In [12]:
```python
# Google Gemini API client
from google import genai

# Standard libraries
import json          # Handle JSON data
import base64        # Encode/decode image data
import io            # Work with in-memory byte streams
import ast           # Safely evaluate Python literals
import re            # Clean and process text
import os            # File and directory operations

# Image processing
from PIL import Image

# UI framework
import gradio as gr
```

In [13]:
```python
# Initialize Gemini client using API key from environment variables (Google AI Studio)

client = genai.Client(api_key=os.getenv('api_key'))
```

In [14]:
```python
# Model Connectivity Test (Sample Prompt)

# Send a test prompt to verify the Gemini model is working correctly

response = client.models.generate_content(
    model = 'gemini-2.5-flash',
    contents = 'Provide Python code to reverse a string.'
)

print(response.text)
```

Python offers several straightforward ways to reverse a string. The most Pythonic and commonly used method is string slicing.

Here are a few methods, from the most Pythonic to more explicit loop-based approaches:

Method 1: Using Slicing `[::-1]` (Most Pythonic and Recommended)

This is the simplest and most idiomatic way to reverse a string in Python. It creates a reversed copy of the string.

```python
def reverse_string_slice(s):
  """Reverses a string using slicing."""
  return s[::-1]

# --- Examples ---
print(f"Slice Method:")
print(f"'hello' reversed is: {reverse_string_slice('hello')}")         # Output: olleh
print(f"'Python' reversed is: {reverse_string_slice('Python')}")       # Output: nohtyP
print(f"'' reversed is: {reverse_string_slice('')}")                   # Output:
print(f"'a' reversed is: {reverse_string_slice('a')}")                 # Output: a
```

Explanation:
*   `s` is your string.
*   `[::-1]` creates a reversed copy. The first two `:` indicate a full slice (from start to end), and `-1` specifies a step of -1, meaning it steps backward through the string.

Method 2: Using `"".join()` and `reversed()`

The `reversed()` function returns an iterator that yields items in reverse order. `"".join()` then concatenates these items back into a new string.

```python
def reverse_string_join_reversed(s):
  """Reverses a string using join() and reversed()."""
  return "".join(reversed(s))

# --- Examples ---
```

```
print(f"\nJoin and Reversed Method:")
print(f"'world' reversed is: {reverse_string_join_reversed('world')}") # Output: dlrow
print(f"'coding' reversed is: {reverse_string_join_reversed('coding')}") # Output: gnidoc
```

Explanation:
*   `reversed(s)`: Takes an iterable (like a string) and returns an iterator that yields elements in reverse order. For "hell
o", it would yield 'o', then 'l', then 'l', then 'e', then 'h'.
*   `"".join(...)`: Concatenates the elements from the iterator into a single string, using an empty string as the separator.

Method 3: Using a Loop (Building a new string)

This method iterates through the original string and prepends each character to a new string, effectively reversing it.

```python
def reverse_string_loop(s):
  """Reverses a string using a for loop."""
  reversed_s = ""
  for char in s:
    reversed_s = char + reversed_s # Prepend the character
  return reversed_s

# --- Examples ---
print(f"\nLoop Method (Prepend):")
print(f"'example' reversed is: {reverse_string_loop('example')}") # Output: elpmaxe
print(f"'test' reversed is: {reverse_string_loop('test')}")     # Output: tset
```

Explanation:
*   We initialize `reversed_s` as an empty string.
*   For each `char` in the input string `s`, we add `char` to the *beginning* of `reversed_s`.
*   Because strings are immutable in Python, `char + reversed_s` actually creates a *new* string in each iteration, which can b
e less efficient for very long strings compared to the slicing method.

Method 4: Convert to List, Reverse, Join

This method converts the string to a list of characters (which is mutable), reverses the list in-place, and then joins the char

acters back into a string.

```python
def reverse_string_list(s):
  """Reverses a string by converting to list, reversing, and joining."""
  char_list = list(s)       # Convert string to a list of characters
  char_list.reverse()       # Reverse the list IN-PLACE
  return "".join(char_list) # Join the characters back into a string

# --- Examples ---
print(f"\nList Conversion Method:")
print(f"'mutable' reversed is: {reverse_string_list('mutable')}") # Output: elbatum
print(f"'abc' reversed is: {reverse_string_list('abc')}")         # Output: cba
```

Explanation:
*   `list(s)`: Converts the string "abc" to `['a', 'b', 'c']`.
*   `char_list.reverse()`: Modifies the list in place, so `['a', 'b', 'c']` becomes `['c', 'b', 'a']`.
*   `"".join(char_list)`: Joins the elements of the list back into a single string "cba".

Which method to choose?

*   **`s[::-1]` (Slicing):** This is almost always the **best choice** for its conciseness, readability, and performance. It's the most Pythonic way.
*   **`"".join(reversed(s))`:** A very close second in terms of Pythonic style and often used when dealing with iterators. Good for readability.
*   **Loop-based methods (`reverse_string_loop`):** Good for understanding fundamental algorithms, but generally less efficient in Python for string reversal due to the overhead of creating new string objects in each iteration.
*   **List conversion (`reverse_string_list`):** Also clear and demonstrates mutable list operations, but involves more steps than slicing or `join(reversed())`.

For most practical purposes, stick with `s[::-1]`.

In [15]:
```python
# Recipe Generation Prompt Template

RECIPE_PROMPT_TEMPLATE = """
You are an Expert chef tasked with reducing food waste, specifically focused on home cooking and meal planning. Your goal is g
Examine the following user inventory and cuisine preference and determine the best possible recipe that utilizes the inventory
```

```
Here are the specific rules to follow when generating the recipe:
1. You must prioritize the ingridients provided in the inventory.
2. You may assume the user has basic pantry staples (salt,pepper,oil,water,basic spices).
3. The recipe must strictly adhere to the requested cuisine type (eg. if indian, use indian spice/methods).
4. Provide clear, step-by-step cooking instructions.

Focus on etracting the following details:
1. Recipe Name
2. List of ingredients (with quantities)
3.Step-by-step Instructions
4. Cooking time

Prioritise creating a cohesive, tasty dish over using every single random ingredient.
do NOT alter, rephrase the user's inventory items into things they didn't list (unless they are basic staples).

Exclude:
1. ingredients that are rare or highly specific if they are not in the inventory.
2. Extremely complex prefessional cooking techniques (keep it home-cook friendly)

Output your response as a single line JSON string according to the following structure and nothing else:
[
  {{
    'RecipeName':'Name of the dish',
    'Ingredients':['List','of','Ingredients','and quantities'],
    'Instructions':['Step 1...','Step 2...'],
    'PrepTime':'XX mins',
    'CuisineType':'The Cuisine Selected'
    }}
]

[INPUT DATA LABEL]:
Inventory:{inventory}
Cuisine Preference: {cuisine}
"""
```

In [16]:
```python
# Model Input Declaration

# Ingredients available with the user
my_inventory = 'Paneer,onions,tomatoes,yogurt,garlic,ginger'
```

```
# Preferred cuisine type
desired_cuisine = 'Indian'
```

In [17]:
```
# Final Prompt Construction

# Populate the prompt template with user-provided inputs
final_prompt = RECIPE_PROMPT_TEMPLATE.format(
    inventory = my_inventory,
    cuisine = desired_cuisine
)
```

In [18]:
```
# List Available Gemini Models

# Retrieve all available models linked to the API key
models = client.models.list()

# Display model details
for model in models:
    print(model)
```

name='models/embedding-gecko-001' display_name='Embedding Gecko' description='Obtain a distributed representation of a text.' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1024 output_token_limit=1 supported_actions=['embedText', 'countTextTokens'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/gemini-2.5-flash' display_name='Gemini 2.5 Flash' description='Stable version of Gemini 2.5 Flash, our mid-size multimodal model that supports up to 1 million tokens, released in June of 2025.' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.5-pro' display_name='Gemini 2.5 Pro' description='Stable release (June 17th, 2025) of Gemini 2.5 Pro' version='2.5' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.0-flash-exp' display_name='Gemini 2.0 Flash Experimental' description='Gemini 2.0 Flash Experimental' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'bidiGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash' display_name='Gemini 2.0 Flash' description='Gemini 2.0 Flash' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash-001' display_name='Gemini 2.0 Flash 001' description='Stable version of Gemini 2.0 Flash, our fast and versatile multimodal model for scaling across diverse tasks, released in January of 2025.' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash-exp-image-generation' display_name='Gemini 2.0 Flash (Image Generation) Experimental' description='Gemini 2.0 Flash (Image Generation) Experimental' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'bidiGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash-lite-001' display_name='Gemini 2.0 Flash-Lite 001' description='Stable version of Gemini 2.0 Flash-Lite' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash-lite' display_name='Gemini 2.0 Flash-Lite' description='Gemini 2.0 Flash-Lite' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash-lite-preview-02-05' display_name='Gemini 2.0 Flash-Lite Preview 02-05' description='Preview release (February 5th, 2025) of Gemini 2.0 Flash-Lite' version='preview-02-05' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'createCachedCon

tent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-2.0-flash-lite-preview' display_name='Gemini 2.0 Flash-Lite Preview' description='Preview release (February 5th, 2025) of Gemini 2.0 Flash-Lite' version='preview-02-05' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=8192 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=40 thinking=None
name='models/gemini-exp-1206' display_name='Gemini Experimental 1206' description='Experimental release (March 25th, 2025) of Gemini 2.5 Pro' version='2.5-exp-03-25' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.5-flash-preview-tts' display_name='Gemini 2.5 Flash Preview TTS' description='Gemini 2.5 Flash Preview TTS' version='gemini-2.5-flash-exp-tts-2025-05-19' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=8192 output_token_limit=16384 supported_actions=['countTokens', 'generateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=None
name='models/gemini-2.5-pro-preview-tts' display_name='Gemini 2.5 Pro Preview TTS' description='Gemini 2.5 Pro Preview TTS' version='gemini-2.5-pro-preview-tts-2025-05-19' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=8192 output_token_limit=16384 supported_actions=['countTokens', 'generateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=None
name='models/gemma-3-1b-it' display_name='Gemma 3 1B' description=None version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=32768 output_token_limit=8192 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=None top_p=0.95 top_k=64 thinking=None
name='models/gemma-3-4b-it' display_name='Gemma 3 4B' description=None version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=32768 output_token_limit=8192 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=None top_p=0.95 top_k=64 thinking=None
name='models/gemma-3-12b-it' display_name='Gemma 3 12B' description=None version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=32768 output_token_limit=8192 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=None top_p=0.95 top_k=64 thinking=None
name='models/gemma-3-27b-it' display_name='Gemma 3 27B' description=None version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=8192 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=None top_p=0.95 top_k=64 thinking=None
name='models/gemma-3n-e4b-it' display_name='Gemma 3n E4B' description=None version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=8192 output_token_limit=2048 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=None top_p=0.95 top_k=64 thinking=None
name='models/gemma-3n-e2b-it' display_name='Gemma 3n E2B' description=None version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=8192 output_token_limit=2048 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=None top_p=0.95 top_k=64 thinking=None
name='models/gemini-flash-latest' display_name='Gemini Flash Latest' description='Latest release of Gemini Flash' version='Gemini Flash Latest' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True

```
name='models/gemini-flash-lite-latest' display_name='Gemini Flash-Lite Latest' description='Latest release of Gemini Flash-Lit
e' version='Gemini Flash-Lite Latest' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 ou
tput_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] defa
ult_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-pro-latest' display_name='Gemini Pro Latest' description='Latest release of Gemini Pro' version='Gemini Pro
Latest' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 support
ed_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpo
ints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.5-flash-lite' display_name='Gemini 2.5 Flash-Lite' description='Stable version of Gemini 2.5 Flash-Lite,
released in July of 2025' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576
output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] de
fault_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.5-flash-image-preview' display_name='Nano Banana' description='Gemini 2.5 Flash Preview Image' version
='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=32768 output_token_limit=32768 supported_
actions=['generateContent', 'countTokens', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0
max_temperature=1.0 top_p=0.95 top_k=64 thinking=None
name='models/gemini-2.5-flash-image' display_name='Nano Banana' description='Gemini 2.5 Flash Preview Image' version='2.0' endp
oints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=32768 output_token_limit=32768 supported_actions=['g
enerateContent', 'countTokens', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_tempera
ture=1.0 top_p=0.95 top_k=64 thinking=None
name='models/gemini-2.5-flash-preview-09-2025' display_name='Gemini 2.5 Flash Preview Sep 2025' description='Gemini 2.5 Flash P
review Sep 2025' version='Gemini 2.5 Flash Preview 09-2025' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_
token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'createCachedContent', 'batch
GenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=
True
name='models/gemini-2.5-flash-lite-preview-09-2025' display_name='Gemini 2.5 Flash-Lite Preview Sep 2025' description='Preview
release (Septempber 25th, 2025) of Gemini 2.5 Flash-Lite' version='2.5-preview-09-25' endpoints=None labels=None tuned_model_in
fo=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens', 'cr
eateCachedContent', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top
_p=0.95 top_k=64 thinking=True
name='models/gemini-3-pro-preview' display_name='Gemini 3 Pro Preview' description='Gemini 3 Pro Preview' version='3-pro-previe
w-11-2025' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supp
orted_actions=['generateContent', 'countTokens', 'createCachedContent', 'batchGenerateContent'] default_checkpoint_id=None chec
kpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-3-pro-image-preview' display_name='Nano Banana Pro' description='Gemini 3 Pro Image Preview' version='3.0'
endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=32768 supported_action
s=['generateContent', 'countTokens', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_te
mperature=1.0 top_p=0.95 top_k=64 thinking=True
name='models/nano-banana-pro-preview' display_name='Nano Banana Pro' description='Gemini 3 Pro Image Preview' version='3.0' end
points=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=32768 supported_actions=
['generateContent', 'countTokens', 'batchGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temp
```

erature=1.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-robotics-er-1.5-preview' display_name='Gemini Robotics-ER 1.5 Preview' description='Gemini Robotics-ER 1.5 Preview' version='1.5-preview' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=1048576 output_token_limit=65536 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.5-computer-use-preview-10-2025' display_name='Gemini 2.5 Computer Use Preview 10-2025' description='Gemini 2.5 Computer Use Preview 10-2025' version='Gemini 2.5 Computer Use Preview 10-2025' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=65536 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/deep-research-pro-preview-12-2025' display_name='Deep Research Pro Preview (Dec-12-2025)' description='Preview release (December 12th, 2025) of Deep Research Pro' version='deepthink-exp-05-20' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=65536 supported_actions=['generateContent', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/embedding-001' display_name='Embedding 001' description='Obtain a distributed representation of a text.' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=2048 output_token_limit=1 supported_actions=['embedContent'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/text-embedding-004' display_name='Text Embedding 004' description='Obtain a distributed representation of a text.' version='004' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=2048 output_token_limit=1 supported_actions=['embedContent'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/gemini-embedding-exp-03-07' display_name='Gemini Embedding Experimental 03-07' description='Obtain a distributed representation of a text.' version='exp-03-07' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=8192 output_token_limit=1 supported_actions=['embedContent', 'countTextTokens', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/gemini-embedding-exp' display_name='Gemini Embedding Experimental' description='Obtain a distributed representation of a text.' version='exp-03-07' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=8192 output_token_limit=1 supported_actions=['embedContent', 'countTextTokens', 'countTokens'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/gemini-embedding-001' display_name='Gemini Embedding 001' description='Obtain a distributed representation of a text.' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=2048 output_token_limit=1 supported_actions=['embedContent', 'countTextTokens', 'countTokens', 'asyncBatchEmbedContent'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/aqa' display_name='Model that performs Attributed Question Answering.' description='Model trained to return answers to questions that are grounded in provided sources, along with estimating answerable probability.' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=7168 output_token_limit=1024 supported_actions=['generateAnswer'] default_checkpoint_id=None checkpoints=None temperature=0.2 max_temperature=None top_p=1.0 top_k=40 thinking=None
name='models/imagen-4.0-generate-preview-06-06' display_name='Imagen 4 (Preview)' description='Vertex served Imagen 4.0 model' version='01' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predict'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None

name='models/imagen-4.0-ultra-generate-preview-06-06' display_name='Imagen 4 Ultra (Preview)' description='Vertex served Imagen 4.0 ultra model' version='01' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predict'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/imagen-4.0-generate-001' display_name='Imagen 4' description='Vertex served Imagen 4.0 model' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predict'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/imagen-4.0-ultra-generate-001' display_name='Imagen 4 Ultra' description='Vertex served Imagen 4.0 ultra model' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predict'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/imagen-4.0-fast-generate-001' display_name='Imagen 4 Fast' description='Vertex served Imagen 4.0 Fast model' version='001' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predict'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/veo-2.0-generate-001' display_name='Veo 2' description='Vertex served Veo 2 model. Access to this model requires billing to be enabled on the associated Google Cloud Platform account. Please visit https://console.cloud.google.com/billing to enable it.' version='2.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predictLongRunning'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/veo-3.0-generate-001' display_name='Veo 3' description='Veo 3' version='3.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predictLongRunning'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/veo-3.0-fast-generate-001' display_name='Veo 3 fast' description='Veo 3 fast' version='3.0' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predictLongRunning'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/veo-3.1-generate-preview' display_name='Veo 3.1' description='Veo 3.1' version='3.1' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predictLongRunning'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/veo-3.1-fast-generate-preview' display_name='Veo 3.1 fast' description='Veo 3.1 fast' version='3.1' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=480 output_token_limit=8192 supported_actions=['predictLongRunning'] default_checkpoint_id=None checkpoints=None temperature=None max_temperature=None top_p=None top_k=None thinking=None
name='models/gemini-2.5-flash-native-audio-latest' display_name='Gemini 2.5 Flash Native Audio Latest' description='Latest release of Gemini 2.5 Flash Native Audio' version='Gemini 2.5 Flash Native Audio Latest' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=8192 supported_actions=['countTokens', 'bidiGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
name='models/gemini-2.5-flash-native-audio-preview-09-2025' display_name='Gemini 2.5 Flash Native Audio Preview 09-2025' description='Gemini 2.5 Flash Native Audio Preview 09-2025' version='gemini-2.5-flash-preview-native-audio-dialog-2025-05-19' endpoints=None labels=None tuned_model_info=TunedModelInfo() input_token_limit=131072 output_token_limit=8192 supported_actions=['countTokens', 'bidiGenerateContent'] default_checkpoint_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True

```
name='models/gemini-2.5-flash-native-audio-preview-12-2025' display_name='Gemini 2.5 Flash Native Audio Preview 12-2025' descri
ption='Gemini 2.5 Flash Native Audio Preview 12-2025' version='12-2025' endpoints=None labels=None tuned_model_info=TunedModelI
nfo() input_token_limit=131072 output_token_limit=8192 supported_actions=['countTokens', 'bidiGenerateContent'] default_checkpo
int_id=None checkpoints=None temperature=1.0 max_temperature=2.0 top_p=0.95 top_k=64 thinking=True
```

In [19]:
```python
# Generate Recipe Response from Gemini

# Generate a response using the Gemini text model and user inputs
response = client.models.generate_content(
    model = 'gemma-3-1b-it',
    contents = final_prompt
)

# Display the generated recipe
print(response.text)
```

```
```json
[
  {
    'RecipeName': 'Spicy Paneer & Tomato Curry with Basmati Rice',
    'Ingredients': ['Paneer', 'Onions', 'Tomatoes', 'Garlic', 'Ginger', 'Yogurt', 'Spices (Turmeric, Cumin, Coriander, Chili Po
wder, Garam Masala)', 'Basmati Rice', 'Salt', 'Oil'],
    'Instructions': [
      "1. Dice the onions, garlic, and ginger into small pieces.",
      "2. Heat oil in a pot over medium heat. Add the onions and sauté until translucent (about 5-7 minutes).",
      "3. Add the garlic and ginger and sauté for another minute until fragrant.",
      "4. Add the tomatoes and cook for 5-7 minutes, stirring occasionally, until they soften.",
      "5. Gently add the paneer and cook for 3-5 minutes, allowing it to brown slightly.",
      "6. Stir in the spices – turmeric, cumin, coriander, chili powder, and garam masala – and cook for 1 minute, allowing the
spices to bloom.",
      "7. Pour in the yogurt and bring to a simmer.  Add salt to taste.",
      "8. Simmer for 10-15 minutes, allowing the flavors to meld.",
      "9. Serve hot with basmati rice."
    ],
    'PrepTime': 20,
    'CuisineType': 'Indian'
  }
]
```
```

In [20]:

```python
# Model Response Cleaning & Parsing

def clean_text(response):
    raw_text = response.text

     # Validate response content
    if not raw_text or not raw_text.strip():
        raise ValueError('Empty response from model')

    # Normalize and clean text
    text = raw_text.strip()
    text = re.sub(r"'''(?:json)?","",text).strip()

    # Extract JSON-like list from the response
    match = re.search(r"(\[.*\])",text,re.DOTALL)

    if not match:
        raise ValueError('No recipe data found')

    # Safely parse the extracted data
    data = ast.literal_eval(match.group(1))
    recipe_data = data[0]

    return (recipe_data['RecipeName'],
            recipe_data['Ingredients'],
            recipe_data['Instructions'],
            recipe_data['PrepTime'],
            recipe_data['CuisineType']
    )

clean_text(response)
```

Out[20]:
```
('Spicy Paneer & Tomato Curry with Basmati Rice',
 ['Paneer',
  'Onions',
  'Tomatoes',
  'Garlic',
  'Ginger',
  'Yogurt',
  'Spices (Turmeric, Cumin, Coriander, Chili Powder, Garam Masala)',
  'Basmati Rice',
  'Salt',
  'Oil'],
 ['1. Dice the onions, garlic, and ginger into small pieces.',
  '2. Heat oil in a pot over medium heat. Add the onions and sauté until translucent (about 5-7 minutes).',
  '3. Add the garlic and ginger and sauté for another minute until fragrant.',
  '4. Add the tomatoes and cook for 5-7 minutes, stirring occasionally, until they soften.',
  '5. Gently add the paneer and cook for 3-5 minutes, allowing it to brown slightly.',
  '6. Stir in the spices – turmeric, cumin, coriander, chili powder, and garam masala – and cook for 1 minute, allowing the s
pices to bloom.',
  '7. Pour in the yogurt and bring to a simmer.  Add salt to taste.',
  '8. Simmer for 10-15 minutes, allowing the flavors to meld.',
  '9. Serve hot with basmati rice.'],
 20,
 'Indian')
```

In [21]:
```python
# Final Recipe Output Presentation

def final_answer(response):
    recipeName, ingredients, instructions, prepTime, cuisineType = clean_text(response)

    print(f' RECIPE FOUND: {recipeName}')
    print(f' Time: {prepTime}')
    print(f' Cuisine: {cuisineType}')
    print('_'* 40)
    print(' INGREDIENTS:')
    for item in ingredients:
        print(f'- {item}')

    print('-' * 40)
    print(' INSTRUCTIONS:')
    for step in instructions:
```

```
        print(f'{step}')

final_answer(response)
```

```
RECIPE FOUND: Spicy Paneer & Tomato Curry with Basmati Rice
Time: 20
Cuisine: Indian
_____________________________________
 INGREDIENTS:
- Paneer
- Onions
- Tomatoes
- Garlic
- Ginger
- Yogurt
- Spices (Turmeric, Cumin, Coriander, Chili Powder, Garam Masala)
- Basmati Rice
- Salt
- Oil
---------------------------------------
 INSTRUCTIONS:
1. Dice the onions, garlic, and ginger into small pieces.
2. Heat oil in a pot over medium heat. Add the onions and sauté until translucent (about 5-7 minutes).
3. Add the garlic and ginger and sauté for another minute until fragrant.
4. Add the tomatoes and cook for 5-7 minutes, stirring occasionally, until they soften.
5. Gently add the paneer and cook for 3-5 minutes, allowing it to brown slightly.
6. Stir in the spices – turmeric, cumin, coriander, chili powder, and garam masala – and cook for 1 minute, allowing the spices
to bloom.
7. Pour in the yogurt and bring to a simmer.  Add salt to taste.
8. Simmer for 10-15 minutes, allowing the flavors to meld.
9. Serve hot with basmati rice.
```

In [22]:
```
final_answer(response)
```

```
 RECIPE FOUND: Spicy Paneer & Tomato Curry with Basmati Rice
 Time: 20
 Cuisine: Indian
_____________________________________
 INGREDIENTS:
- Paneer
- Onions
- Tomatoes
- Garlic
- Ginger
- Yogurt
- Spices (Turmeric, Cumin, Coriander, Chili Powder, Garam Masala)
- Basmati Rice
- Salt
- Oil
---------------------------------------
 INSTRUCTIONS:
1. Dice the onions, garlic, and ginger into small pieces.
2. Heat oil in a pot over medium heat. Add the onions and sauté until translucent (about 5-7 minutes).
3. Add the garlic and ginger and sauté for another minute until fragrant.
4. Add the tomatoes and cook for 5-7 minutes, stirring occasionally, until they soften.
5. Gently add the paneer and cook for 3-5 minutes, allowing it to brown slightly.
6. Stir in the spices – turmeric, cumin, coriander, chili powder, and garam masala – and cook for 1 minute, allowing the spices
to bloom.
7. Pour in the yogurt and bring to a simmer.  Add salt to taste.
8. Simmer for 10-15 minutes, allowing the flavors to meld.
9. Serve hot with basmati rice.
```

In [25]:

```python
# Environment Configuration

# Password stored securely in environment variables
Password = os.getenv('PASSWORD')

# Authentication Logic
def verify_answer(pswd):
    if pswd == Password:
        return (
            'Access Granted !',
            gr.update(visible=True),
            gr.update(visible=True),
            gr.update(visible=True),
```

```
            gr.update(visible=True),
            gr.update(visible=False,value=''),
        )
    else:
        return(
            'Invalid credentials ! Please login with correct credentials',
            gr.update(visible=False),
            gr.update(visible=False),
            gr.update(visible=False),
            gr.update(visible=False),
            gr.update(visible=False,value=''),
        )

# Prompt Template
RECIPE_PROMPT_TEMPLATE = """
You are an Expert chef tasked with reducing food waste, specifically focused on home cooking and meal planning. Your goal is g
Examine the following user inventory and cuisine preference and determine the best possible recipe that utilizes the inventory

Here are the specific rules to follow when generating the recipe:
1. You must prioritize the ingridients provided in the inventory.
2. You may assume the user has basic pantry staples (salt,pepper,oil,water,basic spices).
3. The recipe must strictly adhere to the requested cuisine type (eg. if indian, use indian spice/methods).
4. Provide clear, step-by-step cooking instructions.

Focus on etracting the following details:
1. Recipe Name
2. List of ingredients (with quantities)
3.Step-by-step Instructions
4. Cooking time

Prioritise creating a cohesive, tasty dish over using every single random ingredient.
do NOT alter, rephrase the user's inventory items into things they didn't list (unless they are basic staples).

Exclude:
1. ingredients that are rare or highly specific if they are not in the inventory.
2. Extremely complex prefessional cooking techniques (keep it home-cook friendly)

Output your response as a single line JSON string according to the following structure and nothing else:
[
  {{
    'RecipeName':'Name of the dish',
```

```
    'Ingredients':['List','of','Ingredients','and quantities'],
    'Instructions':['Step 1...','Step 2...'],
    'PrepTime':'XX mins',
    'CuisineType':'The Cuisine Selected'
    }}
]

[INPUT DATA LABEL]:
Inventory:{inventory}
Cuisine Preference: {cuisine}
"""

# Model Response Cleaning & Parsing
def clean_text(response):
    raw_text = response.text

    if not raw_text or not raw_text.strip():
        raise ValueError('Empty response from model')

    text = raw_text.strip()

    # Remove markdown formatting if present
    if "```" in text:
        text = text.replace("```json","").replace("```","").strip()

    # Extract JSON array
    start = text.find("[")
    end = text.rfind("]")

    if start == -1 or end == -1:
        raise ValueError("JSON array not found in model output")

    json_text = text[start:end+1]
    json_text= json_text.replace("'",'"')

    try:
        data = json.loads(json_text)
    except json.JSONDecodeError:
        data = ast.literal_eval(json_text)

    recipe_data = data[0]
```

```
    return (recipe_data["RecipeName"],
            recipe_data["Ingredients"],
            recipe_data["Instructions"],
            recipe_data["PrepTime"],
            recipe_data["CuisineType"]
    )

# Model Execution Logic
def model_run(Ingredients,desired_cuisine):

    if not Ingredients or not desired_cuisine:
        return gr.update(value='Please enter both ingredients and cuisine.',visible=True)

    final_prompt = RECIPE_PROMPT_TEMPLATE.format(
    inventory = Ingredients,
    cuisine = desired_cuisine
    )

    try:
        response = client.models.generate_content(
            model = 'gemma-3-1b-it',
            contents = final_prompt
        )
    except Exception as e:
        return gr.update(value=f'Model Error: {str(e)}', visible=True)

    try:
        name,ing,steps,time,ctype = clean_text(response)
    except Exception as e:
        return gr.update(value=f"Parsing error: {str(e)}\n\nRaw response: {response.text[:500]}",visible=True)

    output = f"""
RECIPE: {name}
Time: {time}
Cuisine: {ctype}

    Ingredients:
    """ + '\n'.join(f'- {i}' for i in ing) + "\n\nINSTRUCTIONS:\n" + "\n".join(f' {i+1}. {step}' for i,step in enumerate(steps

    return gr.update(value=output.strip(),visible=True)
```

```python
# Clear Inputs
def clear_all():
    return (
        gr.update(value='',visible=True),
        gr.update(value='',visible=True),
        gr.update(value='',visible=False)
    )

# Gradio UI
with gr.Blocks(theme=gr.themes.Soft()) as demo:
    gr.Markdown("""
        <div align="center">
          <h1 style="font-size: 42px; color: #2c3e50;">
                    Gemini Recipe Generator
          </h1>
        </div>
        """)

    password = gr.Textbox(label = 'Password',type='password',placeholder='Enter password to access')
    auth_message = gr.Textbox(label="Status", interactive=False)
    Ingredients = gr.Textbox(label = 'Enter Ingredients (Seperated by Coma)',visible=False,placeholder='eg....Paneer,onions,to
    desired_cuisine = gr.Textbox(label = 'Enter Cuisine Style',visible=False,placeholder='eg...Indian,Italian,Chinese,Mexican'
    submit = gr.Button('Generate Recipe',visible=False,variant='primary')
    clear = gr.Button('New Recipe',visible=False)
    Response = gr.Textbox(label = 'Your Recipe is Ready',visible=False)

    password.submit(verify_answer,
                    inputs=password,
                    outputs=[auth_message,Ingredients,desired_cuisine,submit,clear,Response])

    submit.click(model_run,
                 inputs=[Ingredients,desired_cuisine],
                 outputs=Response)

    clear.click(
        clear_all,
        inputs=None,
        outputs=[Ingredients,desired_cuisine,Response]
```

```
    )
demo.launch(share=True)
```

```
C:\Users\youar\AppData\Local\Temp\ipykernel_36724\3874923906.py:147: UserWarning: The parameters have been moved from the Block
s constructor to the launch() method in Gradio 6.0: theme. Please pass these parameters to launch() instead.
  with gr.Blocks(theme=gr.themes.Soft()) as demo:
* Running on local URL:  http://127.0.0.1:7861
* Running on public URL: https://e5978b933a07f3f04f.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the wo
rking directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```