**AI Document Analyzer: Phase 3 - Development Documentation**

**Objective**

The development phase involved implementing the AI Document Analyzer based on the design specifications, creating functional modules for text extraction, NLP analysis, and a web interface. The goal was to build a working prototype ready for testing, addressing multi-format document processing and user interaction requirements.

**Activities**

1. **Environment Setup**:
   * Configured development environment with Python 3.8+, required libraries (pdfplumber, PyMuPDF, python-docx, pytesseract, flask, transformers, vaderSentiment), and Tesseract-OCR.
   * Set up directory structure: static/uploads, static/outputs, logs, templates, and documents for test files.
   * Integrated .env file for storing sensitive credentials (e.g., IBM Watson API key).
2. **Text Extraction Module**:
   * Implemented text extraction for PDFs using pdfplumber with fallback to PyMuPDF for scanned documents.
   * Developed DOCX and TXT file processing using python-docx and file I/O, respectively.
   * Built image text extraction with pytesseract, including preprocessing (grayscale, thresholding) using OpenCV.
3. **NLP Analysis Module**:
   * Integrated IBM Watson NLU for sentiment analysis, keyword extraction, and entity recognition, with retry logic for API failures.
   * Developed local fallbacks: vaderSentiment for sentiment, regex-based keyword/entity extraction, and transformers (BART for summarization, RoBERTa for question answering).
   * Implemented custom keyword extraction based on user input (e.g., “project,” “deadline”).
4. **Web Interface Development**:
   * Built a Flask-based web application with routes for file upload (/analyze), question answering (/ask), and result downloads (/download-txt, /download-json, /download-csv, /download-pdf).
   * Created index.html using Tailwind CSS for styling and Chart.js for visualizing keyword relevance.
   * Added session management to store analysis results for download functionality.
5. **Logging and Error Handling**:
   * Configured logging to capture detailed processing information and errors in logs/analyzer.log and logs/app.log.
   * Implemented error handling for invalid files, large file sizes (>10MB), and API failures, with user-friendly error messages.

**Deliverables**

* **Working Prototype**: Fully functional AI Document Analyzer with text extraction, NLP analysis, and web interface.
* **Module Documentation**: Descriptions of each component’s functionality and integration points.
* **Test Cases**: Initial set of test scenarios (e.g., processing Resume\_Musaib.pdf, analyzing custom keywords, answering questions like “What is the phone number?”).
* **Setup Guide**: Instructions for configuring the environment and running the application.

**Outcomes**

* Successfully implemented a prototype capable of processing PDFs, DOCX, TXT, and images, with NLP-driven insights.
* Integrated cloud and local NLP processing, ensuring robustness against API failures.
* Developed a responsive web interface for user interaction and result visualization.
* Encountered issues (e.g., sentiment score of 0, PDF download errors) to be addressed in testing.

**Next Steps**

* Advance to Phase 4: Testing, to validate functionality, fix issues, and ensure reliability.