SOFTWARE REQUIREMENTS SPECIFICATION

for

DocLayoutAI

Version 1.0

Prepared by:

1. Archit Jaju (IMT2023128)
2. Sanyam Verma (IMT2023040)
3. Pushkar Kulkarni (IMT2023087)
4. Harjot Singh (IMT2023064)

November 12, 2025

Contents

1. Introduction

1.1. Purpose

The purpose of this Software Requirements Specification (SRS) is to define the functional and non-functional requirements for *DocLayoutAI* — a command-line system developed for the Adobe India Hackathon 2024, Challenge 1B: Persona-Driven Document Intelligence.

The system aims to automate the process of identifying and ranking the most relevant sections and subsections from a collection of PDF documents, based on a given user persona and job-to-be-done (JTBD). By parsing both the persona and the documents offline, the system determines contextual relevance and produces structured insights in JSON format.

The SRS provides a clear and detailed reference for developers, testers, and evaluators to understand what the system does, how it behaves, and its operating constraints.

1.2. Intended Audience and Reading Suggestions

This document is intended for the following stakeholders:

- **Developers:** To understand the required functionalities, input/output formats, and system architecture for implementation.
- **Project Evaluators / Hackathon Judges:** To comprehend the system's design choices, scope, and compliance with the challenge requirements.
- **Testers:** To identify measurable and verifiable requirements for system validation.
- **Future Contributors:** To extend or integrate the existing backend into graphical interfaces or larger document processing pipelines.

For ease of reading:

- Chapter 1 (Introduction) provides overall context, motivation, and objectives.
- Chapter 2 (Overall Description) discusses the system perspective, product functions, and user characteristics.
- Chapter 3 (External Interface Requirements) details interfaces and inputs/outputs.
- Chapter 4 (System Features) lists all functional and non-functional requirements.
- Appendices include sample outputs and the glossary.

1.3. Project Scope

DocLayoutAI provides a lightweight, offline analytical solution that can process multiple PDF documents along with a persona definition to automatically surface content relevant to a specific user's information needs.

The system performs PDF text extraction, natural language processing (NLP)-based keyword analysis, and relevance ranking of document sections and subsections. The ranked outputs are

compiled into a structured JSON file containing metadata (persona, input files, timestamp) and a prioritized list of sections/subsections.
The primary objectives of *DocLayoutAI* are:

- To minimize manual document reading and filtering by knowledge workers.
- To provide interpretable, structured insights in JSON form for downstream analysis or visualization.
- To function fully offline, respecting CPU-only and model-size constraints of the hackathon.
- To support modular expansion — e.g., integration with GUI or cloud deployment in future iterations.

The scope of this version is limited to:

- Running as a command-line interface (CLI) tool.
- Processing local PDF files and persona descriptions (text or JSON).
- Generating a single output JSON file summarizing relevant document sections.

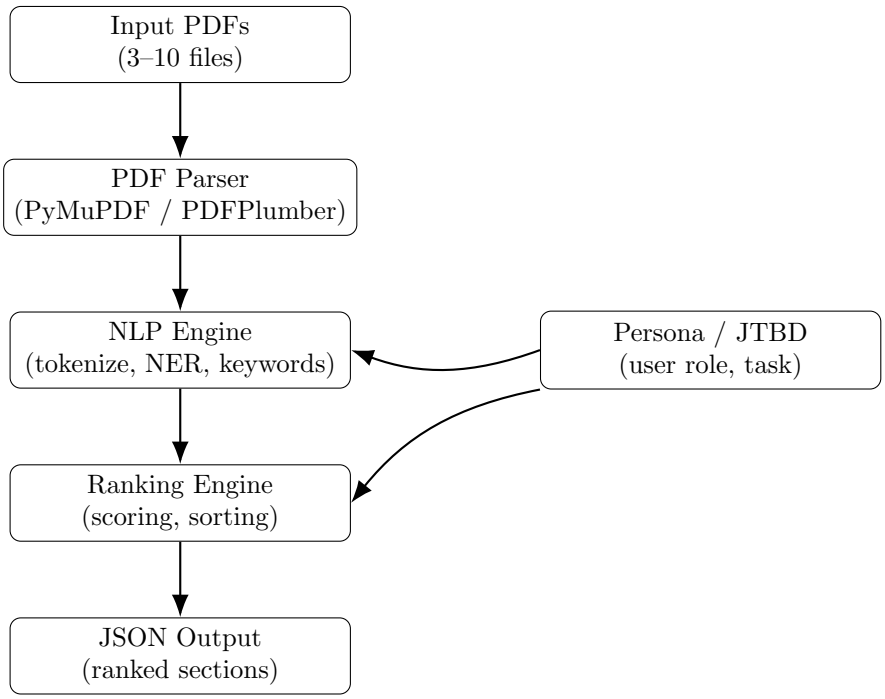


Figure 1.1.: Vertical conceptual workflow of DocLayoutAI.

Figure 1.1 illustrates the high-level workflow of the system: input PDFs and persona/job-to-be-done information are analyzed by the NLP-based engine, which outputs a structured JSON file containing the most relevant document sections.
Overall, *DocLayoutAI* enhances document understanding through persona-driven analysis, enabling efficient and targeted information retrieval for offline use cases.

2. Overall Description

2.1. Product Perspective

DocLayoutAI is a stand-alone, offline analysis system designed for persona-driven document understanding. The tool ingests a collection of PDF documents and a user persona definition (including the job-to-be-done) and outputs a structured JSON file containing the most relevant document sections and subsections.

This project is an independent command-line application — it does not depend on any external web services or databases. Instead, it operates locally within a Dockerized environment using CPU-based natural language processing (NLP) techniques. Its modular design enables clear separation between input processing, NLP analysis, relevance ranking, and JSON output generation.

The main goal of this system is to reduce the manual effort involved in reading and filtering large volumes of text, allowing users to focus on the most relevant information sections automatically identified by the tool.

2.2. User Classes and Characteristics

The system is primarily designed for a single user type — the *Consumer*. However, within that category, different usage contexts may exist:

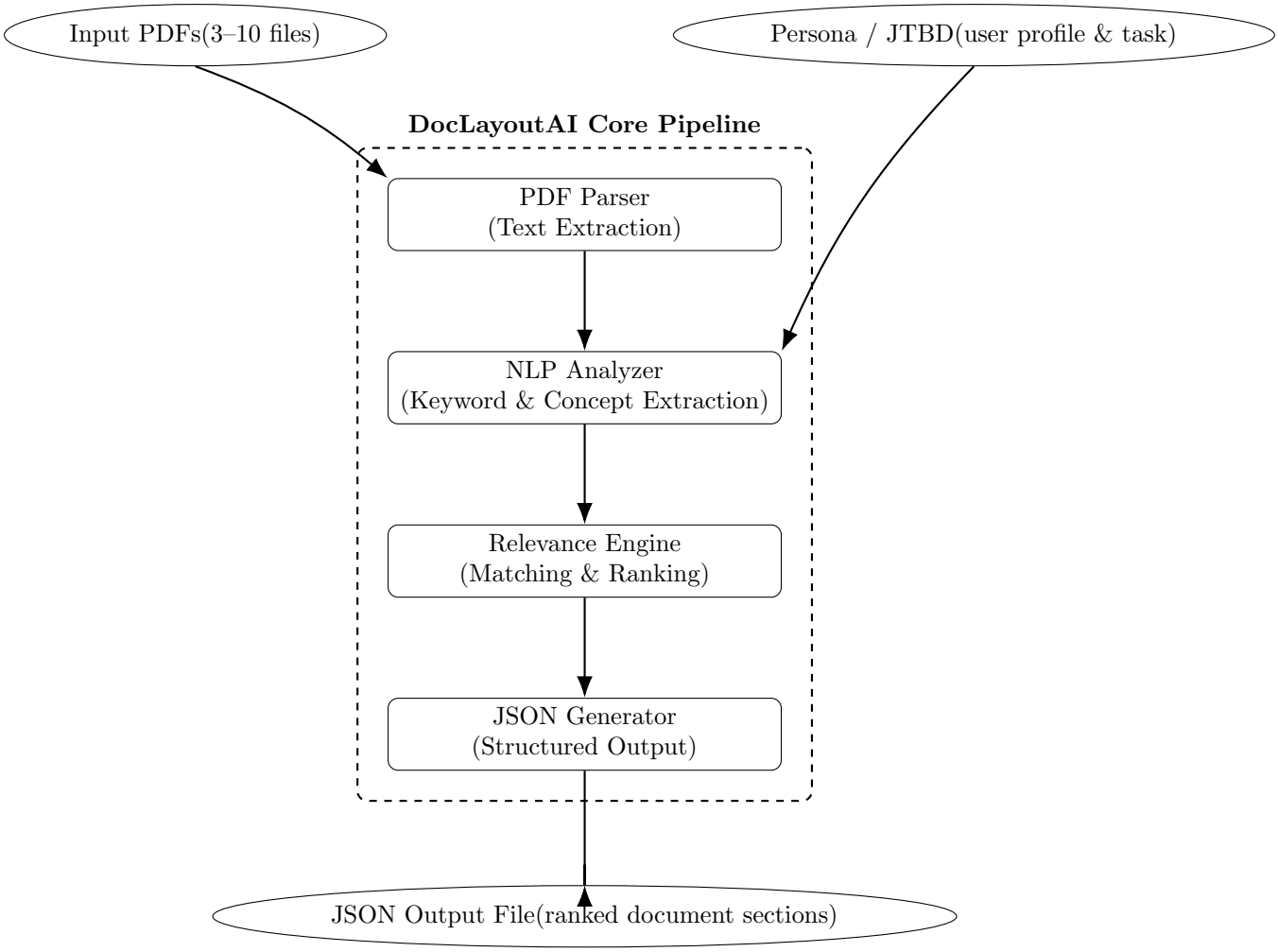
- **Data Analyst / Researcher:** Uses the system to extract key insights from research papers, whitepapers, or technical documents relevant to a given persona or topic.
- **Business Analyst:** Applies the tool to corporate reports or business documents to extract relevant data aligned with strategic roles or objectives.
- **Student / Academic User:** Employs the tool to summarize or identify sections of academic PDFs relevant to a study persona (e.g., "graduate student studying deep learning").
- **Developer / Maintainer:** Extends or integrates the backend into a user interface or web service in future versions.

All users are expected to have basic command-line proficiency and understand how to provide text files and directories as input. No programming background is required for normal use, as all operations are performed via clearly defined CLI options.

2.3. Product Functions

The major functional components of *DocLayoutAI* can be summarized as follows:

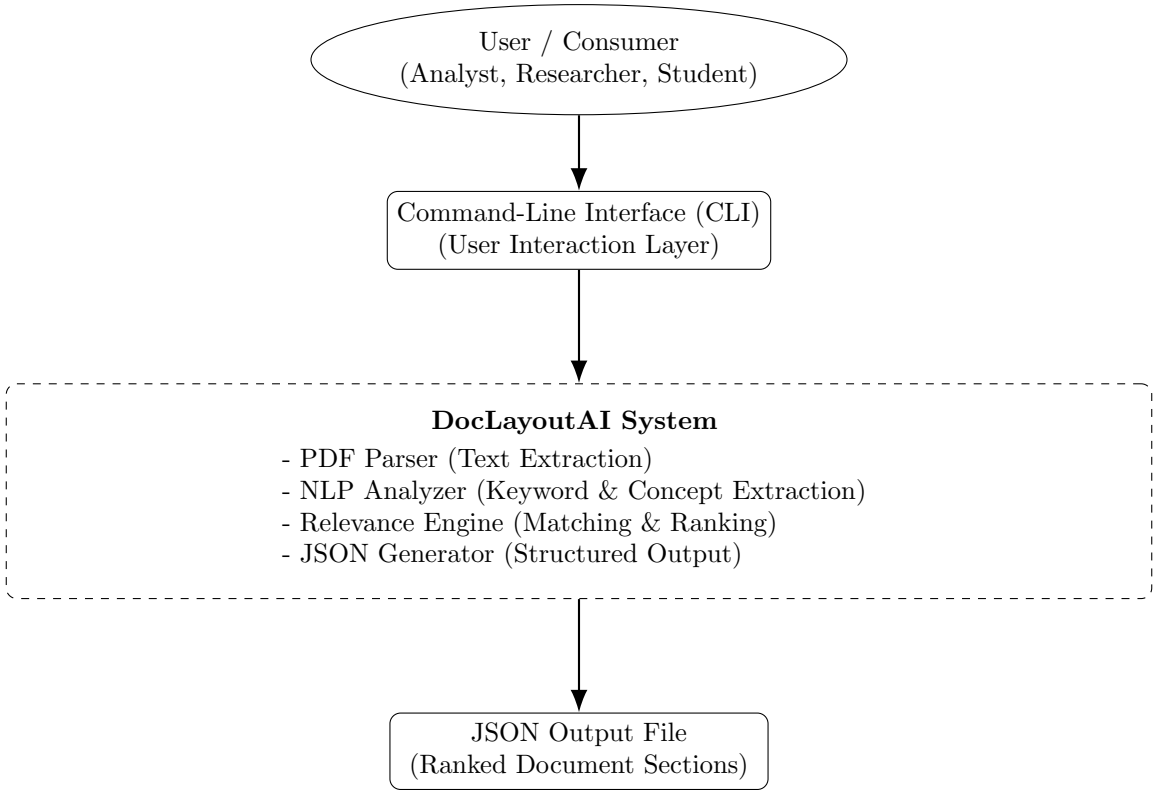
- **Input Handling:** Accepts a directory containing PDF documents and a persona/job description file (in text or JSON).
- **PDF Parsing:** Extracts raw text from each input PDF for further processing.
- **Persona Analysis:** Processes the persona and job-to-be-done to identify key entities, terms, and concepts using NLP.



High-level vertical architecture of *DocLayoutAI*:
Offline, CPU-only modular pipeline with CLI and Docker interfaces.

Figure 2.1.: System architecture overview of DocLayoutAI (vertical layout).

- **Keyword Extraction:** Uses text-processing techniques to extract key terms and topics relevant to the persona.
- **Relevance Matching:** Compares extracted persona keywords against document text to detect semantically aligned sections.
- **Section and Subsection Identification:** Detects structured document headings and associates them with matched content.
- **Ranking Engine:** Ranks sections/subsections based on term frequency, semantic similarity, and contextual relevance.
- **JSON Output Generation:** Compiles all results into a structured JSON format containing metadata (persona, job, documents, timestamp) and ranked outputs.
- **Error Handling and Logging:** Provides informative console logs and clear error messages for invalid inputs or missing files.



User roles and interaction overview in *DocLayoutAI*.

Figure 2.2.: User classes and interaction overview.

2.4. Operating Environment

DocLayoutAI is designed to run in a lightweight, portable environment that adheres to the hackathon constraints. The operating assumptions are:

- Executed on a local machine (Linux recommended, but compatible with macOS and Windows through Docker).
- Runs entirely offline — no network or API calls.
- CPU-only execution; no GPU acceleration required.
- Requires Python 3.x environment with included dependencies (e.g., PyMuPDF, spaCy, or NLTK as applicable).
- Distributed as a Docker image or standalone Python package for reproducibility.

2.5. Design and Implementation Constraints

Several constraints were defined as part of the Adobe Challenge 1B problem statement:

- **Offline Operation:** The system cannot access the internet or rely on cloud-based APIs.
- **Hardware Limitation:** Only CPU resources may be used; GPU or external accelerators are disallowed.

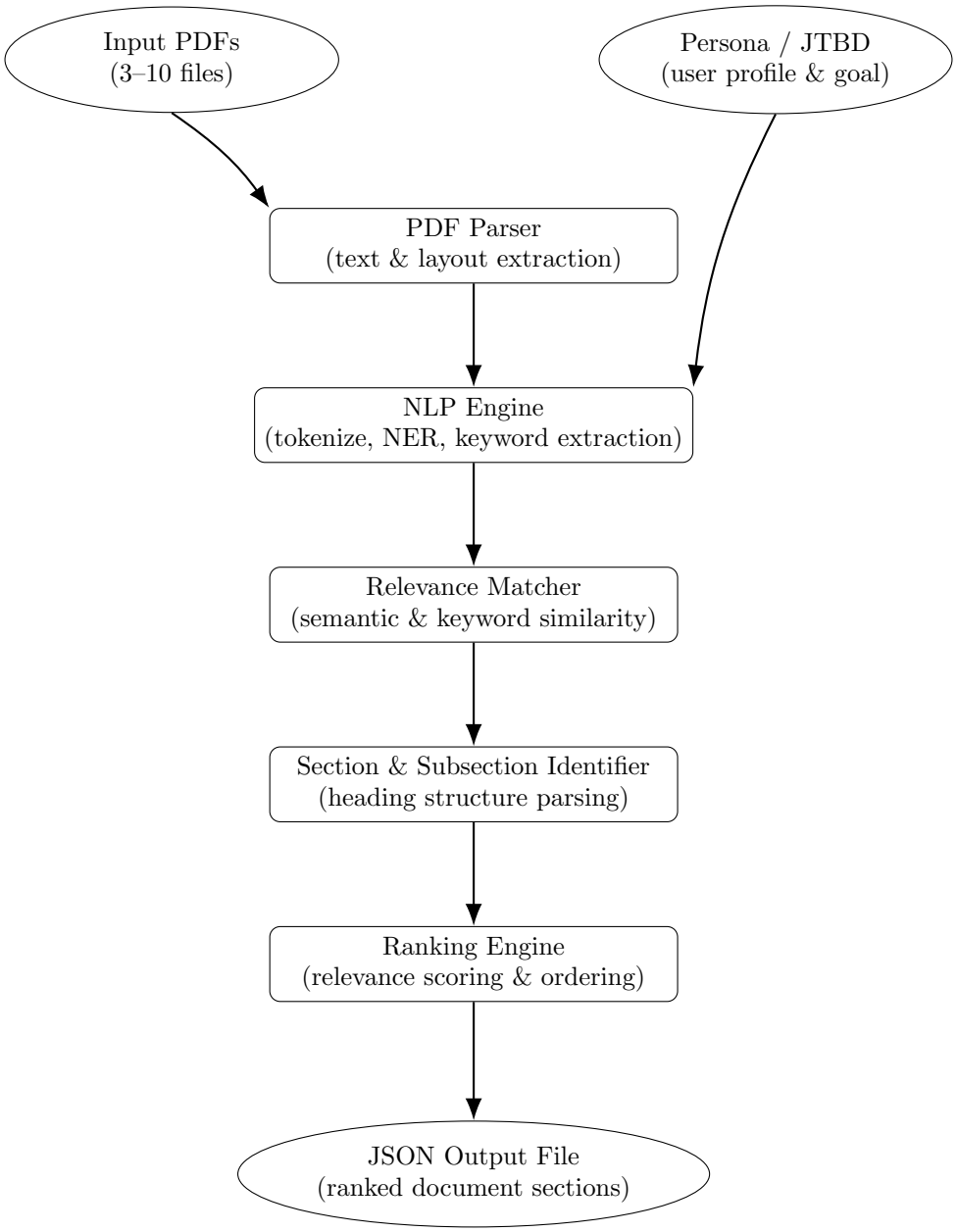
- **Model Size Limit:** All language models used must collectively fit within a 1 GB storage limit.
- **Time Limit:** The system should process 3–5 PDF documents within 60 seconds.
- **Interface Limitation:** The system must be operated entirely through a command-line interface (CLI).

2.6. Design Overview

DocLayoutAI follows a modular design pattern with four core components:

1. **Input Processor:** Validates and reads input persona files and PDF collections.
2. **NLP Engine:** Performs entity extraction and contextual keyword analysis.
3. **Relevance Scoring Module:** Computes similarity and ranking scores across all documents.
4. **Output Formatter:** Structures results and metadata into a standardized JSON schema.

Each component communicates through well-defined intermediate data structures to ensure maintainability and ease of future extension (e.g., web UI or semantic visualization dashboard). In summary, the system emphasizes modularity, clarity, and offline efficiency — making it both hackathon-compliant and extendable for future versions such as interactive interfaces or cloud deployment.



Vertical conceptual data flow of *DocLayoutAI*:
From input PDFs and persona through NLP analysis, matching, and ranking
to final structured JSON output.

Figure 2.3.: Conceptual data flow of DocLayoutAI (vertical layout).

3. System Features

DocLayoutAI is a persona-driven document intelligence system that automates the extraction, ranking, and organization of relevant information from a collection of PDF documents. The system's primary purpose is to analyze multiple documents offline, match them against a given persona and job-to-be-done, and produce a structured JSON summary of the most contextually relevant sections.

3.1. Description and Priority

The features of *DocLayoutAI* are organized by priority, based on their impact on core system functionality and user experience. Each feature directly contributes to the end-to-end analysis pipeline, from data ingestion to JSON generation.

1. **Relevance Extraction and Ranking (Highest Priority)**
   The central functionality of the system. It extracts, analyzes, and ranks document sections/subsections that are most relevant to the provided persona and job-to-be-done description. This feature involves NLP-based keyword extraction, contextual similarity computation, and importance-based ranking.

2. **Persona Parsing and Keyword Generation**
   Processes the input persona description to identify entities, goals, and keywords representing the user's intent. This forms the foundation for all downstream matching and ranking processes.

3. **PDF Text Extraction**
   Reads and extracts machine-readable text from multiple PDF files (using libraries like PyMuPDF or PDFPlumber). Ensures consistent, accurate text parsing even across diverse document layouts.

4. **Section and Subsection Identification**
   Detects structured headings within documents and associates matched text segments to their respective section hierarchy, maintaining logical context for ranking.

5. **JSON Output Generation**
   Compiles all analyzed and ranked information into a well-defined JSON file. The JSON includes metadata (input file names, persona, timestamp) and an ordered list of relevant sections/subsections.

6. **Command-Line Interface (CLI)**
   Enables users to run the analysis using terminal commands with arguments for input and output paths. For example: `doclayoutai -i ./input_pdfs -p persona.txt -o results.json`

7. **Error Handling and Logging**
   Provides clear error messages for invalid file paths, corrupted PDFs, or missing persona inputs. Logs processing steps for debugging and reproducibility.

All features are mandatory to achieve full system functionality. Future versions may include optional features such as graphical visualization or API-based integration.

3.2. Functional Requirements

3.2.1. Overview

DocLayoutAI is implemented as a command-line application with modular components, written primarily in Python. It adheres to hackathon constraints of offline execution and CPU-only computation. The main functional requirements describe what the system must do to fulfill its intended purpose.

3.2.2. List of Functional Requirements

- **FR1: Input Acquisition**
  The system shall accept a directory of PDF files and a persona/job description file as input parameters.
- **FR2: Input Validation**
  The system shall verify that all specified input files exist, are in valid formats, and contain readable content.
- **FR3: PDF Parsing**
  The system shall extract plain text from each PDF for NLP processing.
- **FR4: Persona Analysis**
  The system shall analyze the persona/job description text to extract relevant keywords and entities using NLP techniques.
- **FR5: Relevance Matching**
  The system shall match extracted persona keywords with PDF text content to identify related sections.
- **FR6: Section Identification**
  The system shall detect headings, subheadings, and section boundaries within each document.
- **FR7: Ranking Engine**
  The system shall assign a numerical relevance score to each identified section/subsection, ordering them from highest to lowest relevance.
- **FR8: JSON Output Generation**
  The system shall compile metadata (persona, job, document list, timestamp) and ranked results into a single JSON file that follows the defined schema.
- **FR9: Command-Line Operation**
  The system shall provide CLI options for input and output paths:
  - `-i, --input` : path to folder containing PDF files.
  - `-p, --persona` : path to persona/job description file.
  - `-o, --output` : path for the output JSON file.
- **FR10: Logging and Reporting**
  The system shall log major processing steps to the console for debugging and performance tracking.
- **FR11: Error Handling**
  The system shall handle file errors gracefully and exit with informative error messages without system crash.

3.2.3. Implementation Technologies

DocLayoutAI is implemented using open-source technologies optimized for text processing and offline analysis:

- **Programming Language:** Python 3.x
- **Primary Libraries:** PyMuPDF (PDF parsing), spaCy / NLTK (NLP), JSON (output formatting)
- **Runtime Environment:** Docker container for isolated execution
- **Deployment:** Cross-platform CLI tool compatible with Linux, macOS, and Windows

3.2.4. Dependencies and Integration

DocLayoutAI has minimal external dependencies:

- Local file system access for reading and writing PDFs and JSON.
- Pre-trained NLP model files ($\leq$ 1 GB, stored locally within the Docker container).
- Python standard libraries and open-source NLP packages.

The system does not depend on any network or web APIs. Future versions may include integration points with web-based dashboards or semantic search interfaces.

3.2.5. Feature Summary

Table 3.1 summarizes the main system features and their implementation priority.

Feature	**Description**	**Priority**
Relevance Extraction	Identifies and ranks relevant PDF sections using NLP	High
Persona Parsing	Extracts keywords and context from persona/job input	High
PDF Text Extraction	Converts PDF text to plain text for analysis	High
Section Identification	Detects document headings and structure	Medium
JSON Output	Outputs structured, ranked data with metadata	High
CLI Interface	Command-line access for input/output control	High
Logging and Error Handling	Logs progress and prevents crash on invalid inputs	Medium

Table 3.1.: Summary of DocLayoutAI features and priorities.

Collectively, these features form the core of *DocLayoutAI*, fulfilling its mission to provide offline, persona-driven document analysis and content ranking.

4. Other Nonfunctional Requirements

4.1. Performance Requirements

DocLayoutAI is designed to operate efficiently within the strict resource and time constraints outlined by the hackathon challenge. Since it runs entirely offline and on CPU-only hardware, performance optimization focuses on text parsing, NLP processing, and relevance ranking.

The following performance criteria apply:

- The system shall process 3–5 medium-sized PDF documents (each up to 10 MB) within **60 seconds**.
- All processing — including PDF extraction, persona analysis, and ranking — shall be completed in a single execution cycle without user interaction.
- Memory utilization shall remain below **2 GB RAM** during execution to support compatibility with general-purpose CPUs.
- The total NLP model size shall not exceed **1 GB**, ensuring fast load times and portability.
- The output JSON file shall be generated instantly after ranking, without significant delay.

These constraints ensure the system remains usable even on limited computing environments and maintains predictable runtime behavior under the given workload.

4.2. Security Requirements

Although *DocLayoutAI* is an offline system with no network dependencies, it follows basic data integrity and security practices to ensure safe local execution.

- The system shall not transmit, upload, or log any user data externally.
- Input and output files are accessed strictly from local directories provided by the user.
- File access permissions shall be limited to the executing user environment.
- Sensitive user information (if any) contained in the persona file shall not be stored or cached after execution.
- Logs shall not include document contents or personal data, only filenames and processing summaries.

Since no authentication or online components exist, attack surfaces are minimized by design. This aligns with the hackathon's "offline-only" rule and ensures data privacy throughout execution.

4.3. Software Quality Attributes

The following quality attributes define key expectations for system reliability, usability, and maintainability:

- **Reliability:** The system shall execute deterministically — given the same input documents and persona, it must always produce the same output JSON. Error handling ensures that invalid or unreadable inputs do not cause crashes, and descriptive error messages are displayed on the console.
- **Usability:** The command-line interface shall use intuitive argument names (e.g., `-i`, `-p`, `-o`) and provide clear help text when invoked incorrectly. Documentation and example commands will be included for new users.
- **Maintainability:** The codebase shall follow a modular design, dividing functions into input handling, NLP processing, ranking, and output generation. Each module can be extended independently (for example, to plug in new ranking algorithms or file parsers).
- **Portability:** The system shall be containerized via Docker to ensure consistent behavior across Linux, macOS, and Windows environments.
- **Testability:** The output JSON format and ranking logic are verifiable using test cases on sample personas and document sets. Unit tests for PDF parsing, keyword extraction, and ranking functions shall be included in the codebase.

These attributes collectively ensure that the tool remains stable, usable, and maintainable even beyond the hackathon scope.

4.4. Business Rules

DocLayoutAI was developed in compliance with the rules of the **Adobe India Hackathon 2024 (Challenge 1B: Persona-Driven Document Intelligence)**. The system must function completely offline and be reproducible on any standard computing device.

Key business rules:

- The tool must strictly adhere to challenge constraints — no internet usage, no external APIs, and CPU-only execution.
- The system's output (structured JSON) must meet the competition's required schema format for persona-driven document extraction.
- Future commercial or academic versions may expand beyond the hackathon rules (e.g., by adding APIs or GUIs), but version 1.0 shall remain fully compliant.
- Intellectual property rights and licensing comply with Adobe Hackathon terms — the code is open for educational and research purposes as per submission rules.

These rules define the operational and ethical framework under which the system was designed, ensuring fairness, reproducibility, and compliance with event guidelines.

5. Other Requirements

DocLayoutAI is designed as a modular and extensible offline analysis system. As natural language processing techniques and document understanding models continue to evolve, the system will require periodic maintenance and updates to remain relevant and effective.

5.1. Maintainability

Given that *DocLayoutAI* operates as a research-driven and evolving tool, ongoing maintenance is essential to ensure compatibility with future NLP libraries and document formats. The following guidelines define the maintainability requirements:

- The system codebase shall be modular — divided into clear components such as Input Handling, NLP Analysis, Ranking, and JSON Output — to simplify future modifications or replacements.
- Each core module shall include inline documentation and descriptive comments following Python's PEP 8 and docstring standards.
- Dependencies (e.g., PyMuPDF, spaCy, NLTK) shall be version-controlled using a `requirements.txt` file or equivalent dependency management configuration.
- A changelog or version history shall be maintained for each major update.
- Any refactoring shall ensure backward compatibility with existing input/output formats unless explicitly stated otherwise.

5.2. Scalability and Extensibility

Although version 1.0 of *DocLayoutAI* is built for offline, single-machine use, the design allows for future scalability and feature expansion:

- The current modular design supports easy integration with graphical user interfaces (GUI) or web-based dashboards.
- The NLP engine can be upgraded to use larger or more advanced language models, provided that local resource constraints are respected.
- The ranking mechanism can be extended to include semantic similarity scoring, vector embeddings, or transformer-based context models in future releases.
- Multi-document aggregation and comparison features can be introduced for broader document analysis use cases (e.g., enterprise search or research curation).

5.3. Adaptability

The field of document intelligence and persona-based retrieval is continuously evolving. Therefore:

- The system shall support reconfiguration or replacement of NLP models without requiring structural changes in other modules.
- The JSON output schema shall remain stable but flexible enough to accommodate additional metadata fields in later versions (e.g., confidence scores, source ranking explanations).
- The CLI-based interface may later be integrated into cloud platforms or enterprise pipelines for batch document processing.

5.4. Future Maintenance and Roadmap

DocLayoutAI may require refactoring and optimization as new technologies and user demands emerge. Potential roadmap items include:

- Migration from rule-based to fully semantic or embedding-based ranking.
- Addition of an interactive visualization layer for results (GUI or web-based).
- Integration with LLM-powered summarization modules for refined section extraction.
- Expansion to handle multilingual PDF documents.
- Deployment on lightweight server environments or research clusters.

These enhancements will ensure that the system continues to evolve as document processing and NLP technologies advance, while remaining faithful to its original hackathon objective — enabling persona-driven, offline, and efficient document understanding.

6. Use Cases

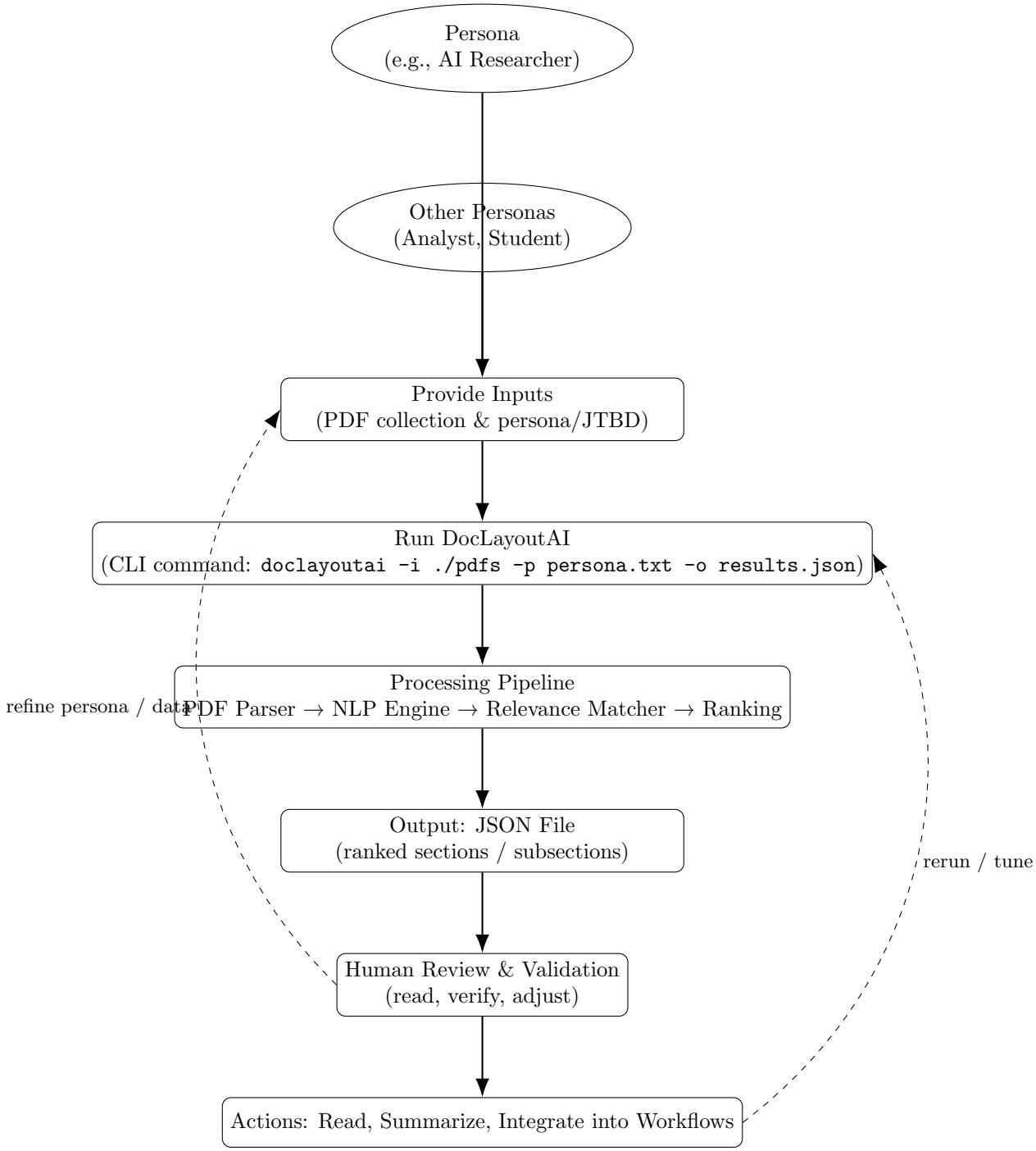
This section provides example use cases that illustrate how *DocLayoutAI* operates for different personas and domains. Each use case highlights how the system extracts and ranks relevant document content based on the persona's job-to-be-done.

- **Academic Research:**
  A researcher persona specializing in Artificial Intelligence provides a collection of PDF research papers with the job description: *"Summarize recent results in reinforcement learning."* The system processes the PDFs, identifies sections such as *"Methods"* and *"Conclusion"* containing reinforcement learning discussions, and ranks them by contextual relevance.

- **Business Analysis:**
  A business analyst persona submits corporate annual reports with the job: *"Identify mentions of sustainability initiatives."* The system scans the documents, detects relevant terms (e.g., *"sustainability," "carbon footprint," "green energy"*), and outputs a ranked list of document sections mentioning these topics.

- **Educational Study:**
  A student persona inputs PDF chapters from a chemistry textbook with the job: *"Review enzyme kinetics."* The system analyzes the PDFs, extracts and ranks sections related to enzyme mechanisms, rate laws, and kinetic models, providing a concise, structured summary.

These use cases demonstrate the persona-driven nature of the system — where the combination of persona, intent, and document corpus guides the extraction and ranking of relevant information.



Persona-driven use-case flow of *DocLayoutAI*:
users supply PDFs and persona, run the system, validate output, and refine iteratively.

Figure 6.1.: Persona-driven use case flow diagram (vertical layout).

A. Sample JSON Output

The following example illustrates the structure and format of the JSON file produced by *DocLayoutAI*. This output reflects the required hackathon format — containing metadata, ranked document sections, and subsections. All values shown are illustrative.

Listing A.1: Example of DocLayoutAI output JSON

```
{
  "documents": ["research_paper1.pdf", "research_paper2.pdf"],
  "persona": "AI Researcher",
  "job": "Summarize neural network advances in 2024",
  "timestamp": "2024-11-12T12:00:00Z",
  "sections": [
    {
      "document": "research_paper1.pdf",
      "page": 3,
      "title": "Deep Learning Techniques",
      "rank": 1
    },
    {
      "document": "research_paper2.pdf",
      "page": 5,
      "title": "Neural Network Architectures",
      "rank": 2
    }
  ],
  "subsections": [
    {
      "document": "research_paper1.pdf",
      "page": 4,
      "text": "Recent advances in convolutional neural networks for image...",
      "rank": 1
    },
    {
      "document": "research_paper2.pdf",
      "page": 6,
      "text": "Transformer-based architectures and cross-modal applications...",
      "rank": 2
    }
  ]
}
```

This JSON schema enables downstream systems (or human readers) to easily parse, visualize, or further analyze the ranked information without re-processing the documents.

B. Glossary

- **Persona:** A user profile describing a role or professional identity (e.g., "AI Researcher" or "Business Analyst") used to guide content relevance.
- **Job-to-be-Done (JTBD):** A short textual statement describing the persona's goal or information need (e.g., "Identify sustainability practices").
- **Section:** A major heading or division in a document, typically representing a primary topic or content area.
- **Subsection:** A subordinate division within a section providing detailed explanations or examples.
- **Relevance Ranking:** The process of ordering document segments based on their semantic similarity to persona intent.
- **NLP (Natural Language Processing):** The field of AI concerned with analyzing and understanding human language using computational models.
- **JSON (JavaScript Object Notation):** A lightweight, structured data format used for representing input and output data in a human- and machine-readable form.