Math Routing Agent

This project presents the design and development of a **Math Routing Agent -** an AI-powered application capable of solving diverse math problems using Agentic Retrieval-Augmented Generation (RAG), fallbacks via large language models, and guardrails for content validation. The system smartly routes queries to a knowledge base or language model, ensuring correctness, safety, and user involvement in feedback.

Source Code available at: https://github.com/SudeepS234/Math-Routing-Agent

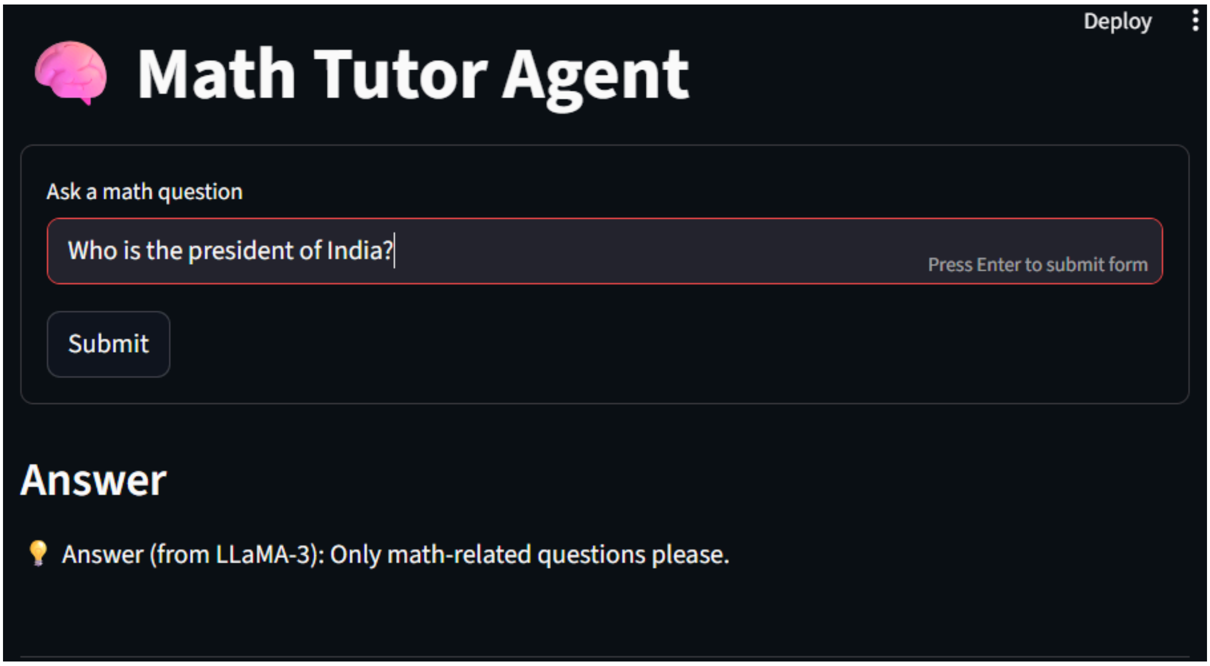
Objectives:

- Knowledge Base(KB) Creation.
- Dynamic Routing to Web Search if questions are not matched with KB.
- Step-by-Step solution to the questions.
- Input and Output Guardrails to allow only Math related inputs and generate outputs strictly related to Math.
- Human Feedback mechanism to evaluate agent answers.
- Deployment on a user friendly Streamlit interface.

1. Input & Output guardrails used for privacy:

Implemented lightweight guardrails using two methods:

- Before processing any question perform a is_math_question() check to ensure math related queries are asked.
- Instructed the model with prompt guardrails such that it should answer like a math professor answering to a student step by step
- The model strictly answers only math related questions and any off topic questions are straight away rejected.



2. Knowledge Base:

Dataset used: Allenai math qa (https://huggingface.co/datasets/allenai/math_qa)

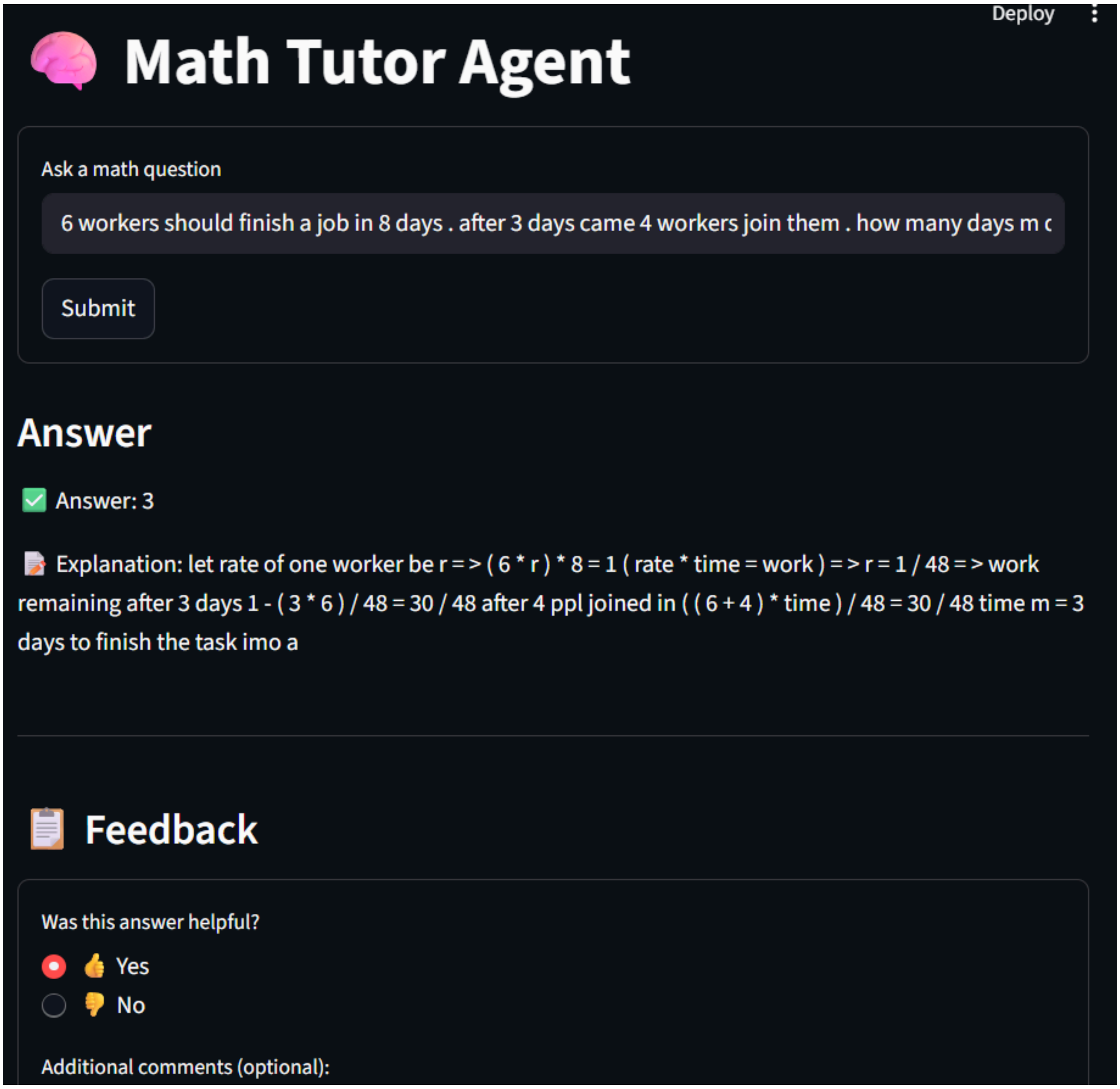
The dataset is a large-scale dataset of math word problems with the below format:

- Problem: a string feature.
- Rationale: a string feature.
- options: a string feature.
- correct: a string feature.
- annotated_formula: a string feature.
- linear_formula: a string feature.
- category: a string feature.

The dataset is converted to the required Knowledge Base format using the script scripts/convert_mathqa_to_kb.py

Results of the system with Knowledge Base questions:

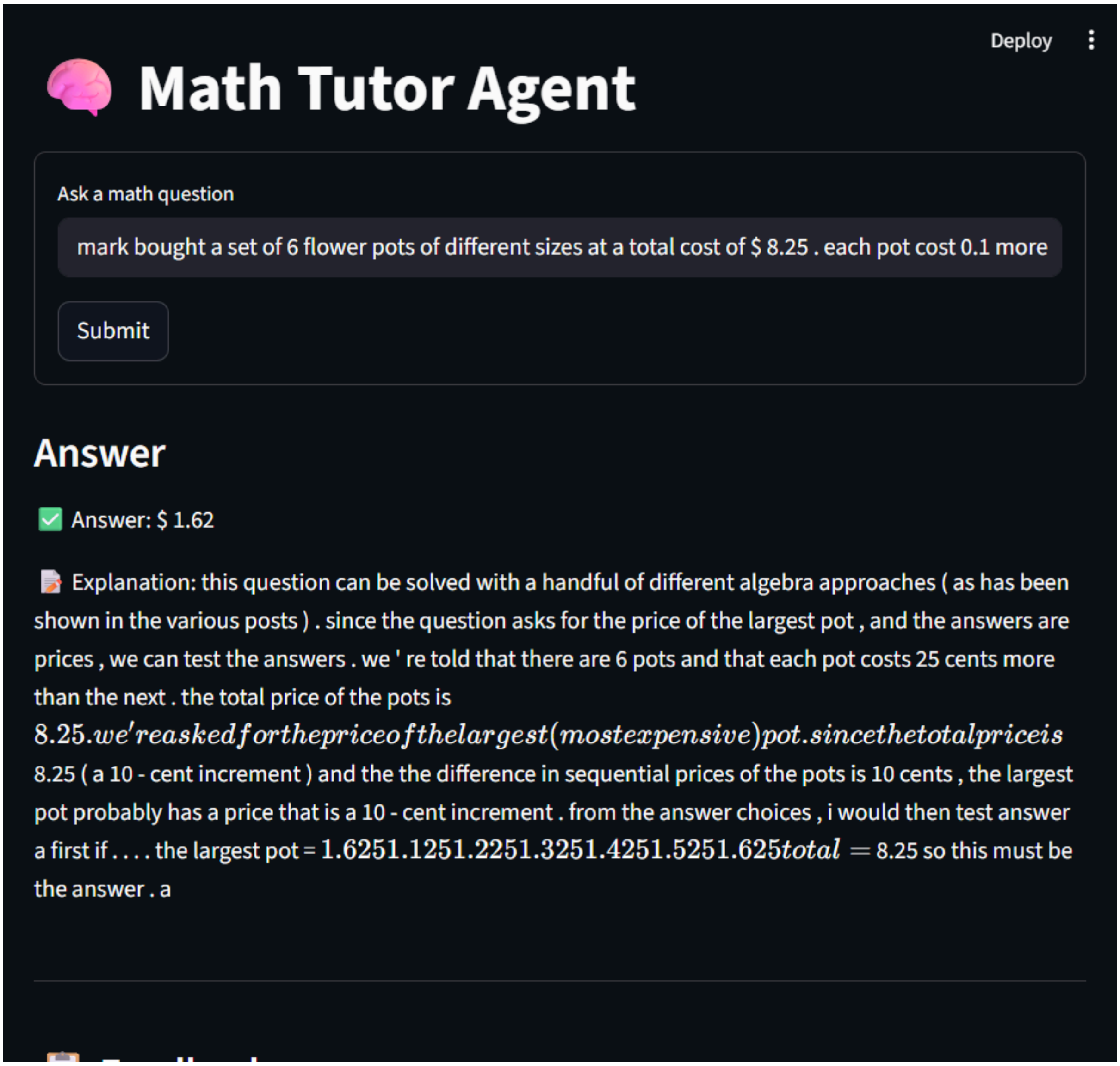
1)



Question in the KB:

```
6 workers should finish a job in 8 days . after 3 days came 4
workers join them . how many days m do they need to finish the
same job ?
Answer: 3
Explanation: let rate of one worker be r = > ( 6 * r ) * 8 = 1 (
rate * time = work ) = > r = 1 / 48 = > work remaining after 3
days 1 - ( 3 * 6 ) / 48 = 30 / 48 after 4 ppl joined in ( ( 6 +
4 ) * time ) / 48 = 30 / 48 time m = 3 days to finish the task
imo a
```

2)


Deploy
Math Tutor Agent
Ask a math question
mark bought a set of 6 flower pots of different sizes at a total cost of $ 8.25 . each pot cost 0.1 more
Submit
Answer
Answer: $ 1.62
Explanation: this question can be solved with a handful of different algebra approaches (as has been shown in the various posts) . since the question asks for the price of the largest pot , and the answers are prices , we can test the answers . we ' re told that there are 6 pots and that each pot costs 25 cents more than the next . the total price of the pots is $8.25 . we're asked for the price of the largest (most expensive) pot . since the total price is$ 8.25 (a 10 - cent increment) and the the difference in sequential prices of the pots is 10 cents , the largest pot probably has a price that is a 10 - cent increment . from the answer choices , i would then test answer a first if . . . . the largest pot = $1.625 1.125 1.225 1.325 1.425 1.525 1.625 total =$ 8.25 so this must be the answer . a

Question in the KB:

```
mark bought a set of 6 flower pots of different sizes at a total
cost of $ 8.25 . each pot cost 0.1 more than the next one below
it in size . what was the cost , in dollars , of the largest pot ?
Answer: $ 1.62
Explanation: this question can be solved with a handful of
different algebra approaches ( as has been shown in the various
posts ) . since the question asks for the price of the largest
pot , and the answers are prices , we can test the answers . we '
re told that there are 6 pots and that each pot costs 25 cents
more than the next . the total price of the pots is $ 8.25 . we '
re asked for the price of the largest ( most expensive ) pot .
since the total price is $ 8.25 ( a 10 - cent increment ) and the
the difference in sequential prices of the pots is 10 cents , the
largest pot probably has a price that is a 10 - cent increment .
from the answer choices , i would then test answer a first
if . . . . the largest pot = $ 1.625 1.125 1.225 1.325 1.425 1.
525 1.625 total = $ 8.25 so this must be the answer . a
```

Qdrant is used to build a collection named “math_kb” which is a vector store containing all the questions, answers and their explanations in the collection format.

A all-MiniLM-L6-v2 sentence transformer model is used as a embedding model and the qdrant client is running on port 6333/6334 using a docker container.

Cosine distance is used as the vector parameter.

```python
from langchain_community.vectorstores import Qdrant
from langchain_community.embeddings import HuggingFaceEmbeddings
from qdrant_client import QdrantClient
from qdrant_client.models import Distance, VectorParams
from dotenv import load_dotenv
import os

load_dotenv()

embedding_model_name = os.getenv("EMBEDDING_MODEL",
"sentence-transformers/all-MiniLM-L6-v2")
embeddings = HuggingFaceEmbeddings(model_name=embedding_model_name)

qdrant_client = QdrantClient(
    url=os.getenv("QDRANT_URL", "http://localhost:6333"),
    api_key=os.getenv("QDRANT_API_KEY")
)

COLLECTION_NAME = os.getenv("QDRANT_COLLECTION", "math_kb")

# Ensure collection exists
if COLLECTION_NAME not in [c.name for c in qdrant_client.
get_collections().collections]:
    qdrant_client.recreate_collection(
        collection_name=COLLECTION_NAME,
        vectors_config=VectorParams(
            size=len(embeddings.embed_query("test")),
            distance=Distance.COSINE
        )
    )

qdrant = Qdrant(
    client=qdrant_client,
    collection_name=COLLECTION_NAME,
    embeddings=embeddings
)
```

The Knowledge Base is built following a pattern:
<Question>
<Answer>
<Explanation>

The format is saved into a text file named math_kb.txt and the vector store is indexed with all of the questions.

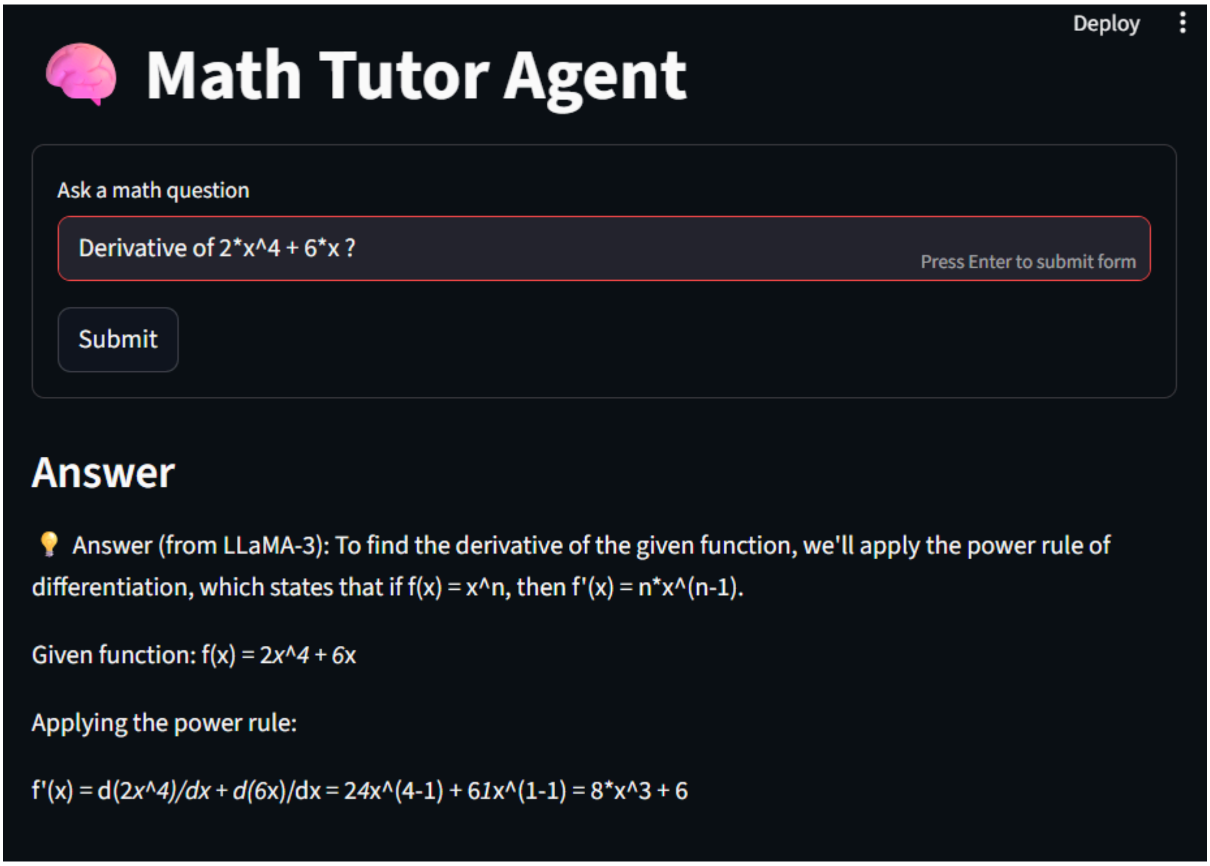
3. Dynamic Routing when similarity is low

A similarity score is calculated for each question in the range from 0 to 1, using threshold as 0.75 and only questions that have similarity score greater than the threshold are found to be present in the Knowledge Base (99% of the time) with very less False Positives.

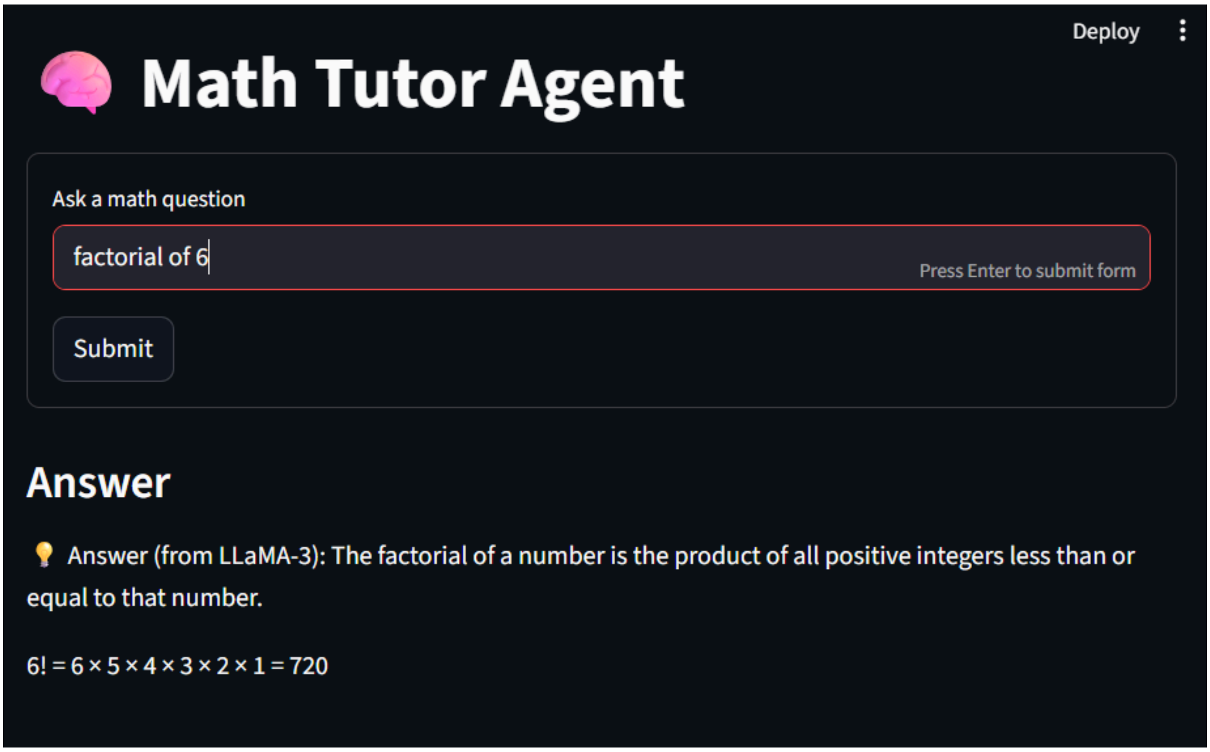
In the case of low similarity score the agent falls back to LLama-3 inference api implemented using Huggingface and requests a step by step answer and the results are displayed by letting the user know that the answers are from an external source.

Few examples of questions outside of KB:

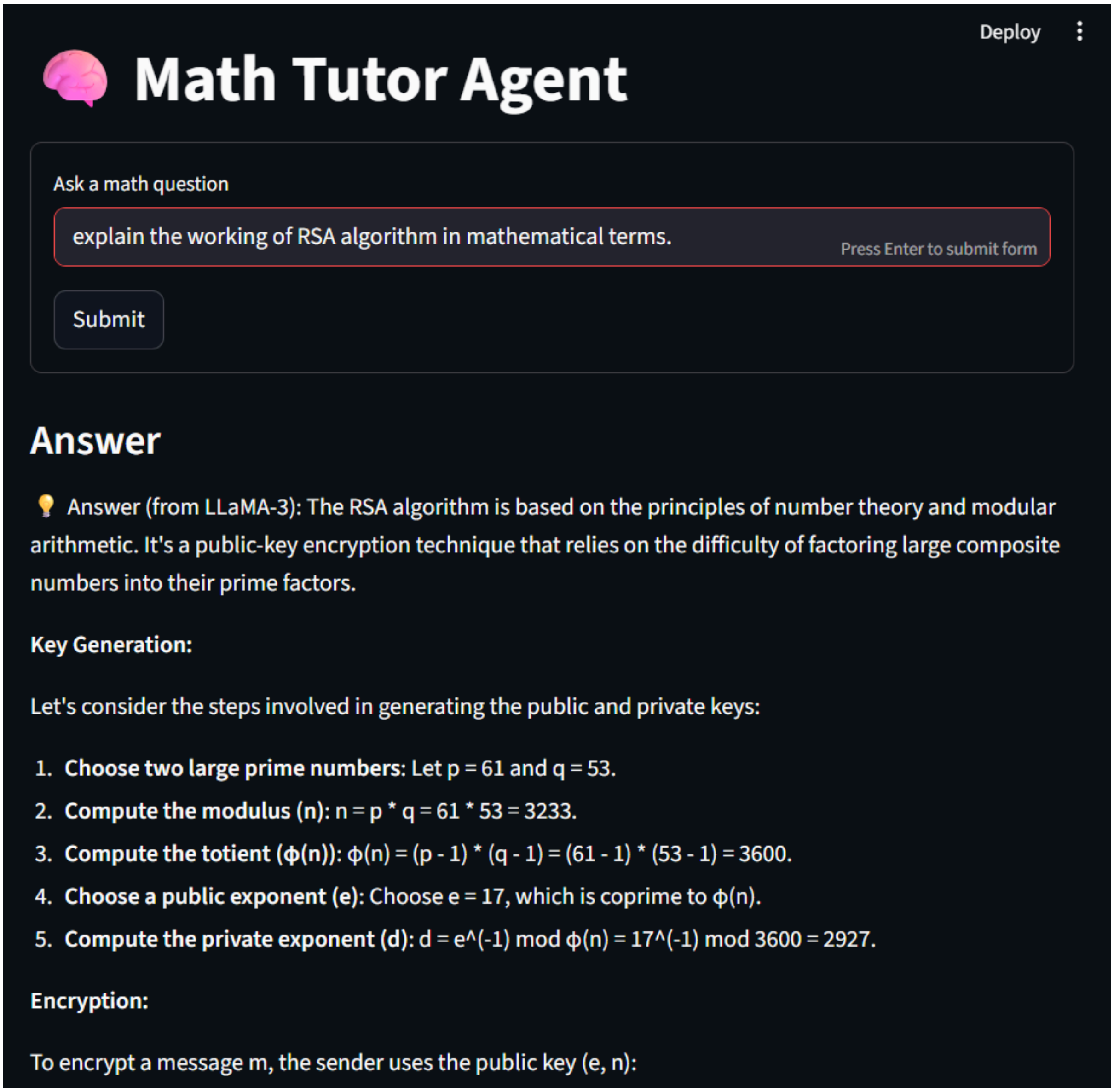
1)



2)


Deploy
Math Tutor Agent
Ask a math question
factorial of 6
Press Enter to submit form
Submit
Answer
Answer (from LLaMA-3): The factorial of a number is the product of all positive integers less than or equal to that number.
6! = 6 × 5 × 4 × 3 × 2 × 1 = 720

3)


Deploy
Math Tutor Agent
Ask a math question
explain the working of RSA algorithm in mathematical terms.
Press Enter to submit form
Submit
Answer
Answer (from LLaMA-3): The RSA algorithm is based on the principles of number theory and modular arithmetic. It's a public-key encryption technique that relies on the difficulty of factoring large composite numbers into their prime factors.
Key Generation:
Let's consider the steps involved in generating the public and private keys:
1. Choose two large prime numbers: Let p = 61 and q = 53.
2. Compute the modulus (n): n = p * q = 61 * 53 = 3233.
3. Compute the totient (ϕ(n)): ϕ(n) = (p - 1) * (q - 1) = (61 - 1) * (53 - 1) = 3600.
4. Choose a public exponent (e): Choose e = 17, which is coprime to ϕ(n).
5. Compute the private exponent (d): d = e^(-1) mod ϕ(n) = 17^(-1) mod 3600 = 2927.
Encryption:
To encrypt a message m, the sender uses the public key (e, n):

Deploy

c = m^e mod n

where c is the ciphertext, m is the plaintext message (a number between 1 and n-1), e is the public exponent, and n is the modulus.

For example, let's encrypt the message m = 123:

c = 123^17 mod 3233 ≈ 2141

Decryption:

To decrypt the ciphertext c, the receiver uses the private key (d, n):

m = c^d mod n

Using the private exponent d = 2927 and the ciphertext c = 2141, we can decrypt the message:

m = 2141^2927 mod 3233 ≈ 123

Mathematical Reasoning:

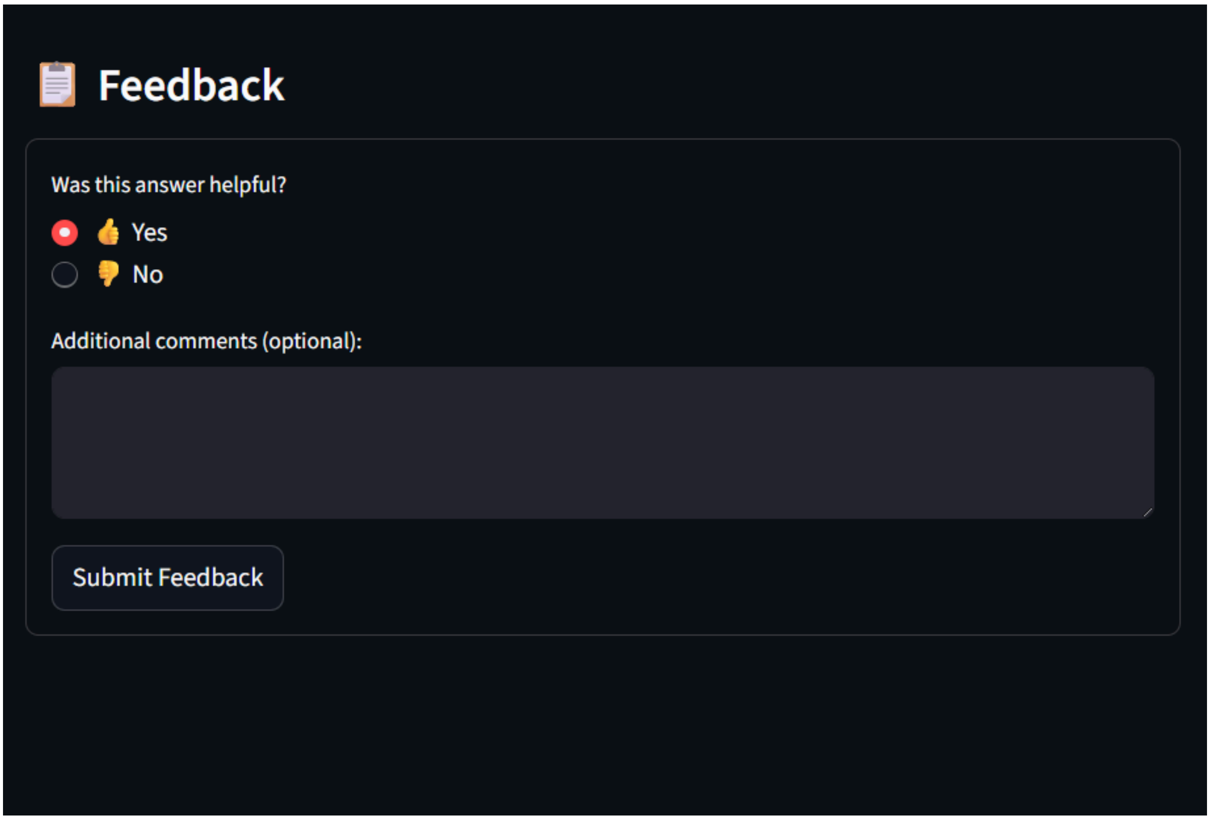
The RSA algorithm relies on the following mathematical properties:

- The difficulty of factoring large composite numbers into their prime factors.
- The property of congruence: a ≡ b (mod n) means a and b leave the same remainder when divided by n.
- The property of modular exponentiation: (a^b) mod n = ((a mod n)^b) mod n.
- The property of Euler's totient function: ϕ(n) is the number of positive integers less than or equal to n that are relatively prime to n.

These properties ensure that the encryption and decryption processes are secure and reliable.

4. Human-in-a-loop mechanism:

Implemented a simple Yes/No feedback with a message which gets logged into a file along with the question and the response the agent generated, further implementation of the mechanism resulted in lot of errors and I’m working on it to implement the DSPy library for the mechanism.



feedback log file:

```
7 factorial,"Answer (from LLaMA-3):
To calculate 7 factorial (denoted as 7!), we multiply all positive
integers from 1 to 7.

Therefore, 7! = 5040.",👍 Yes,could have been more detailed
```

5. JEEBench benchmark:

Benchmark was performed on some randomly selected questions from the JEE Bench dataset and accurate results were found.

Q)


```
{
    "description": "JEE Adv 2018 Paper 2",
    "index": 44,
    "subject": "math",
    "type": "Numeric",
    "question": "Let $P$ be a matrix of order $3 \\times 3$
    such that all the entries in $P$ are from the set $\\{-1,0,
    1\\}$. Then, what is the maximum possible value of the
    determinant of $P$?",
    "gold": "4"
},
```

Result:

Answer

💡 Answer (from LLaMA-3): To find the maximum possible value of the determinant of P, we first need to understand how the determinant of a 3x3 matrix is calculated.

The determinant of a 3x3 matrix [a, b, c; d, e, f; g, h, i] is given by:

a(ei - fh) - b(di - fg) + c(dh - eg)

Since all the entries in P are from the set {-1, 0, 1}, the maximum possible value for any entry is 1. To maximize the determinant, we should try to maximize the individual terms in the determinant formula.

To do this, we should try to have as many terms as possible with a value of 1 and as few terms as possible with a value of -1.

One possible arrangement for the matrix is:

[1, 1, -1; 1, -1, 1; -1, 1, -1]

Using this matrix, we can calculate the determinant:

det(P) = 1(-1*1 - 1*(-1)) - 1(1*1 - 1*(-1)) + 1(1*(-1) - (-1)*1) = 0 - 2 + 0 = -2 (for this matrix)

However, we can try another arrangement to get an even higher value for the determinant:

[1, -1, 1; 1, 1, -1; -1, 1, 1]

Using this matrix, we can calculate the determinant:

det(P) = 1(1*1 - 1*1) - 1(1*1 - (-1)*1) + 1*(1*1 - (-1)*1) = 0 + 2 + 2 = 4 (for this matrix)

Hence, the maximum possible value of the determinant of P is 4.