VISHWAKARMA
UNIVERSITY
Maximising Human Potential

Prompt Engineering Project-2

Name: Siddarth Rajeev

Roll no.: 47

Div: D

SRN: 31240836

🧠 AI-Powered Math Problem Solver

Project Report

1. Project Overview

The AI-Powered Math Problem Solver is an intelligent application that leverages the capabilities of large language models (LLMs) to assist students, educators, and enthusiasts in solving a wide range of mathematical problems. This tool can interpret user queries, understand mathematical expressions, and generate either step-by-step solutions or direct answers based on user preference.

2. Objectives

- To build an interactive, intelligent system that can solve engineering-level math problems.

- To integrate natural language understanding for accepting flexible math queries.

- To support two modes of output: step-by-step explanation and direct answer.

- To enhance accessibility to math help without reliance on traditional calculators or search engines.

3. Technologies Used

- Frontend: React.js + TailwindCSS

- Backend: Node.js

- LLM Integration: OpenAI GPT-4 / Anthropic Claude / Cohere

- Prompt Engineering: Custom-designed prompts based on query type

- Math Formatting: KaTeX / MathJax

- Deployment: Vercel / Netlify / Render

4. Core Features

- Natural Language Input

- Solution Mode Toggle: Step-by-Step or Direct Answer

- Equation Support with LaTeX-style input and output

- Dynamic Re-prompting for clarity and formatting

5. Prompt Engineering Approach

Prompts are dynamically constructed based on query type, user-selected output mode, and include few-shot examples for better context understanding.

Example Prompt (Step-by-Step):
"You are a math tutor. Solve the following step-by-step and format your answer with LaTeX: Integrate $x^2 * \sin(x)\, dx$"

Example Prompt (Direct):
"What is the solution to: Integrate $x^2 * \sin(x)\, dx$? Only provide the final answer in LaTeX."

6. System Architecture

User → React Frontend → Backend API (Node.js) → LLM API (OpenAI)
↑ ↓
Math Renderer ←—- Formatted LLM Output

7. Challenges Faced

- Ensuring math accuracy in responses

- Handling ambiguous or poorly-phrased queries

- Maintaining consistency in step formatting

- Balancing prompt verbosity with response clarity

-Obtaining Api key

8. Conclusion

This project demonstrates how modern AI can revolutionize education by offering intelligent, adaptive, and user-friendly tools for learning and problem-solving. With further enhancements, this math solver could become a powerful companion for learners across the globe.

9. Streamlit App Code (Frontend)

This is the Streamlit-based UI that allows users to type in math problems and get AI-powered answers. It includes modern styling, handles user inputs, toggles between solution modes, and interacts with the OpenRouter API.

```
import streamlit as st
import requests
import os
import textwrap
from dotenv import load_dotenv
load_dotenv()

st.set_page_config(page_title="🧠 Math Solver", layout="centered")
st.markdown("""
  <style>
  html, body, [class*="css"]  {
    font-family: 'Segoe UI', sans-serif;
    background-color: #f5f5f5;
    color: #333333;
  }
  .main {
    padding-top: 30px;
  }
  .stTextInput>div>div>input {
    padding: 12px;
    border-radius: 10px;
  }
  .stButton>button {
    background-color: #4CAF50;
```

```
        color: white;
        padding: 10px 20px;
        border-radius: 10px;
        border: none;
    }
    .stSelectbox>div>div {
        border-radius: 10px;
    }
    </style>
""", unsafe_allow_html=True)

st.title("🧮 Engineering Math Problem Solver")
st.markdown("""
Type any engineering math problem below and get a solution using AI.
Choose whether you want a **step-by-step explanation** or just the **direct answer**.
""")

question = st.text_area("Enter your math problem:", height=150, placeholder="e.g. Solve the differential equation dy/dx + y = e^x")
mode = st.selectbox("Solution Type:", ["Step-by-Step", "Direct Answer"])

api_key = os.getenv("OPENROUTER_API_KEY")
model = "mistralai/mixtral-8x7b-instruct"

headers = {
    "Authorization": f"Bearer {api_key}",
    "HTTP-Referer": "your-app-id-on-openrouter",
    "Content-Type": "application/json"
}

if mode == "Step-by-Step":
    prompt_prefix = "Solve this engineering math problem step-by-step:"
else:
    prompt_prefix = "Give a direct concise answer to this engineering math problem:"

if st.button("🔍 Solve"):
    if not question.strip():
        st.warning("⚠️ Please enter a math problem before clicking Solve.")
    elif not api_key:
        st.error("❌ API key is missing. Please set the OPENROUTER_API_KEY environment variable.")
    else:
        with st.spinner("Solving..."):
            try:
                response = requests.post(
                    "https://openrouter.ai/api/v1/chat/completions",
                    headers=headers,
                    json={
                        "model": model,
                        "messages": [
                            {"role": "user", "content": f"{prompt_prefix} {question}"}
                        ]
                    },
```

```
            timeout=60
        )

        if response.status_code == 200:
            result = response.json()
            answer = result["choices"][0]["message"]["content"]
            answer_wrapped = "\n\n".join(textwrap.wrap(answer, width=100))
            st.markdown("### ✅ Solution")
            st.markdown(f"""<div style='white-space: pre-wrap;'>{answer_wrapped}</div>""",
unsafe_allow_html=True)
        else:
            st.error(f"⚠️ API Error {response.status_code}: {response.text}")
    except requests.exceptions.RequestException as e:
        st.error(f"❌ Request failed: {str(e)}")
```

10. Prompt Engineering for Code Generation

These are prompts that have been used to generate or enhance different parts of the application using ChatGpt:

1. Create a Streamlit app that takes a math problem as input and solves it using an API. Add an option to toggle between step-by-step and direct answers.

2. Build a modern UI in Streamlit for a math solver with styled input boxes and a nice layout.

3. Generate a Python script using Streamlit to call OpenRouter's Mixtral-8x7B model to solve math queries.

4. Write an HTML form that submits a math problem to a Flask backend and displays the returned result cleanly.

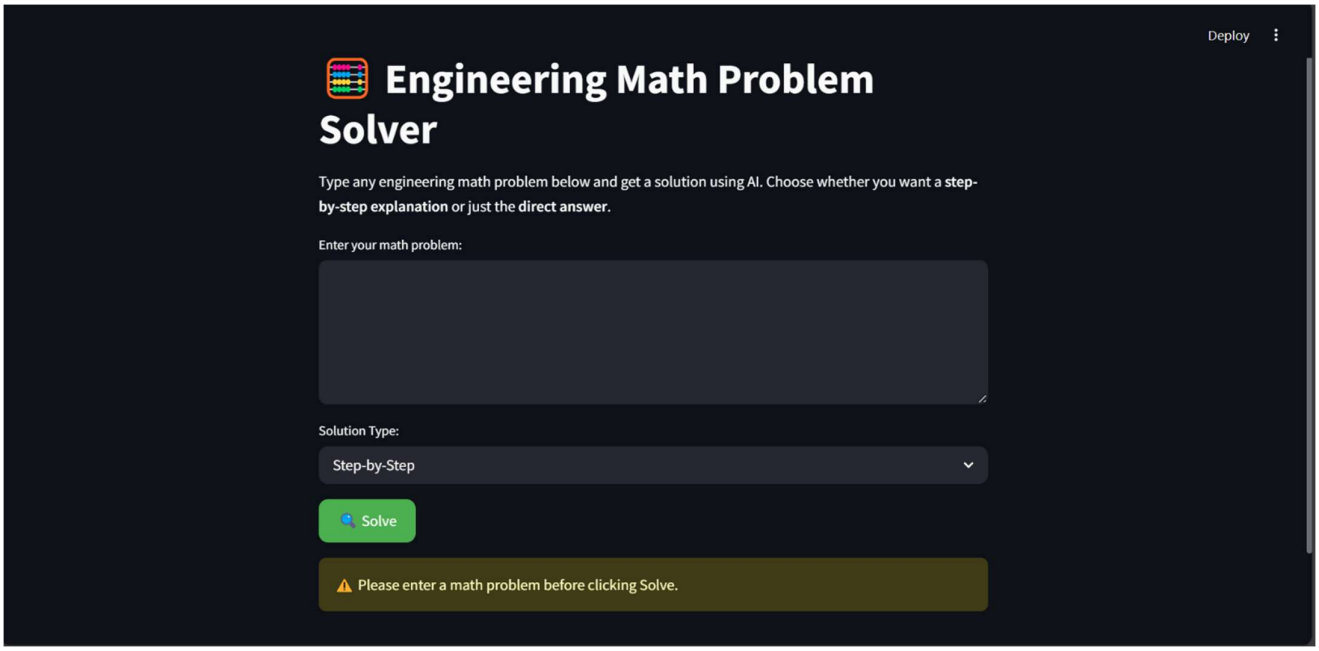
5. Add custom CSS styling to a basic Flask HTML template for a math app with centered layout and styled buttons.

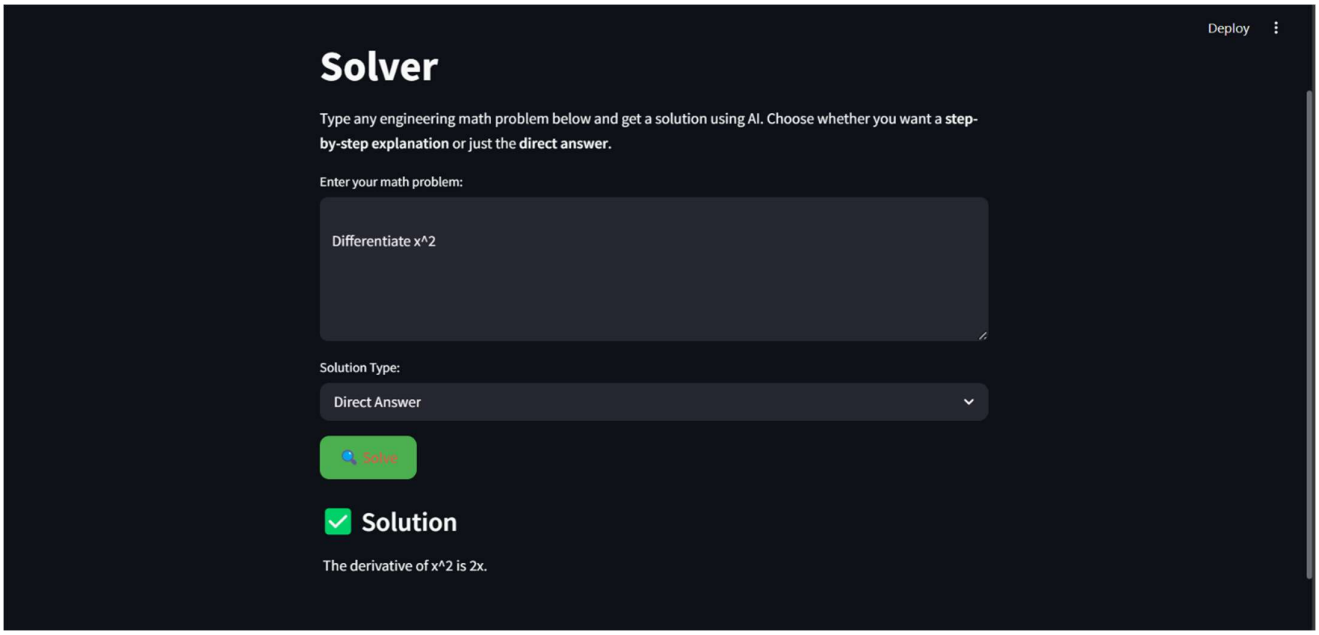
6. Improve the performance and user feedback of a Streamlit app that calls an external API with a spinner and error handling.

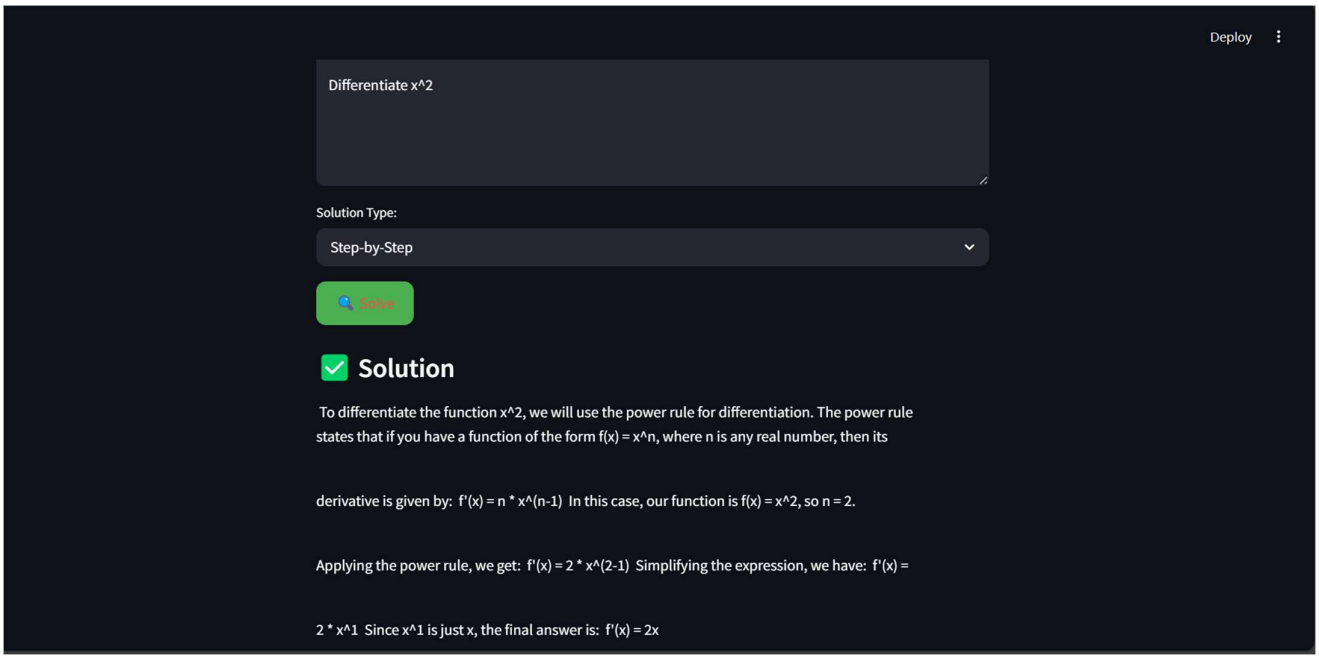
7. Show how to load environment variables (API keys) securely in a Python web app using dotenv.

8. Generate a Streamlit math solver with API error handling, response formatting, and Markdown rendering.

11.App screenshots







YT VIDEO LINKS

https://youtu.be/FdrAJt_cx1o?si=RU9GdoCJtD0TEF6Y