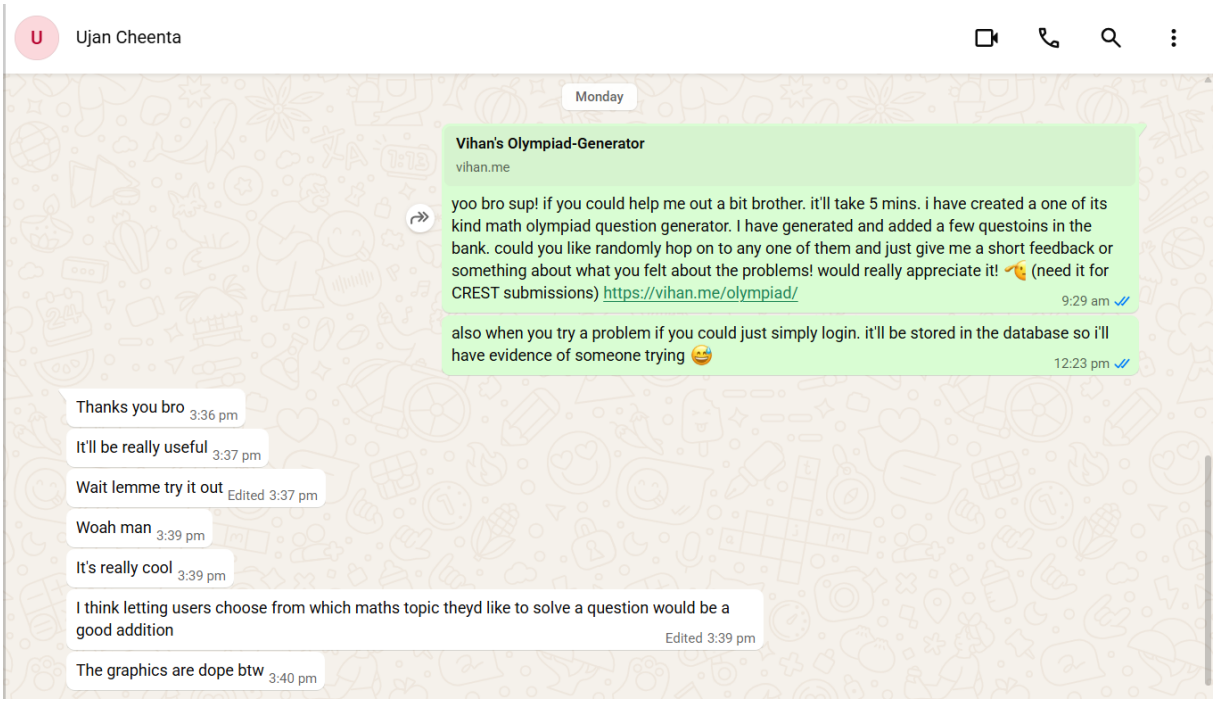


Figure 14: Student 1

This student described the platform as "really cool" and liked the graphics. He also suggested that the Olympiad generator allow students to choose the topic they would like to practice.

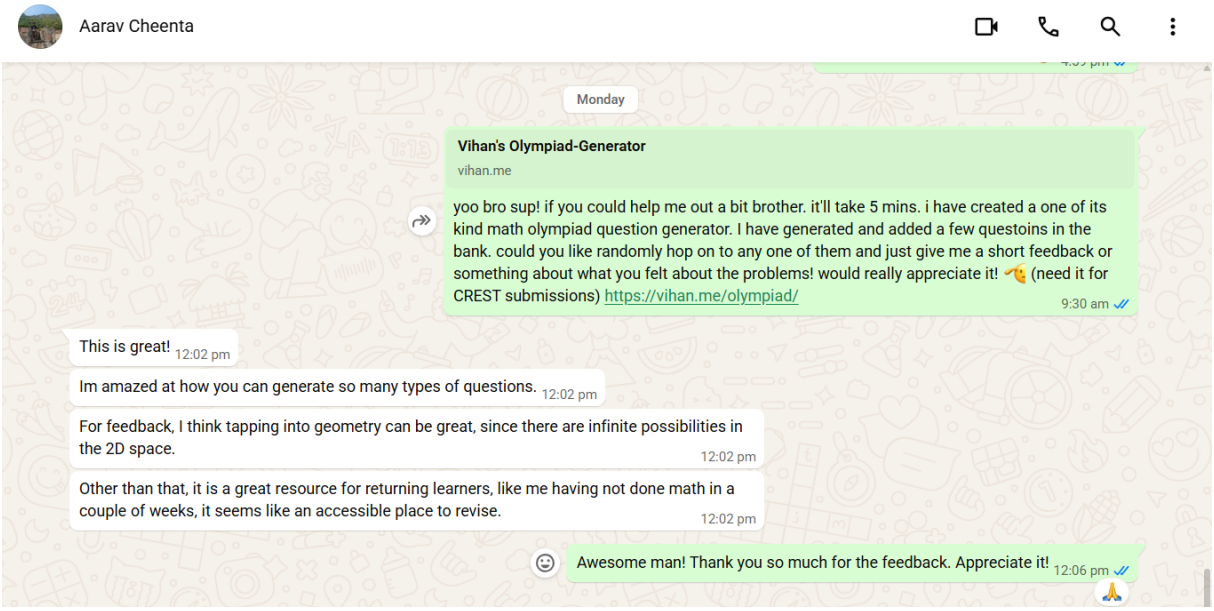


Figure 15: Student 2

A second student found value in this generator as a revision tool. He liked the range of question types the system could generate. A useful suggestion was extending it to Geometry in particular. He reasoned that 2D geometry offers many possibilities for construction.

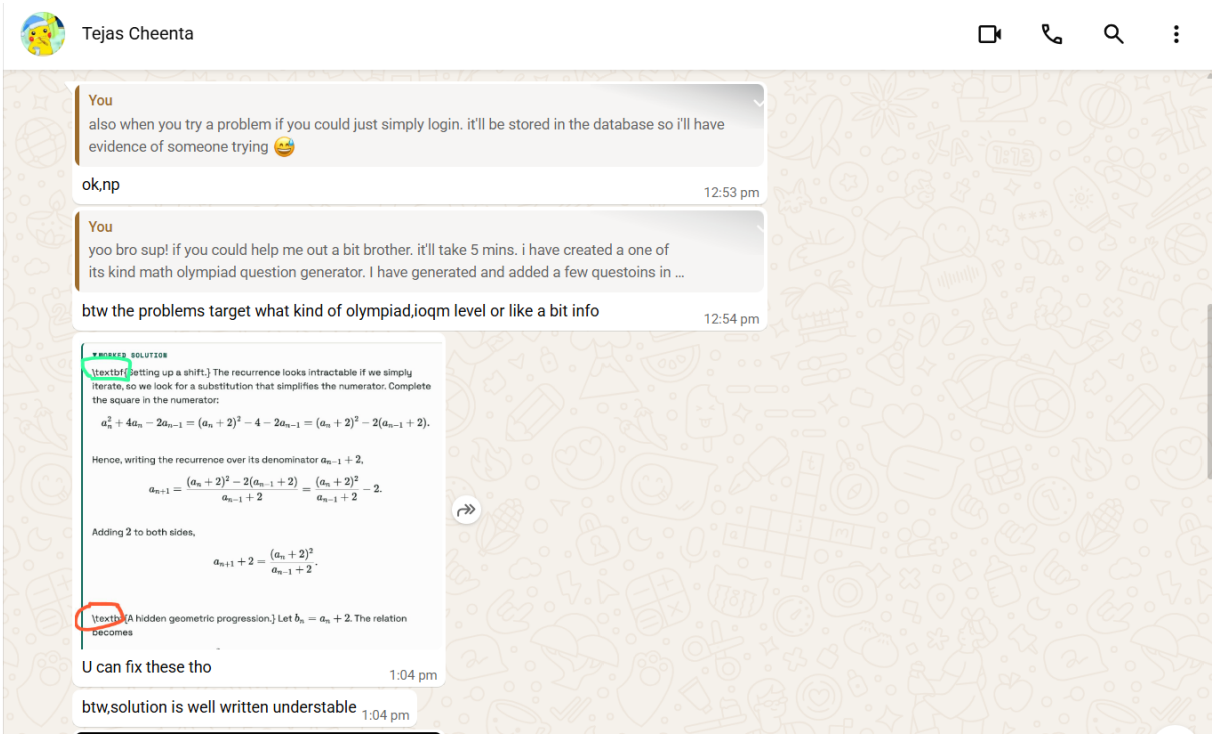


Figure 16: Student 3

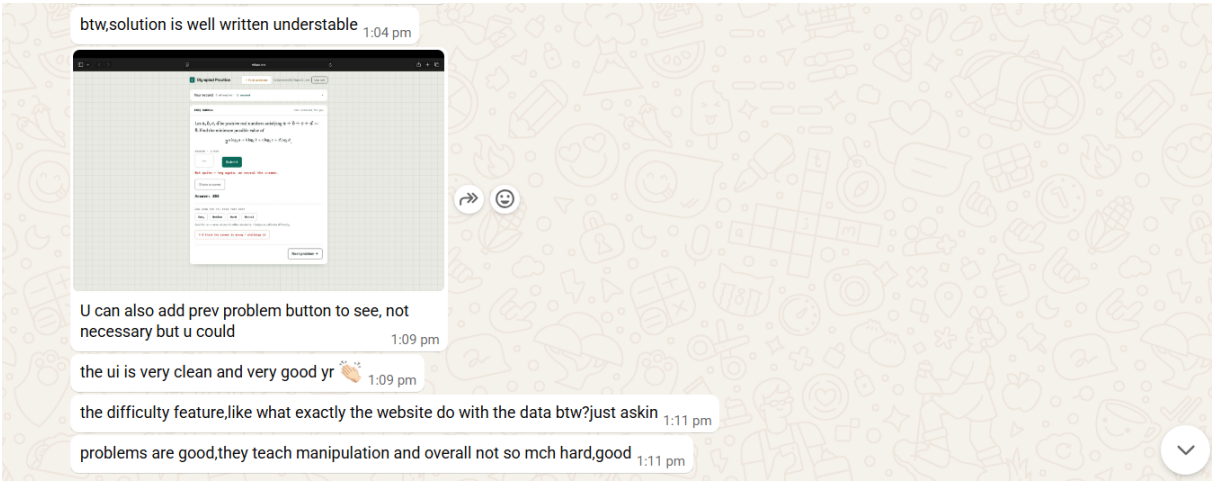


Figure 17: Student 3

A third student had more operational feedback. His assessment of the problems was that they were "good" in the sense that they required true algebraic manipulation, and were "not so much hard". This aligns with the project's intended difficulty. He also caught a rendering bug, which was a stray LaTeX command.

This was a small pilot, and I don't go on to claim that it demonstrates broad community adoption. What it does say is that people outside the project were able to use the generator, found the questions at the intended level, and offered concrete and usable suggestions.

10 Ethical Considerations

My project raises a set of genuine ethical considerations. It requires financial cost, depends on AI systems, and involves other people's time, judgment, and trust.

Financial Sustainability.

Every problem generated costs real money, since it makes multiple calls to the Anthropic API. The pipeline with steps means that it makes around six to seven calls per accepted problem. At this prototype's scale, this comes to 25 to 30 cents per problem, which is not a meaningful constraint at the moment. Having 100 questions generated at the cost of one dollar, assuming that the cost per problem scales linearly and not exponentially, seems to be a reasonable amount. The donor role for this purpose might be sufficient to let the project thrive. However, I still don't know how the costs rise depending on increased requests and user base.

Reliability of AI.

AI models are extremely powerful, no doubt. However, a generated problem can be wrong. If an incorrect solution or a problem makes it to the bank, then it genuinely hampers the quality and has direct ethical consequences. A student practicing with a flawed problem or an unsound proof could internalize incorrect mathematics without realizing it, which is a worse outcome than simply having fewer problems available. This is why I have tried tightening this throughout the project.

One of the steps of the pipeline is the verification. While this is expected to filter out a lot of errors, the verification is itself done by an AI model. This circles me back to the issue of AI creeping in errors. The role of a Math expert comes precisely in at this point. Human review by mathematically qualified people isn't a formality step. They are the actual safeguard against the verification pipeline's blind spots. To add another layer of safeguard, I added a challenge section for the students. After going through a problem and its solution, if the student doesn't find the answer correct or has a better solution, then he/she is welcome to 'Challenge' the question. This challenge directly goes to the curator who approved it. I have made the 'challenging' fairly simple and professional. Exchange of mathematical ideas between students and experts is the foundation of a strong community.

So through this three-step verification process, first by the AI, then by the expert, and followed by any student, I have tried to tackle the single most 'eyebrow-raising' ethical concern of using AI models.

Student encouragement.

The entire purpose of the project is to expand the math community and bring in fresh quality problems. But as with any field, there will always be people at different ends of the spectrum. A problem might be too difficult for student A, but too easy for student B. Student A, looking at the easy rating given by student B, will feel discouraged at the fact that he/she couldn't solve a simple question. This is the reason why the ratings will be hidden from students. It will only be used for the generator feedback and learning purposes.

11 Decisions and Their Impact on the Project

I have, throughout the report, evaluated and justified the decisions and approaches I took. However, in this section, I explicitly make clear the three most important and consequential decisions and how they would have looked different down the line.

Solo tool versus community tool.

The decision to reframe how the generator was to be used was crucial. The three-role structure is what shifted my project from 'a tool that I built for myself' to 'something I designed that is useful to, and usable by, the people beyond me.'

AI Intervention to refine intuitions.

I think this decision was one of the single most crucial ones in terms of technical aspects. My intuitions, as they read, were made for a human to comprehend and internalize. The AI refinement to make them read as a 'seed' rather than an 'instruction' is the reason why the quality of the bank is much higher than what it would have been.

Multi-seed versus single seed

CogAtom [2] utilized the large database it had to create a graph of intuitions it would connect. This decision, personally, wasn't clear to me. I had no idea how two intuitions would stack upon each other. Let alone a chain of them. So I had rejected this decision even without trying, as it wasn't clear to me. However, this could have gone either way with the impact. If the model somehow (still not clear to me) had been able to bring together more than two intuitions, the quality of the problems would have genuinely been a level higher. In hindsight, it could have made the entire pipeline messier and more cluttered.

12 Timeline

The work on this project spanned approximately 13 months. The initial idea came to me in June 2025 and I made this submission in July 2026. I had a lot of other commitments as an IBDP 11th-grade student. So there were technically a few months where I didn't touch the project at all. Table 1 gives an overall shape of the period. Table 2 gives a week-by-week breakdown of the sprint in June and July 2026, where the bulk of the work took place.

Table 2: Project phases, June 2025 – July 2026

Period	**Phase**	**Key Activity**
June 2025	Initial idea	Project concept first considered. Nothing consolidated yet.
July 2025	Research	Encountered the CogAtom paper [2]. Began logging intuitions from personal Olympiad notes.
August 2025	Intuitions logging	Continued logging intuitions from existing notes.
September 2025	*No CREST work — IBDP Y1 Semester 1 examinations.*	
October 2025	Approach 1	Direct-feed generation attempted and discarded.
November 2025	Approach 2	Structured two-step input (formula + intuition + tagging) attempted.
December 2025	Approach 2 (continued)	Structured approach further tested and diagnosed as producing worse output.
January–March 2026	*No CREST work — IBDP Y1 Finals and Internal Assessments.*	
April–May 2026	Approach 3	Intuition rewrite with Claude. Scope narrowed to Algebra and Inequalities. Single-user framing reconsidered toward a three-role platform model.
June 2026	Build and UI refinement	Pipeline implementation and interface work in Claude Code.
July 2026	Report writing and submission	This report drafted, reviewed, and finalized.

Table 3: Detailed breakdown, June–July 2026 sprint

Week	Focus	Output
Wk 1 (early June)	Intuition finalization	Four-part template applied across full dataset. Final classification pass completed.
Wk 2 (mid June)	Pipeline design	Five-step generation pipeline specified. Novelty audit and self-duplication check added as designed additions.
Wk 3 (late June)	Algebra module build	Seed selection, backward construction, banking, and duplication-report steps implemented in Claude Code.
Wk 4 (early July)	Verification	Independent verification step implemented. Hidden numerical verification layer extended to the Inequalities module.
Wk 5 (mid July)	UI refinement	Curator/student-facing interface refined. Problem lifecycle states enforced at the database level.
Wk 6 (late July)	Report writing	Outreach to Olympiad students. Drafted this report and finalized.

13 Conclusions and Limitations

I set out with this project to design and build a working prototype of an AI-assisted system for generating novel math Olympiad problems, as the technical foundation of a free community platform. Judged against that aim, I seem to have succeeded at what it set out to do at this stage. I have also been honest, throughout the report, about what it hasn't been able to do yet. I feel that I have iterated the working and technical aspects of the project sufficiently. Hence, I go ahead with the limitations directly.

First, the depth of the intuition seeds isn't even at all. I would like to make it explicitly clear that the intuition bank has an uneven quality distribution, with half of them being genuinely non-obvious entries and the other half being standard Olympiad results. So with a dataset having uneven depth and a pipeline randomly picking from it, the unevenness in quality of problems generated becomes two-fold. However, even the bad problems are at the par level and hence I would worry less about this. Though it is a genuine limitations.

Second, the scoping of this to only Algebra from the four main ones (others being Combinatorics, Geometry, and Number Theory) is a huge limitation. Covering only a quarter section from the math Olympiad world seems unfair to the community from my end. However, this is a prototype and not the full idea operating at full scale.

If the prototype can hold up, the implications for the wider math community are on the positive side. A working demonstration that a modest, hand-curated and AI refined intuition library can be turned into a source of fresh practice problems bodes well for the community. There is a scope for them to expand it beyond Algebra to their area of specialization (like Algebra for me) by following the same pipeline and design structuring.

Within these honestly stated bounds, the project has done what it set out to do. I have built a working, verifiable, novel-checked generator, on a foundation designed to be extended rather than treated as finished.

14 Personal Reflection

I have learned a great deal of things through this CREST project that go beyond math. I learned through the CogAtom paper how differently the same underlying idea looks when the goal is training a machine learning model versus contributing to the community.

If I did this again, I would have thought earlier and harder about who the platform will actually be for, before I designed anything. I changed the direction of the project twice. First, from a full research build to a leaner one, and second, from a solo tool to a three-role platform. I also realized how both the changes came from the same cause. I started building without having a decent idea of what 'success' meant for this project. I just set out building a library and prompting AI to generate questions. While it is true that every project will deviate a bit from its own idea of 'success', having a mapped-out plan earlier would have helped.

The wider impact is real and modest at the same time with the current prototype. In the future, I would like to extend this further beyond Algebra. Eventually recruit a proper community to curate problems and start spreading the project among more and more students.

References

[1] Art of Problem Solving. 2025 AIME II Problems/Problem 13. AoPS Wiki, 2025. URL https://artofproblemsolving.com/wiki/index.php?title=2025_AIME_II_Problems/Problem_13. Accessed 2026.

[2] Zhuofan Chen, Jiyuan He, Yichi Zhang, Xing Hu, Haoxing Wen, Jun Bai, and Wenge Rong. CogAtom: From cognitive atoms to olympiad-level mathematical reasoning in large language models. *arXiv preprint arXiv:2509.17318*, 2025. URL https://arxiv.org/abs/2509.17318.