**Automatic Form Filling using GPT**  
Welcome to the world of Automatic Form Filling powered by GPT technology! In this project, we utilize Python to streamline the process of filling out various forms effortlessly.

Table of Contents

[Getting Started: I](#_Toc162607583)

[Project Structure: II](#_Toc162607584)

[Privacy and Customization: II](#_Toc162607585)

[Functionality Overview: II](#_Toc162607586)

[Supplementary Methods Explored: IV](#_Toc162607587)

[Appendix: VI](#_Toc162607588)

# Getting Started:

Before diving in, ensure you have all the necessary packages installed. The requirement for the complete project is provided in the requirements.txt. Some additional packages might be needed depending on the functionalities of the modules I have imported. Please make sure to install these packages in your Python environment using pip or any package manager you prefer. You can do this by running the following command:

pip3 install streamlit-extras streamlit-authenticator PyYAML fuzzywuzzy pandas pdf2docx openai gradio retrying flask opencv-python Pillow pytesseract opencv-python-headless textblob

If you're using Python 2, replace pip3 with pip.

# Project Structure:

The primary Python executable file responsible for orchestrating the system's operations is named FormFiller.py.

# Privacy and Customization:

In adherence to privacy regulations, users are urged to employ customized API keys. Moreover, the selection of the GPT model is adaptable to individual requirements.

# Functionality Overview:

###### Data Collection and Preparation -

* The algorithm is designed to handle data collection and preparation from a variety of input and data file formats.
* The Algorithm uses conditional statements (if, elseif) to determine the file format,
* Supported formats: Word files (.docx, .doc), Excel files (.xls, .xlsx), and PDF files. If an unsupported file format is encountered, the code informs the user with a message stating, "Invalid form file format. Supported formats: Word Files (.docx, .doc), Excel Files (.xls, .xlsx), and PDF Files."
* For data file the algorithm includes a versatile function named "**read\_data\_file**."

###### Answer Generation -

* The function **‘generate\_answers\_for\_questions’** puts together the process of generating answers to a set of extracted questions based on the context and a persona profile.
* The **‘if hasattr()’** function is used to check whether the ‘question’ that is extracted has a ‘text’ attribute. This enhances the versatility of the function, allowing it to work with both Excel text and Word file texts.
* Following that, the generated answers are refined using the function **‘answer\_cleanup’**. This step involves removing extra information and formatting readable answers.

###### Communication with OpenAI -

* This function serves as a crucial piece of code in the communication process with OpenAI. I have designed this function with retry logic to enhance the robustness of API calls, especially in situations where there are poor network or server issues.
* Retry Logic: This function uses a while loop to manage the retry mechanism, while the loop is active the code tracks the number of retires attempted using the ‘retries’ variable. The maximum number of retries is set by the **‘max\_retries’** parameter, which defaults to 3 but can be adjusted based on the requirements.
* After the communication is secured, a valid response is received, the function returns the JSON response using **‘response.json()’** and the response is stored there for further processing.

###### Answer Cleaning -

* This function is an essential part after the response is received and is stored in **‘response.jason()’**.
* If the content extraction is successful and no errors occur, the function will return the extracted content. However, in the event of an error, it returns a generic error message.

Persona Generation -  
 This feature is used to generate the answers based on the characteristics of the person.

The process of generating a persona begins with the extraction of the data provided. Once the data is provided, it analyses and identifies key attributes or themes that inform the person’s characteristics.

the **‘generate\_persona\_for\_form\_filler’** function determines the persona based on the context such as doctor, sports, finance, or student, which tailor the persona to match with the specific domains of the application.

If in case the given context isn’t identified as a persona that is already defined, I have included a more generalized form filler as a persona

Once the persona is generated based on the information, the particular persona is extracted using **‘extract\_persona\_info’** and is used in the prompt for generating answers. So that means when the GPT answers a particular question, it keeps in mind that the characteristics of the model are the ones that are extracted.

###### Form Filling -

The system encompasses functions tailored to filling forms across various formats:

* **fill\_in\_answers\_excel\_openpyxl:** For Excel Spreadsheets.
* **fill\_in\_answers\_word:** For Word and PDF files.

These are the functions responsible for filling in the answers.  
they work by searching for the questions and filling in its corresponding answers in the next empty space. The excel function ignores the questions that have read only cells, which filters again to avoid filling in for texts that does not require answers. It prints out the ignored questions too. Since PDF files are converted into word, it uses the same function to perform the filling.

# Supplementary Methods Explored:

In addition to the above functionalities, several alternative methods were explored during the system's development phase. Here are some of them:

<extract_coordinates.py>:Initially tried to extract the coordinates of the extracted text to place the answers fixing a certain distance between them. This code was used to extract coordinates of the text.

[**extract\_text\_around.py**](extract_text_around.py)**:**

Also tried to extract the text around the identified boxes, but the code used to not extract the complete word even though the distance threshold was adjusted.

[**extract\_text\_insidebox.py**](extract_text_insidebox.py)**:**

This code extracts the texts inside the identified boxes in the form. Also prints out the boxes that is identified.

[**extract\_text\_using\_:\_?\_.\_.py**](file:////Users/hope/Desktop/Final%20Project/Supplementary%20Material/extract_text_using_:_%3f_._.py)**:**

This piece of code extracts all the texts that has these characters at the end of their questions, mostly indicating them as questions. The characters are, ‘:’, ‘?’, ‘.’.

[**input\_word\_box.py**](input_word_box.py)**:**

This was tired to insert random texts inside the boxes while also trying to preserve the format of the document.

PDF\_OCR.py:

Both the files [PDF\_OCR\_1](PDF_OCR_1.py) and [PDF\_OCR\_2](PDF_OCR_2.py) was used to extract the texts using ORC method from pdf files.

[**pdf\_to\_json.py**](pdf_to_json.py)**:**

When trying to convert from pdf to docx, considered converting into json which would make it easier to map the answers.

[**Question\_LLM.py**](Question_LLM.py)**:**

This was one of the other methods tried to interact with the OpenAI without using HTTP request and API endpoint directly, instead making use of OpenAI Python SDK tools.

[**Sentiment\_analysis.py**](Sentiment_analysis.py)**:**

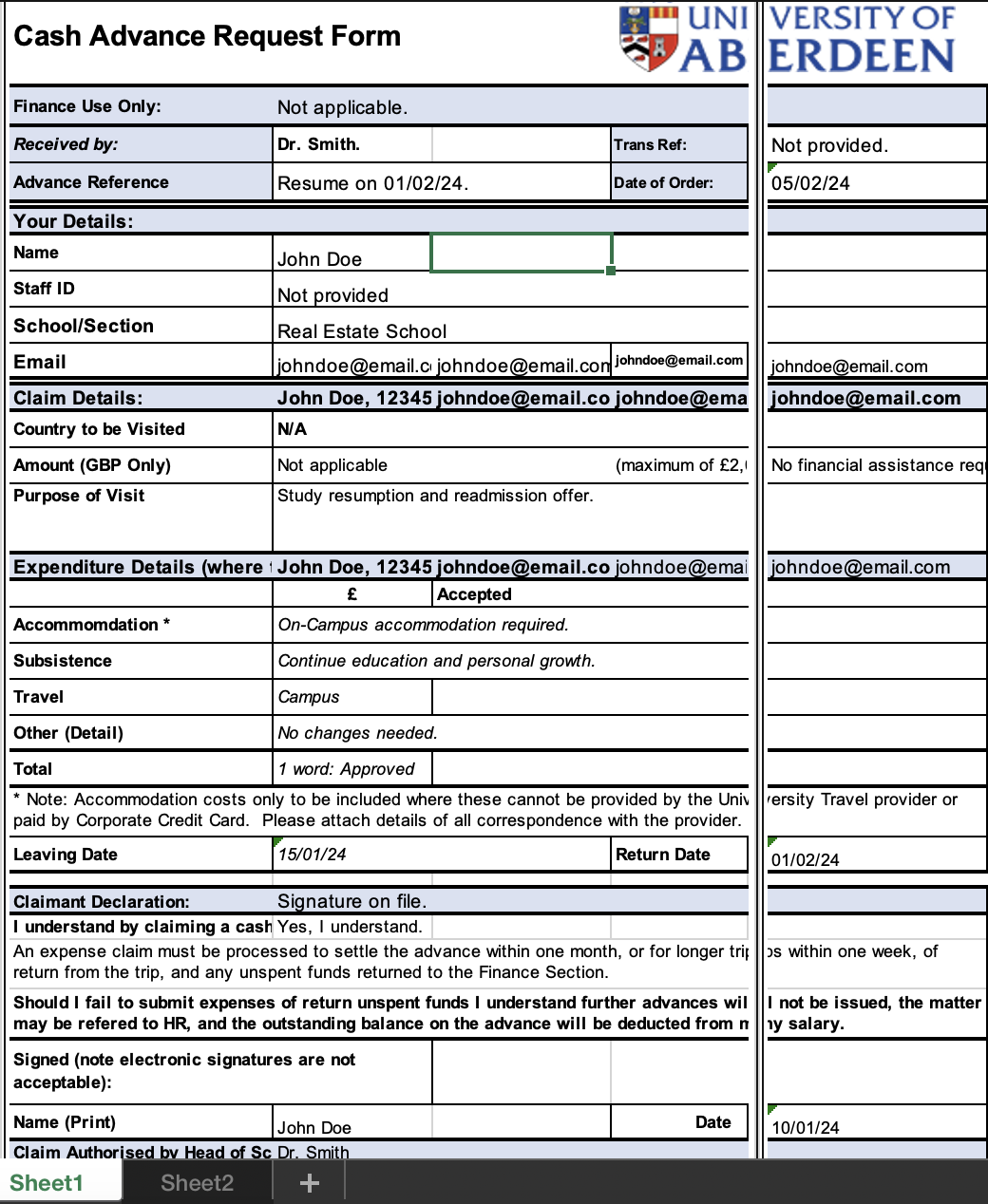
Sentiment analysis was performed on the extracted text to determine if the text was a question. This was done using the python library ‘textblob’ after trying the ‘nltk’ library.

The folders [**FormFiller**](FormFiller) and [**FormFiller App**](FormFiller%20App) both contain the methods tried using **‘flask’** library in creating a web application for this project.   
The files are separated as templates and the main **‘app.py’.**

The folder named **‘**[**FormFiller App SL’**](FormFiller%20App%20SL)has all the files that was used in making the current web application. Instruction to run that can be accessed in **‘**[**README’**](FormFiller%20App%20SL/Read%20Me.docx)

# Appendix:

Here are some examples of the filled form.  
The algorithm would fill better if provided accurate data to it.

* 
* A screen shot of a parking permit application

  Description automatically generated
* A document with text and a logo

  Description automatically generated with medium confidence
* A blue and white project report

  Description automatically generated with medium confidence
* A blue and white project report

  Description automatically generated with medium confidence
* A document with text on it

  Description automatically generated
* A application form with text and a blue and white text

  Description automatically generated

**Thank You**