Python Developer Intern Assignment: Building a Code Interpreter

Objective: Create a code interpreter using Python that takes various file formats (PDF, XLSX, CSV, DOCX) as input, reads the content, generates relevant Python code using an LLM (like GPT-3.5 API), executes the generated code, and returns the result to the user. The solution should be robust and handle a wide range of user actions.

Requirements:

1. **File Reading:**
    - Implement functionality to read content from the following file formats:
        - PDF
        - XLSX
        - CSV
        - DOCX
2. **Code Generation:**
    - Use GPT-3.5 API to generate relevant Python code based on the content of the file and the user's prompt.
3. **Code Execution:**
    - Execute the generated Python code and capture the output.
4. **Response Handling:**
    - Provide the output back to the user, ensuring that the system can handle various user requests.

Steps to Complete the Assignment:

1. **File Reader Module:**
    - Implement separate functions to read content from PDF, XLSX, CSV, and DOCX files.
    - Use libraries like PyPDF2 or pdfplumber for PDF, openpyxl for XLSX, pandas for CSV, and python-docx for DOCX.
2. **Code Writer Module:**
    - Use OpenAI's GPT-3.5 API to generate Python code based on the content read from the files and the user's prompt.
    - Ensure that the prompt sent to GPT-3.5 API is well-structured and informative to get accurate code outputs.
3. **Code Executor Module:**
    - Implement a safe and secure way to execute the generated Python code.
    - Use `exec()` or `eval()` functions with proper security measures to avoid code injection or malicious execution.
4. **Integration with Vanilla LLM:**
    - Implement the logic to pass the generated code to the LLM for execution and then process the output.
5. **Final Answer Module:**
    - Collect the output from the code execution and format it for the user.
    - Ensure the output is clear, concise, and addresses the user's request.

Deliverables:

- **Source Code:** Provide the full source code with comments and documentation.
- **README:** Include a README file with instructions on how to set up and run the project.
- **Test Cases:** Provide test cases for different file types and user prompts to demonstrate the robustness of the solution.

Evaluation Criteria:

1. **Functionality:** The code should successfully read different file formats, generate relevant Python code, execute it, and return the correct output.
2. **Robustness:** The solution should handle a wide range of user inputs and edge cases.
3. **Code Quality:** Code should be clean, well-documented, and follow best practices.
4. **Security:** Proper measures should be taken to ensure the safe execution of generated code.
5. **User Experience:** The final output should be user-friendly and informative.

Tools and Resources:

- Python Libraries: PyPDF2, pdfplumber, openpyxl, pandas, python-docx
- OpenAI API: OpenAI GPT-3.5 Documentation

Example Workflow:

1. **User Input:** User uploads a PDF file and inputs a prompt like "Summarize the content of this PDF."
2. **File Reader:** The PDF content is read and extracted.
3. **Code Writer:** GPT-3.5 API generates Python code to summarize the PDF content.
4. **Code Executor:** The generated Python code is executed to produce a summary.
5. **Final Answer:** The summary is formatted and returned to the user.

Additional Notes:

- Ensure the system is scalable and can be extended to support more file formats or functionalities in the future.
- Consider implementing error handling and logging to assist with debugging and maintenance.

Submission Deadline: 11th June 6pm IST