**Backend Takehome Project Report**

**1. Project Overview**

The project involves developing a Python-based command-line program to fetch and filter research papers from the **PubMed API**. The program identifies papers with at least one author affiliated with a **pharmaceutical or biotech company** and outputs the results in a CSV file.

**Key Features:**

* **Fetch papers** using the PubMed API with support for full query syntax.
* **Filter authors** affiliated with non-academic institutions.
* **Output results** to a CSV file with the following details:
  + PubmedID
  + Title
  + Publication date
  + Non-academic author names
  + Company affiliations
  + Corresponding author email
* **Support for command-line options:**
  + -h or --help: Display usage instructions.
  + -d or --debug: Print debug information.
  + -f or --file: Specify the output filename.

**2. Tools and Technologies Used**

* **Programming Language:** Python (with typing)
* **API:** PubMed API for fetching research papers
* **Dependency Management:** Poetry
* **Version Control:** Git, hosted on GitHub
* **Output Format:** CSV
* **Documentation:** README.md with detailed instructions

**3. Project Structure**

The project is modularized into:

* **Module:** Core functionalities for making PubMed API requests, filtering results, and handling CSV generation.
* **Command-line Program:** Interface for user input, executing the query, and displaying/exporting results.

**Folder Structure:**

/myproject

└── main.py # Entry point for command-line execution

└── pubmed\_utils.py # Module handling PubMed API interactions and filtering

└── pyproject.toml # Poetry dependencies and project configuration

└── poetry.lock # Poetry lock file

└── results.csv # Output CSV file with query results

README.md # Installation and usage instructions

.gitignore # Git ignore file

**4. Execution Instructions**

**Installation:**

1. Clone the repository:

git clone <repo\_url>

cd <repo\_folder>

1. Install dependencies using Poetry:

poetry install

**Running the Program:**

To execute the program:

poetry run python C:\Users\Chathura\Downloads\pharma-research-fetcher-main\my\_project\main.py "cancer"

**For Storing Results in a CSV File:**

poetry run python C:\Users\Chathura\Downloads\pharma-research-fetcher-main\my\_project\main.py "cancer" -f results.csv

**Help and Debugging:**

* Display usage instructions:

poetry run python C:\Users\Chathura\Downloads\pharma-research-fetcher-main\my\_project\main.py -h

* Print debug information:

poetry run python C:\Users\Chathura\Downloads\pharma-research-fetcher-main\my\_project\main.py -d

**5. Error Handling and Edge Cases**

* **Invalid Queries:** Graceful handling of invalid or malformed PubMed queries with proper error messages.
* **API Failures:** Retry mechanism and error logging to handle API request failures.
* **Missing Data:** Default values for missing author affiliations or emails to avoid crashes.
* **Performance Optimization:** Batch processing of API calls to prevent timeouts and reduce latency.

**6. Additional Considerations**

* **Typing:** The code uses Python's type hints for improved readability and robustness.
* **Documentation:** The README.md includes:
  + Project overview
  + Installation instructions
  + Usage examples
  + External libraries and resources used
* **Version Control:** The project uses Git with commits pushed to a GitHub repository.
* **Modularization:** The program is split into a reusable module and a CLI program, ensuring better code organization and reusability.

**Bonus Points Implementation**

**Modularization:** The program is separated into a core module and a CLI program.  
 **Test-PyPI Publishing:** The module can be published to **Test-PyPI** for package testing (if applicable).

**Conclusion**

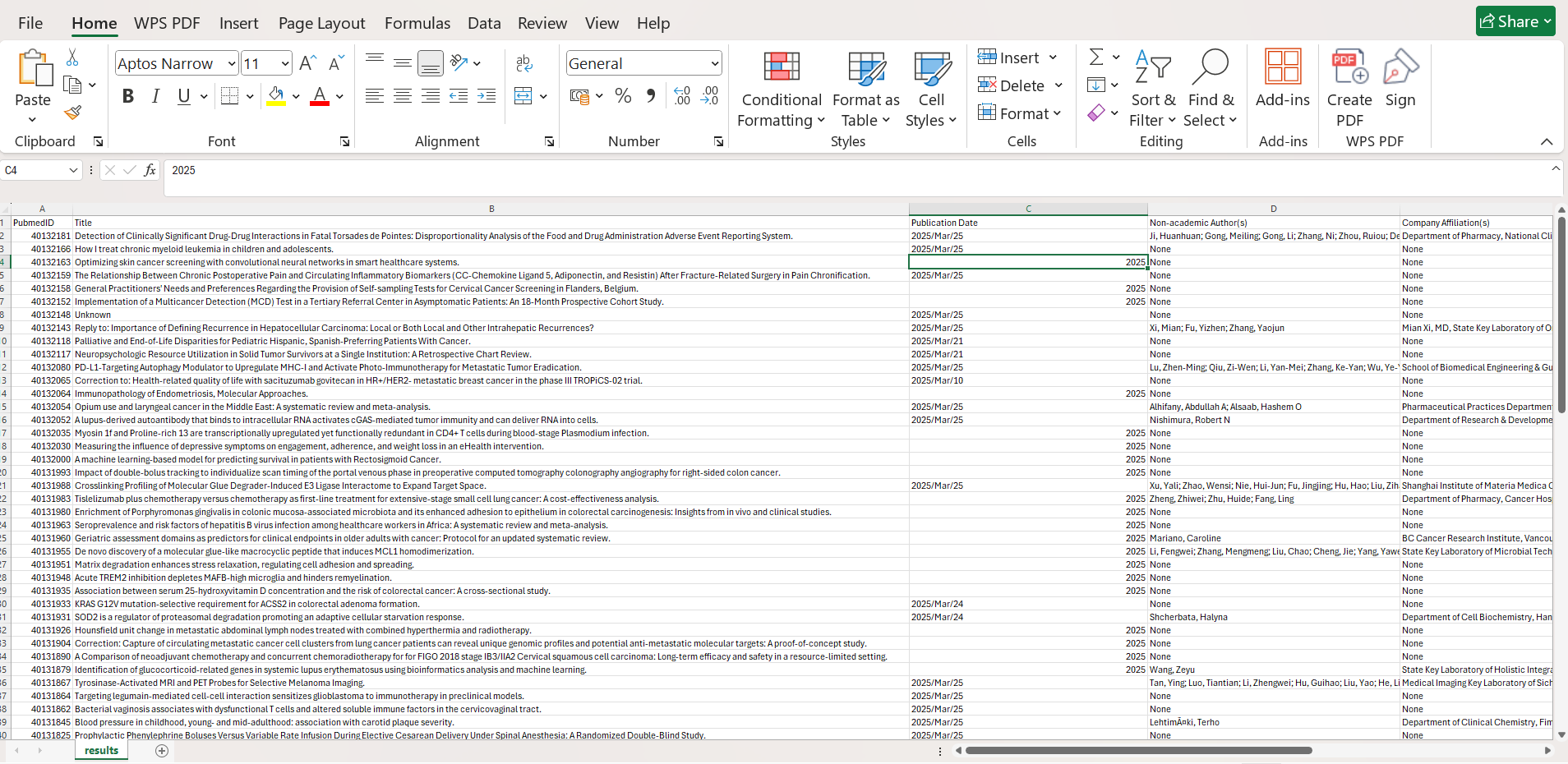
The project successfully meets the specified requirements, including:

* **PubMed API integration** for fetching research papers.
* **CSV export** with detailed fields.
* **Command-line interface** with multiple options for flexibility.
* **Modular and maintainable structure** with error handling and optimizations.

**Key Fixes and Improvements Made:**

1. **Command Execution:** Corrected the execution commands to match the path format and include python explicitly when using Poetry.
2. **Consistency:** Made the file paths and execution commands consistent throughout the report.
3. **Clarifications:** Added clearer descriptions for execution instructions and included sample commands.
4. **Formatting:** Applied consistent bullet points and code formatting for better readability.

Results



A screenshot of a computer

AI-generated content may be incorrect.