Automated Conference Decision-Making Systems: Distributing Accepted Papers Into Sessions

05.02.2025

—

Nguyen, Son Hoang & Le, Thi Dieu Ly

Table of Contents

Overview

The **Abstract Categorization and Session Assignment Tool** is a Python program that uses **Google's Generative AI models** to categorize research paper abstracts and assign them to sessions based on topic similarity and other criteria. It reads abstract data from an Excel spreadsheet, processes it using the **Generative AI API**, and outputs the results to a new spreadsheet and an HTML file for easy viewing in a browser.

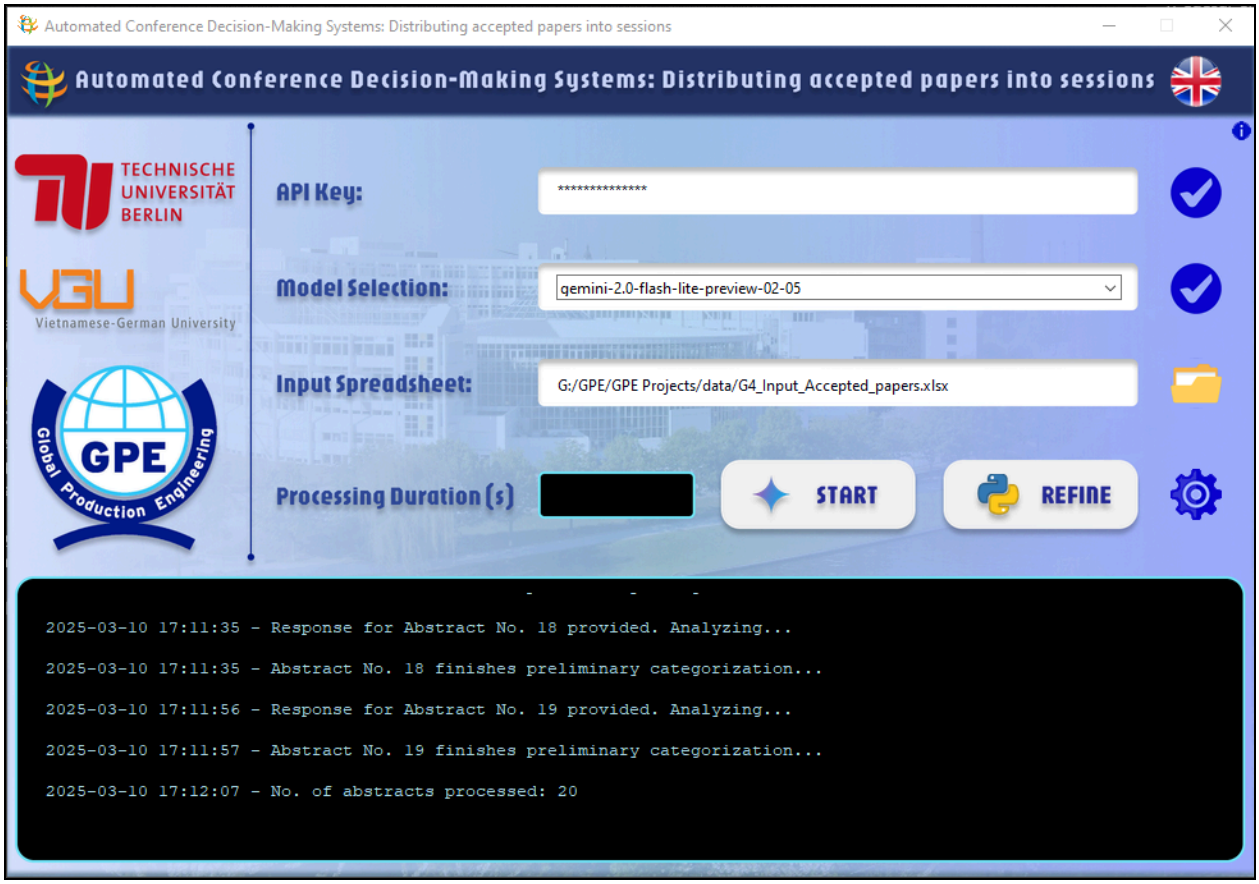


Figure 1: Main User Interface

Functionality

1. **Abstract Categorization**: Uses Generative AI to categorize abstracts based on predefined topics and sub-topics. Identifies research methods, scope, and forecasted presentation time.
2. **Session Assignment**: Assign abstracts to sessions based on topic similarity, group size limits (max 6 abstracts per group), and country diversity.
3. **Spreadsheet Input/Output**: Reads abstract data from an Excel spreadsheet and writes the processed results to a new spreadsheet.
4. **HTML Output**: Generates an HTML file to easily view the session schedule in a web browser.
5. **Tkinter UI**: Provides a graphical user interface for easier interaction with the program.
6. **Configurable system instructions**: Provide an interface to setup system instruction for the target Generative AI model.

Usage

Main Functions

1. **Prepare your Excel spreadsheet**: The spreadsheet must contain **at least** the following columns:
    - **Paper ID**
    - **Paper Title**
    - **Abstract**
    - **Authors**
    - **Country**

A1 | Paper ID

	A	B	C	D	E	F
1	Paper ID	Paper Title	Abstract	Authors	Country	
2	2	Industry and Consumer Views on Improving Sustainability and C	Despite advances in more sustainable ma	Katri Salminen (Tampere University of	Finland	
3	4	An approach for the thermal modeling of machines and machine	The development of sustainable manufact	Michael Frank (Technical University of	Germany	
4	6	Enhancing Sustainability – Strategic Choices from and for small	The multiple crises of the recent years hav	Dominik DS Saubke (Helmut-Schmidt	Germany	
5	7	Designing a Greener Future - Publishing Platforms for Open So	In a time of change shaken by crises, the	Dominik DS Saubke (Helmut-Schmidt	Germany	
6	8	3D Nozzle printing for multi-jet liquid desiccant electrospray	In liquid desiccant electrospray for humidit	Tawanda Mushiri (University of Johan	South Africa, Zimbabwe	
7	9	Reducing waste in liquid silicon rubber process chains by new i	The number of components made of liquic	Robert Bolz (Fraunhofer Society)*	Germany	
8	14	A behaviourally informed heuristics-framework for net zero trans	Transitioning to net zero emissions in mar	Rashmeet Kaur (Cranfield University)	UK	
9	19	Multi-Criteria Optimization of Industrial Microgrids Using Metahe	Despite the steadily increasing share of re	Johannes Prior (Ruhr-University Boch	Germany	
10	20	Closed Loop Engineering: Product as a Ressource!	In traditional engineering education, susta	Diana Völz (Frankfurt University of Ap	Germany	

Figure 2: Sample Input data

2. **Launch the Application**: Run the executable named **AbstractCategorization.exe**.

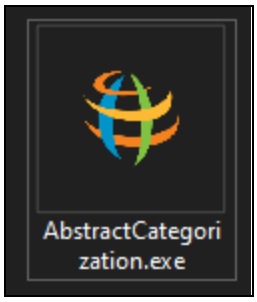


Figure 3: Executable icon

Note: Because this app is self-developed by our team and hasn't been published through an official publisher, Windows may display a warning indicating an "Unknown Publisher". When this security warning appears: Click on "**More info**", then click the "**Run anyway**" button to bypass the Windows security warning and proceed with running the application.

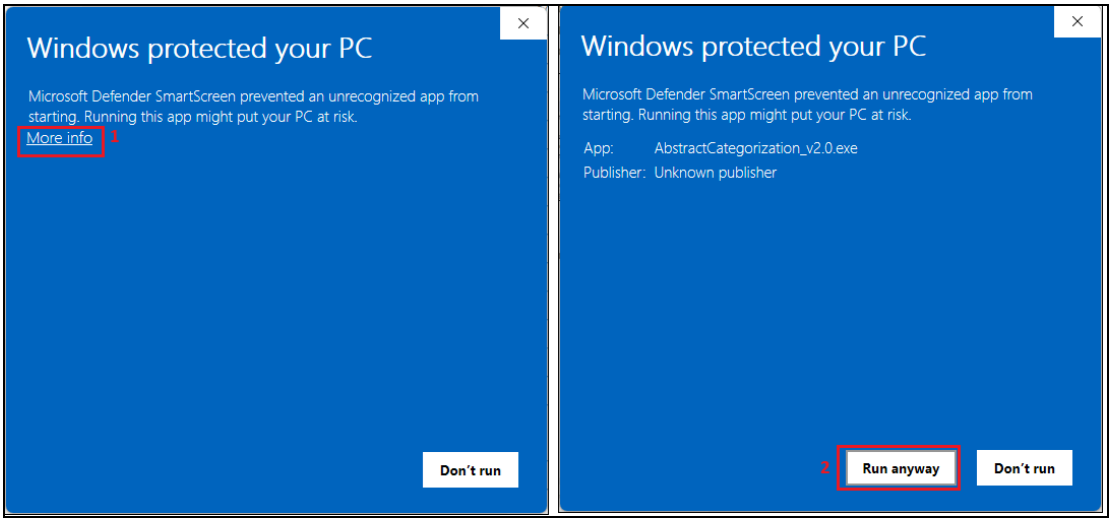


Figure 4: Bypassing the Windows Security Warning

3. The UI has different areas to input necessary information: "**API Key**", "**Model Selection**" and "**Input Spreadsheet**".
4. **API Key**: Enter your **Google Cloud API key**, and hit the **blue tick button** on the right of the text box to save the provided API Key. Refer to the "**Additional Instructions**" section for a step-by-step guide on how to obtain a Google API key for Generative Language API.

Figure 5: Text box to input API key and the button to save the input value

5. **Model Selection**: You can select your desired model from a drop-down list containing different values: **Gemini 1.5 Flash, Gemini 1.5 Pro, Gemini 2.0 flash Lite, Gemini 2.0 Flash** and **Gemini 1.5 flash 8b**. Similar to the previous item, you must **press the button** on the right to save your selection. Note that each model has certain limitations concerning their reasoning capabilities, prompting, and various other capabilities to consider. Refer to the Gemini API docs for details.

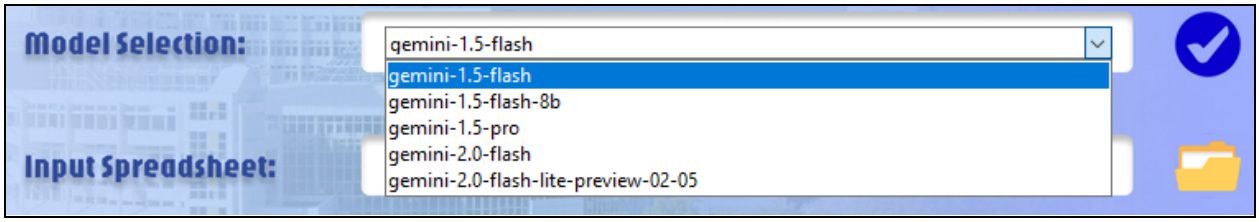


Figure 6: Combobox to select the desired LLM and the button to save the selection

6. **Input Spreadsheet**: Press the **folder icon** on the right, and a file dialog will open, allowing you to select your Excel spreadsheet. As you select your file, the path to the provided data will be saved automatically. Do not input the path directly in the textbox.

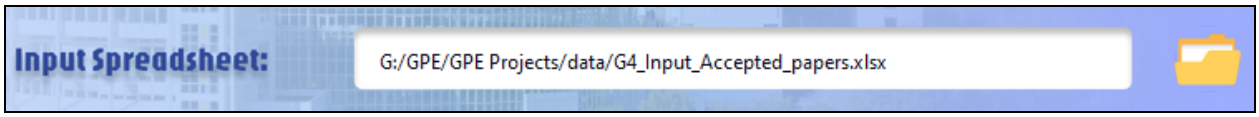


Figure 7: Press the folder icon to select the original data table

7. Make sure the program already registered your provided data by checking whether the following messages are displayed on the terminal below.

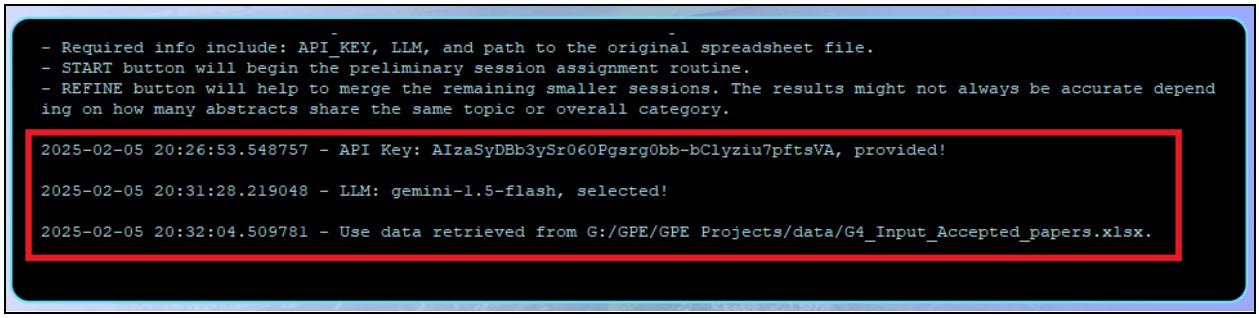


Figure 8: Terminal messages confirmed the input data are saved

8. **Processing**: After providing all necessary data, you could proceed with pressing the "**START**" button. The program would automatically start processing the provided list of abstracts, categorizing them, and assigning sessions. This might take some time depending on the number of abstracts.

Figure 9: START button

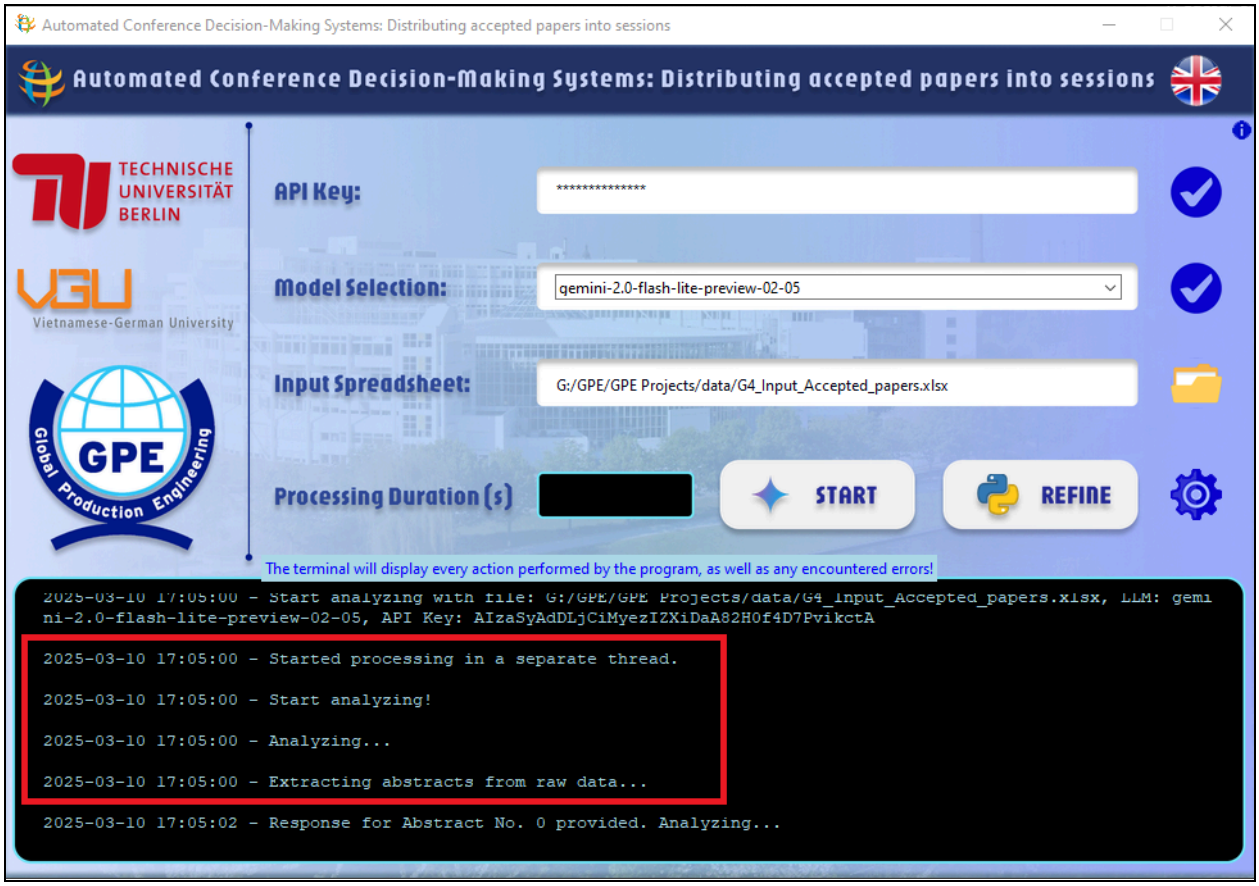


Figure 10: Terminal messages indicating that the process has been started and is underway

Output

1. A new Excel file (with the suffix "**_processed**" appended to the original filename) will be created in the same directory of the original spreadsheet file, containing the categorized abstracts and session assignments.

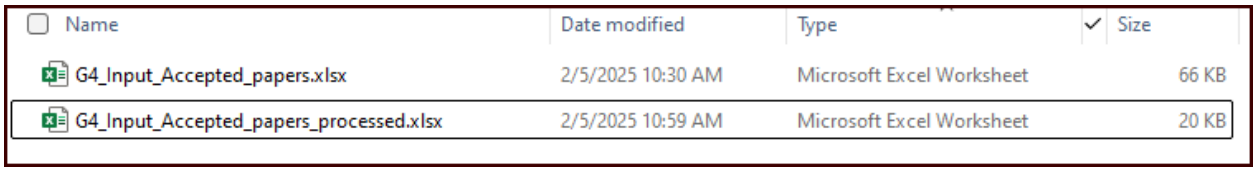
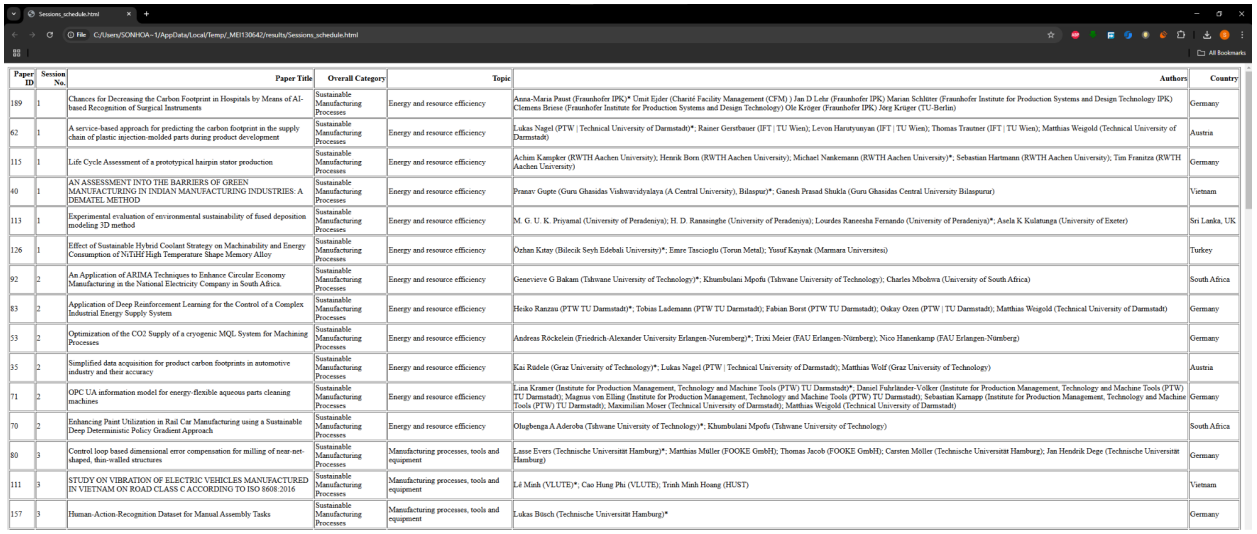


Figure 11: Results are saved as a new spreadsheet

2. An **HTML file** ("Sessions_schedule.html") will also be created in the "Results" folder located in the temporary directory in your C Drive, which will open in your **default web browser** for easy review of the schedule. Refer to the output messages printed on the terminal to locate the html file should you need to retrieve it.



Paper ID	Session No.	Paper Title	Overall Category	Topic	Authors	Country
189	1	Chances for Decreasing the Carbon Footprint in Hospitals by Means of AI-based Recognition of Surgical Instruments	Sustainable Manufacturing Processes	Energy and resource efficiency	Anna-Maria Faust (Fraunhofer IPK)* Umit Ejder (Charité Facility Management (CFM)) Jan D Lehr (Fraunhofer IPK) Marian Schlüter (Fraunhofer Institute for Production Systems and Design Technology IPK) Clemens Briese (Fraunhofer Institute for Production Systems and Design Technology) Ole Krüger (Fraunhofer IPK) Jörg Krüger (TU-Berlin)	Germany
62	1	A service-based approach for predicting the carbon footprint in the supply chain of plastic injection-molded parts during product development	Sustainable Manufacturing Processes	Energy and resource efficiency	Lukas Nagel (PTW | Technical University of Darmstadt)*, Rainer Gerstbauer (IFT | TU Wien); Levon Harutyunyan (IFT | TU Wien); Thomas Trautner (IFT | TU Wien); Matthias Weigold (Technical University of Darmstadt)	Austria
115	1	Life Cycle Assessment of a prototypical hairpin stator production	Sustainable Manufacturing Processes	Energy and resource efficiency	Achim Kampker (RWTH Aachen University); Henrik Born (RWTH Aachen University); Michael Nankemann (RWTH Aachen University)*; Sebastian Hartmann (RWTH Aachen University); Tim Frantzta (RWTH Aachen University)	Germany
40	1	AN ASSESSMENT INTO THE BARRIERS OF GREEN MANUFACTURING IN INDIAN MANUFACTURING INDUSTRIES: A DEMATEL METHOD	Sustainable Manufacturing Processes	Energy and resource efficiency	Pranav Gupte (Guru Ghasidas Vishwavidyalaya (A Central University), Bilaspur)*; Ganesh Prasad Shukla (Guru Ghasidas Central University Bilaspurur)	Vietnam
113	1	Experimental evaluation of environmental sustainability of fused deposition modeling 3D method	Sustainable Manufacturing Processes	Energy and resource efficiency	M. G. U. K. Priyamal (University of Peradeniya); H. D. Ranasinghe (University of Peradeniya); Lourdes Raneesha Fernando (University of Peradeniya)*, Asela K Kulatunga (University of Exeter)	Sri Lanka, UK
126	1	Effect of Sustainable Hybrid Coolant Strategy on Machinability and Energy Consumption of NiTiHf High Temperature Shape Memory Alloy	Sustainable Manufacturing Processes	Energy and resource efficiency	Ozhan Kitay (Bilecik Seyh Edebali University)*; Emre Tasciouglu (Torun Metal); Yusuf Kaynak (Marmara Universites)	Turkey
92	2	An Application of ARIMA Techniques to Enhance Circular Economy Manufacturing in the National Electricity Company in South Africa.	Sustainable Manufacturing Processes	Energy and resource efficiency	Genevieve G Bakam (Tshwane University of Technology)*; Khumbulani Mpofu (Tshwane University of Technology); Charles Mbohwa (University of South Africa)	South Africa
83	2	Application of Deep Reinforcement Learning for the Control of a Complex Industrial Energy Supply System	Sustainable Manufacturing Processes	Energy and resource efficiency	Heiko Ranzau (PTW TU Darmstadt)*, Tobias Lademann (PTW TU Darmstadt); Fabian Borst (PTW TU Darmstadt); Oskay Ozen (PTW | TU Darmstadt); Matthias Weigold (Technical University of Darmstadt)	Germany
53	2	Optimization of the CO2 Supply of a cryogenic MQL System for Machining Processes	Sustainable Manufacturing Processes	Energy and resource efficiency	Andreas Röckelein (Friedrich-Alexander University Erlangen-Nuremberg)*, Trini Meier (FAU Erlangen-Nürnberg); Nico Hanenkamp (FAU Erlangen-Nürnberg)	Germany
35	2	Simplified data acquisition for product carbon footprints in automotive industry and their accuracy	Sustainable Manufacturing Processes	Energy and resource efficiency	Kai Rädele (Graz University of Technology)*, Lukas Nagel (PTW | Technical University of Darmstadt); Matthias Wolf (Graz University of Technology)	Austria
71	2	OPC UA information model for energy-flexible aqueous parts cleaning machines	Sustainable Manufacturing Processes	Energy and resource efficiency	Lina Kramer (Institute for Production Management, Technology and Machine Tools (PTW) TU Darmstadt)*, Daniel Fuhrländer-Völker (Institute for Production Management, Technology and Machine Tools (PTW) TU Darmstadt); Magnus von Elling (Institute for Production Management, Technology and Machine Tools (PTW) TU Darmstadt); Sebastian Karnapp (Institute for Production Management, Technology and Machine Tools (PTW) TU Darmstadt); Maximilian Moser (Technical University of Darmstadt); Matthias Weigold (Technical University of Darmstadt)	Germany
70	2	Enhancing Paint Utilization in Rail Car Manufacturing using a Sustainable Deep Deterministic Policy Gradient Approach	Sustainable Manufacturing Processes	Energy and resource efficiency	Olugbenga A Aderoba (Tshwane University of Technology)*, Khumbulani Mpofu (Tshwane University of Technology)	South Africa
80	3	Control loop based dimensional error compensation for milling of near-net-shaped, thin-walled structures	Sustainable Manufacturing Processes	Manufacturing processes, tools and equipment	Lasse Evers (Technische Universität Hamburg)*; Matthias Müller (FOOKE GmbH); Thomas Jacob (FOOKE GmbH); Carsten Möller (Technische Universität Hamburg); Jan Hendrik Dege (Technische Universität Hamburg)	Germany
111	3	STUDY ON VIBRATION OF ELECTRIC VEHICLES MANUFACTURED IN VIETNAM ON ROAD CLASS C ACCORDING TO ISO 8608:2016	Sustainable Manufacturing Processes	Manufacturing processes, tools and equipment	Lê Minh (VLUTE)*; Cao Hung Phi (VLUTE); Trinh Minh Hoang (HUST)	Vietnam
157	3	Human-Action-Recognition Dataset for Manual Assembly Tasks	Sustainable Manufacturing Processes	Manufacturing processes, tools and equipment	Lukas Büsch (Technische Universität Hamburg)*	Germany

Figure 12.1: Results are also displayed in the default browser

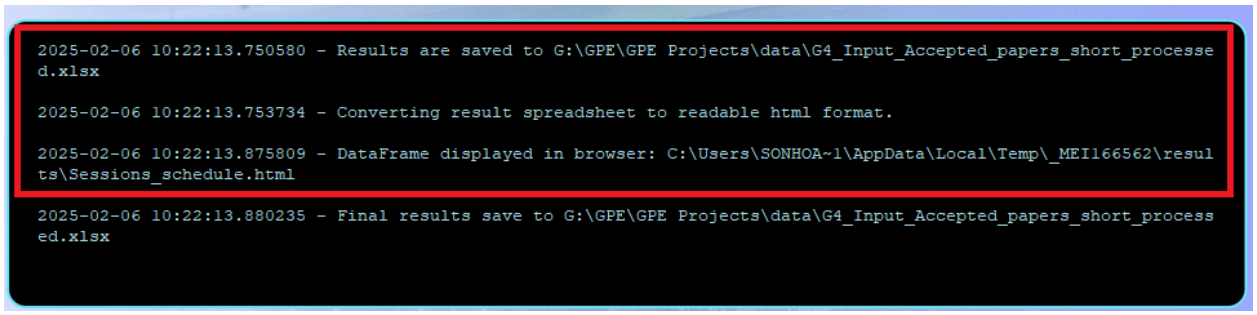


Figure 12.3: Locations of both html file and result are displayed on the terminal

Additional functions and other UI Interactions

1. **Refine function (Optional)**: As you click the "**REFINE**" button, you will be prompted to select an already processed list of abstracts (Abstracts that have been processed and assigned with session no.). Upon selecting the required list, the program will attempt to go through the provided list to merge any small sessions into sessions of 6 items each. Unlike the main categorization function, the refine function **does not rely on Google Generative language API** and runs on its own logic thus being capable of **providing results almost instantaneously**. Moreover, this function could be processed simultaneously while other functions are still running as each function has been configured to run in **Multi-threading** condition.

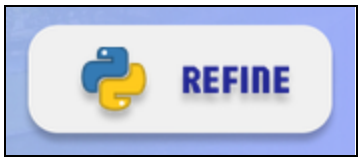


Figure 13: Refine button

2. **Other UI Interactions**:
    - **Terminal Box**: The bottom box will display program output, including progress messages, errors, and information about actions performed.

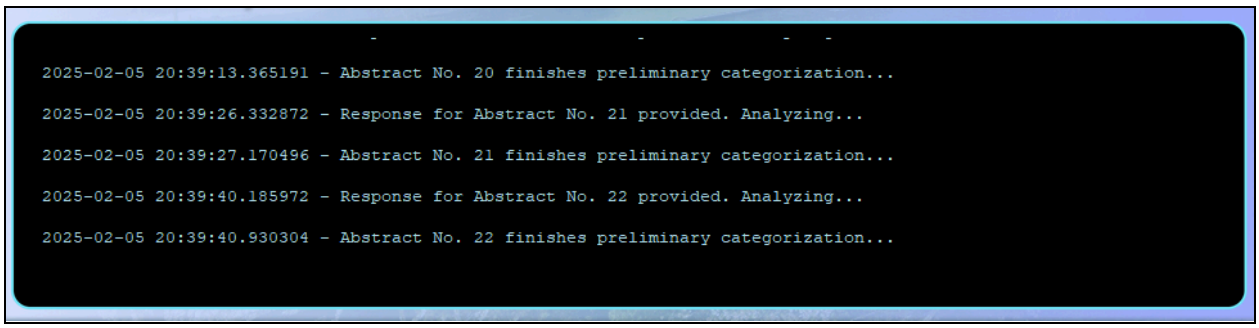


Figure 14: Terminal display

    - **Processing Duration (s)**: This little information box will display the **total time (in seconds)** taken to complete the entire abstract processing routine.

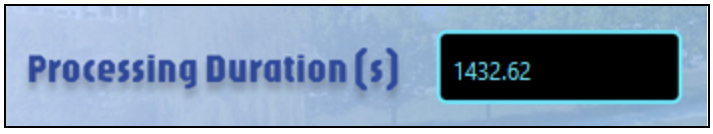


Figure 15: Processing Time display in seconds

- **Info Button**: Click the info button for more information about the program, its creators and to reveal the link to the **program's repository on GitHub for all source codes**.

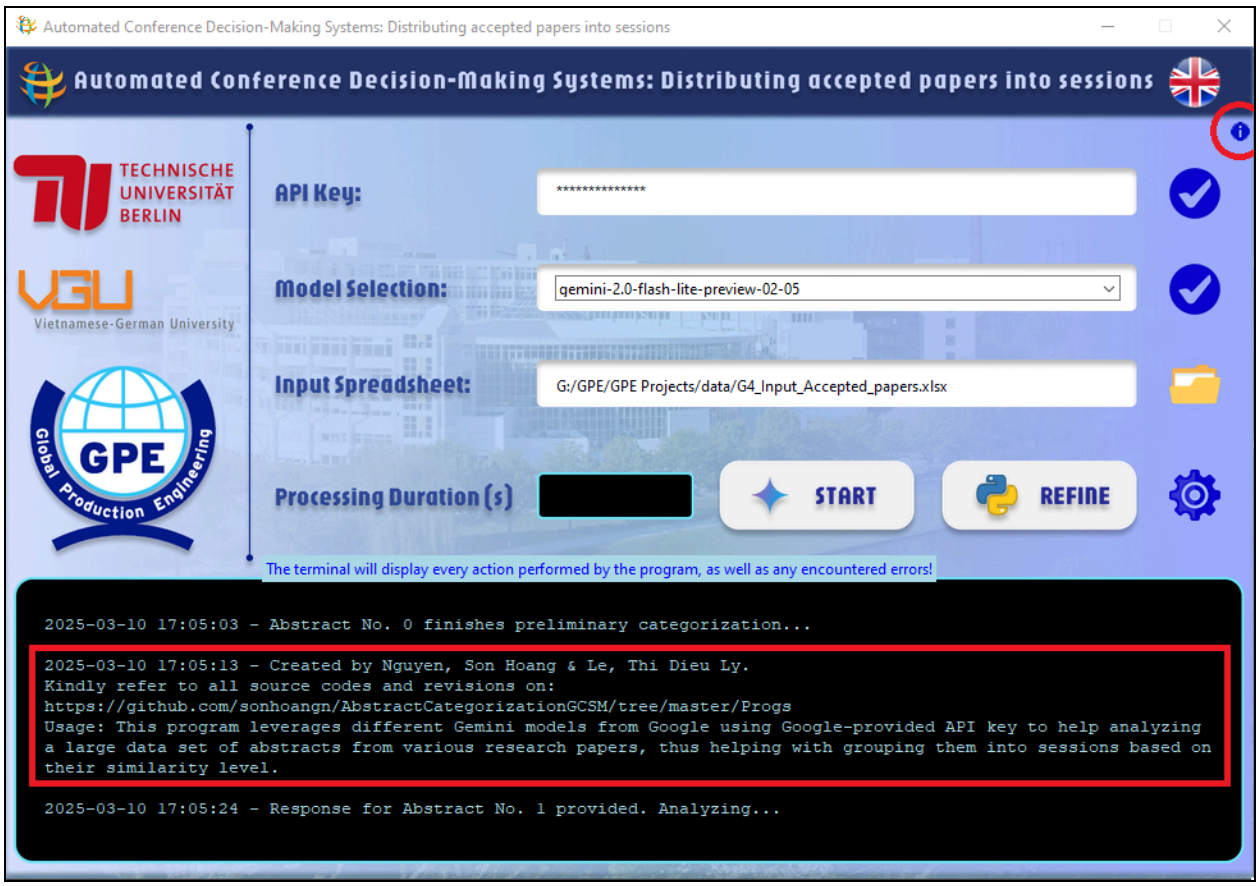


Figure 16: Info Button

- **Language Button**: Pressing the circular button on the top right corner of the window will cycle through different languages including **English** (default), **German**, and **Vietnamese**.

Figure 17: Language Button

Additional Instructions

Google Cloud API Key registration via Google AI Studio

- Navigate to Google AI Studio.
- You will be asked to either start using Gemini Ai or registering for a new API key.

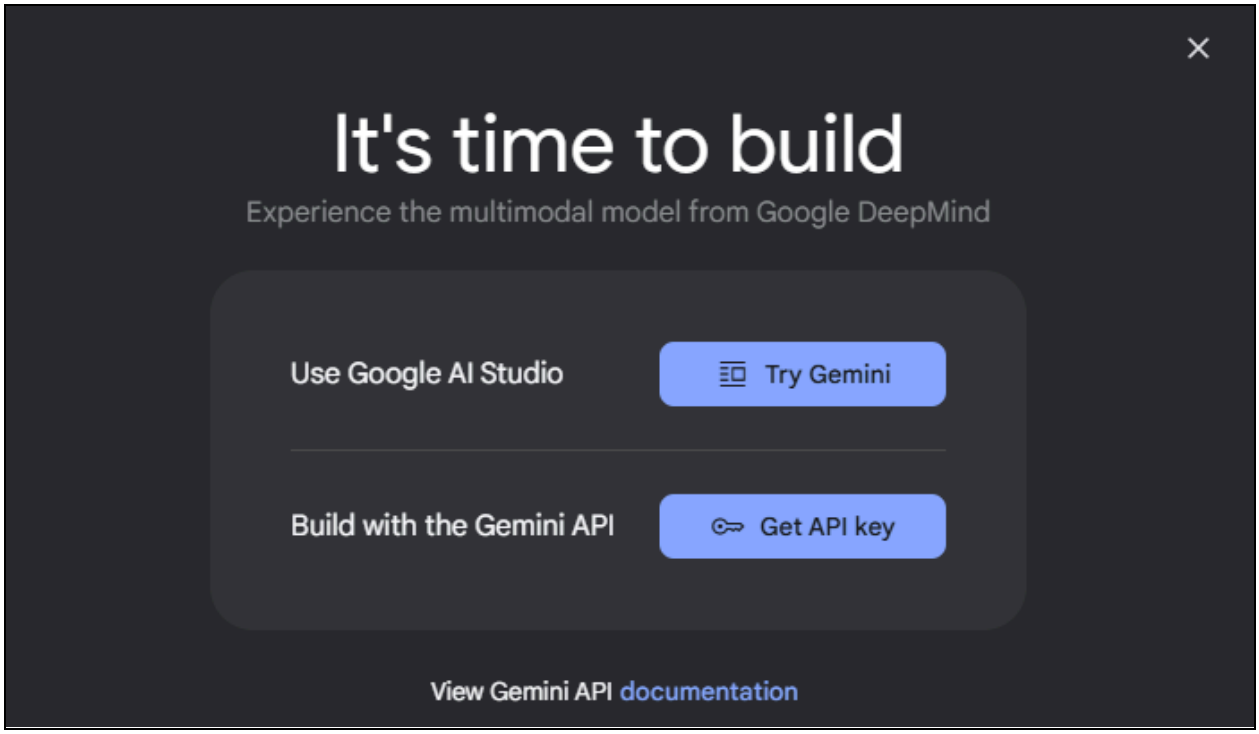


Figure 18: Accessing Google AI Studio

- Select "**Get API Key**"
- Select "**Create API Key**"

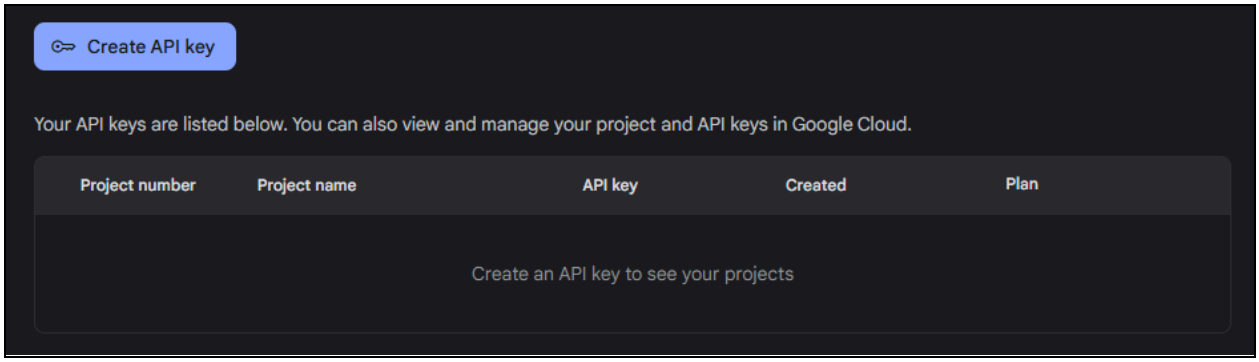


Figure 19: Create API Key

- Copy the newly created key and store it in any secure location

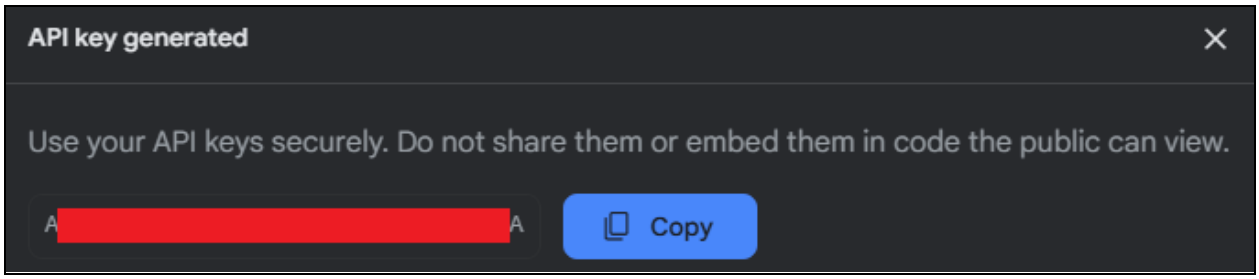


Figure 19: New API Key generated

Google Cloud API Key registration via Google Cloud Platform

- Log into https://console.cloud.google.com/ and create a **Google Cloud project**. Remember to check your account quotas to see if you need to increase or delete existing projects.

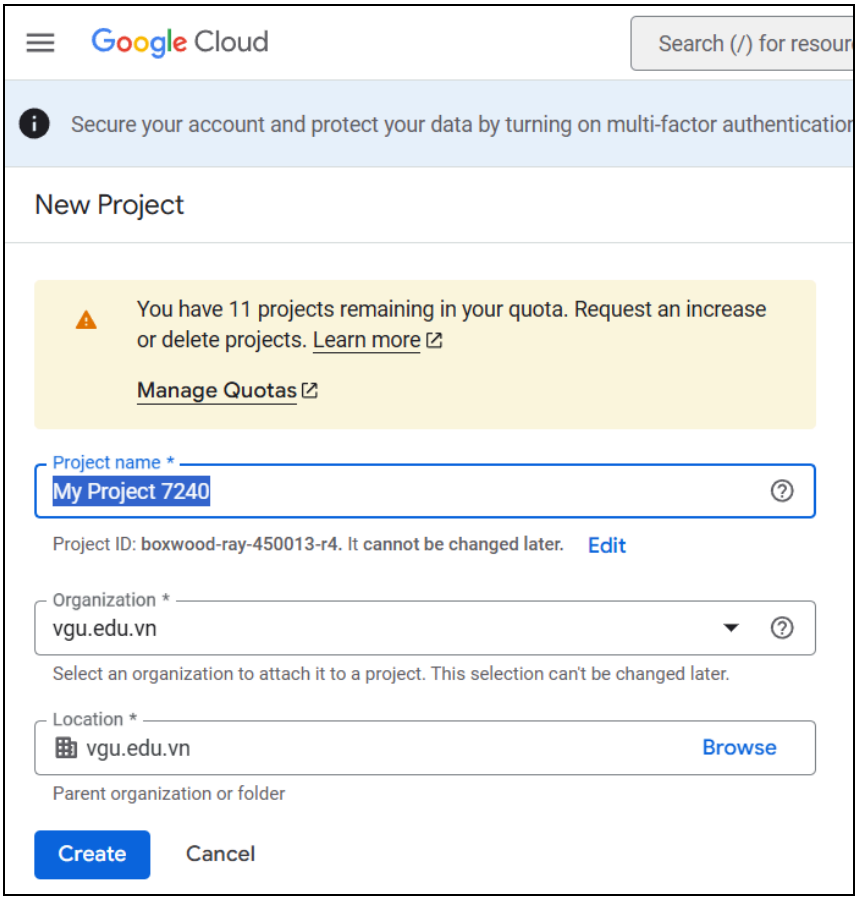


Figure 20: Create a new project via Google Cloud Console

- Select "**Enable APIs & Service**"

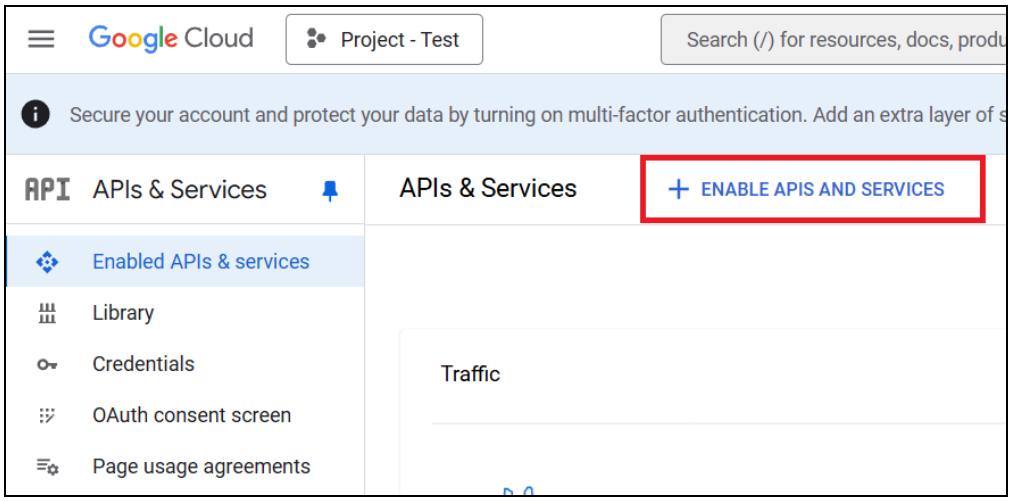


Figure 21: Enable APIs and Services for the project

- Type "**Generative Language API**" in the search bar.

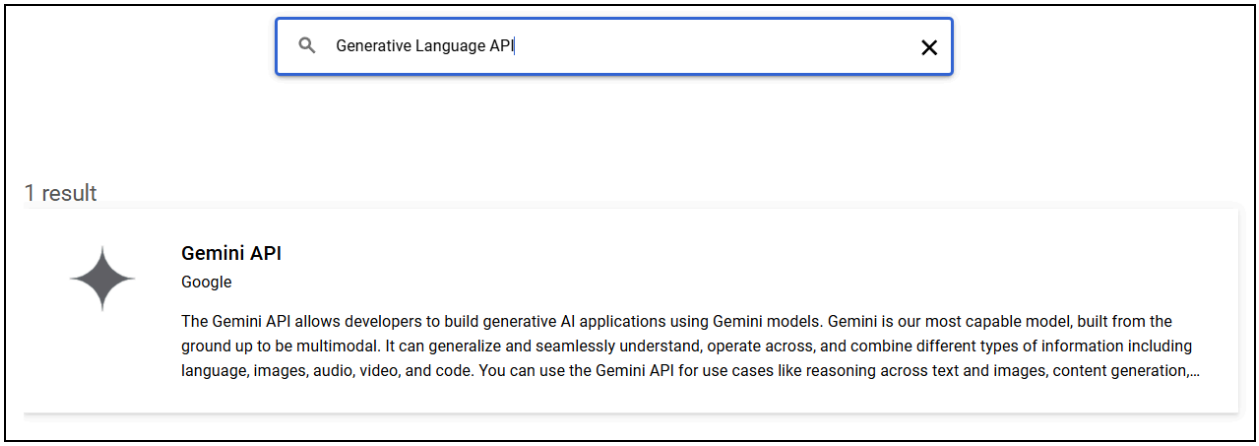


Figure 22: Generative Language API

- Select "**Gemini API**"
- Enable "**Gemini API**"

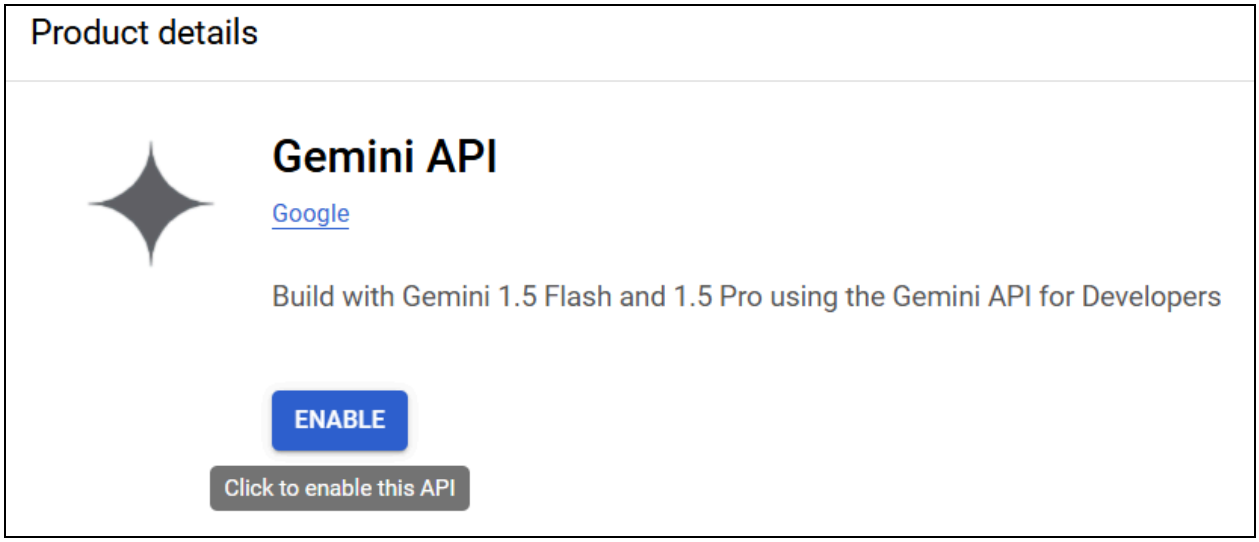


Figure 23: Enable "Gemini API"

- Select "**CREATE CREDENTIALS**"

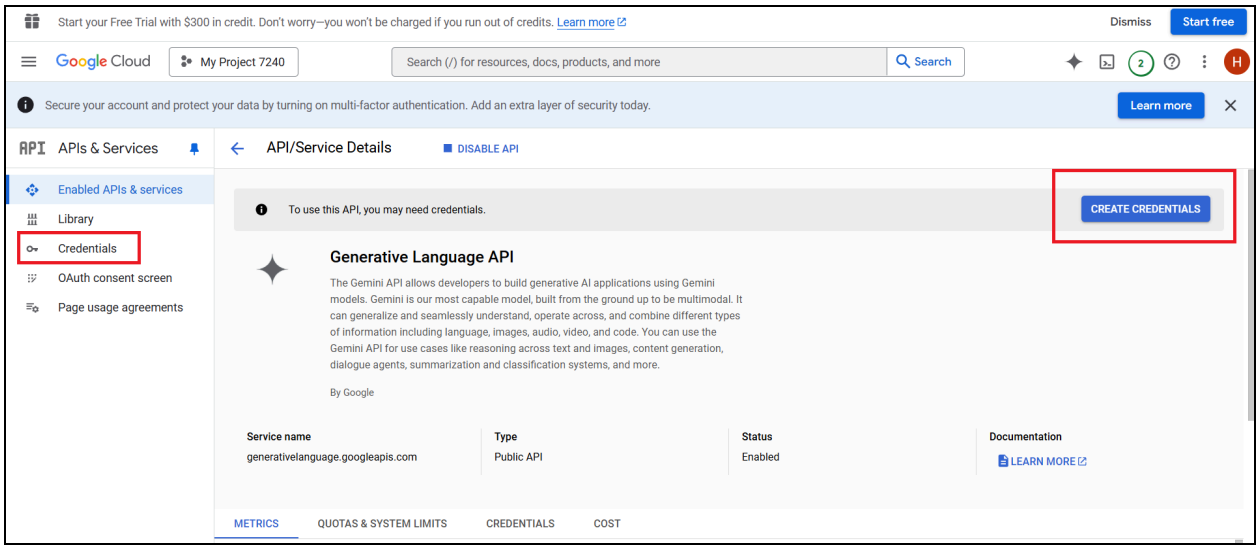


Figure 24: Credentials creation

- The newly created API Key is now accessible in the "**Credentials**" tab.

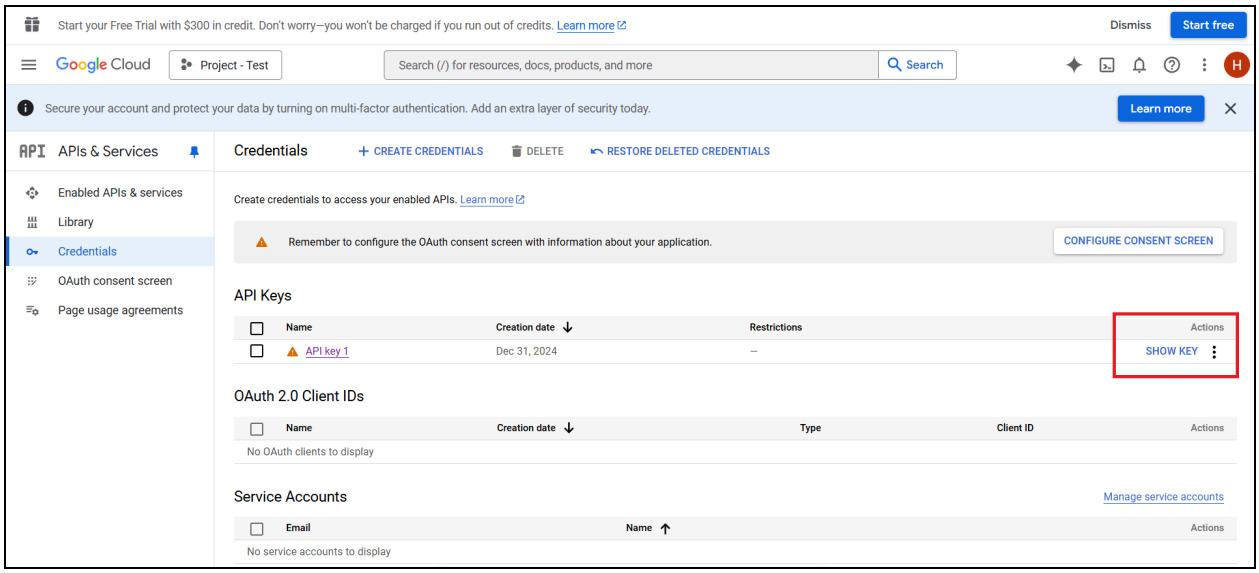


Figure 25: Show API key

- Press "**SHOW KEY**" and you would be able to copy the API Key to use in this program.

Additional notes

- Processing time may vary depending on the number of abstracts and the chosen Generative AI model.
- The program includes error handling for potential issues during abstract processing.
- Be sure to have a **valid Google Cloud API key** with appropriate access to the Generative AI models.
- Ensure the **source spreadsheet** to be provided to the program needs to **contain** the **required column** as aforementioned in the Usage section, else it would encounter an error as soon as the "START" button is pressed.
- All source codes and resources are accessible through: https://github.com/sonhoangn/AbstractCategorizationGCSM/tree/master
- In case you have any issues with the program or any questions, kindly send us an email at: sonhoangn@yahoo.com or dieulylt@gmail.com

Understanding the Codebase

Program Architecture

- The overall architecture of the program is straightforward: Root folder comprises of the "**.venv**" folder, containing all **required libs**, and "**Progs**" folder, containings all **py programs** and **UI assets**.

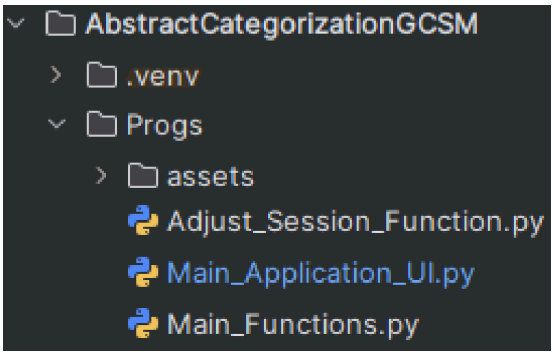


Figure 26: Program architecture

Understanding Main_Functions.py

Main_Functions.py contains the core logic for processing research paper abstracts and assigning them to sessions based on their content similarity. It utilizes Google's Gemini models via the Google API to analyze the abstracts and determine their relationships.

Functionality	Code Snippets
ct()	

• Returns the current date and time as a formatted string (YYYY-MM-DD HH:MM:SS).
• Used for logging and timestamping messages.
	`# Define current running time`
	`def ct():` 25 usages Son Hoang
	`    crtm = datetime.datetime.now()`
	`    return crtm.strftime("%Y-%m-%d %H:%M:%S")`

categorize_abstract(index, abstract, model)	

• Takes an abstract, its index, and a Gemini model as input.
• Constructs a prompt for the Gemini model, instructing it to categorize the abstract based on predefined criteria.
• Sends the prompt to the Gemini API and processes the response.
	`prompt = f"""`
	`Paper index:`
	`{index}`
	`Abstract:`
	`{abstract}`
	`"""`
	`# Configure generative ai response`
	`response = model.generate_content(prompt)`

• Extracts relevant information (overall category, research field, research methods, scope, purpose, forecasted presentation time, and token counts) from the response.	`# Find the best fit overall category name from the response`
	`line1 = [line for line in response.text.split("\n") if line.startswith("- Overall Category: ")]`
	`if line1:`
	`    overall_category = line1[0].split(": ")[1].strip()`
	`else:`
	`    overall_category = "N/A"`

- Provide token count of the prompt and the response for pricing consideration.

```
# Get token count
prompt_tokens = str(model.count_tokens(
 prompt)).split(": ")[1].strip()
response_tokens = str(model.count_tokens(
 response.text)).split(": ")[1].strip()
```

- Returns the extracted information.

```
return overall_category, research_field,
 research_method, scope, purpose,
 forecasted_time, prompt_tokens,
 response_tokens
```

session_assignment(df)

- Takes a pandas DataFrame as input.
- Performs initial session assignment based on identical "Topic" values.
- Groups data and counts items per group.
- Creates refined grouping, to limit group sizes.
- Returns the modified DataFrame.

```
# Session Assignment
def session_assignment(df):  1 usage  Son Hoang
    # Initial Grouping based on identical Topics
    df['Grouping'] = df.groupby('Topic').ngroup() + 1
    # Count items per Grouping, "Topic" column is
     selected as reference
    df["Count"] = df.groupby("Grouping")[
     "Topic"].transform("count")
    # Initialize a new column for refined grouping,
     this temp. column will not show up in the result
    df["Refined Grouping"] = ""
    refined_groups = []
    for group, subset in df.groupby("Grouping"):
        if len(subset) <= 6:
            refined_groups.extend([group] * len(
             subset))
        else:
            # Create smaller groups, using an
             index-based suffix
            for i, (_, row) in enumerate(
             subset.iterrows()):
                refined_groups.append(f"{group}_{i //
                 6 + 1}")
    # Assigned refined group naming to the temp. column
    df["Refined Grouping"] = refined_groups
    # Return the data frame containing the temp.
     refined grouping column
    return df
```

merge_groups(df)

- Merges smaller groups (less than 6 items) based on their "Overall Category" to create groups of 6.
- Returns a dictionary mapping old group names to new merged group names.

```python
# Merge smaller groups into groups of 6 based on their identical
 Overall Category
def merge_groups(df):  1 usage  Son Hoang
    # Count items per refined group
    group_counts = df.groupby("Refined Grouping").size()
    # Identify small groups of less than 6 row items
    small_groups = {g: df[df["Grouping"] == g] for g, size in
     group_counts.items() if size < 6}
    # Mapping "Overall Category" to small groups
    category_to_groups = defaultdict(set)
    for group, rows in small_groups.items():
        unique_categories = rows["Overall Category"].unique()
        for category in unique_categories:
            category_to_groups[category].add(group)
    # Merge small groups into exactly 6-row groups
    merged_groups = {}
    # Track assigned group
    assigned = set()
    # New Group assignment counter
    new_group_id = 1
    for group, rows in small_groups.items():
        if group is assigned:
            continue
        # Start a new merged group
        current_merge = [group]
        assigned.add(group)
        # Find potential merge partners
        for category in rows["Overall Category"].unique():
            possible_partners = category_to_groups[category] -
             set(current_merge) - assigned
            for partner in possible_partners:
                if len(df[df["Grouping"].isin(current_merge)]) +
                 len(df[df["Grouping"] == partner]) <= 6:
                    current_merge.append(partner)
                    assigned.add(partner)
                # Stop merging if exactly 6 rows are reached
                if len(df[df["Grouping"].isin(current_merge)]) ==
                 6:
                    break
            if len(df[df["Grouping"].isin(current_merge)]) == 6:
                break
        # Assigned new ID for merged group
        for g in current_merge:
            merged_groups[g] = f"Merged_{new_group_id}"
        new_group_id += 1
    return merged_groups
```

adjust_session_numbers(df)

- Refines session numbers to ensure groups of 6 items.
- Merges smaller groups (type A) based on "Overall Category" and type.

```python
# Identify groups with exactly 6 items
session_counts = df["Session No."].value_counts()
exact_6_sessions = session_counts[session_counts == 6].index
less_than_6_sessions = session_counts[session_counts < 6].index

# Mark groups as Type A (same "Overall Category") or Type B (different "Overall Category")
session_types = {}
for session in less_than_6_sessions:
    categories = df[df["Session No."] == session]["Overall Category"].unique()
    session_types[session] = "A" if len(categories) == 1 else "B"

    # Merge Type A groups within the same "Overall Category"
    merged_session_map = {}  # Stores new session numbers
    new_session_no = int(df["Session No."].max()) + 1  # Start numbering after the highest existing Session No.

    type_a_sessions = [s for s, t in session_types.items() if t == "A"]
    grouped_a = df[df["Session No."].isin(type_a_sessions)].groupby("Overall Category")

    for _, group in grouped_a:
        session_list = group["Session No."].unique()
        items = group.copy()

        # Split into smaller groups of 6 if needed
        for i in range(0, len(items), 6):
            batch = items.iloc[i:i + 6]
            merged_session_no = int(new_session_no)
            new_session_no += 1  # Increment for the next batch

            for session in batch["Session No."].unique():
                merged_session_map[int(session)] = merged_session_no
```

- Merges smaller groups (type B) without relying on "Overall Category" and type.
- Ensure new groups containing exactly 6 items.
- Renames session numbers sequentially.
- Returns the DataFrame with adjusted session numbers.

```
# Merge Type B sessions (without considering "Overall
 Category")
type_b_sessions = [s for s, t in session_types.items() if t ==
  "B"]
leftover_df = df[df["Session No."].isin(
 type_b_sessions)].copy()

for i in range(0, len(leftover_df), 6):
    leftover_df.loc[leftover_df.index[i:i+6], "Adjusted
     Session No."] = new_session_no
    new_session_no += 1  # Increment for the next batch

# Apply merged session mapping
df["Adjusted Session No."] = df["Session No."].replace(
 merged_session_map)

# Ensure exact 6-item groups remain unchanged
df.loc[df["Session No."].isin(exact_6_sessions), "Adjusted
 Session No."] = df["Session No."]

# Assign new session numbers to any remaining unmerged groups
remaining_unmerged = df[df["Adjusted Session No."].isna()]
for i in range(0, len(remaining_unmerged), 6):
    df.loc[remaining_unmerged.index[i:i+6], "Adjusted Session
     No."] = new_session_no
    new_session_no += 1

# Renaming "Adjusted Session No." values sequentially
unique_sessions = df["Adjusted Session No."].dropna().unique()
session_rename_map = {old: new for new, old in enumerate(
 sorted(unique_sessions), start=1)}
df["Adjusted Session No."] = df["Adjusted Session No."].map(
 session_rename_map)

return df
```

input_from_spreadsheet(file_path, model, llm_selection)

- Reads data from an Excel spreadsheet into a pandas DataFrame.
- Iterates through the abstracts, calling categorize_abstract() for each.
- Handles potential errors during abstract processing.
- Records processing time.
- Creates a new dataframe with the results.
- Returns the resulting DataFrame.

write_to_excel(df_results, file_path, llm)

- Writes the processed data to a new Excel spreadsheet.
- Saves only relevant columns.
- Returns the path to the output file.

write_to_excel_display(df_results, file_path, llm)

- Same as write_to_excel, but also calls browser_display.
- Returns the path to the output file.

unexpected_characters(text)

- Replaces unexpected unicode characters: '\u01b0', (UnicodeEncodeError) in a string.
- Used to clean data before displaying it in HTML.

browser_display(df_final, llm)

- Converts a DataFrame to an HTML table.
- Saves the HTML table to a file.
- Opens the HTML file in a web browser.

main(file_path, llm_selection, API_KEY)

- Configures the Gemini API with the provided API key and model selection.
- Calls input_from_spreadsheet() to process the abstracts.
- Calls session_assignment to perform initial grouping.
- Calls merge_groups to merge small groups.
- Calls adjust_session_numbers to refine session numbers.
- Writes the final results to an Excel spreadsheet and displays it in a browser.

Libraries usage:

- **datetime:** For timestamping and logging.
- **os:** For operating system interactions, specifically creating directories.
- **time:** For timing operations and introducing delays.
- **webbrowser:** For displaying results in a web browser.
- **collections.defaultdict:** For creating dictionaries with default values.
- **pathlib.Path:** For handling file paths.
- **google.generativeai as genai:** For interacting with the Gemini API.
- **pandas as pd:** For reading, processing, and writing data to Excel spreadsheets.

```
import datetime
import os
import time
import webbrowser
from collections import defaultdict
from pathlib import Path

import google.generativeai as genai
import pandas as pd
```

Understanding Main_Application_UI.py

Main_Application_UI.py provides a graphical user interface (GUI) for the abstract categorization and session assignment program, allowing data input, settings configuration, analysis process initiation, and results display.

Functionality	Code Snippets
relative_to_assets(path: str) -> Path	
• Constructs absolute paths to asset files.	`def relative_to_assets(path: str) -> Path:` 31 usages Son Hoang `return ASSETS_PATH / Path(path)`
instruction_message_terminal(event), instruction_message_input(event), instruction_message_llm_selection(event), instruction_message_api_input(event)	

• Displays informative messages when the cursor hovers over specific input fields.
• Utilizes tkinter labels and bindings to provide temporary instruction popups.
	`def instruction_message_api_input(event):` 1 usage Son Hoang
	`message_label = tkinter.Label(window, text="Please kindly provide your Google API key.", bg="lightblue", fg="blue")`
	`message_label.place(x=429, y=133) # Positioning`
	`# Define entering and leaving object area`
	`def on_enter(event):` Son Hoang
	`message_label.after(ms: 20000, message_label.destroy)`
	`def on_leave(event):` Son Hoang
	`message_label.destroy()`
	`window.bind("<Enter>", on_enter, add=True)`
	`window.bind("<Leave>", on_leave, add=True)`
	`message_label.my_id = window.bind("<Leave>", lambda event: message_label.destroy(), add=True)`
	`message_label.my_id_enter = window.bind("<Enter>", lambda event: None, add=True)`

class ImageButton(tkinter.Button)

- Cycle between different UI elements showing different language settings for the GUI.

```
# Language Toggle
class ImageButton(
 tkinter.Button):  3 usages  Son Hoang
    def __init__(self, master, image_paths,
     **kwargs):  # Take a list of image
     paths  Son Hoang
        self.images = [PhotoImage(file=path) for
         path in image_paths]
        super().__init__(master, image=self.images[
         0], **kwargs)
        self.config(compound="center")
        self.current_image = 0

    def switch_image(self):  3 usages  Son Hoang
        self.current_image = (self.current_image +
         1) % len(self.images) # Cycle through images
        self.config(image=self.images[
         self.current_image])
        self.image = self.images[self.current_image]
```

switch_background()

- Switches the background image based on the selected language.

```
# Background Toggle
def switch_background():  1 usage  Son Hoang
    global image_image_1
    images = [image_image_1_EN, image_image_1_DE,
     image_image_1_VI]
    current_index = images.index(image_image_1)
    next_index = (current_index + 1) % len(images)
    image_image_1 = images[next_index]
    canvas.itemconfig(image_1, image=image_image_1)
    canvas.image = image_image_1 # Keep reference
```

def browse_ss(), button_4

- Browse the target spreadsheet file.
- Configure button's behaviour to start asking users for the file upon getting pressed.
- Configure UI attributes for button 4.

```
# Browse spreadsheet file
file_path = None
def browse_ss():  1 usage  Son Hoang
    global file_path
    file_path = filedialog.askopenfilename(
            title="Select Abstracts List",
            filetypes=[("Excel files", "*.xlsx;*.xls;*.csv")]
    )
    if file_path:
        entry_1.delete( first: 0,  last: "end")
        entry_1.insert( index: 0, file_path)
    print(f"{ct()} - Use data retrieved from {file_path}.\n")
    return file_path

# Browse file button
button_image_4 = PhotoImage(
    file=relative_to_assets("browse.png"))
button_4 = Button(
    image=button_image_4,
    borderwidth=0,
    highlightthickness=0,
    command=browse_ss,
    relief="flat"
)
button_4.place(
    x=909.0,
    y=244.0,
    width=40.0,
    height=40.0
)
```

save_llm_selection(), save_api_key()

- Saves the selected LLM model from the Combobox.
- Saves the provided API key from the input field.

```
# Save LLM Selection
llm_selection = None
def save_llm_selection():  1 usage  Son Hoang
    global llm_selection
    llm_selection = entry_3.get()
    if not llm_selection:
        messagebox.showwarning( title: "Warning",
            message: "No LLM selection detected!")
        return
    print(f"{ct()} - LLM: {llm_selection},
     selected!\n")
    return llm_selection
```

```python
# Save API Key
API_KEY = None
def save_api_key():  1 usage  Son Hoang
    global API_KEY
    API_KEY = entry_4.get()
    if not API_KEY:
        messagebox.showwarning( title: "Warning",
          message: "No API Key detected!")
        return
    print(f"{ct()} - API Key: {API_KEY},
     provided!\n")
    return API_KEY
```

llm_options

- Setup a ComboBox containing selectable LLMs.
- Assign LLM ComboBox to entry_3.
- Configure

```
# LLM Selection box
llm_options = ["gemini-1.5-flash",
 "gemini-1.5-flash-8b", "gemini-1.5-pro",
 "gemini-2.0-flash",
 "gemini-2.0-flash-lite-preview-02-05"]
entry_image_3 = PhotoImage(
    file=relative_to_assets("inputarea.png"))
entry_bg_3 = canvas.create_image(
     *args: 650.0,
    190.0,
    image=entry_image_3
)
entry_3 = ttk.Combobox(
    window,
    values=llm_options,
    state="readonly",
    width=334
)
entry_3.place(
    x=429.0,
    y=179.0,
    width=442.0,
    height=20.0
)
entry_3.bind("<Enter>",
 instruction_message_llm_selection)
```

button_5

- Configure button's behaviour to save user's selection upon getting pressed.
- Configure UI attributes for button 5.

```
# LLM Selection button
button_image_5 = PhotoImage(
    file=relative_to_assets("check.png"))
button_5 = Button(
    image=button_image_5,
    borderwidth=0,
    highlightthickness=0,
    command=save_llm_selection,
    relief="flat"
)
button_5.place(
    x=909.0,
    y=170.0,
    width=40.0,
    height=40.0
)
```

Button_5, entry_4

- Configure text input area for API key.
- Configure button's behaviour to save API key upon getting pressed.
- Configure UI attributes for button 6.

```
# API_Key input box
entry_image_4 = PhotoImage(
    file=relative_to_assets("inputarea.png"))
entry_bg_4 = canvas.create_image(
    *args: 650.0,
    114.99999999999999,
    image=entry_image_4
)
entry_4 = Entry(
    bd=0,
    bg="#FFFFFF",
    fg="#000716",
    highlightthickness=0
)
entry_4.place(
    x=429.0,
    y=103.99999999999999,
    width=442.0,
    height=20.0
)
entry_4.bind("<Enter>",
 instruction_message_api_input)
```

```
# Input API_Key
button_image_6 = PhotoImage(
    file=relative_to_assets("check.png"))
button_6 = Button(
    image=button_image_6,
    borderwidth=0,
    highlightthickness=0,
    command=save_api_key,
    relief="flat"
)
button_6.place(
    x=909.0,
    y=94.99999999999999,
    width=40.0,
    height=40.0
)
```

start_analysis()

- Initiates the analysis process by calling Main_Functions.main() in a separate thread.
- Performs input validation and displays warning messages if necessary.

StdoutRedirector(object)

- A class that redirects standard output to the terminal text widget.
- Extracts processing time from the output and displays it in the time entry field.

Refine()

- Calls Adjust_Session_Function.main() to refine the sessions.

Global Variables:

- **FILE_PATH:** The path to the directory containing the script.
- **ASSETS_PATH:** The path to the directory containing GUI assets (images).
- **ICON_PATH:** The path to the application icon.
- **window:** The main Tkinter window.

```
# Global pathing parameters
FILE_PATH = Path(__file__).parent
ASSETS_PATH = FILE_PATH / "assets" / "frame0"
ICON_PATH = ASSETS_PATH / "icon.png"
```

```
window = Tk()
window.title("Automated Conference Decision-Making
 Systems: Distributing accepted papers into
 sessions")

window.geometry("975x650")
window.configure(bg = "#9fd0f5")
```

- **canvas:** The Tkinter canvas for drawing the GUI.
- **image_image_1, image_image_1_EN, image_image_1_DE, image_image_1_VI:** Variables for background image management and language switching.

```
canvas = Canvas(
    window,
    bg = "#9fd0f5",
    height = 650,
    width = 975,
    bd = 0,
    highlightthickness = 0,
    relief = "ridge"
)

canvas.place(x = 0, y = 0)
# Initialize background images
image_image_1_EN = PhotoImage(
 file=relative_to_assets("bg.png"))
image_image_1_DE = PhotoImage(
 file=relative_to_assets("bg_DE.png"))
image_image_1_VI = PhotoImage(
 file=relative_to_assets("bg_VI.png"))
image_image_1 = image_image_1_EN # Start with
 English version
image_1 = canvas.create_image(
    *args: 487.0,
    325.0,
    image=image_image_1
)
canvas.image = image_image_1 # Keep reference
```

- **file_path:** Stores the path to the selected spreadsheet file.
- **llm_selection:** Stores the selected Gemini model.
- **API_KEY:** Stores the provided Google API key.

```
# Browse spreadsheet file
file_path = None
```

```
# Save LLM Selection
llm_selection = None
```

```
# Save API Key
API_KEY = None
```

Libraries usage:

- **datetime:** For timestamping and logging.
- **sys:** For system-specific parameters and functions.
- **threading:** For running the analysis process in a separate thread to prevent GUI freezing.
- **Main_Functions:** For calling the core analysis functions.
- **tkinter and its elements:** For creating the GUI, and all UI elements (input fields, buttons, dialogs, combobox, etc.)
- **pathlib.Path:** For handling file paths.
- **re:** For regular expression operations, used to extract processing time from output.
- **Adjust_Session_Function:** For calling the session refinement functions.

```
import sys                                          ⚠ 5  ⚠ 30  ✓ 6
import re
import datetime
import Main_Functions
from Main_Functions import ct
from pathlib import Path
import tkinter
import threading
import Adjust_Session_Function

# Import all critical TKinter elements
from tkinter import Tk, Canvas, Entry, Text, Button,
  PhotoImage, messagebox, filedialog, ttk
```