Project: First Principles Problem Solving App

Author: Miro Zilaji, 31.7.2025

1.Brief description of the problem are / outline of the topic

1.1 Content

Thinking based on First Principles has proven to solve many complex problems. Not many people are familiar with First Principle Thinking. People mainly think based on analogy, which leads to "old solutions" for new problems. Using First Principle Thinking we can boost innovation (and also ease solving everyday challenges).

2. Reproducible Steps

This section summarizes the development and design of an interactive Streamlit application to facilitate first principles problem solving for end-users.

2.1 State of Research / Literature Review

Which aspects were examined, and which were not?

- Examined:
  - The effectiveness of stepwise digital guides for first-principles reasoning.
  - Integration of large language models (e.g., OpenAI GPT) as assistants for reasoning, assumption-challenging, and creativity.
  - UI/UX best practices for advancing users through structured thinking (minimizing distractions such as keyboard hints, focusing on button-driven progress, auto-saving outputs).
  - Persistent session-based approaches (using st.session_state) for safe, repeatable workflows.
- Not examined:
  - The long-term educational impact of repeated app use.
  - Direct comparison to human mentors or classic classroom settings.
  - Quantitative user experience studies.

Controversies and Methods Used So Far:

- It remains debated whether AI is most effective as a “creative collaborator” or as a strictly analytical tool when guiding users through first principles thinking.
- Many previous digital apps focus on mindmaps or static frameworks, while recent approaches (like ours) apply generative AI to suggest, question, and provoke.

2.2 Research Question

How can a web-based app, powered by stepwise AI guidance, enable users to apply first principles thinking more effectively to their real-world challenges?

2.3 State of Research

- There are hundreds of web resources and articles on first principles thinking, but relatively few interactive apps that:
    - Integrate an AI model to analyze, challenge, and generate solutions in live user sessions.
    - Guide users with a clear, streamlined UI that reinforces focus and clarity.
    - Address reproducibility by auto-saving session outputs for future analysis or study.

2.4 Knowledge Gap

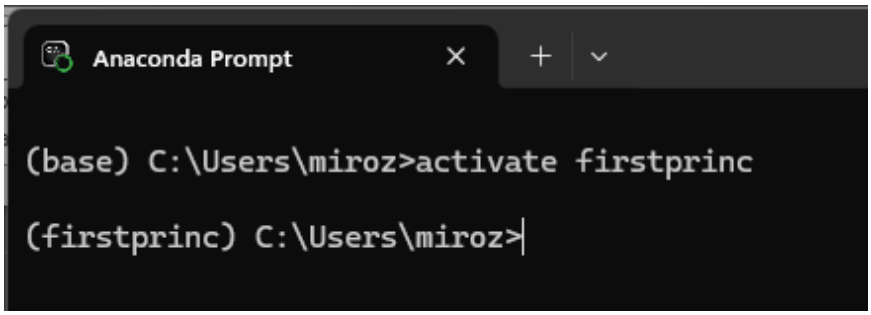
- Existing digital tools for problem solving are either too generic or lack AI-driven, scaffolded guidance that starts from raw problem to assumptions, facts, elements, and creative solutions—all in a reproducible, user-friendly workflow.
- Little published work describes UI/UX techniques specifically for removing cognitive distractions (e.g., keyboard shortcuts, untimely field hints) in reasoning apps.

2.5 Methodology

- **App Design:** Modular, step-by-step UI built with Streamlit and Python, all state managed explicitly for repeatability.
- **AI Integration:** OpenAI's GPT-based API used for deep reasoning, assumption challenge, and creative alternative generation.
- **User Flow:** Users initiate with a problem statement, receive an AI analysis, submit an assumption, trigger a challenge, add facts/elements, and receive AI-generated solutions.
- **Session Saving:** Each session is saved automatically as a timestamped file (*.txt) for easy reproducibility and later review.
- **Accessibility:** No specialized input methods or knowledge required. The app is self-explanatory, with all action moved to explicit buttons. No accidental advancement, and visual cues minimized distractions.
- **Deployment:** Runs locally; code and resulting session files are portable and inspectable. Can be modified for cloud or desktop distribution.

Preparation

Creating new environment (firstprinc) for development of the Application.



The Script

Importing necessary Liabraries

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 30 17:20:46 2025

@author: miroz
"""

# FINAL Version - DON`T TOUCH!!!!!!!


# --- Imports: Core Python, environmental variables, UI, APIs, and date/time for filenames ---
import os
from dotenv import load_dotenv
import streamlit as st
import openai
from datetime import datetime
```

Libraries used:

- **os** - The os module provides a way for Python code to interact with the operating system.
- **dotenv** - allows Python code to load environment variables from a .env file
- **streamlit** - open-source Python framework that lets you build interactive, user-friendly web apps and dashboards
- **openai** - allows you to interact programmatically with OpenAI's language models—including GPT-3, GPT-3.5, and GPT-4—directly from your Python code
- **datetime** - provides classes for manipulating dates and times.

Printing the Top-of-App Infographic (generated by AI for this App)

```
# --- Top-of-app Infographic/Workflow Graphic ---
# Shows the main user flow visually at the very top
st.image("generated-image.png", use_container_width=True)
# CSS tweak to move title up closer under the image
st.markdown("""
    <style>
    .block-container img {
        margin-bottom: 0.5rem !important;
    }
    </style>
""", unsafe_allow_html=True)
```

Image: generated_image.png

Loading environment variables (API key, in this case) and creating a folder for storing session reports

```
# --- Load environment variables (like API keys) from .env file in the current directory ---
dotenv_path = os.path.join(os.getcwd(), '.env')
load_dotenv(dotenv_path=dotenv_path)

# --- Create a folder to store session reports if it doesn't exist ---
SAVE_DIR = r"C:\Users\miroz\first_principles_sessions"
if not os.path.exists(SAVE_DIR):
    os.makedirs(SAVE_DIR)
```

Giving Prompts to the model

```python

# --- The core "FirstPrinciplesSolver" class: Handles all OpenAI API interactions ---
class FirstPrinciplesSolver:
    def __init__(self):
        # Try to get the OpenAI API key from environment. Halt app if not present.
        api_key = os.getenv("OPENAI_API_KEY")
        if not api_key:
            st.error(
                "OPENAI_API_KEY is missing. "
                "Your .env must be in the SAME folder and contain: OPENAI_API_KEY=sk-...")
            raise RuntimeError("OPENAI_API_KEY missing")
        # Set up OpenAI client for later prompt calls
        self.client = openai.OpenAI(api_key=api_key)

    def analyze_problem(self, problem):
        # Compose detailed prompt for first-principles analysis.
        prompt = f"""
        You are an expert in first-principles thinking. Analyze this problem: "{problem}"
        Provide structured analysis with:
        1. Problem Domain: What category does this belong to?
        2. Fundamental Elements: 4-5 core components that drive this problem
        3. Hidden Assumptions: 2-3 assumptions the person might not realize they're making
        4. Key Questions: 3 specific questions to help them think deeper
        Be practical and specific. Format your response clearly. If you don´t
        know the answer, say so. Don´t improvise.

        """
        try:
            # Call OpenAI API, get model response, extract content only (not full metadata)
            response = self.client.chat.completions.create(
                model="gpt-3.5-turbo",
                messages=[{"role": "user", "content": prompt}],
                max_tokens=400, temperature=0.7
            )
            return response.choices[0].message.content
        except Exception as e:
            return f"Error analyzing problem: {e}"
```

The model utilizes OpenAI gpt-3.5-turbo Model (through API Key). I have given the model a role (“expert in first-principles thinking”) and delegated following tasks:

Provide structured analysis with:

1. **Problem Domain: What category does this belong to?**
2. **Fundamental Elements: 4-5 core components that drive this problem**
3. **Hidden Assumptions: 2-3 assumptions the person might not realize they're making**
4. **Key Questions: 3 specific questions to help them think deeper**

And given guidance:

Be practical and specific. Format your response clearly.

I have specifically addressed the situation if the model does not know the answer. In that case it should just say so. Under no circumstances it is allowed to make something up (and/or improvise):

If you don´t know the answer, say so. Don´t improvise.

Using AI to challenge assumptions given by user and generate creative solutions

```
83
84      def challenge_assumption(self, assumption, context):
85          # Prompt to critically challenge an assumption via first principles
86          prompt = f"""
87          Challenge this assumption: "{assumption}"
88          In the context of: "{context}"
89          Generate 2-3 thought-provoking questions that:
90          - Question why this assumption exists
91          - Explore what would happen if it weren't true
92          - Suggest alternative perspectives
93          Be specific and practical.
94          """
95          try:
96              response = self.client.chat.completions.create(
97                  model="gpt-3.5-turbo",
98                  messages=[{"role": "user", "content": prompt}],
99                  max_tokens=200, temperature=0.8
100             )
101             return response.choices[0].message.content
102         except Exception as e:
103             return f"Error challenging assumption: {e}"
104
105     def generate_solutions(self, problem, facts, elements):
106         # Prompt the AI for unconventional, creative, first-principles solutions
107         prompt = f"""
108         Using first-principles thinking, generate creative solutions for: "{problem}"
109         Given these facts: {facts}
110         And these key elements: {elements}
111         Suggest 3-4 unconventional approaches that:
112         1. Question the problem definition
113         2. Eliminate unnecessary constraints
114         3. Recombine elements in new ways
115         4. Draw inspiration from other domains
116         Be specific and actionable.
117         """
118         try:
119             response = self.client.chat.completions.create(
120                 model="gpt-3.5-turbo",
121                 messages=[{"role": "user", "content": prompt}],
122                 max_tokens=400, temperature=0.8
123             )
124             return response.choices[0].message.content
125         except Exception as e:
126             return f"Error generating solutions: {e}"
127
```

Challenging Assumptions

Generate 2-3 thought-provoking questions that:

- Question why this assumption exists

- Explore what would happen if it weren't true

- Suggest alternative perspectives

Be specific and practical.

Generating Creative Solutions based on First Principles Thinking

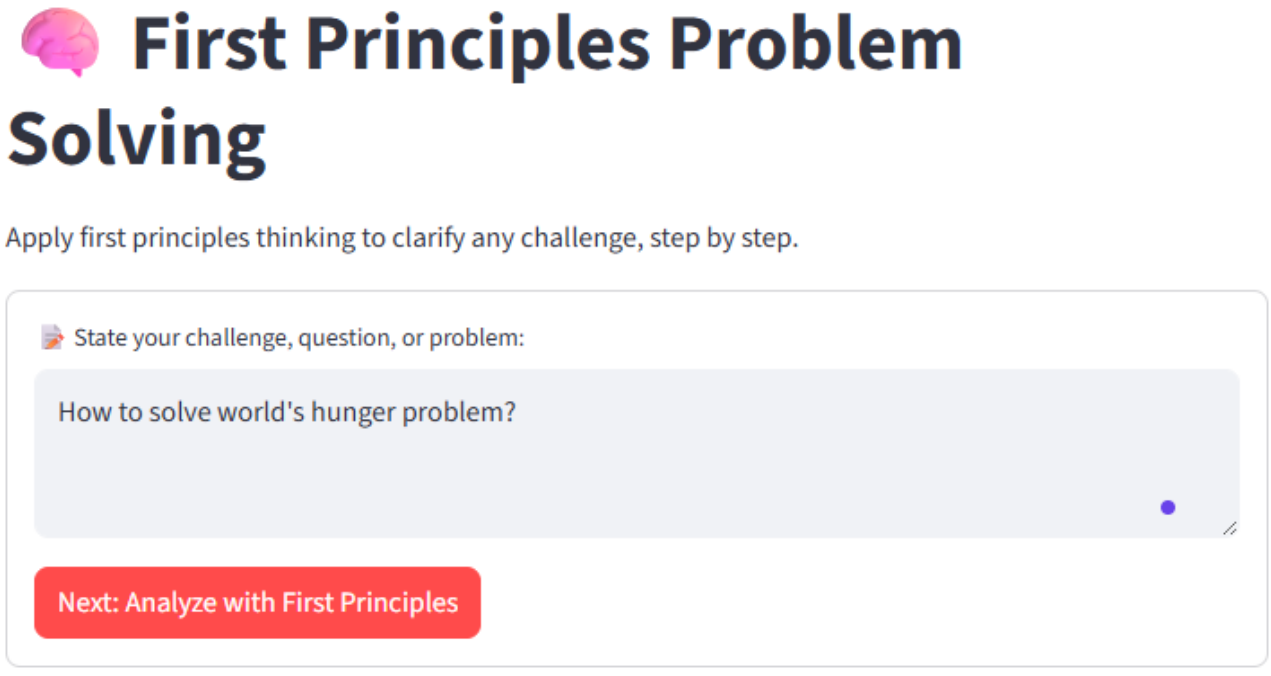
Suggest 3-4 unconventional approaches that:

1. **Question the problem definition**
2. **Eliminate unnecessary constraints**
3. **Recombine elements in new ways**
4. **Draw inspiration from other domains**

Be specific and actionable.

Setting up UI (User Interface) using streamlit

```
127
128  # --- Streamlit page config and headers ---
129  st.set_page_config(page_title="First Principles Problem Solving", page_icon="🧠", layout="centered")
130  st.title("🧠 First Principles Problem Solving")
131  st.write("Apply first principles thinking to clarify any challenge, step by step.")
132
133  # --- Set up and verify the solver (API connection) ---
134  try:
135      solver = FirstPrinciplesSolver()
136  except Exception:
137      st.stop()  # If missing key, stop the app, error already shown
```



Defining variables for different states

```
138
139  # --- Define all session state variables needed for full dialog flow and reset ---
140  defaults = {
141      "problem_input": "",            # The user's core problem statement
142      "analysis": None,               # AI's first-principles analysis output
143      "assumption_input": "",         # User's stated hidden assumption
144      "challenge": None,              # AI's assumption challenge
145      "facts_input": "",              # User's list of relevant facts
146      "elements_input": "",           # User's list of main elements/drivers
147      "solutions": None,              # AI's creative solutions
148      "show_exit": False,             # Whether to display exit instructions
149  }
150  for k, v in defaults.items():
151      if k not in st.session_state:
152          st.session_state[k] = v
153
```

Added option to run the application again

```
153
154  # --- Cross-version helper: rerun the app (after updating session state) ---
155  def rerun():
156      try:
157          st.rerun()    # Streamlit >=1.27
158      except AttributeError:
159          st.experimental_rerun()  # Streamlit <1.27
160
```

User input „Problem/Challenge“

```
160
161  # --------------- Multi-step interactive UI ---------------
162  # STEP 1: User enters the problem statement
163  if not st.session_state.analysis:
164      with st.form("problem_form", clear_on_submit=False):
165          problem = st.text_area(
166              "📂 State your challenge, question, or problem:",
167              value=st.session_state.problem_input,
168              help=" "  # disables input hint
169          )
170          # User clicks button to submit the problem (no accidental submit on Enter)
171          next1 = st.form_submit_button("Next: Analyze with First Principles", type="primary")
172          if next1 and problem.strip():
173              # Show spinner while AI processes input
174              with st.spinner("Analyzing your problem with first principles..."):
175                  analysis = solver.analyze_problem(problem)
176              # Store everything; clear downstream session state for proper step flow
177              st.session_state.problem_input = problem
178              st.session_state.analysis = analysis
179              st.session_state.assumption_input = ""
180              st.session_state.challenge = None
181              st.session_state.facts_input = ""
182              st.session_state.elements_input = ""
183              st.session_state.solutions = None
184              st.session_state.show_exit = False
185              rerun()  # Immediately rerun, now displaying the next step
```

Application shows general overview of the problem from the First Principles standpoint

```
# STEP 2: Show analysis, ask for one assumption, collect and handle via button
elif st.session_state.analysis and not st.session_state.challenge:
    st.markdown("### First Principles Analysis")
    st.write(st.session_state.analysis)
    st.markdown("---")
    with st.form("assumption_form", clear_on_submit=False):
        assumption = st.text_input(
            "🤔 State one assumption you're making about this problem:",
            value=st.session_state.assumption_input,
            help=" "
        )
        next2 = st.form_submit_button("Next: Challenge Assumption", type="primary")
        if next2 and assumption.strip():
            with st.spinner("Challenging your assumption (first principles)..."):
                challenge = solver.challenge_assumption(assumption, st.session_state.problem_input)
            st.session_state.assumption_input = assumption
            st.session_state.challenge = challenge
            st.session_state.facts_input = ""
            st.session_state.elements_input = ""
            st.session_state.solutions = None
            st.session_state.show_exit = False
            rerun()
```

General Analysis of the problem

🧠 First Principles Problem Solving

Apply first principles thinking to clarify any challenge, step by step.

First Principles Analysis

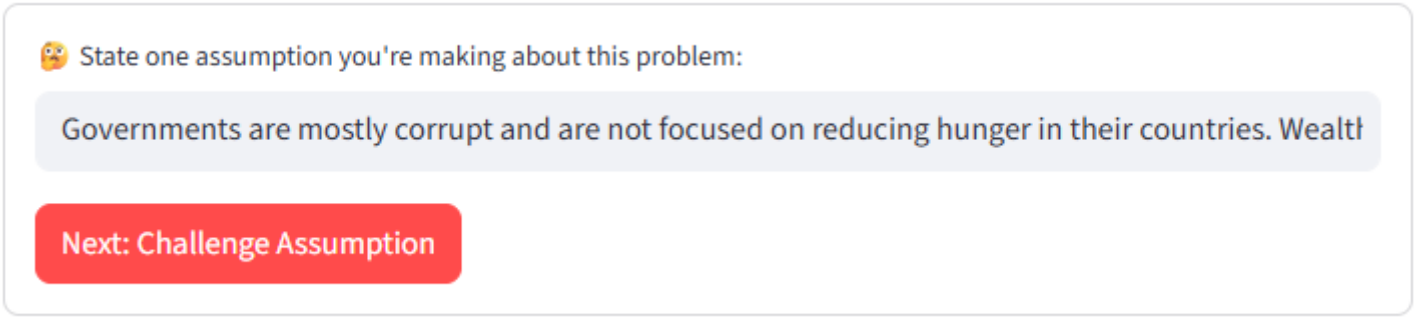
1. Problem Domain: This problem belongs to the domain of global food insecurity and poverty.

2. Fundamental Elements: a. Limited access to nutritious food: Many people around the world do not have access to enough food or lack the resources to purchase nutritious food. b. Unequal distribution of resources: Food production and distribution systems are often inefficient or unequal, leading to food waste in some areas while others suffer from hunger. c. Poverty and economic disparities: Poverty is a major driving factor behind food insecurity, as people living in poverty often cannot afford an adequate diet. d. Climate change and environmental factors: Environmental degradation, climate change, and natural disasters can disrupt food production and lead to food shortages in vulnerable regions.

3. Hidden Assumptions: a. Assumption: There is a scarcity of food resources globally. b. Assumption: Solving hunger is solely a matter of increasing food production. c. Assumption: The problem can be solved through charity and short-term aid rather than addressing root causes.

4. Key Questions: a. How can we address the systemic issues of poverty and unequal distribution of resources that contribute to food insecurity? b. What sustainable agricultural practices and technologies can be implemented to increase food production without harming the environment? c. How can we empower communities to become self-sufficient in food production and reduce their dependence on external aid?

User enters assumptions she/he is having



Application challenges assumptions, user enters relevant facts about the situation and list of main elements/drivers.

```
                rerun()

# STEP 3: Show challenge, ask for facts/elements; collect user input, process on button
elif st.session_state.challenge and not st.session_state.solutions:
    st.markdown("### Assumption Challenge")
    st.write(st.session_state.challenge)
    st.markdown("---")
    with st.form("facts_form", clear_on_submit=False):
        facts = st.text_area(
            "📊 List relevant facts you know about this situation:",
            value=st.session_state.facts_input,
            help=" "
        )
        elements = st.text_area(
            "🧩 List the main elements or drivers involved:",
            value=st.session_state.elements_input,
            help=" "
        )
        next3 = st.form_submit_button("Next: Generate First Principles Solutions", type="primary")
        if next3 and facts.strip() and elements.strip():
            with st.spinner("Generating creative first-principles solutions..."):
                solutions = solver.generate_solutions(
                    st.session_state.problem_input,
                    facts, elements
                )
            st.session_state.facts_input = facts
            st.session_state.elements_input = elements
            st.session_state.solutions = solutions
            rerun()
```

🧠 First Principles Problem Solving

Apply first principles thinking to clarify any challenge, step by step.

Assumption Challenge

1. Why do we automatically assume that governments are mostly corrupt when it comes to addressing hunger in their countries? What evidence or examples can we look at to challenge this assumption and highlight instances where governments have successfully tackled hunger issues?
2. What would happen if we shifted our focus away from the belief that wealthy countries are solely focused on increasing wealth, and instead looked at the various humanitarian efforts and initiatives they have undertaken to address hunger on a global scale?
3. How can we reframe our perspective on corruption in humanitarian organizations to acknowledge the vast majority of organizations that are dedicated to genuinely reducing hunger and improving global food security, and how can we support and amplify their efforts in solving the world's hunger problem?

📊 List relevant facts you know about this situation:

One of main reasons for hunger in the world is lack of natural resources in those countries. Developed countries are sponsoring help, but often there is not proper control how (and where) the money was spent.

🧩 List the main elements or drivers involved:

Lack of natural resources in those countries.
Global warming is making the situation worse.
Corrupt national governments in those countries.
Profit driven developed countries (and global corporations).

Next: Generate First Principles Solutions

Application presents a set of executable creative solutions for the problem

```
237
238     # STEP 4: All steps complete: show all results/answers and save .txt report, with extra options/buttons
239     elif st.session_state.solutions:
240         # Display all results (analysis, challenge, solutions) for review
241         st.markdown("### First Principles Analysis")
242         st.write(st.session_state.analysis)
243         st.markdown("### Assumption Challenge")
244         st.write(st.session_state.challenge)
245         st.markdown("### Creative Solutions")
246         st.write(st.session_state.solutions)
247         st.markdown("---")
248
249         # Format the full report for downloading and saving to disk
250         session_txt = f"""First Principles Problem Solving Session
251     ======================================
```

Creative Solutions

1. Question the problem definition: Instead of focusing solely on providing food aid to solve hunger in the world, we could shift our perspective to focus on sustainable agricultural practices and infrastructure development in these countries. By investing in long-term solutions that empower local communities to grow their own food, we can address the root causes of hunger.

2. Eliminate unnecessary constraints: Instead of relying solely on government aid and funding, we could explore alternative sources of funding such as impact investing and crowdfunding. This would enable more transparency and accountability in how the funds are being utilized, reducing the risk of corruption.

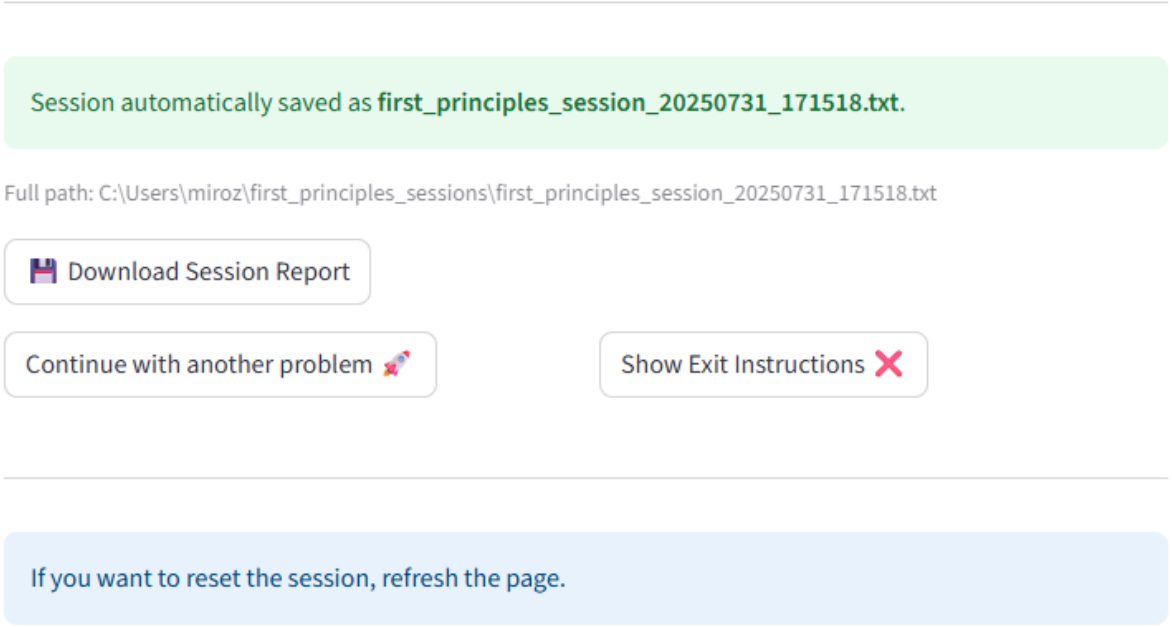
3. Recombine elements in new ways: We could leverage technology such as blockchain to track the flow of funds and resources in real-time, ensuring that they are reaching the intended recipients and being used effectively. This would also enable donors to have more visibility and control over their contributions.

4. Draw inspiration from other domains: We could draw inspiration from the concept of food sovereignty, which emphasizes the right of people to define their own food and agriculture systems. By empowering local communities to take control of their food production and distribution, we can create more sustainable and resilient food systems that are less vulnerable to external factors such as global warming and profit-driven interests.

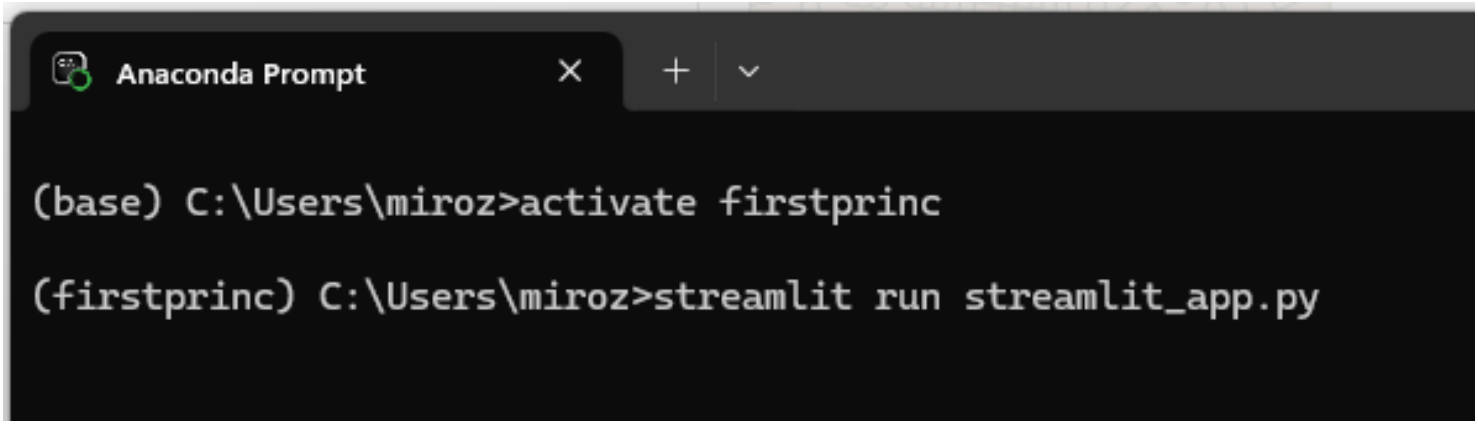
Utilities (writing report in txt file, allow user to download the report, click to proceed with another problem or asks for instructions how to close the application).

```
252
253     Problem: {st.session_state.problem_input}
254
255     First Principles Analysis:
256     {st.session_state.analysis}
257
258     Challenged Assumption: {st.session_state.assumption_input}
259     Challenge Response:
260     {st.session_state.challenge}
261
262     Creative Solutions:
263     {st.session_state.solutions}
264     """
265         # Write .txt session to the fixed folder with a time-stamped filename
266         timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
267         filename = f"first_principles_session_{timestamp}.txt"
268         abs_path = os.path.join(SAVE_DIR, filename)
269         try:
270             with open(abs_path, "w", encoding="utf-8") as f:
271                 f.write(session_txt)
272             st.success(f"Session automatically saved as **{filename}**.")
273             st.caption(f"Full path: {abs_path}")
274         except Exception as e:
275             st.error(f"Could not save file: {e}")
276
277         # Let user also download it instantly from browser if desired
278         st.download_button("💾 Download Session Report", session_txt, file_name=filename
279
280         # Two columns: continue (reset session_state), exit (show instructions)
281         col1, col2 = st.columns(2)
282         with col1:
283             if st.button("Continue with another problem 🚀"):
284                 for k in list(st.session_state.keys()):
285                     st.session_state[k] = defaults.get(k, None)  # reset to initial stat
286                 rerun()
287         with col2:
288             if st.button("Show Exit Instructions ❌"):
289                 st.session_state.show_exit = True
290
291         # If "exit" is pressed, explain how to close the tab and stop Streamlit
292         if st.session_state.show_exit:
293             st.info(
294                 "To close the app: **close this browser tab/window**. "
295                 "Then, in your terminal, press **Ctrl+C** to stop the server."
296             )
297
298     # --- UI/UX tip at the bottom at all times ---
299     st.markdown("---")
300     st.info("If you want to reset the session, refresh the page.")
301
```

Each report is stored with a unique name (with date and time stamp).

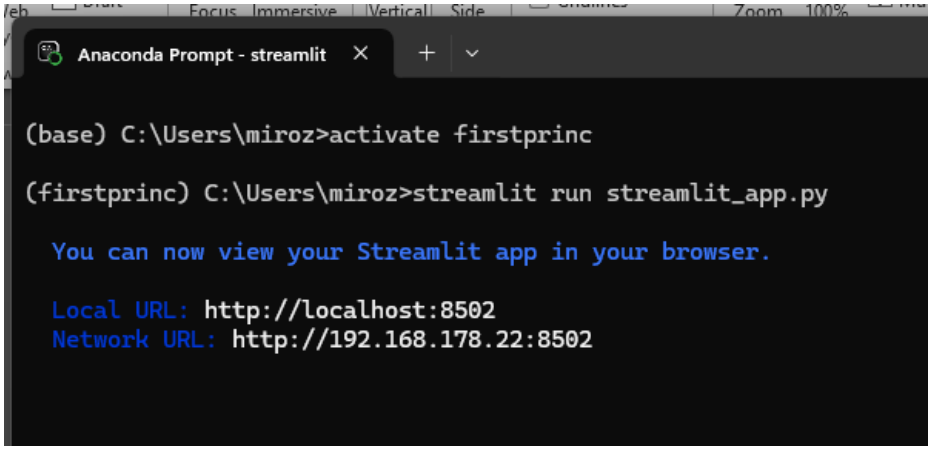


The App is run by command: streamlit run streamlit_app.py

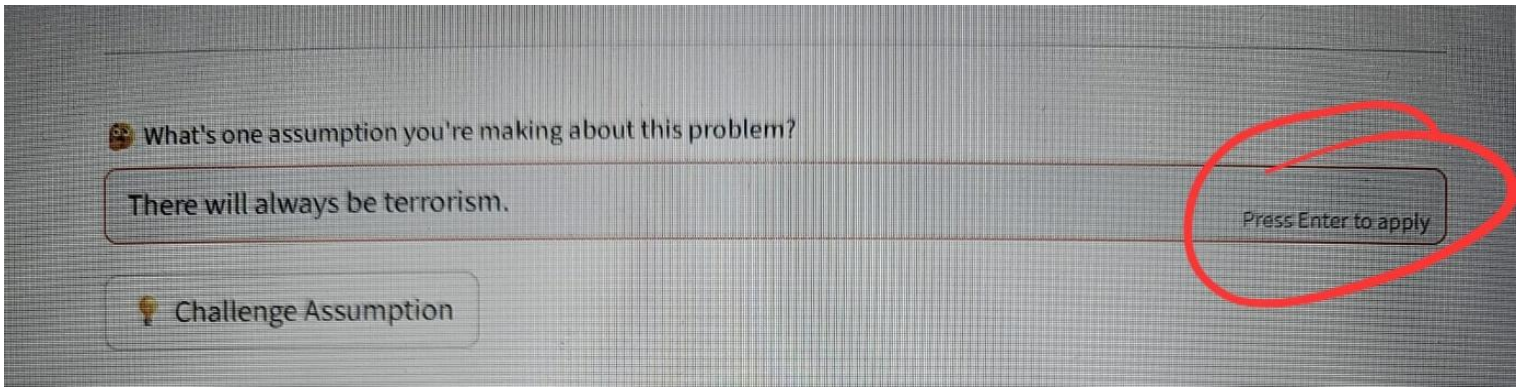


Challenges:

There is no way that application can be closed within application itself, so it has to be done in a Terminal (prompt). Application is also run from the Terminal.



Getting rid of unwanted prompt




```
# --- CSS to remove Streamlit's "Ctrl+Enter" helper hints from all text fields ---
st.markdown("""
    <style>
    /* Target hints under all text input and text area fields */
    .e1bc0bcm0, .stTextArea [data-baseweb="textarea"]+div, .stTextInput [data-baseweb="input"]+div {
        visibility: hidden; height: 0px !important; pointer-events: none;
    }
    </style>
""", unsafe_allow_html=True)
```

Streamlit does not allow to remove the text (totally). The only option was to “hide it”.