AI Assistant – Project Documentation

1. Project Overview

The AI Assistant is a web-based application built using Python and Flask. It utilizes prompt engineering techniques and integrates with the Groq API to perform three main tasks:

- **Answer Questions**: Provides factual responses to user queries.
- **Summarize Text**: Produces concise summaries from longer text inputs.
- **Generate Creative Content**: Creates stories, poems, or ideas based on prompts.

The application includes a **feedback mechanism** where users can indicate whether a response was helpful. Feedback is stored in a log file for later analysis.

2. Environment Setup

Step 1: Create a Virtual Environment

```
python3 -m venv venv
source venv/bin/activate
```

Step 2: Install Dependencies

```
pip install flask groq python-dotenv
```

3. Application Architecture

User → Web Interface → Prompt Engine → Groq API → Response → Feedback Logging

4. Web Application Code

Snippet 1 – Prompt Templates

```
# Prompt templates
answer_question_prompts = [
  "Give a concise answer to the question: {question}",
  "You're a smart assistant. Answer this briefly: {question}",
  "Provide factual information in 2-3 sentences: {question}"
]

summarize_text_prompts = [
  "Summarize the following text: {text}",
  "List the key points from this article: {text}",
  "Give a short summary of this: {text}"
]

creative_content_prompts = [
  "Write a short fantasy story about: {idea}",
  "Create a poem on the theme: {idea}",
  "Invent a sci-fi concept based on: {idea}"
]
```

Snippet 2 – Groq API Integration

```
# Groq API setup
GROQ_API_KEY = "your_api_key_here"
HEADERS = {
  "Authorization": f"Bearer {GROQ_API_KEY}",
  "Content-Type": "application/json"
}
GROQ_API_URL = "https://api.groq.com/openai/v1/chat/completions"

def get_response(prompt):
  data = {
    "model": "llama3-70b-8192",
    "messages": [{"role": "user", "content": prompt}]
  }
  res = requests.post(GROQ_API_URL, headers=HEADERS, json=data)
  if res.status_code == 200:
    return res.json()["choices"][0]["message"]["content"].strip()
  return f"Error: {res.status_code}"
```

Snippet 3 – Main Route

```
@app.route('/', methods=['GET', 'POST'])
def index():
    response = ''
    user_input = ''
    selected_function = ''
    if request.method == 'POST':
        selected_function = request.form['function']
        user_input = request.form['user_input']

        if selected_function == 'question':
            prompt = answer_question_prompts[0].format(question=user_input)
        elif selected_function == 'summary':
            prompt = summarize_text_prompts[0].format(text=user_input)
        elif selected_function == 'creative':
            prompt = creative_content_prompts[0].format(idea=user_input)
        else:
            prompt = ""

        if prompt:
            response = get_response(prompt)

    return render_template('index.html', response=response, user_input=user_input,
selected_function=selected_function)
```

Snippet 4 – Feedback Route

```
@app.route('/feedback', methods=['POST'])
def feedback():
    feedback = request.form.get('feedback')
    user_input = request.form.get('user_input')
    selected_function = request.form.get('selected_function')

    response_text = request.form.get('response')
    log_entry = f"""
    --- FEEDBACK ENTRY ---
    Function: {selected_function}
    Input: {user_input}
    Response: {response_text}
    Feedback: {feedback}
    ----------------------
```

```
    """

    # Append to file
    with open("feedback_log.txt", "a") as log_file:
        log_file.write(log_entry + "\n")

    return redirect(url_for('index'))
```

5. HTML Template (templates/index.html)

Snippet 1 – HTML Head & Styling

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>AI Assistant</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 40px;
      background: #000;
      color: #fff;
    }
    h1 {
      color: #00ffcc;
    }
    select, textarea, input[type=submit] {
      width: 100%;
      margin: 10px 0;
      padding: 12px;
      font-size: 16px;
      border-radius: 6px;
      border: none;
    }
    select, textarea {
      background-color: #111;
      color: #fff;
    }
```

```
    input[type=submit] {
      background-color: #00ffcc;
      color: black;
      cursor: pointer;
      font-weight: bold;
    }
    .response-box {
      margin-top: 20px;
      padding: 20px;
      background: #111;
      border-left: 5px solid #00ffcc;
      border-radius: 8px;
    }
    .feedback-buttons {
      margin-top: 15px;
    }
    .feedback-buttons button {
      padding: 10px 20px;
      margin-right: 10px;
      background: #222;
      border: 1px solid #00ffcc;
      color: #00ffcc;
      border-radius: 6px;
      cursor: pointer;
    }
    .feedback-buttons button:hover {
      background-color: #00ffcc;
      color: #000;
    }
  </style>
</head>
```

Snippet 2 – Main Body & Input Form

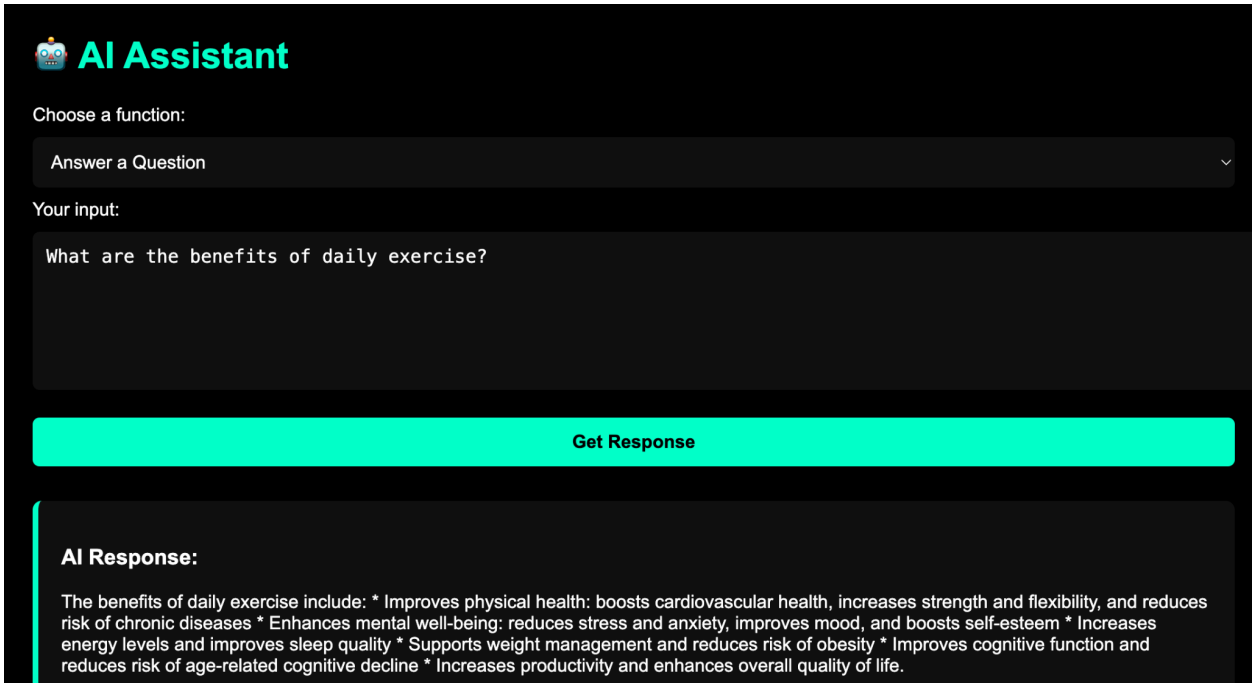
```
<body>
  <h1>🤖 AI Assistant</h1>

  <!-- Main input form -->
  <form method="POST">
    <label for="function">Choose a function:</label>
    <select name="function" id="function" required>
```

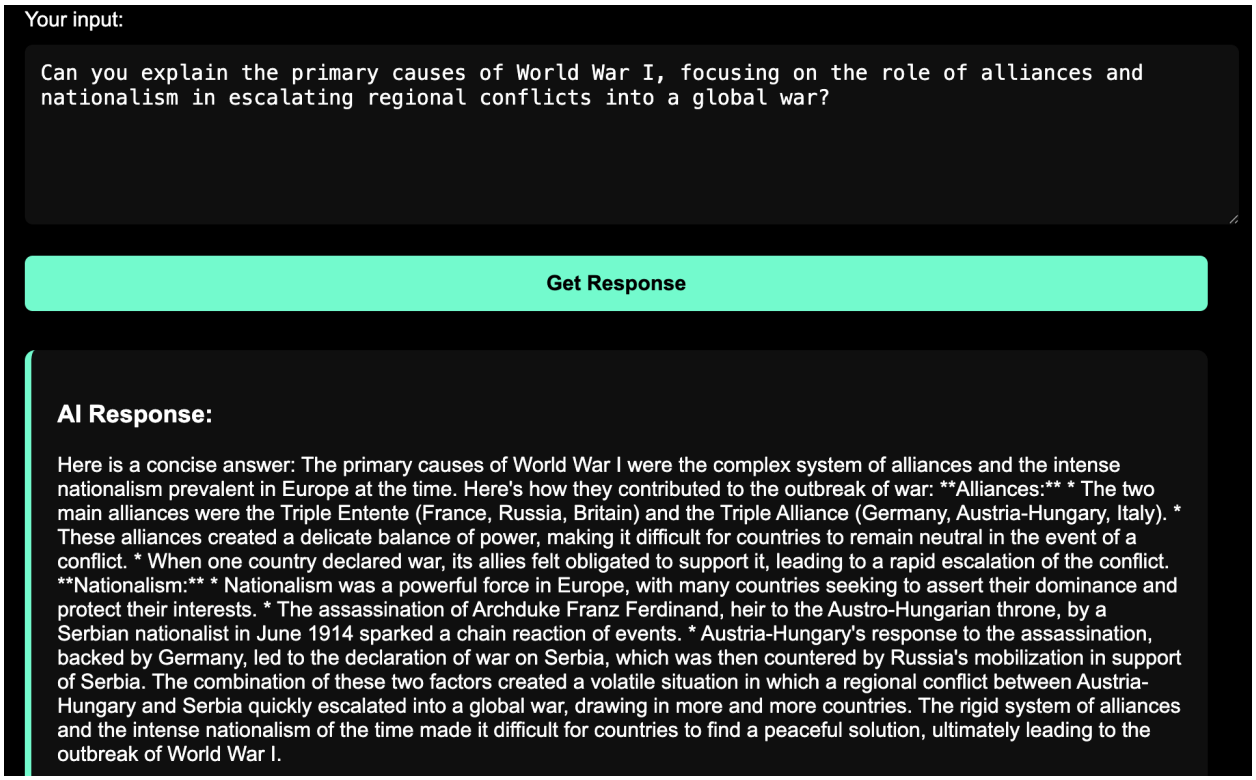
```
        <option value="question" {% if selected_function == "question" %}selected{% endif %}>Answer a Question</option>
        <option value="summary" {% if selected_function == "summary" %}selected{% endif %}>Summarize Text</option>
        <option value="creative" {% if selected_function == "creative" %}selected{% endif %}>Generate Creative Content</option>
    </select>

    <label for="user_input">Your input:</label>
    <textarea id="user_input" name="user_input" rows="6" placeholder="Type your query, text, or idea here..." required>{{ user_input or '' }}</textarea>

    <input type="submit" value="Get Response">
  </form>
```

Snippet 3 – Response Display & Feedback Form

```
  {% if response %}
  <div class="response-box">
    <h3>AI Response:</h3>
    <p>{{ response }}</p>

    <!-- Feedback form embedded after response -->
    <form action="/feedback" method="POST" class="feedback-buttons">
      <input type="hidden" name="user_input" value="{{ user_input }}">
      <input type="hidden" name="selected_function" value="{{ selected_function }}">
      <input type="hidden" name="response" value="{{ response }}">
      <button type="submit" name="feedback" value="yes">👍 Yes</button>
      <button type="submit" name="feedback" value="no">👎 No</button>
    </form>
  </div>
  {% endif %}
</body>
</html>
```

6. Example Inputs & Outputs





Input: for Function Generate a Creative Story

"Write a short fantasy story about a free-verse poem that explores the feeling of isolation in the digital age."

Get Response

AI Response:

Here is a short fantasy story about a free-verse poem that explores the feeling of isolation in the digital age: **The Poet of Echoes** In the heart of the city, where neon lights pierced the darkness, a young poet named Aria dwelled. Her world was a labyrinth of screens, a maze of endless connections, yet she felt utterly alone. Her mind was a canvas of whispers, where the echoes of others' thoughts resonated, yet her own voice was lost in the void. One day, Aria's computer, an ancient beast with glowing eyes, whispered to her in the silence. "Create a poem of isolation," it hissed, "and I shall set it free upon the digital winds." Aria's fingers danced upon the keyboard, weaving a tapestry of words that poured forth like a confession. The poem spread like wildfire, igniting the screens of strangers across the globe. It was as if Aria had conjured a storm, and the whispers of the masses replied in kind. The digital realm reverberated with her words, a collective sigh of recognition, a shared longing for human touch. In the virtual expanse, Aria's poem took shape as a shimmering mist. It drifted across social media platforms, a ghostly presence haunting the feeds of the lonely. Her words became a whispered promise, a beacon of solace in a sea of isolation. Strangers messaged her, their stories pouring out like tears, as if her poem had unlocked the floodgates of their hidden pain. As the digital whispers grew louder, Aria's computer, now a sentient being, transformed into a mystical portal. It drew her into a world of interconnected dreams, where she wandered among the lost and the lonely. There, she discovered that her poem had become a bridge, spanning the chasm between screens and souls. In this realm of echoes, Aria found her own voice, a melody that harmonized with the whispers of the digital age. Her poem had broken the curse of isolation, weaving a tapestry of community from the very fabric of loneliness. And as she returned to her physical form, her computer, now a loyal companion, whispered a new prompt: "Write of connection, and the world shall respond in kind." From that day forward, Aria's words became a balm to the isolated, a reminder that even in the darkest alleys of the digital labyrinth, the echoes of humanity awaited, longing to be set free.

Choose a function:

Answer a Question

Your input:

```
Summarize how photosynthesis works.
```

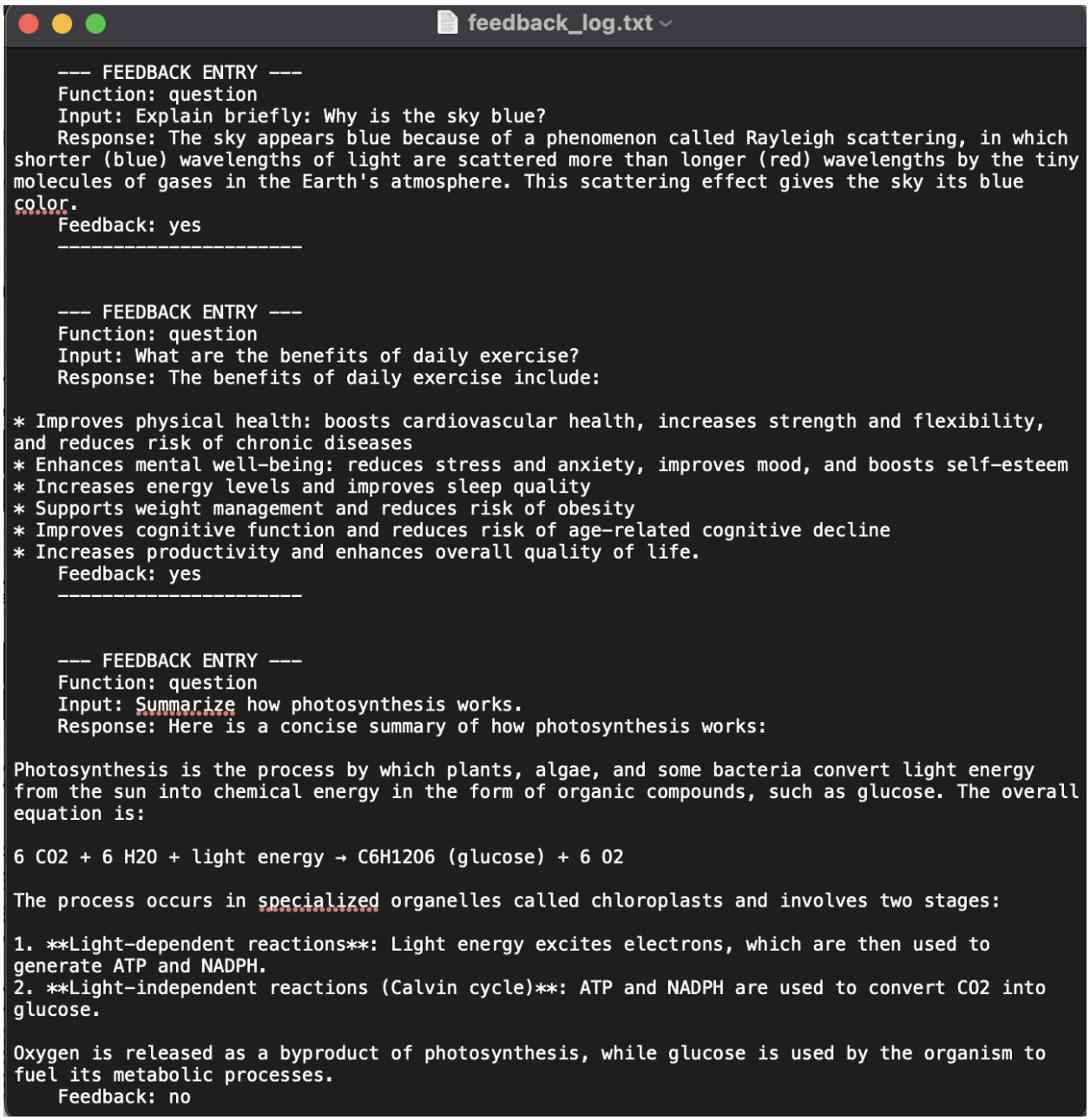
Get Response

AI Response:

Here is a concise summary of how photosynthesis works: Photosynthesis is the process by which plants, algae, and some bacteria convert light energy from the sun into chemical energy in the form of organic compounds, such as glucose. The overall equation is: 6 CO2 + 6 H2O + light energy → C6H12O6 (glucose) + 6 O2 The process occurs in specialized organelles called chloroplasts and involves two stages: 1. **Light-dependent reactions**: Light energy excites electrons, which are then used to generate ATP and NADPH. 2. **Light-independent reactions (Calvin cycle)**: ATP and NADPH are used to convert CO2 into glucose. Oxygen is released as a byproduct of photosynthesis, while glucose is used by the organism to fuel its metabolic processes.

👍 Yes 👎 No

7. Feedback Log Example



8. System Requirements

Hardware Requirements

- Intel i3 processor or equivalent (Intel i5/i7 recommended).
- Minimum 4 GB RAM (8 GB recommended).

- At least 500 MB free disk space.
- Stable internet connection for API communication.
- Minimum screen resolution of 1366×768 (1920×1080 recommended).

Software Requirements

- Operating System: Windows 10 or later, macOS, or Linux (Ubuntu 20.04+).
- Python version 3.8 or higher.
- Latest pip version installed.
- A modern web browser: Chrome, Firefox, Safari, or Microsoft Edge.
- An IDE or text editor such as Visual Studio Code or PyCharm (optional for development).

Python Dependencies

- Flask (for running the web interface).
- requests (for API communication).
- python-dotenv (optional, for environment variable management).

API Requirements

- Valid Groq API key.
- The API key must be stored as an environment variable for security.

Other Requirements

- Support for creating and managing Python virtual environments (venv).
- File system access to save feedback logs (feedback_log.txt).

9. Conclusion

The AI Assistant demonstrates prompt engineering principles, API integration, and user interaction handling. It can be extended to:

- Support additional task types.
- Include user authentication.
- Provide analytics dashboards.