Multimodal Reasoning AI Agent 🧠

Upload an image and provide a reasoning-based task for the AI Agent. The AI Agent will analyze the image and respond based on your input.

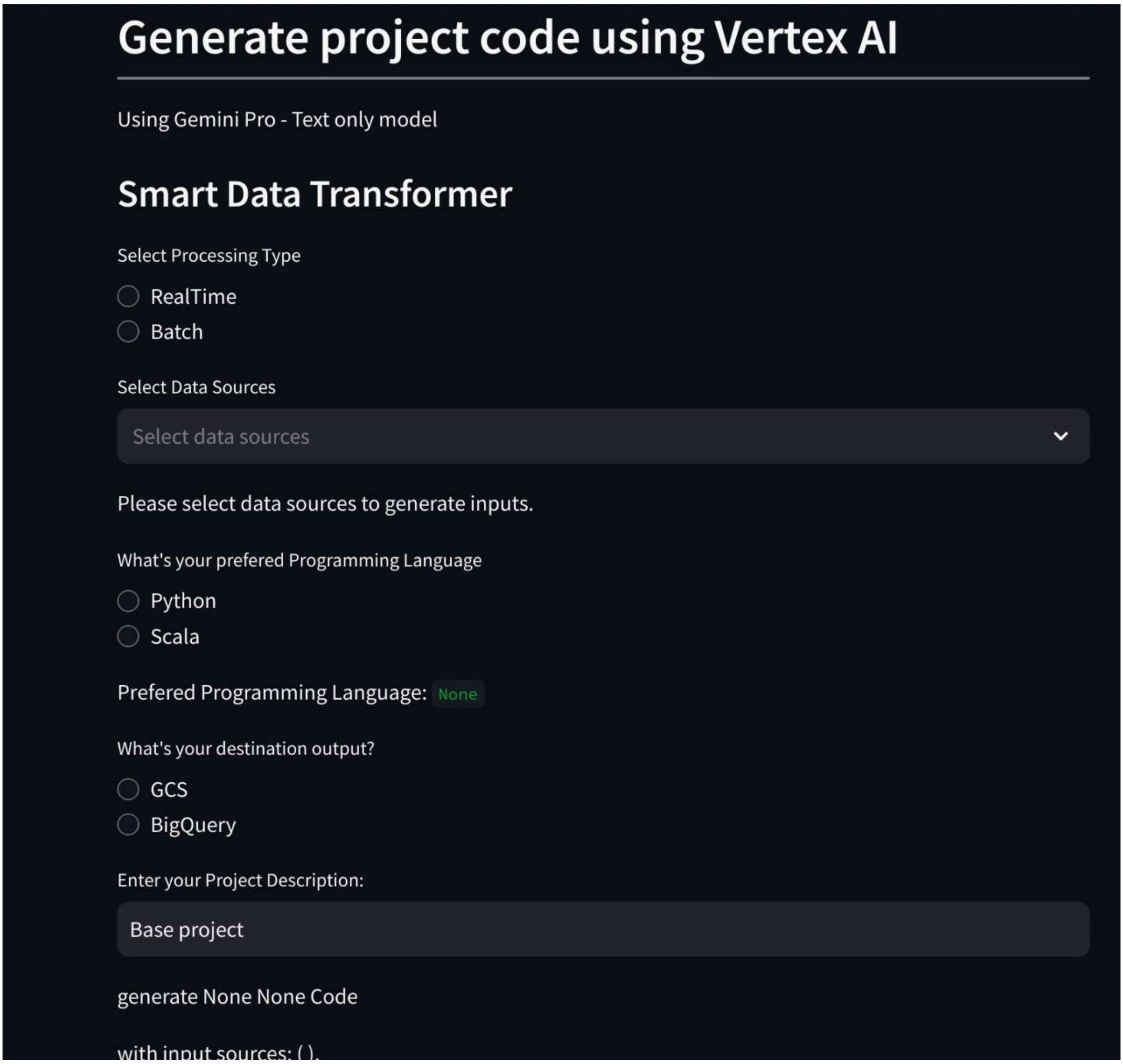
Upload Image

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

Streamlit_code.jpg 163.6KB ✕



Generate project code using Vertex AI

Using Gemini Pro - Text only model

Smart Data Transformer

Select Processing Type

○ RealTime
○ Batch

Select Data Sources

Select data sources

Please select data sources to generate inputs.

What's your prefered Programming Language

○ Python
○ Scala

Prefered Programming Language: `None`

What's your destination output?

○ GCS
○ BigQuery

Enter your Project Description:

Base project

generate None None Code

with input sources: ( ).

Uploaded Image

Enter your task/question for the AI Agent:

generate 3 versions of the code for this UI 1. Python 2. Streamlit 3. HTML & Javascript. Pls suggest which is best

Analyze Image

AI Response:

~~~
## UI Code Generation

Here are three versions of the code for the UI depicted in the image:

**1. Python 2 (CLI)**

```python
# Python 2

def get_user_input():
    print "Generate project code using Vertex AI"
    print "Using Gemini Pro - Text only model\n"
    print "Smart Data Transformer\n"

    processing_type = raw_input("Select Processing Type (RealTime/Batch): ").lower
    while processing_type not in ["realtime", "batch"]:
        processing_type = raw_input("Invalid input. Select Processing Type (RealTi

    data_sources = raw_input("Select Data Sources (comma-separated): ")

    programming_language = raw_input("What's your preferred Programming Language (
    while programming_language not in ["python", "scala"]:
        programming_language = raw_input("Invalid input. What's your preferred Pro

    destination_output = raw_input("What's your destination output (GCS/BigQuery):
    while destination_output not in ["gcs", "bigquery"]:
        destination_output = raw_input("Invalid input. What's your destination out

    project_description = raw_input("Enter your Project Description: ")

    print "\ngenerate None None Code"
    print "with input sources: ({})".format(data_sources)

    print "\nSummary:"
    print "Processing Type:", processing_type.capitalize()
    print "Data Sources:", data_sources
~~~

```
    print "Preferred Programming Language:", programming_language.capitalize()
    print "Destination Output:", destination_output.upper()
    print "Project Description:", project_description

if __name__ == "__main__":
    get_user_input()
```

Explanation:

- This version provides a basic command-line interface.
- It prompts the user for each input sequentially using `raw_input()`.
- Simple input validation is included for the radio button choices.
- It then prints a summary of the collected inputs.

2. Streamlit (Python 3)

```
# Python 3 (Streamlit)
import streamlit as st

st.title("Generate project code using Vertex AI")
st.caption("Using Gemini Pro - Text only model")
st.header("Smart Data Transformer")

st.subheader("Select Processing Type")
processing_type = st.radio("", ["RealTime", "Batch"])

st.subheader("Select Data Sources")
data_sources = st.selectbox("Select data sources", ["Source A", "Source B", "Sourc

st.text("Please select data sources to generate inputs.")

st.subheader("What's your preferred Programming Language")
programming_language = st.radio("", ["Python", "Scala"], index=0)

st.text(f"Preferred Programming Language: {programming_language if programming_lan

st.subheader("What's your destination output?")
destination_output = st.radio("", ["GCS", "BigQuery"])

st.subheader("Enter your Project Description:")
project_description = st.text_area("", value="Base project")

st.write("generate None None Code")
st.write(f"with input sources: ({data_sources})")

st.write("\n**Summary:**")
st.write(f"Processing Type: {processing_type}")
st.write(f"Data Sources: {data_sources}")
st.write(f"Preferred Programming Language: {programming_language}")
st.write(f"Destination Output: {destination_output}")
st.write(f"Project Description: {project_description}")
```

Explanation:

- This version uses the Streamlit library to create an interactive web application.
- `st.title`, `st.header`, `st.subheader`, `st.caption`, and `st.text` are used for displaying text elements.
- `st.radio` creates radio button groups.
- `st.selectbox` creates a dropdown menu.
- `st.text_area` creates a multiline text input field.
- Streamlit automatically handles the UI rendering and updates based on user interaction.

3. HTML & Javascript

```
<!DOCTYPE html>
<html>
<head>
    <title>Generate project code using Vertex AI</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #1e1e1e;
            color: #ffffff;
            padding: 20px;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
        }
        h1, h2, h3 {
            color: #ffffff;
        }
        hr {
            border-top: 1px solid #555;
        }
        .form-group {
            margin-bottom: 15px;
        }
        label {
            display: block;
            margin-bottom: 5px;
        }
        input[type="radio"] {
            margin-right: 5px;
        }
        select {
            width: 100%;
            padding: 8px;
            background-color: #333;
            color: #ffffff;
            border: 1px solid #555;
            border-radius: 4px;
        }
        textarea {
            width: 100%;
            padding: 8px;
            background-color: #333;
            color: #ffffff;
            border: 1px solid #555;
            border-radius: 4px;
            resize: vertical;
        }
        .output {
            margin-top: 20px;
            padding: 10px;
            background-color: #282c34;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Generate project code using Vertex AI</h1>
        <p>Using Gemini Pro - Text only model</p>
        <hr>
        <h2>Smart Data Transformer</h2>

        <div class="form-group">
            <h3>Select Processing Type</h3>
            <label><input type="radio" name="processing_type" value="realtime"> Re
            <label><input type="radio" name="processing_type" value="batch"> Batch
        </div>

        <div class="form-group">
            <h3>Select Data Sources</h3>
            <select id="data_sources">
                <option value="">Select data sources</option>
                <option value="source_a">Source A</option>
                <option value="source_b">Source B</option>
                <option value="source_c">Source C</option>
            </select>
        </div>
        <p>Please select data sources to generate inputs.</p>

        <div class="form-group">
            <h3>What's your preferred Programming Language</h3>
            <label><input type="radio" name="programming_language" value="python">
            <label><input type="radio" name="programming_language" value="scala">
        </div>
        <p>Preferred Programming Language: <span id="preferred_language">None</spa

        <div class="form-group">
            <h3>What's your destination output?</h3>
            <label><input type="radio" name="destination_output" value="gcs"> GCS<
            <label><input type="radio" name="destination_output" value="bigquery">
        </div>

        <div class="form-group">
            <h3>Enter your Project Description:</h3>
            <textarea id="project_description">Base project</textarea>
        </div>

        <div class="output">
            <p>generate <span id="output_language">None</span> <span id="output_de
            <p>with input sources: (<span id="input_sources"></span>)</p>
        </div>
    </div>

    <script>
        const processingRadios = document.querySelectorAll('input[name="processing
        const langRadios = document.querySelectorAll('input[name="programming_lang
        const destRadios = document.querySelectorAll('input[name="destination_outp
        const dataSourceSelect = document.getElementById('data_sources');
        const preferredLanguageSpan = document.getElementById('preferred_language'
        const outputLanguageSpan = document.getElementById('output_language');
        const outputDestinationSpan = document.getElementById('output_destination'
        const inputSourcesSpan = document.getElementById('input_sources');
        const projectDescriptionTextarea = document.getElementById('project_descri

        langRadios.forEach(radio => {
            radio.addEventListener('change', () => {
                preferredLanguageSpan.textContent = radio.value.charAt(0).toUpperC
                outputLanguageSpan.textContent = radio.value.charAt(0).toUpperCase
            });
        });

        destRadios.forEach(radio => {
            radio.addEventListener('change', () => {
                outputDestinationSpan.textContent = radio.value.toUpperCase();
            });
        });
        dataSourceSelect.addEventListener('change', () => {
            inputSourcesSpan.textContent = dataSourceSelect.value;
        });
    </script>
</body>
</html>
```

Explanation:

- This version uses standard HTML for the structure of the UI elements.
- CSS is included within the `<style>` tags to provide basic styling resembling the image.
- Javascript is used to add basic interactivity:
    - Updating the "Preferred Programming Language" text when a radio button is selected.
    - Updating the output code placeholders based on selections.

Which is Best?

The "best" option depends on the context and your goals:

- **Streamlit (Python 3)** is likely the **best choice for rapid prototyping and internal tools**, especially if you are comfortable with Python. It allows you to create interactive web applications with minimal code and without needing to write HTML, CSS, or Javascript directly. It's excellent for data science workflows and sharing results quickly.

- **HTML & Javascript** provides the **most flexibility and control** over the UI's appearance and behavior. It's the standard for web development and allows for complex interactions and integrations. However, it requires more effort to set up and manage, especially for simple interfaces.

- **Python 2 (CLI)** is the **least visually appealing and interactive**. It's suitable for simple scripting tasks or when a graphical interface is not necessary. It would be the least desirable option for a user interface intended for regular interaction.

In summary:

- **For quick, interactive prototypes and data science tools:** Streamlit (Python 3) is highly recommended.
- **For full control, complex interactions, and standard web applications:** HTML & Javascript is the way to go.
- **For very basic scripting and no GUI requirement:** Python 2 (CLI) might suffice.

Based on the visual nature of the provided image, **Streamlit (Python 3) would be the most efficient and effective way to recreate that UI with interactivity using Python**.