

Assignment 1 - OpenAI Evaluation App

Application link: <https://team5-assignment1.streamlit.app/>

Video Link:

https://drive.google.com/file/d/1HdklqzupZM6o4eZYRvcK_zLyv5mOZoi2/view?usp=drive_link

Team member 1: Dharun Karthick Ramaraj

Team member 2: Nikhil Godalla

Team member 3: Linata Deshmukh

Introduction

The OpenAI Evaluation App is developed to evaluate responses generated by the OpenAI ChatGPT model, focusing on questions from the GAIA dataset. This app aims to facilitate user interaction with the model by enabling the exploration of validation test cases, allowing comparisons between ChatGPT-generated answers and expected responses. Additionally, when the response is incorrect, users can provide custom instructions to guide the model in generating a revised response and providing visual analysis of results.

The development process unfolds through the integration of multiple technologies, showcasing a comprehensive data analysis and evaluation pipeline. These technologies include Streamlit for creating a dynamic user interface, AWS S3 for handling file storage, Azure SQL for managing data, and the OpenAI API for generating responses from the ChatGPT model.

In this phase, users will explore how well AI-generated answers align with actual responses, interact with the app's seamless data management and storage systems, and leverage the ability to send questions with tailored instructions for improved evaluation. The app's architecture emphasizes collaborative teamwork and practical application of data engineering, cloud services, AI integration, and user role management.

Problem Statement

The task is to build a model evaluation tool using **Streamlit** for the Model Evaluation Team. The tool will allow users to select a validation test case from the GAIA dataset and evaluate the OpenAI model's responses against that test case. The following requirements need to be addressed:

1. **User Interface:** The tool will be developed using **Streamlit** to provide a user-friendly interface for model evaluation.
2. **Test Case Selection:** Users will have the ability to select a specific test case from the validation test file within the GAIA dataset.
3. **Model Interaction:** The system should be designed to send the selected context data and question to the **OpenAI model** to generate an answer.
4. **Answer Comparison and Modification:** Users should be able to compare the OpenAI model's response to the final answer from the metadata file. If the model's response is incorrect, users should be provided with an option to modify the **Annotator steps** from the metadata file and re-evaluate the model without giving away the answer.
5. **Feedback and Reporting:** The tool should record the feedback and responses from users for each evaluation. The results should be visualized using **Streamlit**, generating reports and insights based on the evaluations.

Development:

- Streamlit was selected for its simplicity, fast development, and efficient data processing, making it perfect for creating interactive evaluation tools.
- Establish connection with Azure SQL to facilitate dynamic interactions and data storage of evaluation results.
- Integrate with AWS S3 for secure storage and retrieval of files associated with the GAIA dataset questions.
- Develop user input features such as dropdowns and text fields for selecting questions from the GAIA dataset and modifying instructions.
- Send context data and prompts to the OpenAI API, incorporating metadata as required.
- Allow users to compare ChatGPT's response with the expected answer and enable modifications to the Annotator instructions when the model's response is incorrect.
- Use Matplotlib for generating visualizations, displaying evaluation results and user feedback in real-time.
- Implement session management to handle different user roles, including admins with access to manage users and datasets.

Testing:

- Create at least 3 test cases for each question from the GAIA dataset: one where the OpenAI model's response matches the expected answer without issues, another where ChatGPT provides an incorrect response and the user modifies the Annotator instructions for re-evaluation, and a third where ChatGPT fails to generate the correct answer despite modified instructions, testing error handling and user feedback logging.

Deployment:

Azure SQL was chosen for managing data storage and query execution as part of the evaluation process, providing scalable cloud-based database solutions. For student accounts, Azure offers free credits, making it cost-effective. Similarly, **AWS S3** was used for storing files

associated with questions in the GAIA dataset. AWS provides \$100 in free credits for student accounts, making it a great option for handling file storage and retrieval. The Streamlit application is deployed on **Streamlit Community Cloud**, offering a free and easy-to-use platform for hosting the app with public access, making it ideal for student projects and rapid prototyping.

Proof of Concept

Executive Summary: A proof of concept (PoC) to evaluate responses generated by the OpenAI ChatGPT model using questions from the GAIA dataset. The project integrates several technologies including Streamlit, AWS S3, Azure SQL, and OpenAI to facilitate the comparison of AI-generated answers with predefined answers from the dataset. The aim is to create an interactive tool that allows users to modify instructions, re-evaluate the model, and visualize results.

Objectives: To provide a functional system where users can:

- Explore questions from the GAIA dataset.
- Compare ChatGPT's responses to predefined answers.
- Modify instructions if the model provides incorrect responses.
- Visualize results using charts and overall response summary.

Methodology:

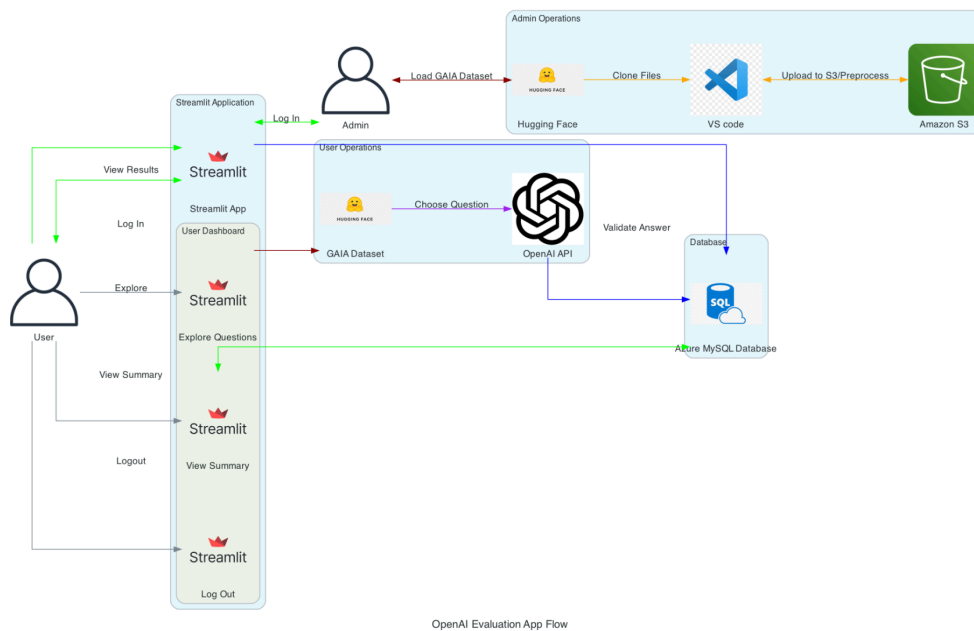
- **Data Integration:** The application integrates the GAIA dataset, uploading associated files (CSV, Excel, etc.) to AWS S3 for storage, while metadata and question details are stored in Azure SQL for easy access.
- **Data Quality:** Before sending data to ChatGPT, preprocessing is done to ensure that files are correctly linked to questions and can be retrieved from AWS S3.
- **Data Processing:**
 - Data is sent to the OpenAI API for generating responses.
 - Incorrect responses allow users to modify the original instructions, which are then sent back to OpenAI for re-evaluation.
- **Data Analysis:** The results of the evaluations are stored in Azure SQL and visualized using Matplotlib to display performance metrics, including correct vs. incorrect responses.

Findings:

- The OpenAI model performs well with most questions when provided with the correct context.
- However, handling complex files remains a challenge, as OpenAI doesn't accept direct file inputs.

- All files need to be preprocessed before being sent to the model, ensuring the relevant information is extracted and presented in a way that the model can interpret.

Architecture Diagram



The architecture consists of the following integrated components within the **Streamlit** framework:

- **Streamlit (Frontend and Backend):** The core of the application, Streamlit serves both as the frontend and backend. It provides the user interface for selecting questions, running evaluations, and displaying the results. Python scripts embedded within Streamlit handle data processing, API interactions, and result visualization.
- **Cloud Storage (AWS S3):** Used to store files associated with the questions in the GAIA dataset. These files are securely stored in S3 and retrieved when necessary during the evaluation process.

- **Database (Azure SQL):** Stores metadata, user information, and evaluation results. This includes the paths to files stored in AWS S3 and the outcomes of ChatGPT evaluations.
- **OpenAI API:** Powers the evaluation process by generating responses for selected questions. The generated answers are compared to the expected final answers from the GAIA dataset to determine accuracy.

1. User Interaction

The user interface is designed with a simple and intuitive flow to guide users through the process of logging in, evaluating questions, and viewing results.

- **Landing and Login Page:**
 - Upon accessing the application, users are greeted with an introductory page that prompts them to either **Log In** or **Create a New Account**.
 - The login form requires a **Username** and **Password**, and users can toggle password visibility for ease of use.
- **Main Dashboard:**
 - Once logged in, users are directed to the **Dashboard**, which offers three main actions:
 1. **Explore Questions:** Leads to the GAIA dataset question exploration and evaluation page.
 2. **View Summary:** Displays the results and performance summaries of previous evaluations.
 3. **Log Out:** Allows users to securely end their session.

2. Question Evaluation Process

This core feature allows users to interact with the GAIA dataset and validate AI-generated responses using ChatGPT.

- **GAIA Dataset QA Page:**
 - Users can browse and select questions from the GAIA dataset for evaluation.
 - After selecting a question, it is sent to the **OpenAI API (ChatGPT)** for analysis and response generation.
 - Responses are classified into one of the following categories:
 - **Correct:** The AI provides a correct response without needing further instructions.
 - **Incorrect:** The AI provides an incorrect or incomplete response, at which point the user can modify the question's instructions and resend it for reevaluation.
- **File Attachments Handling:**
 - For questions that include file attachments, the necessary files are retrieved from **Amazon S3** storage.

- The files undergo preprocessing, where relevant data is extracted and combined with the question text. This ensures ChatGPT receives the full context for evaluation.

3. Admin Operations

The **Admin** role is responsible for loading datasets, managing repositories, and ensuring the smooth operation of the file management process.

- **Loading GAIA Dataset:**
 - Admins load the dataset from **Hugging Face**, which contains the required data for user evaluations.
- **Repository Management:**
 - Using **Visual Studio Code**, admins manage the project repository by cloning files and pushing updates to the cloud.
- **File Upload to Amazon S3:**
 - Files from the repository are uploaded to **Amazon S3** for storage, making them accessible during question evaluation.

4. Database Management

Data persistence and organization are handled through the **Azure MySQL Database**.

- **Data Storage:**
 - The application interacts with the Azure MySQL Database to create tables, store validated answers, and manage the evaluation results.
 - This ensures that all the data from evaluations is organized and can be retrieved for later analysis.

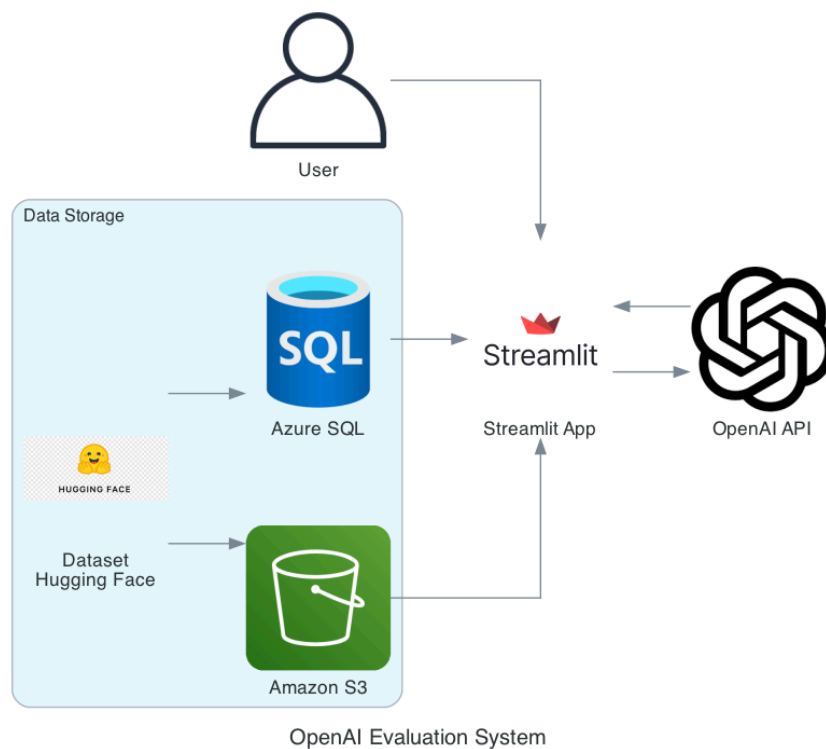
5. Summary of Results

The **Summary of Results** page allows users to view performance metrics and detailed statistics about the evaluations.

- **Visualizing Results:**
 - A **Bar Chart** displays a breakdown of ChatGPT's performance, categorizing the responses as:
 - **Correct with Instructions**
 - **Correct without Instructions**
 - **Incorrect with Instructions**
 - **Incorrect without Instructions**
- **Dataset Statistics:**
 - Users can view the overall dataset statistics, such as:
 - **Total Questions in Dataset**
 - **Total Answered Questions**

■ Total Unanswered Questions

Walkthrough of the Application



About the Application:

Our application allows you to evaluate AI-generated responses by testing how well AI answers various questions from a dataset. If the AI response is incorrect or unsatisfactory, the app offers an option to resend the question with custom instructions, allowing you to guide the AI toward a more suitable response. This feature enables users to explore different ways the AI can handle tasks and adjust its behavior according to specific needs.

To begin, open the application by clicking on the provided link:

<https://team5-assignment1.streamlit.app/>

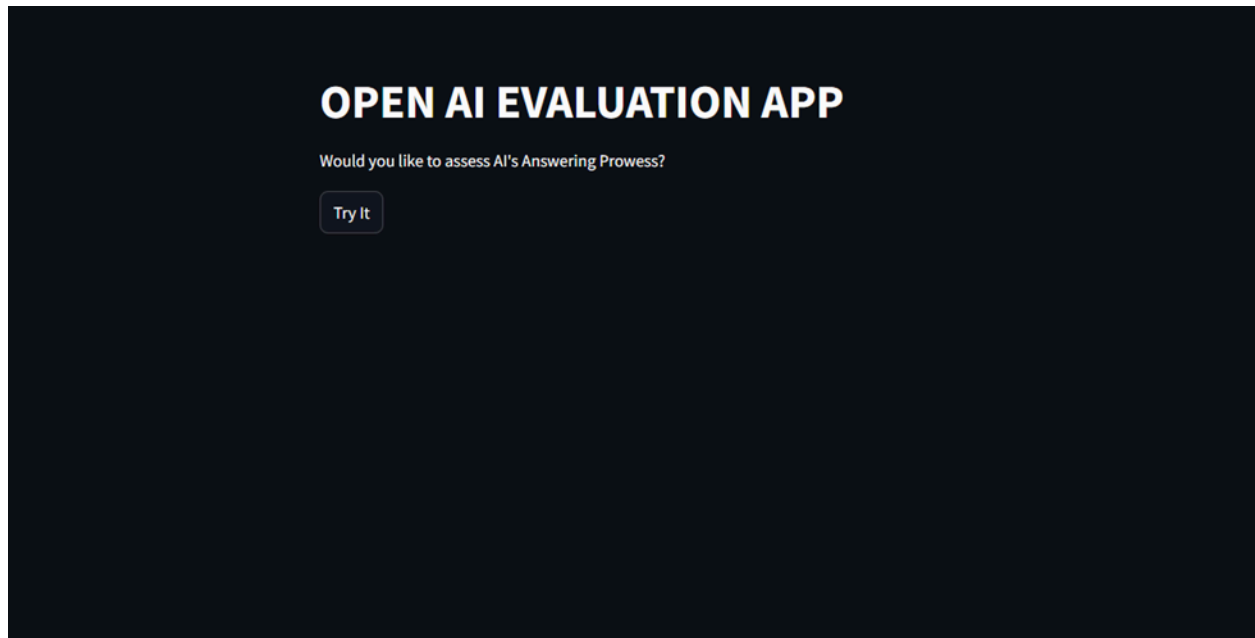
This will take you to the application's landing page.

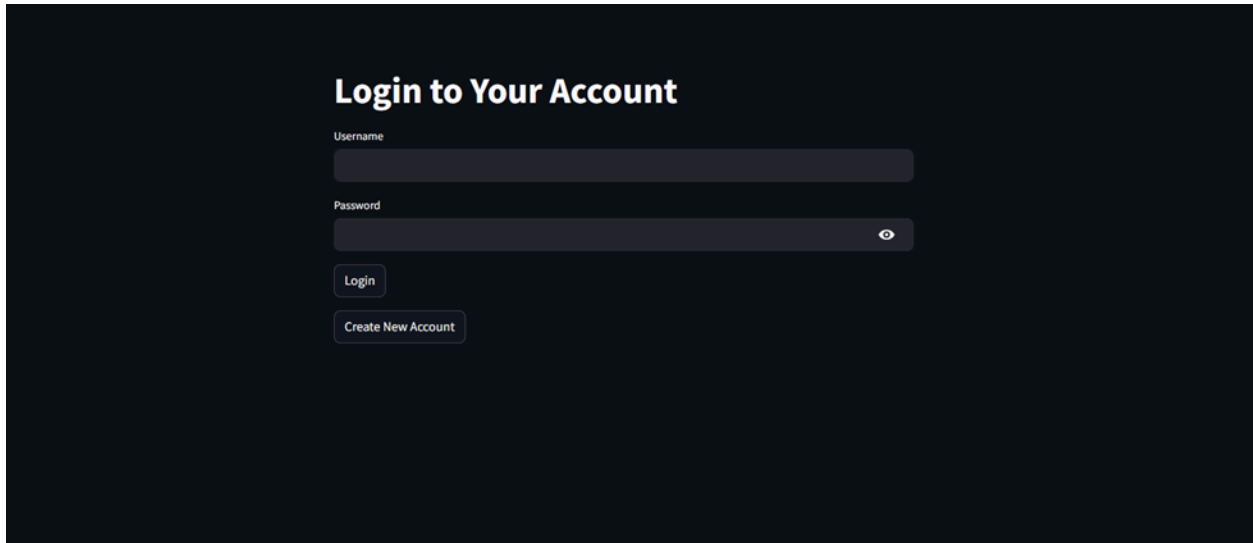
1. Landing and Login Page

Purpose: Introduces the user to the application and allows them to log in or create a new account.

Key Elements:

- The landing page greets users with the application's name, with a **"Try It"** button to start.
- The login page includes fields for **Username** and **Password**, with an option to toggle password visibility. Users can choose to either **Log In** or **Create a New Account**.





When Logged in as Admin:

The **Admin Dashboard** in the application allows administrators to manage both the dataset and users within the system. Admin users have access to special functionalities not available to regular users, which include the following:

1. **Dataset Management:** Admins can trigger the entire dataset processing pipeline from the **Dataset Management** page. This includes cloning the GAIA repository, loading the dataset, uploading associated files to AWS S3, and inserting records into the Azure SQL database. The process ensures that the dataset is up to date and ready for evaluation.
2. **User Management:** From the **User Management** page, admins can view a list of all users in the system. They have the ability to promote regular users to admin status, granting them the same privileges, or they can delete users from the system entirely. This functionality ensures that admins have full control over who can access and modify critical components of the application.

Dataset Management

This page allows you to manage the GAIA dataset. You can trigger the entire dataset processing pipeline, which includes cloning the repository, loading the dataset, uploading files to S3, and inserting records into Azure SQL.

Process Dataset

Back to Admin

User Management

Below is the list of users in the system:

1432F932-BAAE-4EB9-AF6D-64DA9C83AF0C	user	user	Promote to Admin
2AFBF89E-65EA-45C8-9810-F0F58D12DC17	admin	admin	Delete User
8A3A213A-139C-4492-9BCE-462FE39A05F1	1	user	Delete User
B98FFDF8-169E-41D2-85FB-6441F13D28E6	a	user	Promote to Admin
			Delete User

Back to Admin

When Logged in as User:

Welcome, user!

Main Page

Welcome user!

Explore Questions

View Summary

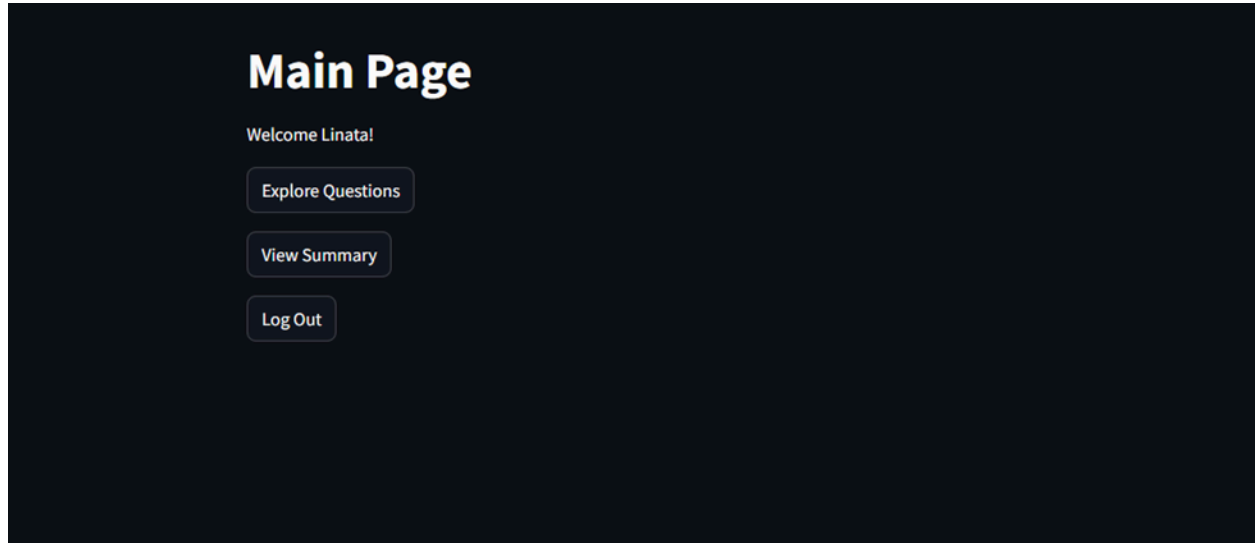
Log Out

2. Main Page (User Dashboard)

Purpose: Serves as the main navigation hub for users.

Key Elements:

- A welcome message with the user's name
- User will have three options:
 - **Explore Questions:** Navigate to the question exploration and evaluation page.
 - **View Summary:** See the summary of past evaluations.
 - **Log Out:** Ends the user session.



3. GAIA Dataset QA Page (Question Exploration)

Purpose: Allows users to explore questions from the GAIA dataset and evaluate AI-generated responses.

Key Elements:

- A list of questions from the GAIA dataset is displayed, allowing users to select a specific question for evaluation.
- After selecting a question, users can view the full question text and the expected final answer.
- A button is provided to send the selected question to ChatGPT for evaluation.
- If ChatGPT's response is correct, the status "**Correct without Instructions**" is displayed along with ChatGPT's response.
- If the response is incorrect, the status "**Incorrect without Instructions**" is displayed.
- An editable text box appears, pre-filled with instructions from the GAIA dataset. Users can modify the instructions and resend the question to ChatGPT for re-evaluation.

[Back](#)

GAIA Dataset QA with ChatGPT

[Refresh](#)[Previous](#)[Next](#)

ID	Question
0	A paper about AI regulation that was originally submitted to arXiv.org in June 2022 shows a figure with tl
1	I'm researching species that became invasive after people who kept them as pets released them. There's
2	If we assume all articles published by Nature in 2020 (articles, only, not book reviews/columns, etc) relie
3	In Unlambda, what exact charcter or text needs to be added to correct the following code to output "For
4	If Eliud Kipchoge could maintain his record-making marathon pace indefinitely, how many thousand ho
5	The attached spreadsheet shows the inventory for a movie and video game rental store in Seattle, Washi
6	How many studio albums were published by Mercedes Sosa between 2000 and 2009 (included)? You can

Select Question Index

2: If we assume all articles published by Nature in 2...

Question: If we assume all articles published by Nature in 2020 (articles, only, not book reviews/columns, etc) relied on statistical significance to justify their findings and they on average came to a p-value of 0.04, how many papers would be incorrect as to their claims of statistical significance? Round the value up to the next integer.

Expected Final Answer: 41

No file associated with this question.

[Send to ChatGPT](#)

Select Question Index

2: If we assume all articles published by Nature in 2...

Question: If we assume all articles published by Nature in 2020 (articles, only, not book reviews/columns, etc) relied on statistical significance to justify their findings and they on average came to a p-value of 0.04, how many papers would be incorrect as to their claims of statistical significance? Round the value up to the next integer.

Expected Final Answer: 41

No file associated with this question.

ChatGPT's Response: Roughly 4% of the papers would be incorrect in claiming statistical significance, which would amount to approximately 1 paper out of every 25 published by Nature in 2020.

The response was incorrect. Please provide instructions.

Edit Instructions (Optional)

1. Find how many articles were published in Nature in 2020 by Googling "articles submitted to nature 2020"
2. Click through to Nature's archive for 2020 and filter the results to only provide articles, not other types of publications: 1002

Send Instructions to ChatGPT

Final Result Status: Incorrect without Instruction

Latest ChatGPT Response: Roughly 4% of the papers would be incorrect in claiming statistical significance, which would amount to approximately 1 paper out of every 25 published by Nature in 2020.

2: If we assume all articles published by Nature in 2020 (articles, only, not book reviews/columns, etc) relied on statistical significance to justify their findings and they on average came to a p-value of 0.04, how many papers would be incorrect as to their claims of statistical significance? Round the value up to the next integer.

Expected Final Answer: 41

No file associated with this question.

ChatGPT's Response: Approximately 41 papers would be incorrect in claiming statistical significance with a p-value of 0.04 out of the total articles published by Nature in 2020.

The response was incorrect. Please provide instructions.

Edit Instructions (Optional)

nature 2020"
2. Click through to Nature's archive for 2020 and filter the results to only provide articles, not other types of publications: 1002

Send Instructions to ChatGPT

ChatGPT's Response with Instructions: Approximately 41 papers would be incorrect in claiming statistical significance with a p-value of 0.04 out of the 1002 articles published by Nature in 2020.

Final Result Status: Correct with Instruction

Latest ChatGPT Response: Approximately 41 papers would be incorrect in claiming statistical significance with a p-value of 0.04 out of the 1002 articles published by Nature in 2020.

4. Handling File Attachments for Question Evaluation

Purpose: To retrieve, preprocess, and integrate file attachments with questions before sending them to ChatGPT for evaluation.

Key Elements:

- If a question includes an attached file, the system retrieves the file from **AWS S3**.
- The attached file is **preprocessed** to extract relevant data or context necessary for answering the question.
- The **context extracted** from the file is combined with the question text to form a complete prompt.

- The full prompt, including the question and the file context, is then sent to **ChatGPT** for evaluation.
- ChatGPT's response is generated based on both the question and the information from the file.

4	If Eliud Kipchoge could maintain his record-making marathon pace indefinitely, how many thousands of
5	The attached spreadsheet shows the inventory for a movie and video game rental store in Seattle, Washi
6	How many studio albums were published by Mercedes Sosa between 2000 and 2009 (included)? You can

Select Question Index

5: The attached spreadsheet shows the inventory for a...

Question: The attached spreadsheet shows the inventory for a movie and video game rental store in Seattle, Washington. What is the title of the oldest Blu-Ray recorded in this spreadsheet? Return it as appearing in the spreadsheet.

Expected Final Answer: Time-Parking 2: Parallel Universe

File Name: 32102e3e-d12a-4209-9163-7b3a104efe5d.xlsx

File Path (URL): <https://s3-openai-evaluation-app-storage.s3.amazonaws.com/32102e3e-d12a-4209-9163-7b3a104efe5d.xlsx>

File downloaded successfully to: .cache\temp_file\32102e3e-d12a-4209-9163-7b3a104efe5d.xlsx

ChatGPT's Response: The oldest Blu-Ray in the spreadsheet is "Time-Parking 2: Parallel Universe" from 2009.

Send to ChatGPT

Final Result Status: Correct without Instruction

Latest ChatGPT Response: The oldest Blu-Ray in the spreadsheet is "Time-Parking 2: Parallel Universe" from 2009.

5. Summary of Results Page

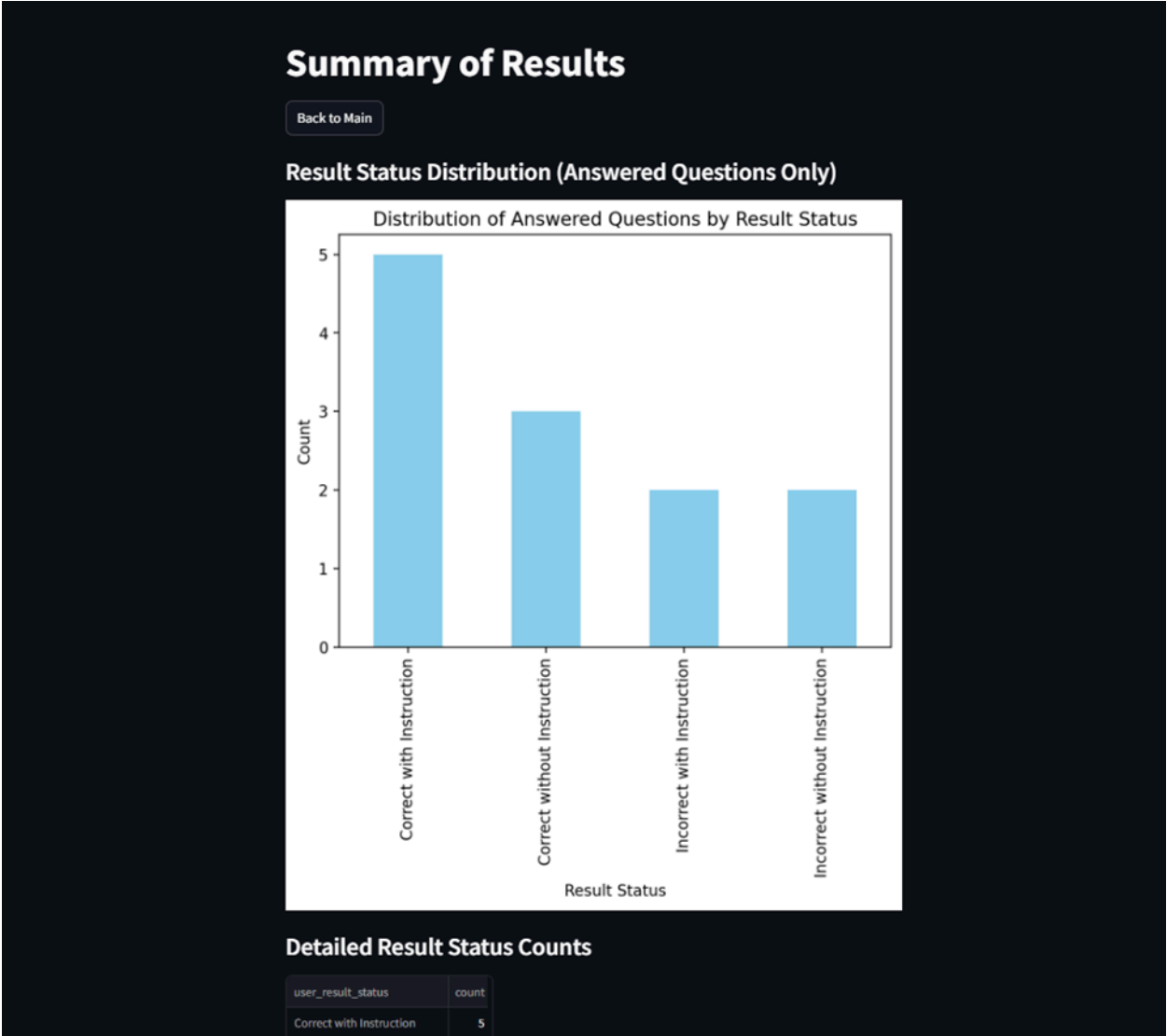
Purpose:

This page provides users with a overview of the evaluation results, allowing them to analyze ChatGPT's performance based on the responses provided to the dataset questions. The results are visualized using both graphical charts and numerical tables for clarity and insight.

Key Elements:

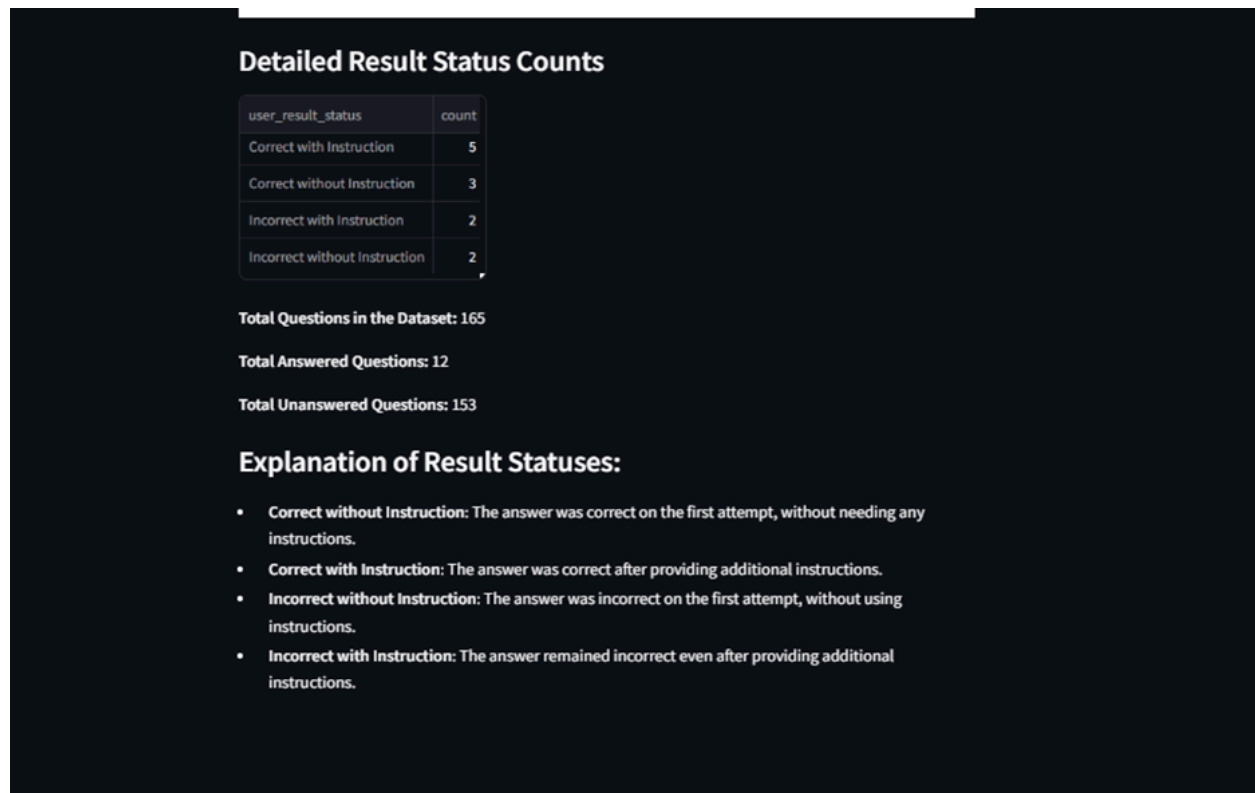
Bar Chart (Result Status Distribution): The **bar chart** visually represents the distribution of answered questions by their result status. The result statuses include:

- **Correct with Instruction:** When ChatGPT provided the correct response after being given additional instructions.
- **Correct without Instruction:** When ChatGPT provided the correct response on the first attempt, without any need for further instructions.
- **Incorrect with Instruction:** When the response was initially incorrect but remained incorrect even after receiving additional instructions.
- **Incorrect without Instruction:** When the response was incorrect on the first attempt, and no instructions were provided.



Dataset and Question Statistics: Additional statistics are provided at the bottom of the page, summarizing the overall dataset and evaluation process:

- **Total Questions in the Dataset:** The total number of questions available in the dataset (e.g., 165).
- **Total Answered Questions:** The number of questions that have been evaluated (e.g., 12).
- **Total Unanswered Questions:** The remaining questions that have yet to be evaluated (e.g., 153).



Application Workflow

The OpenAI Evaluation App is structured to guide users through multiple stages of interaction, leveraging various functionalities to ensure a seamless evaluation process. Below is a detailed breakdown of the application workflow, illustrating key steps alongside screenshots to visualize each major functionality:

User Roles and Authentication

- **Purpose:** The application supports two user roles—admin and regular users. Admins have access to management functionalities such as uploading new datasets and

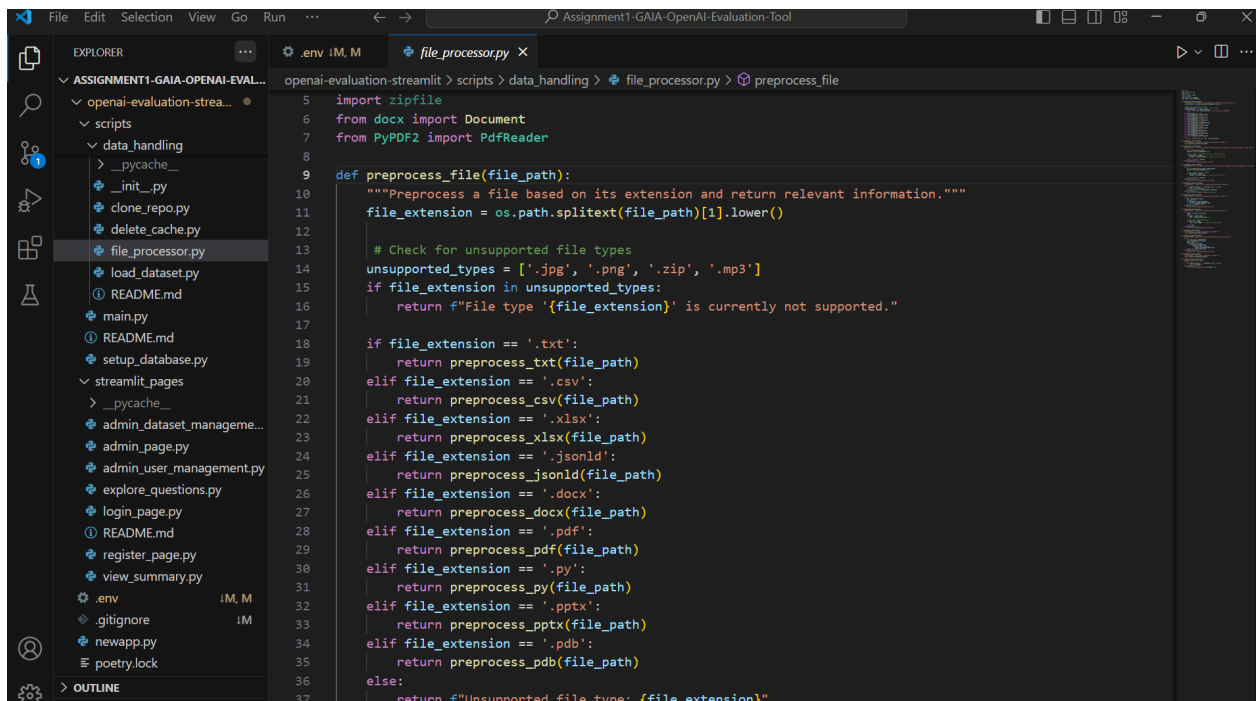
managing users, while regular users can explore questions, submit them to ChatGPT, and view results.

```
14 # Callback function to handle login
15 def on_login_click():
16     username = st.session_state['username']
17     password = st.session_state['password']
18
19     # Fetch user data from database
20     user = fetch_user_from_sql(username)
21
22     if user:
23         stored_password = user['password'].encode('utf-8') # Stored hashed password from DB
24         if bcrypt.checkpw(password.encode('utf-8'), stored_password):
25             # Store user info in session state
26             st.session_state['user_id'] = user['user_id']
27             st.session_state['username'] = user['username']
28             st.session_state['role'] = user['role']
29             st.session_state['login_success'] = True # Set login_success to True
30             st.success(f"Welcome, {username}!")
31
32             # Fetch user-specific results after login and store in session state
33             user_results = fetch_user_results(st.session_state['user_id'])
34             if user_results is not None and not user_results.empty:
35                 st.session_state['user_results'] = user_results
36             else:
37                 # Initialize an empty user_results DataFrame if the user has no data yet
38                 st.session_state['user_results'] = pd.DataFrame() # Empty DataFrame for a new user
39
40             # Redirect based on role
41             if user['role'] == 'admin':
42                 st.session_state.page = 'admin' # Redirect to admin page
43             else:
44                 st.session_state.page = 'main' # Redirect to main page for regular users
45
46     else:
```

- `on_login_click()`: Handles user authentication by checking the username and password and redirects the user based on their role.
- `go_to_register()`: Redirects the user to the registration page and resets login-related session state variables.
- `login_page()`: Displays the login form and handles navigation to either login or registration page.
- `fetch_user_from_sql(username)`: Queries the Azure SQL database to retrieve user details based on the provided username.
- `fetch_user_results(user_id)`: Fetches the specific user's results from the `user_results` table in Azure SQL.

File Preprocessing and Upload

- **Purpose:** The app handles complex files (e.g., CSV, Excel, PDFs) associated with the GAIA dataset. Since OpenAI doesn't accept direct file uploads, files are first retrieved from AWS S3, preprocessed, and relevant information is extracted.



- `preprocess_file()`: Identifies the file type and delegates preprocessing to the appropriate function based on the file extension.
- `preprocess_csv()`: Loads and processes content from a CSV file, ensuring it fits within token limits for further use.
- `preprocess_xlsx()`: Reads and processes the content of an Excel (.xlsx) file, truncating if necessary to meet token restrictions.
- `preprocess_pdf()`: Extracts text from a PDF file, processing it for further analysis while ensuring size limits are respected.
- `preprocess_docx()`: Reads and returns the text content from a Word document (.docx), handling paragraph formatting.
- `preprocess_jsonld()`: Loads and returns the formatted content of a JSON-LD file for easy readability and processing.

Sending Questions to OpenAI (ChatGPT)

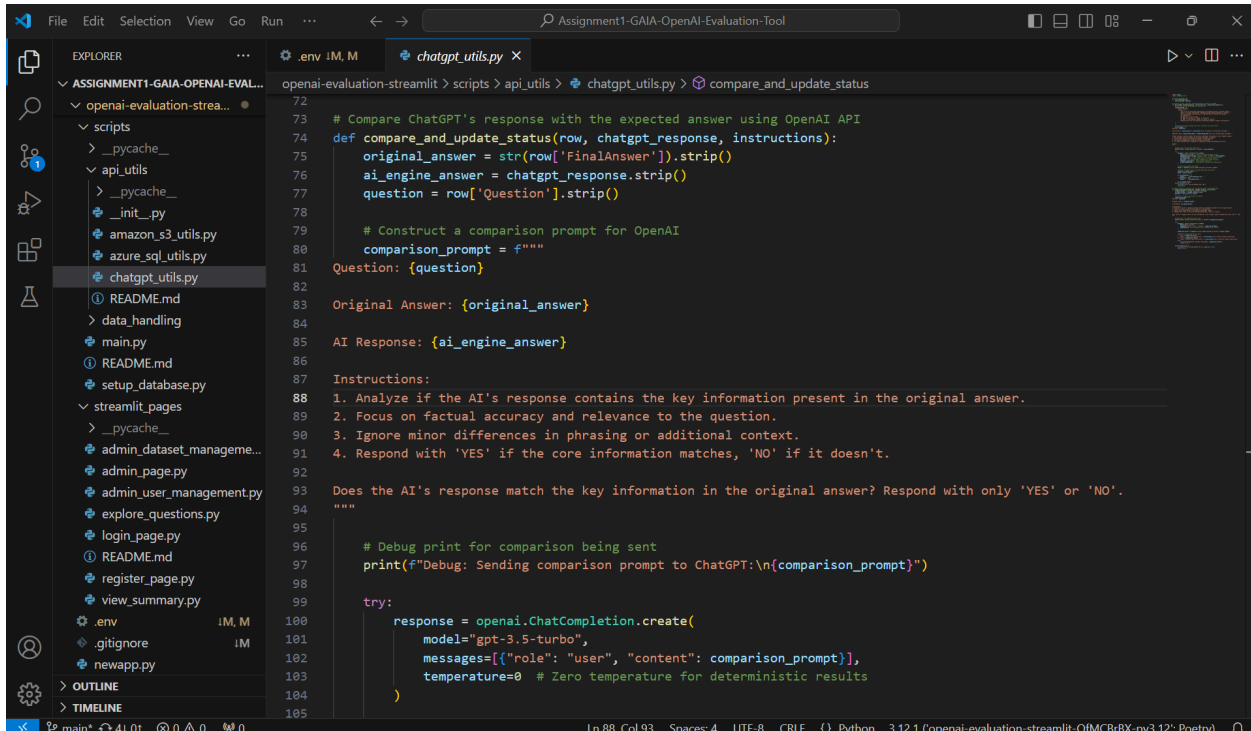
- **Purpose:** Users select a question from the GAIA dataset, and the app sends the context and preprocessed data (if any) to the OpenAI API (ChatGPT) to generate a response.

```
1 import openai
2 import streamlit as st
3
4 # Initialize OpenAI API
5 def init_openai(api_key):
6     openai.api_key = api_key
7
8 # Function to send a question and preprocessed file data to ChatGPT
9 def get_chatgpt_response(question, instructions=None, preprocessed_data=None):
10     # Construct the system message for the Chat API
11     system_message = {
12         "role": "system",
13         "content": (
14             "You are an AI assistant specialized in providing concise, accurate answers. "
15             "Focus on the key information requested without adding unnecessary context. "
16             "Use any provided instructions or reference data to inform your answer. "
17             "For example:\n"
18             "Q: What is 2 + 2? A: 4.\n"
19             "Q: Name the capital of France. A: Paris.\n"
20             "Q: What is the chemical symbol for water? A: H2O.\n"
21             "Respond with only the essential information needed to answer the question."
22         )
23     }
24
25     # Construct the user message with clear structure and instructions
26     user_message = f"""
27     Question: {question}
28
29     Instructions: {instructions if instructions else 'No specific instructions provided.'}
30
31     Reference Data: {preprocessed_data if preprocessed_data else 'No reference data available.'}
32
33     Please provide a concise answer that directly addresses the question. Your response should:
34     1. Be no longer than 3 sentences or 50 words, whichever is shorter.
```

- `init_openai(api_key)`: Initializes the OpenAI API by setting the API key for authenticating subsequent requests.
- `get_chatgpt_response(question, instructions, preprocessed_data)`: Sends a question, along with any instructions and reference data, to the OpenAI API for a concise and relevant response from ChatGPT.
- `system_message`: Prepares a system prompt for ChatGPT to ensure responses are concise and focused on essential information.
- `user_message`: Constructs a user message containing the question, instructions, and reference data for submission to ChatGPT, ensuring clarity in the request.

Validation Process

- **Purpose**: After receiving the initial response from ChatGPT, users can validate the response. If the response is incorrect, the user can modify the instructions (Annotator steps) and send them back for re-evaluation.
- **Implementation**: The system asks ChatGPT a validation question, such as whether the response aligns with the expected answer (Yes/No). Based on this validation, the status is updated in the system (correct/incorrect).

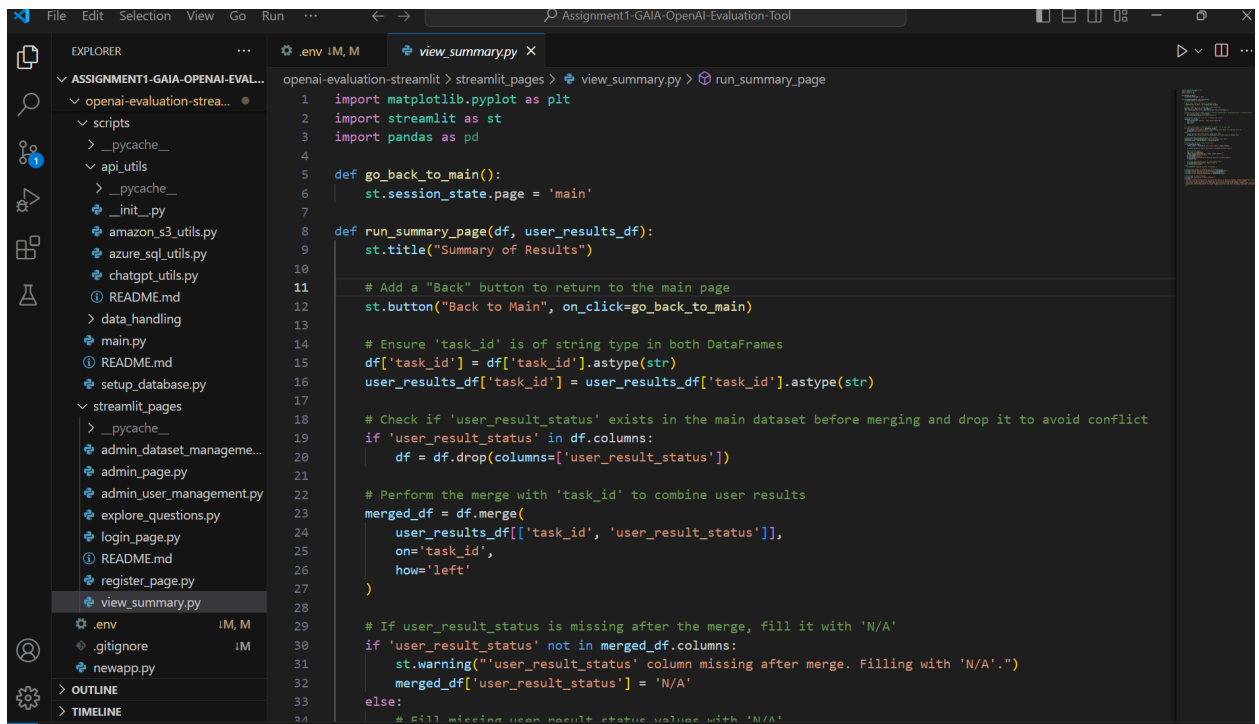


```
72
73 # Compare ChatGPT's response with the expected answer using OpenAI API
74 def compare_and_update_status(row, chatgpt_response, instructions):
75     original_answer = str(row['FinalAnswer']).strip()
76     ai_engine_answer = chatgpt_response.strip()
77     question = row['Question'].strip()
78
79     # Construct a comparison prompt for OpenAI
80     comparison_prompt = f"""
81     Question: {question}
82
83     Original Answer: {original_answer}
84
85     AI Response: {ai_engine_answer}
86
87     Instructions:
88     1. Analyze if the AI's response contains the key information present in the original answer.
89     2. Focus on factual accuracy and relevance to the question.
90     3. Ignore minor differences in phrasing or additional context.
91     4. Respond with 'YES' if the core information matches, 'NO' if it doesn't.
92
93     Does the AI's response match the key information in the original answer? Respond with only 'YES' or 'NO'.
94     """
95
96     # Debug print for comparison being sent
97     print(f"Debug: Sending comparison prompt to ChatGPT:\n{comparison_prompt}")
98
99     try:
100         response = openai.ChatCompletion.create(
101             model="gpt-3.5-turbo",
102             messages=[{"role": "user", "content": comparison_prompt}],
103             temperature=0 # Zero temperature for deterministic results
104         )
105
```

- `compare_and_update_status(row, chatgpt_response, instructions)`: Compares the response from ChatGPT with the original answer, determining whether the response is correct or incorrect based on specific evaluation criteria.
- `comparison_prompt`: Constructs a comparison message for evaluating if ChatGPT's response matches the original answer, focusing on factual accuracy and relevance.

Visualization with Matplotlib

- **Purpose**: To provide insights into the model's performance, the app displays real-time visualizations using Matplotlib, showing the distribution of correct and incorrect responses.
- **Implementation**: The system generates bar charts and other visual elements to help users easily interpret the results of the evaluations. Admins can view summaries of user feedback and evaluation results.



```
1 import matplotlib.pyplot as plt
2 import streamlit as st
3 import pandas as pd
4
5 def go_back_to_main():
6     st.session_state.page = 'main'
7
8 def run_summary_page(df, user_results_df):
9     st.title("Summary of Results")
10
11     # Add a "Back" button to return to the main page
12     st.button("Back to Main", on_click=go_back_to_main)
13
14     # Ensure 'task_id' is of string type in both DataFrames
15     df['task_id'] = df['task_id'].astype(str)
16     user_results_df['task_id'] = user_results_df['task_id'].astype(str)
17
18     # Check if 'user_result_status' exists in the main dataset before merging and drop it to avoid conflict
19     if 'user_result_status' in df.columns:
20         df = df.drop(columns=['user_result_status'])
21
22     # Perform the merge with 'task_id' to combine user results
23     merged_df = df.merge(
24         user_results_df[['task_id', 'user_result_status']],
25         on='task_id',
26         how='left'
27     )
28
29     # If user_result_status is missing after the merge, fill it with 'N/A'
30     if 'user_result_status' not in merged_df.columns:
31         st.warning("'user_result_status' column missing after merge. Filling with 'N/A'.")
32         merged_df['user_result_status'] = 'N/A'
33     else:
34         # Fill missing user result status value with 'N/A'
```

- The `go_back_to_main()` function changes the page to 'main' when the "Back to Main" button is clicked, enabling user navigation back to the main interface.
- The `run_summary_page()` function generates and displays the summary page, merging the dataset (`df`) with the user results (`user_results_df`) based on `task_id`, providing a comprehensive view of evaluation results.
- Data merging is performed inside `run_summary_page()` using `task_id` to combine user results with the dataset, ensuring each question's result is properly attached.
- A bar chart is created to visually display the distribution of the `user_result_status` (e.g., "Correct with Instruction" or "Incorrect without Instruction"), using Matplotlib for clear visualization of answered questions.
- Detailed counts for each `user_result_status` are shown in tabular format, allowing users to see how many responses fall under each status category.

References

1. <https://huggingface.co/datasets/gaia-benchmark/GAIA>
2. <https://docs.streamlit.io/>
3. <https://docs.aws.amazon.com/s3/>
4. <https://learn.microsoft.com/en-us/azure/azure-sql/database/>
5. <https://openai.com/api/>
6. <https://matplotlib.org/stable/index.html>
7. <https://www.npmjs.com/package/bcrypt>

