**Private & Confidential: WASSERSTOFF INNOVATION & LEARNING LABS PRIVATE LIMITED**

**Position: Ai Engineer Intern DEADLINE: 4 Days(96 hrs)**

**Task Sheet:Internship Task: Domain-Specific PDF Summarization & Keyword Extraction Pipeline**

**Objective** :

Design and implement a dynamic pipeline that processes multiple PDF documents from a desktop folder, generates domain-specific summaries and keywords, and stores them in a MongoDB database. The system must efficiently handle documents of varying lengths, from short to long, and update the database with summary and keyword data after each document is processed.

Your solution should prioritize concurrency, speed, and innovation. The use of pre-built frameworks will result in up to a 25 point deduction,so applicants are encouraged to create custom solutions wherever possible.

You have been provided with a dataset in the github link provided at the end of the document to use for the task.Please use that for the task. At the end you need to submit an exported MongoDB Collection along with the code in a Github repository. **Do not use any pre-built pipeline, as that will not be considered or judged at all.**

Github for Dataset

Github link: “https://github.com/Devian158/AI-Internship-Task.git”

**Task Details**

**1. PDF Ingestion & Parsing**

Requirement : The pipeline should be able to process multiple PDFs from a folder on the desktop, i.e. if the path of a folder is provided in the code, it should ingest all the documents in that folder. It must handle documents of varying lengths:

- Short PDFs (1-10 pages)

- Medium PDFs (10-30 pages)

- Long PDFs (30+ pages)

Concurrency: Ensure the system can process multiple documents in parallel, managing system resources efficiently. The pipeline must handle large files and high volumes without crashing.

**2. MongoDB Dataset Storage & JSON Updates**

Initial Storage: When each PDF is ingested, its metadata (document name, path, size, etc.) must be stored in a MongoDB collection.

Post-Processing Update: After summarization and keyword extraction, the MongoDB entry for each document must be updated with the JSON output, including the generated summary and extracted keywords.

**3. Summarization & Keyword Extraction**

Summarization: Dynamically generate summaries that are relevant to the domain you have chosen. The summary length and detail should correspond to the document length (e.g., a detailed summary for long documents, concise summaries for short ones).

Keyword Extraction : Extract non-generic , domain-specific keywords that reflect key ideas or themes of the document. Avoid common or irrelevant keywords.

**4. JSON Structure & MongoDB Updates**

JSON Format : Summaries and keywords must be formatted in JSON, which will then be stored in the MongoDB document. Make sure to handle updates efficiently after processing each document.

Error Handling : Log any errors (e.g., for corrupted PDFs or unsupported formats) and ensure that MongoDB records are not affected by such issues.

**5. Concurrency & Performance**

Concurrency : The pipeline should be designed to handle multiple documents simultaneously, leveraging parallel processing to improve speed.

Performance : Provide data on how well the system scales, especially in terms of how quickly it processes large and multiple PDFs concurrently.

**Additional Requirements**

Document Variety : The pipeline must handle PDFs of varying lengths. Provide examples of short (1-2 pages), medium (10-12 pages), and long (30+ pages) documents in your testing. Framework Restrictions : The use of pre-built libraries or frameworks (such as Langchain or others) is permitted, but doing so will result in a deduction of up to 15 points . Custom solutions will receive higher marks.

Testing & Performance : You are expected to provide test results, especially regarding performance metrics such as concurrency, memory usage, and processing speed.

Error Handling : Handle all edge cases, such as corrupted or encrypted PDFs, without interrupting the pipeline. All errors should be logged and MongoDB records updated accordingly.

**Marking Scheme**

**Total Marks: 100**

**1. PDF Ingestion & Parsing (20 marks)**

- Concurrency (10 marks): Efficiently processing multiple PDFs in parallel.

- Handling Document Variety (5 marks): Adapting the pipeline for short, medium, and long documents.

- Error Handling & Memory Management (5 marks): Proper handling of corrupted files, memory management, and resource optimization.

**2. Summarization & Keyword Extraction (40 marks)**

- Summarization Quality (15 marks): Summaries should be clear, concise, and relevant, varying based on document length.

- Keyword Quality (15 marks): Keywords should be non-generic, specific to the domain, and accurately reflect document content.

- Handling Document Lengths (10 marks): Properly adjusting the process for different document lengths and types.

**3. JSON Structure & MongoDB Update (15 marks)**

- Initial Storage (5 marks): Properly store document metadata in MongoDB.

- Post-Processing Updates (5 marks): Efficiently update MongoDB with the JSON summaries and keywords after each document is processed.

- Error Handling (5 marks): Ensure that errors like corrupted files do not break the pipeline, with appropriate logging and recovery mechanisms.

**4. Concurrency & Performance (15 marks)**

- Concurrency (10 marks): Efficient multi-document parallel processing.

- Performance Metrics (5 marks): Provide performance data, such as time taken per document and concurrency speed.

**5. Documentation & Innovation (10 marks)**

- Documentation (5 marks): A clear `README.md` with detailed setup instructions and usage guide. Extra points for Docker configuration.

- Innovation (5 marks): Creative and innovative approaches to document parsing, keyword extraction, or summarization will be rewarded. Over-reliance on pre-built frameworks will result in a deduction of up to 15 marks.

**Submission Guidelines**

* Your time starts at receipt of the email.
* Complete and Submit your project as described in the PDF attached to this email
* Push all changes to your repository while keeping it public on GitHub and name the repository as follows: rishi-pawar/wasserstoff/AiInternTask
* Host your project publicly and provide a link to a Vercel or another publicly hosted version of your project.
* Please make sure that you understand the problem statement completely.
* Please make sure that your code is well commented.
* Any kind of plagiarism would result in immediate disqualification of your application.
* It is necessary for you to attempt the task and take it to completion with all mentioned functionality to qualify for the next round.
* Codebase : Submit a well-commented and structured codebase.
* Documentation : Provide a `README.md` file that includes setup instructions, system requirements, and an explanation of your solution.
* Testing : Include unit tests and performance benchmarks to validate the efficiency and accuracy of your system.
* Optional Docker Setup : If possible, include a Docker configuration for simplified deployment. Bonus points will be awarded for this.
* Performance Reports : Attach metrics showing concurrency, speed, and resource utilization.

**Github Folder Structure Guidelines:**

IPYNB/Jupyter Notebook submissions will not be accepted, the task should follow a proper clean code architecture in Python or Javascript ( Whatever your preferred language is)

Code that is clean with proper formatting and annotations will be ranked higher if the proper syntax is followed.

**Email Submission:**

Send an email to hr@thewasserstoff.com with the subject "Ai Internship Task Submission" and include the link to your GitHub repository in the email.

**Video Explanation:**

Please create a short video explaining your approach to the problem and your solution strategy. This video is crucial as it will give us insights into your thought process and problem-solving abilities.

**Important Notes**

- Pre-built Libraries : Use of pre-built libraries (e.g., Langchain) is allowed but will result in a deduction of up to 25 marks . Custom-built solutions will be rewarded more highly.

- Keyword Extraction : Ensure keywords are domain-specific and relevant. Generic or overly broad keywords will result in a deduction.

- Document Variety : Ensure that your pipeline dynamically adjusts for short, medium, and long documents, adapting the summarization and keyword extraction accordingly.

**Github**

Github link: “https://github.com/Devian158/AI-Internship-Task.git”

**Contact**

For any doubts, you can contact me at [divyansh.sharma@thewasserstoff.com](mailto:divyansh.sharma@thewasserstoff.com) and hr@thewasserstoff.com with a mail including your query and details.