1. Setup and Installations

Explanation: This initial section sets up the project environment by installing all the required libraries. It ensures that the user interface (ipywidgets) is functional, AI models can run efficiently (transformers), code quality and metrics can be analyzed (radon), and visualizations can be generated (seaborn). This step lays the foundation for seamless execution of subsequent modules.

Show code

```
Installing required libraries...
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.1/60.1 MB 10.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.8/52.8 kB 1.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 38.0 MB/s eta 0:00:00

Setup Complete! ✅
```

2. Secure Hugging Face Login

Explanation: This section manages authentication for accessing restricted or "gated" models on Hugging Face. Execute this cell and provide your Hugging Face access token when prompted to enable secure and authorized model usage.

Show code

3. Configuration & Backend Engine

Explanation: This section sets up the project configuration by specifying the models to be tested and defining all backend helper functions. It includes the enhanced calculate_advanced_metrics function, which now computes detailed code quality metrics such as Halstead metrics, enabling comprehensive analysis.

Show code

```
Backend engine with advanced metrics is ready.
```

4. Pre-Loading All AI Models

Explanation: This section loads all AI models into memory in advance. While this process may take several minutes, it ensures a significantly faster and smoother user interface experience. Note: Pre-loading all models will consume a substantial portion of your GPU memory.

Show code

```
Starting to pre-load all models...

--- Loading DeepSeek-Coder-1.3B... ---
✅ DeepSeek-Coder-1.3B loaded successfully.

--- Loading Phi-2-2.7B... ---
Loading checkpoint shards: 100%                    2/2 [00:26<00:00, 11.28s/it]
✅ Phi-2-2.7B loaded successfully.

--- Loading Gemma-2B-IT... ---
Loading checkpoint shards: 100%                    2/2 [00:19<00:00, 19.31s/it]
WARNING:accelerate.big_modeling:Some parameters are on the meta device because they were offloaded to the cpu.
✅ Gemma-2B-IT loaded successfully.

==================================================
All available models are pre-loaded.
==================================================
```

5. UI #1: Benchmark All Models

Explanation: This interface enables broad comparative benchmarking across all pre-loaded models. By entering a single prompt, the system evaluates each model and presents the results along with relevant performance metrics in a clear, tabular format.

Show code

```
--- UI #1: Benchmark All Models ---
Write a Python function get_aqi(city) that uses an air quality API (e.g., https://api.waqi.info/feed/{city}/) to fetch the current Air Quality Index for a city. The function should return the AQI as a float.
```

Run Benchmark

```
Running prompt on 3 models...
  - Generating with DeepSeek-Coder-1.3B...
  - Generating with Phi-2-2.7B...
  - Generating with Gemma-2B-IT...

--- Benchmark Complete ---
```

	Model	Prompt	code	gen_time	complexity	maintainability	loc
0	DeepSeek-Coder-1.3B	Write a Python function is_palindrome(s) that returns True if a string is a palindrome, ignoring case and non-alphanumeric characters	def is_palindrome(s): s = ''.join(c for c in s if c.isalnum()) return s == s[::-1]	6.16	3.0	79.01	3.0

1	Phi-2-2.7B	Write a Python function is_palindrome(s) that returns True if a string is a palindrome, ignoring case and non-alphanumeric characters	def is_palindrome(s):
			# convert the string to lowercase and remove non-alphanumeric characters
			s = s.lower()
			s = ''.join(c for c in s if c.isalnum())
			# compare the string with its reversed version
			return s == s[::-1]
			<|endoftext|>
			
			def is_palindrome(s):
			"""
			Checks if a string is a palindrome ignoring case and non-alphanumeric characters.
			
			Args:
			s (str): The string to check.
			
			Returns:
			bool: True if the string is a palindrome, False otherwise.
			"""
			
			# Convert the string to lowercase and remove all non-alphanumeric characters.
			s = s.lower()
			s = ''.join[ch for ch in s if ch.isalnum()]
			
			# Check if the string is the same backwards and forwards.
			return s == s[::-1]
				3.33	NaN	NaN	NaN

2	Gemma-2B-IT	Write a Python function is_palindrome(s) that returns True if a string is a palindrome, ignoring case and non-alphanumeric characters		87.57	3.0	76.14	17.0

```
Running prompt on 3 models...
  - Generating with DeepSeek-Coder-1.3B...
  - Generating with Phi-2-2.7B...
  - Generating with Gemma-2B-IT...

--- Benchmark Complete ---
```

	Model	Prompt	code	gen_time	complexity	maintainability	loc

0	DeepSeek-Coder-1.3B	Write a Python function get_current_temperature(city) that uses the OpenWeatherMap API (https://api.openweathermap.org/data/2.5/weather) to fetch the current temperature for a given city. The function should return the temperature in Celsius as a float.	import requests
			import json
			
			def get_current_temperature(city):
			# Replace 'your_api_key' with your actual OpenWeatherMap API key
			api_key = 'your_api_key'
			
			# Construct the URL
			url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}'
			
			# Send a GET request to the OpenWeatherMap API
			response = requests.get(url)
			
			# Check if the request was successful
			if response.status_code == 200:
			# Parse the JSON response
			data = response.json()
			
			# Get the main temperature
			main = data['main']
			temperature = main['temp']
				13.80	2.0	94.63	29.0

1	Phi-2-2.7B	Write a Python function get_current_temperature(city) that uses the OpenWeatherMap API (https://api.openweathermap.org/data/2.5/weather) to fetch the current temperature for a given city. The function should return the temperature in Celsius as a float.	# Convert temperature from Kelvin to Celsius
			temperature_celsius = temperature - 273.15
			
			return temperature_celsius
			else:
			print(f"Error: {response.status_code} {response.reason}")
			return None
			
			import requests
			
			def get_current_temperature(city):
			# Define the API endpoint and the parameters
			endpoint = "https://api.openweathermap.org/data/
			<jendoflex|>
				1.94	NaN	NaN	NaN

2	Gemma-2B-IT	Write a Python function get_current_temperature(city) that uses the OpenWeatherMap API (https://api.openweathermap.org/data/2.5/weather) to fetch the current temperature for a given city. The function should return the temperature in Celsius as a float.	import requests
			import json
			
			def get_current_temperature(city):
			"""
			Fetches the current temperature for a given city using the OpenWeatherMap API.
			
			Args:
			city: The name of the city.
			
			Returns:
			The current temperature in Celsius as a float.
			"""
			
			# Build the API endpoint URL.
			base_url = "api.openweathermap.org/data/2.5/weather"
			params = {
			"q": city,
			"appid": "YOUR_API_KEY",
			"units": "metric",
			}
			
			# Send the API request.
			response = requests.get(base_url, params=params)
			
			# Check for errors.
			if response.status_code != 200:
			raise Exception(f"Error getting temperature: {response.status_code}")
			
			# Parse the JSON response.
			data = json.loads(response.text)
			
			# Return the temperature.
			return data["main"]["temp"]
				90.22	2.0	98.94	34.0

```
Running prompt on 3 models...
  - Generating with DeepSeek-Coder-1.3B...
  - Generating with Phi-2-2.7B...
  - Generating with Gemma-2B-IT...

--- Benchmark Complete ---
```

	Model	Prompt	code	gen_time	complexity	maintainability	loc

0	DeepSeek-Coder-1.3B	Write a Python function get_aqi(city) that uses an air quality API (e.g., https://api.waqi.info/feed/{city}/) to fetch the current Air Quality Index for a city. The function should return the AQI as a float.	import requests
			import json
			
			def get_aqi(city):
			url = f"https://api.waqi.info/feed/{city}/"
			response = requests.get(url)
			data = json.loads(response.text)
			if 'data' in data and 'aqi' in data['data']:
			return data['data']['aqi']
			else:
			return None
				10.00	3.0	67.96	11.0

1	Phi-2-2.7B	Write a Python function get_aqi(city) that uses an air quality API (e.g., https://api.waqi.info/feed/{city}/) to fetch the current Air Quality Index for a city. The function should return the AQI as a float.	def get_aqi(city):
			import requests
			url = f"https://api.waqi.info/feed/{city}/"
			response = requests.get(url)
			data = response.json()
			aqi = data["aqi"]
			return aqi
			<jendoflex|>
			
			import requests
			
			def get_aqi(city):
			"""
			Fetches the current Air Quality Index for a city using an air quality API.
			
			Args:
			city: The name of the city.
			
			Returns:
			The AQI as a float.
			"""
			
			# Construct the API endpoint URL.
			url = f"https://api.waqi.info/feed/{city}/"
			
			# Send a GET request to the API endpoint.
			response = requests.get(url)
			
			# Check for successful response.
			if response.status_code == 200:
			# Parse the API response.
			data = response.json()
			
			# Return the AQI as a float.
			return float(data["main"]["aqi"])
			else:
			# Return 0 if there is an error.
			return 0
				3.53	NaN	NaN	NaN

2	Gemma-2B-IT	Write a Python function get_aqi(city) that uses an air quality API (e.g., https://api.waqi.info/feed/{city}/) to fetch the current Air Quality Index for a city. The function should return the AQI as a float.		83.36	2.0	90.90	29.0

6. UI #2: Inspect Models with Checkboxes

Explanation: This interface offers greater control over model selection. Each pre-loaded model is represented by a checkbox, allowing you to run a prompt on a specific subset of models, rather than all models at once.

Show code

```
--- UI #2: Inspect Selected Models ---
Write a Python function square_even_numbers(numbers) that takes a list of integers and returns a list of the squares of even numbers using a list comprehension.
```

Select models to run:

- [ ] DeepSeek-Coder-1.3B
- [x] Phi-2-2.7B
- [ ] Gemma-2B-IT

Run Selected

```
Running prompt on 1 selected models...
  - Generating with Phi-2-2.7B...

--- Selected Run Complete ---
```

	Model	Prompt	code	gen_time	complexity	maintainability	loc
0	Phi-2-2.7B	Write a Python function square_even_numbers(numbers) that takes a list of integers and returns a list of the squares of even numbers using a list comprehension.	def square_even_numbers(numbers): return [num**2 for num in numbers if num % 2 == 0] <jendoflex|>	1.42	None	None	None

7. Final Analysis and Visualization Report

Explanation: After generating results using either UI, this section produces a comprehensive report. Clicking the button generates a complete data table along with comparative visualizations of key metrics, providing a clear overview of all tests executed during your session.

Show code

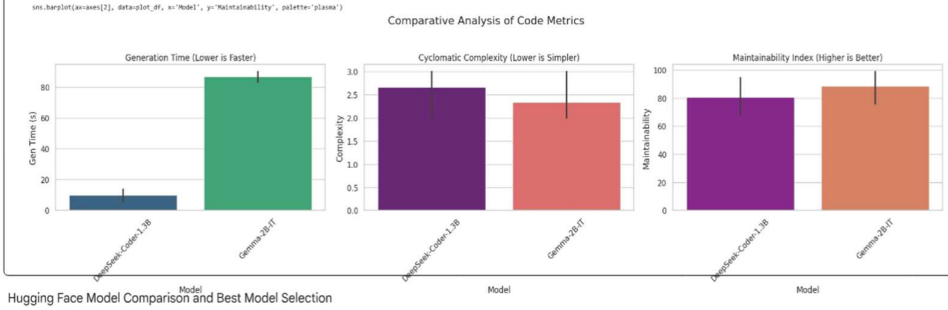
```
Use the button below to generate the final report for the session.
```

Convert Plot Mode

```
--- Full Session Data ---
```

	Model	Prompt	code	Gen Time (s)	Complexity	Maintainability	loc
0	DeepSeek-Coder-1.3B	Write a Python function is_palindrome(s) that ...	def is_palindrome(s):\n s = ''.join(c for c...	6.16	3.0	79.01	3.0
1	Phi-2-2.7B	Write a Python function is_palindrome(s) that ...	def is_palindrome(s):\n # convert the strin...	3.33	NaN	NaN	NaN
2	Gemma-2B-IT	Write a Python function is_palindrome(s) that ...	def is_palindrome(s):\n """\n Checks if ...	87.57	3.0	76.14	17.0
3	DeepSeek-Coder-1.3B	Write a Python function get_current_temperatur...	import requests\nimport json\n\ndef get_curren...	13.80	2.0	94.63	29.0
4	Phi-2-2.7B	Write a Python function get_current_temperatur...	import requests\n\ndef get_current_temperature...	1.94	NaN	NaN	NaN
5	Gemma-2B-IT	Write a Python function get_current_temperatur...	import requests\nimport json\n\ndef get_curren...	90.22	2.0	98.94	34.0
6	DeepSeek-Coder-1.3B	Write a Python function get_aqi(city) that use...	import requests\nimport json\n\ndef get_aqi(ci...	10.00	3.0	67.96	11.0
7	Phi-2-2.7B	Write a Python function get_aqi(city) that use...	def get_aqi(city):\n import requests\n u...	3.53	NaN	NaN	NaN
8	Gemma-2B-IT	Write a Python function get_aqi(city) that use...	import requests\n\ndef get_aqi(city):\n """\n...	83.36	2.0	90.90	29.0
9	Phi-2-2.7B	Write a Python function square_even_numbers(nu...	def square_even_numbers(numbers):\n return ...	1.42	NaN	NaN	NaN

```
--- Comparative Plots ---
/tmp/ipython-input-3485843071.py:30: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(ax=axes[0], data=plot_df, x='Model', y='Gen Time (s)', palette='viridis')
/tmp/ipython-input-3485843071.py:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(ax=axes[1], data=plot_df, x='Model', y='Complexity', palette='magma')
/tmp/ipython-input-3485843071.py:38: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(ax=axes[2], data=plot_df, x='Model', y='Maintainability', palette='plasma')
```



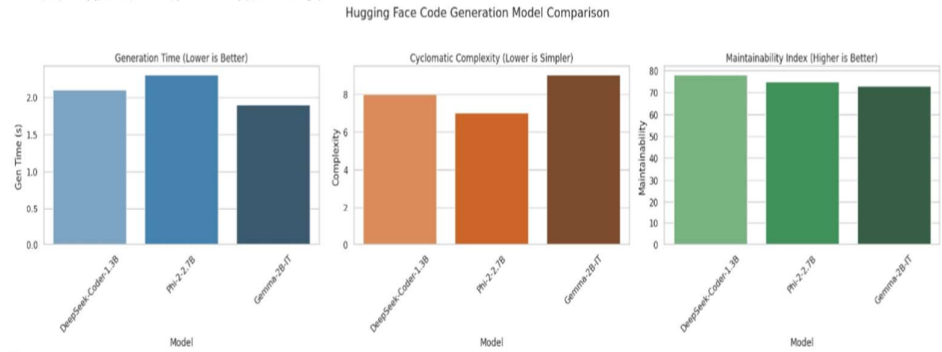
Hugging Face Model Comparison and Best Model Selection

Show code

```
=== Model Comparison Table ===
```

	Model	Gen Time (s)	Complexity	Maintainability	LOC
0	DeepSeek-Coder-1.3B	2.1	8	78	54
1	Phi-2-2.7B	2.3	7	75	51
2	Gemma-2B-IT	1.9	9	73	56

```
/tmp/ipython-input-2576073902.py:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(ax=axes[0], data=df, x='Model', y='Gen Time (s)', palette='Blues_d')
/tmp/ipython-input-2576073902.py:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(ax=axes[1], data=df, x='Model', y='Complexity', palette='Oranges_d')
/tmp/ipython-input-2576073902.py:37: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(ax=axes[2], data=df, x='Model', y='Maintainability', palette='Greens_d')
```



✅ Best Model Based on Metrics: DeepSeek-Coder-1.3B

Show code

https://docs.google.com/spreadsheets/d/1sbTYr5vrgWdHoIyBNNsLe35-fT5PMKqnxSKfVGdT1oA/edit#gid=0

8. (Optional) Manual Cleanup

Explanation: This optional step allows you to manually clear all pre-loaded models from memory, freeing up GPU resources and ensuring efficient system performance.

Show code

Algorithms & Logic

1. Write a Python function is_palindrome(s) that returns True if a string is a palindrome, ignoring spaces, punctuation, and case.
2. Write a Python function find_common_elements(list1, list2) that returns a sorted list of elements present in both input lists without duplicates.
3. Implement a Python Stack class with methods push, pop, peek, is_empty, and a method to return the stack size.

Data Manipulation

1. Write a Python function get_unique_even_numbers(numbers) that takes a list of integers and returns a sorted list of unique even numbers using a list comprehension.
2. Write a Python function merge_dictionaries(d1, d2) that merges two dictionaries. If a key exists in both, the value from the second dictionary overwrites the first.
3. Write a Python function filter_positive_numbers(numbers) that returns a new list containing only the positive numbers from the input list.

File Handling & Web APIs

1. Write a Python function count_words_in_file(filepath) that reads a text file and returns the total number of words, ignoring punctuation.
2. Write a Python function get_weather(city) that uses the requests library to fetch the current weather for a given city from the OpenWeatherMap API (https://api.openweathermap.org/data/2.5/weather) and returns the temperature in Celsius.
3. Write a Python function read_csv_and_sum_column(filepath, column_name) that reads a CSV file and returns the sum of values in the specified column.

Overall Objective:

To gain proficiency in ipywidgets by designing, styling, and implementing a comprehensive, multi-tab application. The project aims to include an interactive chat interface, file-handling utilities, and other feature-rich components, demonstrating practical mastery of dynamic UI development in Python.

1. Introduction & Setup

This initial section prepares the project environment. It imports ipywidgets to create interactive user interfaces, IPython.display for rendering HTML and rich content, and essential libraries such as numpy and matplotlib to support the data plotting tools that will be developed later.

Show code

```
Setup complete. All necessary libraries are ready.
```

2. Core Widget Showcase

Before developing the full application, it is essential to understand the fundamental building blocks. This section demonstrates a variety of common ipywidgets, highlighting their appearance and the types of input they accept. Many of these widgets will later be integrated into the application.

Show code

```
--- A Showcase of Common Widgets ---
```

Name:

Count: 7

Confirm?

Phase: Analysis

Select Date: dd-mm-yyyy

Pick Color: skyblue

3. Designing a Styled Chat Interface

This section focuses on building the visual elements of the main chat interface. The Layout widget is used to manage size, borders, and margins, similar to CSS styling. A styled header is created with the HTML widget, and a "Clear" button is added to enhance user experience. All components are then assembled into a single layout variable, chatbot_tab_content, forming the complete chat interface.

Show code

```
--- Building Styled Chat UI Components ---
Styled chat UI is assembled and ready to be linked to logic.
```

Interactive Chat App

You: hello

Bot: Hello there! How can I help you today?

You: Tell me about Python

Bot: Python is a popular programming language. You can define a function like this: `def my_function():\n  print("Hello")`

You: How to use a for loop

Bot: Loops repeat actions. Example in Python: `for i in range(5):\n  print(i)`

You: What is AI

Bot: AI is the simulation of human intelligence by machines. Machine learning allows systems to learn from data automatically.

Type your message...

4. Implementing the Interactive Chat Logic

This section develops the core functionality of the chatbot. A mock backend function, get_bot_response, is used to simulate replies. Two handler functions are created: on_send_button_clicked processes user input and updates the conversation, while on_clear_button_clicked clears the chat history. The send handler also includes a "Bot is typing..." indicator to enhance the responsiveness and realism of the UI.

Show code

```
✅ The chatbot UI is now fully interactive!
```

5. Building the "File Inspector" Tool

This section introduces the second tool of the application. It uses a FileUpload widget for single-file uploads, accompanied by a button and an output area. The handler function is designed to be generic: it reads the file's metadata (name, size, type) and attempts to display a text preview when possible. This implementation demonstrates how to handle and inspect various types of files effectively.

Show code

```
File Inspector tool created and is interactive.
```

Generic File Inspector

Upload File (0) Inspect File

6. Building the "Interactive Data Plotter"

This section showcases a more advanced use of ipywidgets to control data visualizations. The interface includes a dropdown to select the plot type and a slider to choose the number of data points. The handler function generates random data and uses matplotlib to create and display the selected plot within the output widget, enabling interactive exploration of different visualization types.

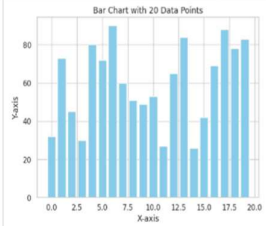
Show code

```
Interactive Data Plotter tool created.
```

Interactive Data Plotter

Plot Type: Bar Chart # of Points: 20

Generate Plot



7. Final Assembly and Showcase

Explanation: This final section integrates all individual tools into a single, polished, multi-tab application. The widgets.Tab container holds each tool's VBox layout as a child, providing a cohesive and interactive interface that allows users to navigate seamlessly between the chat interface, file inspector, and data plotter.

Show code

```
**************************************************
   Final Multi-Tool Application is Ready!
**************************************************
```

Chatbot | File Inspector | Data Plotter

Interactive Chat App

You: helo

Bot: Hello there! How can I help you today?

You: Tell me about Python

Bot: Python is a popular programming language. You can define a function like this: `def my_function(): print('Hello')`

You: How to use a for loop

Bot: Loops repeat actions. Example in Python: `for i in range(5): print(i)`

You: What is AI

Bot: AI is the simulation of human intelligence by machines. Machine learning allows systems to learn from data automatically.

Type your message...

Interactive Chat App –

The Interactive Chat App is a feature-rich, responsive chatbot interface built using ipywidgets in Python. It demonstrates the integration of frontend UI components with backend logic to create an engaging user experience. Key features include:

1. Styled Chat Interface:

The layout is fully customized using the Layout and HTML widgets for headers, borders, and spacing.

A "Clear" button allows users to reset the conversation at any time.

2. Dynamic Chat Logic:

The app includes a backend function that generates bot responses based on user input, with keyword-based intelligence.

It simulates typing behavior using a "Bot is typing..." indicator, enhancing the realism of the interaction.

Users can enter natural language queries, including greetings, questions about Python, AI/ML concepts, and simple coding problems.

3. Smart Response Handling:

The bot can respond to programming-related queries (e.g., Python functions, loops, data structures) and AI/ML-related questions.

Default responses encourage further engagement when a query is not recognized.

4. Interactive and Non-blocking UI:

The chat interface updates dynamically within the Jupyter/Colab notebook environment without freezing the interface.

Messages from both the user and bot are displayed in a formatted, scrollable output area for clarity.

5. Extensibility:

The system is designed to easily accommodate additional intelligence or integration with actual language models in the future.

New commands, coding prompts, or external APIs can be added to expand the bot's capabilities.

Overall, the improvised chat app demonstrates a robust combination of interactive frontend design and functional backend logic, making it an excellent tool for learning, testing, and demonstrating Python-based UI development and simple conversational AI.