BFSI- OCR OF BANK STATEMENTS

Milestone 4 Report

PREPARED BY:
Jasneet Arora

Table of Contents

1. Introduction

The BFSI (Banking, Financial Services, and Insurance) OCR (Optical Character Recognition) project aims to develop a comprehensive system to automate the extraction, visualization, and analysis of financial document data. Utilizing the meta-llama/Llama-3.2-11B-Vision-Instruct-Turbo model, the system processes bank statements, salary slips, and other financial records, ensuring accuracy and efficiency. Through integrated modules for data acquisition, quality control, OCR processing, and visualization, the project delivers a streamlined solution, culminating in a unified, user-friendly interface.

2. Objectives

Milestone 4 focuses on creating a unified system that integrates functionalities developed earlier into a seamless workflow. The goal is to enhance user experience by enabling manual and cloud-based image uploads through a Streamlit interface, implementing the Meta Llama OCR pipeline for text extraction, and providing dynamic data visualization with Plotly alongside querying capabilities using the Meta Llama model for interactive financial document analysis.

3. Methodology

The methodology for Milestone 4 integrates existing functionalities into a cohesive system. A Streamlit interface supports both manual image uploads and cloud-based retrieval via Cloudinary. The meta-llama/Llama-3.2-11B-Vision-Instruct-Turb model powers OCR processing, extracting data from financial documents with high accuracy. Visualization uses Plotly to create interactive graphs, while querying capabilities leverage the Meta Llama model for context-aware insights. This ensures an efficient, user-friendly workflow for financial data analysis.

4. Technical Implementation

4.1 Unified Streamlit Interface

The Streamlit interface serves as the central user interaction platform, offering:

- **Manual Uploads**: Users can upload images directly via a drag-and-drop interface.
- **Cloud-Based Retrieval**: Images stored in Cloudinary can be fetched using the Cloudinary API, providing flexibility for large-scale usage.

4.2 OCR Pipeline Integration

The OCR processing pipeline integrates:

- **Image Preprocessing**: Encoding images in Base64 format for compatibility with the Meta Llama model.
- **Data Extraction**: The meta-llama/Llama-3.2-11B-Vision-Instruct-Turbo model extracts key financial data, with custom prompts designed to capture structured information from complex document layouts.

- **Data Cleaning**: Pandas is used to organize extracted data, removing extraneous symbols and formatting issues using regular expressions.

4.3 Visualization and Insights

- **Dynamic Visualizations**: Interactive bar and pie charts created using Plotly enable users to analyse extracted data effectively.
- **Query Feature**: Users can ask document-specific questions through a chatbot interface, powered by the meta-llama/Llama-3.2-11B-Vision-Instruct-Turbo model, which provides context-sensitive answers.

4.4 File Management

Temporary files are managed efficiently using Python's tempfile and os libraries, ensuring seamless processing and system stability.

5. Results and Deliverables

5.1 Results

The integrated system successfully combines functionalities for:

- **Image Management**: Uploading and retrieving images from multiple sources.
- **Data Extraction**: Accurate text extraction and data organization for financial documents.
- **User Interaction**: Real-time data visualization and querying capabilities, enhancing user experience.

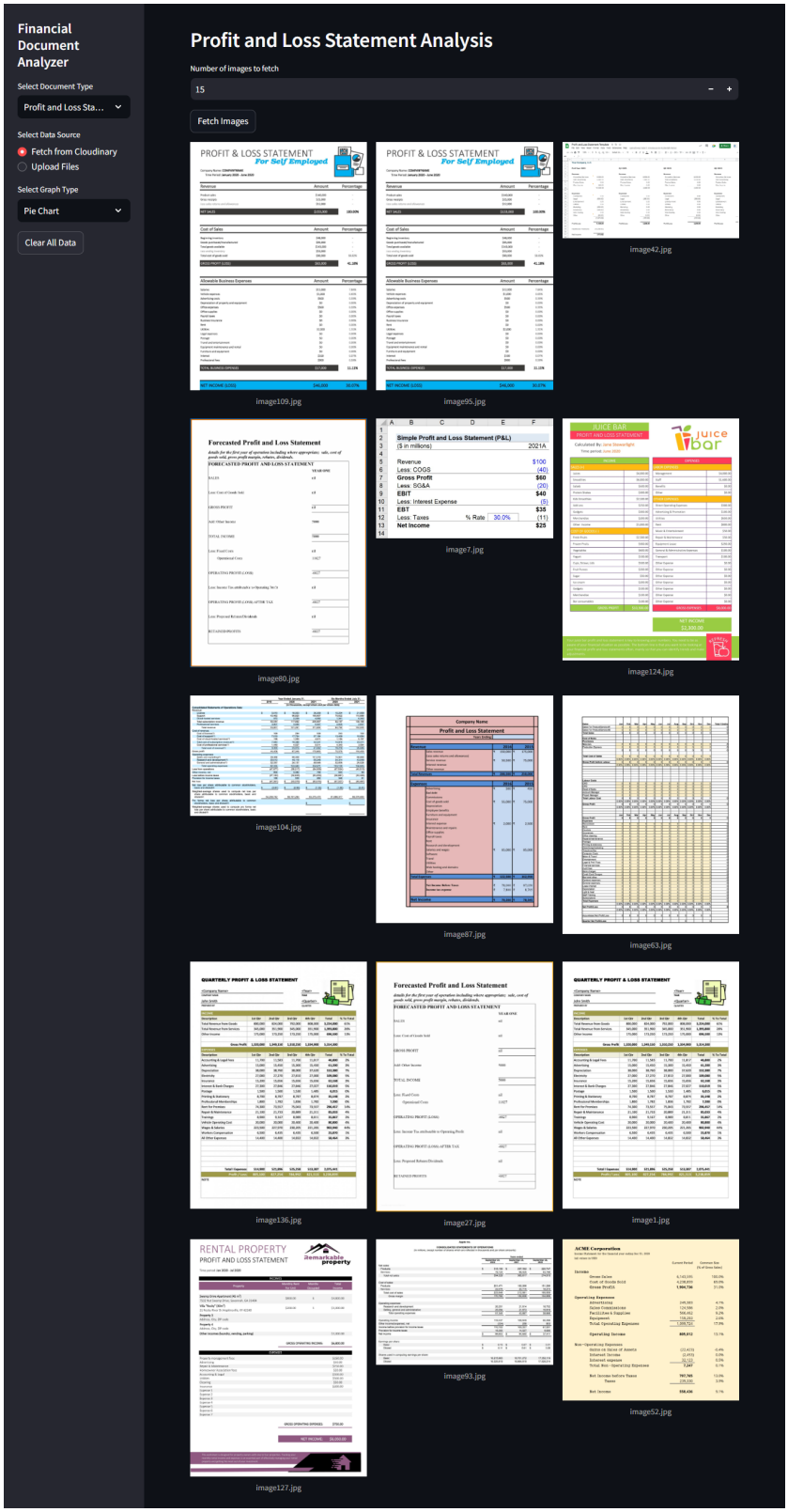
5.2 Deliverables

- **Streamlit Interface**: A unified interface supporting manual image uploads, cloud-based retrieval, data visualization, and document querying.
- **OCR Processing**: Fully functional OCR pipeline using the meta-llama/Llama-3.2-11B-Vision-Instruct-Turbo model.
- **Data Visualization**: Dynamic and interactive tools powered by Plotly.
- **Querying Features**: AI-driven interface for deeper insights into document content.

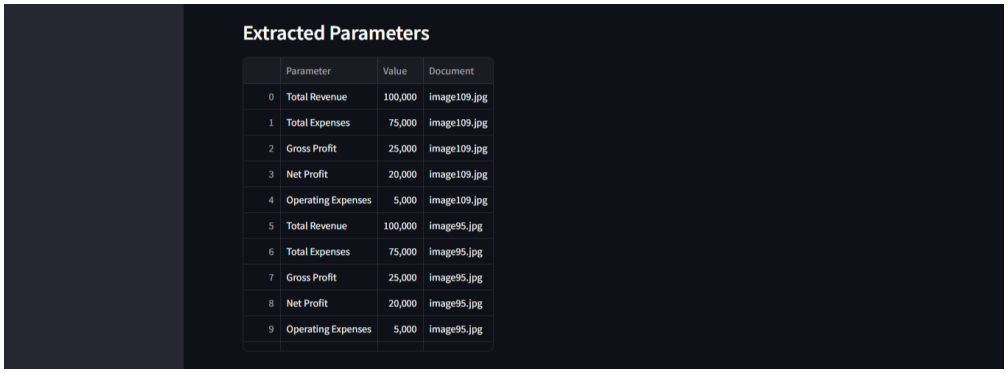
6. Dashboard Snapshots

6.1 Example 1: Profit and Loss Statement

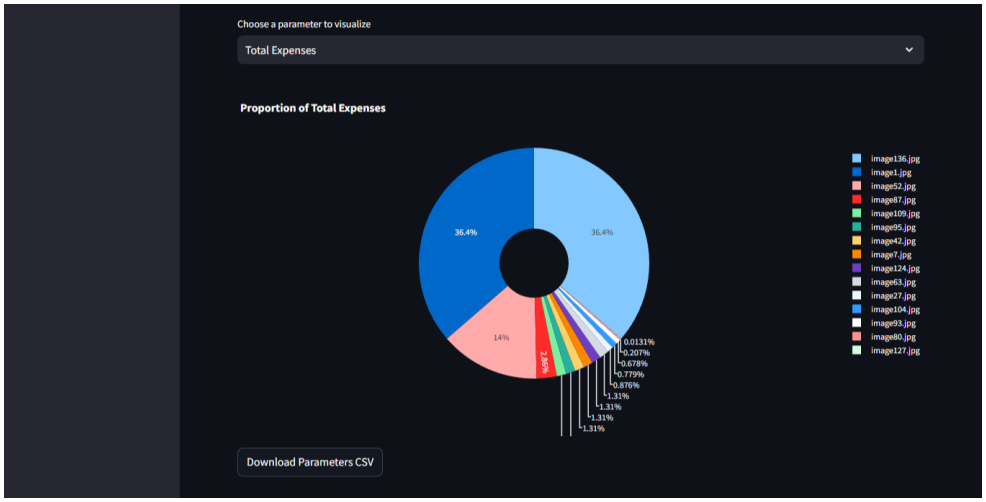
6.1.1 Updated Images Using Cloudinary



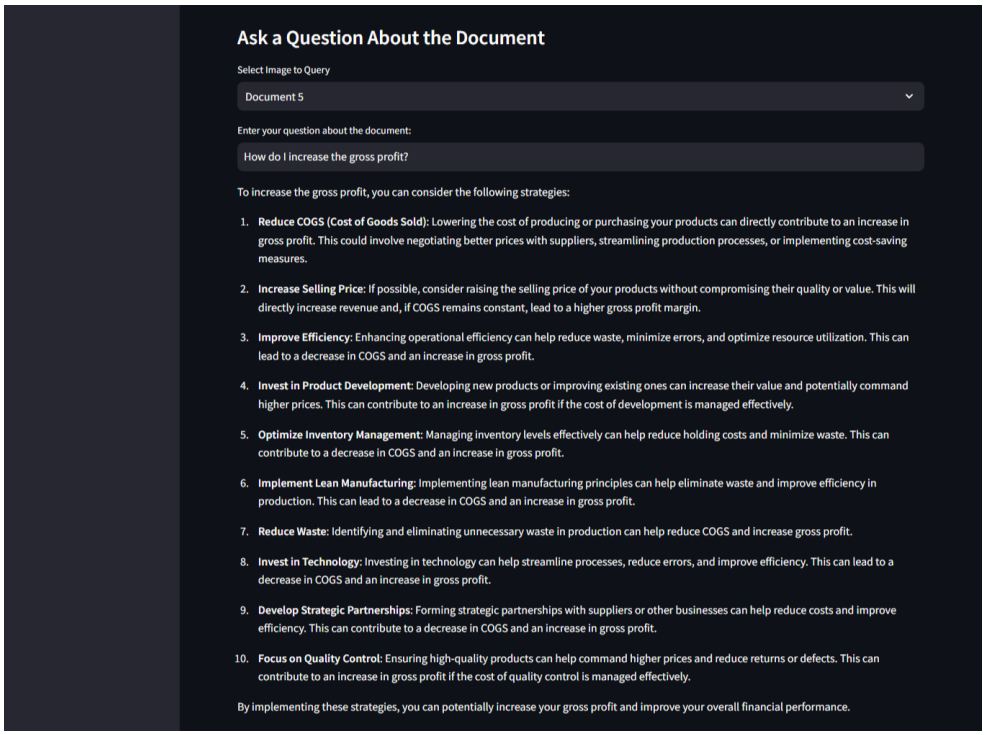
6.1.2 Parameter Extraction



6.1.3 Visualization: Pie Chart

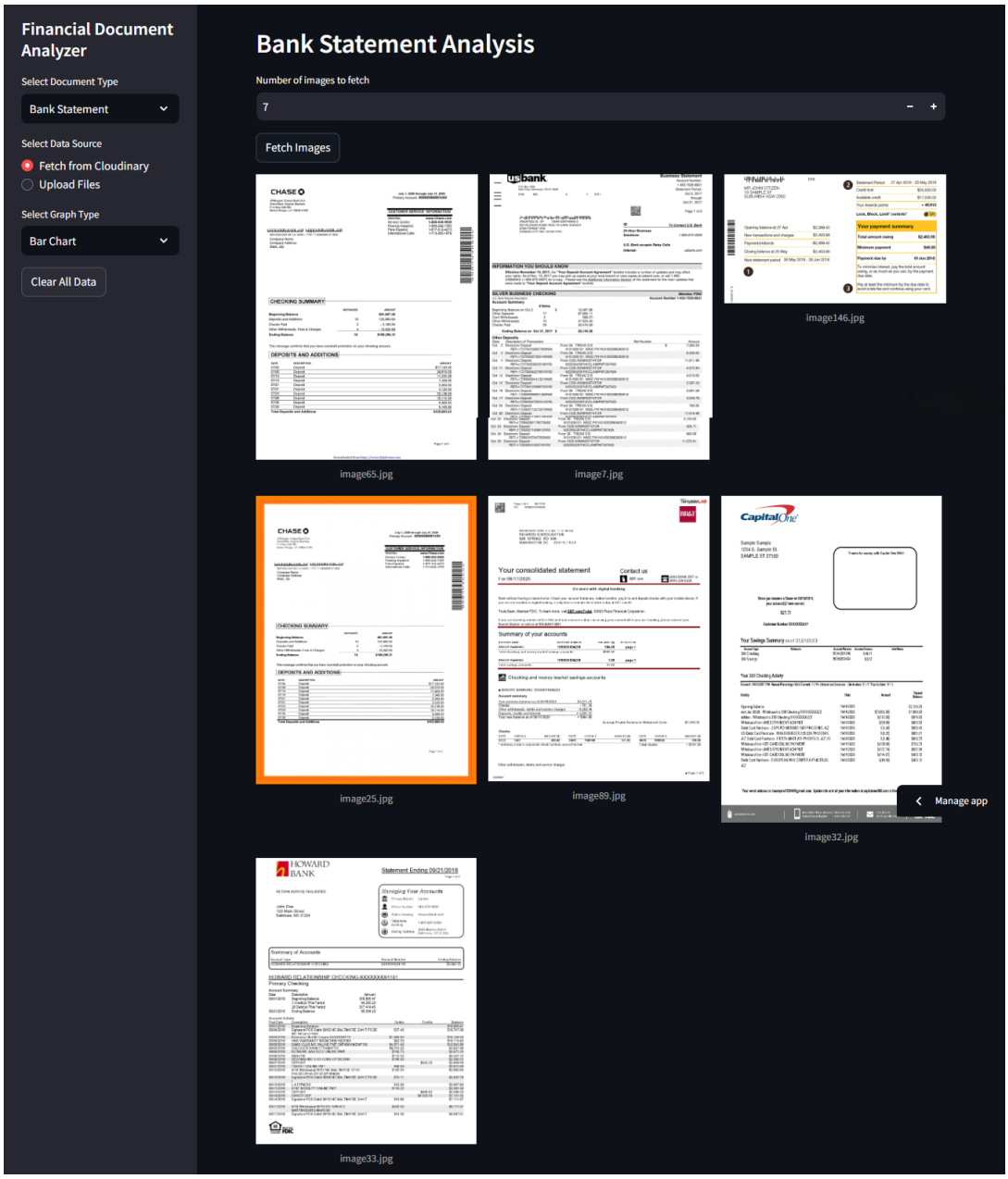


6.1.4 Ask a Question About the Document

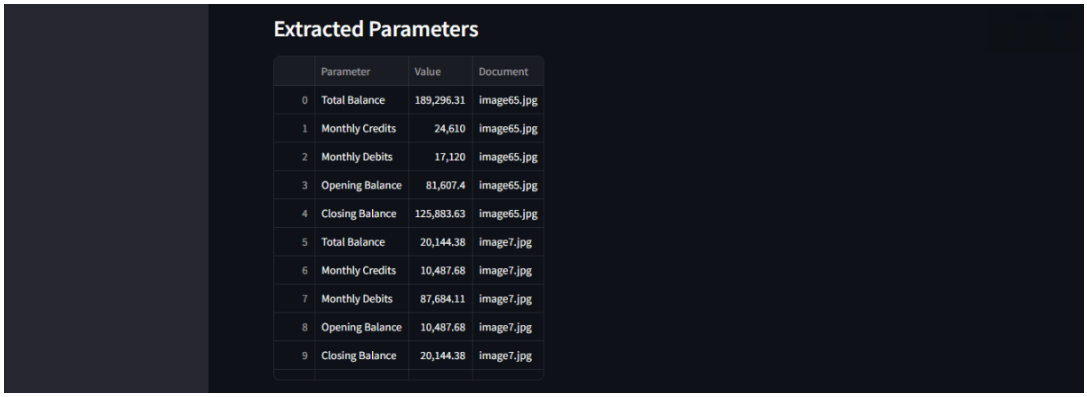


6.2 Example 2: Bank Statement

6.2.1 Updated Images Using Cloudinary



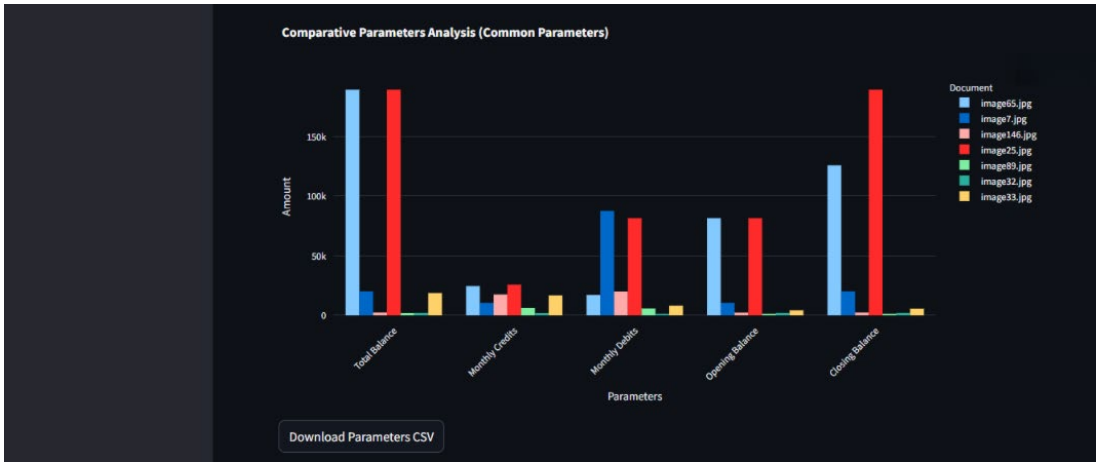
6.2.2 Parameter Extraction



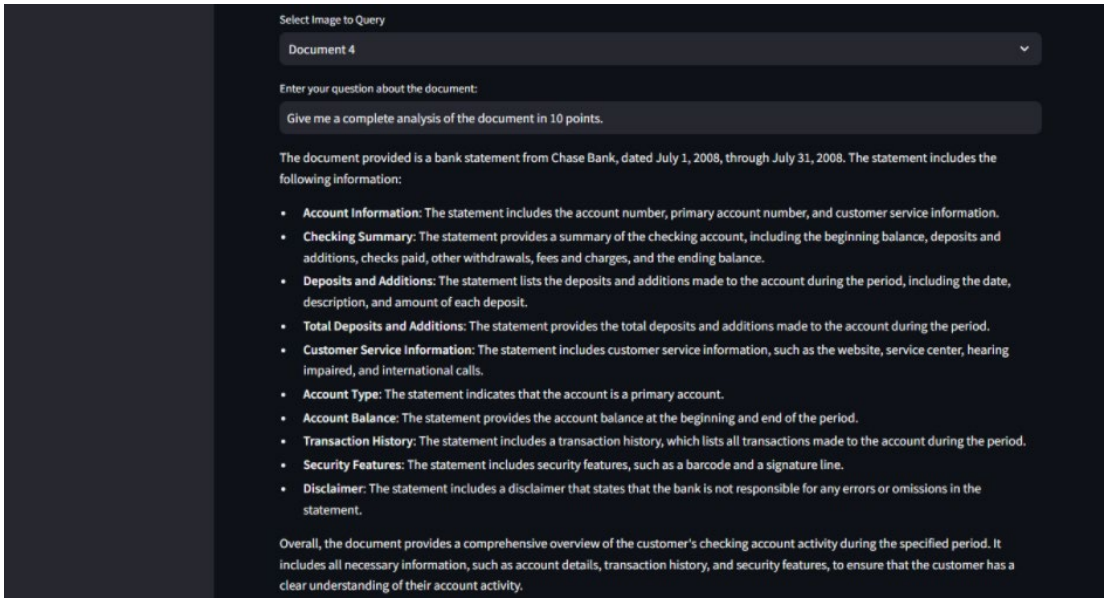
Extracted Parameters

	Parameter	Value	Document
0	Total Balance	189,296.31	image65.jpg
1	Monthly Credits	24,610	image65.jpg
2	Monthly Debits	17,120	image65.jpg
3	Opening Balance	81,607.4	image65.jpg
4	Closing Balance	125,883.63	image65.jpg
5	Total Balance	20,144.38	image7.jpg
6	Monthly Credits	10,487.68	image7.jpg
7	Monthly Debits	87,684.11	image7.jpg
8	Opening Balance	10,487.68	image7.jpg
9	Closing Balance	20,144.38	image7.jpg

6.2.3 Visualization: Bar Chart



6.2.4 Ask a Question About the Document



7. Challenges and Solutions

7.1 Integration Challenges

- **Challenge**: Harmonizing functionalities like Cloudinary integration, OCR processing, and data visualization without compromising performance.
- **Solution**: Modular architecture ensured seamless integration, while extensive testing verified system reliability.

7.2 API Reliability

- **Challenge**: Occasional connectivity and retrieval issues with the Cloudinary API.
- **Solution**: Implemented robust error-handling mechanisms, including retries and fallback operations.

8. Next Steps

Future enhancements will focus on:

- Deploying the system in a production environment for real-time financial document analysis.
- Optimizing performance to handle various document formats and larger datasets.
- Incorporating additional features such as multi-language support and automated report generation.

9. Appendices

Appendix A: Technical Architecture

- **Frontend**: Streamlit for user interaction, including image uploads and retrieval.
- **PDF/Image Processing**: PyMuPDF (fitz) for extracting content from PDFs and Pillow for handling image formats.
- **Data Processing**: Pandas for organizing and manipulating extracted data with regular expressions for cleaning.
- **Visualization**: Plotly for dynamic bar and pie charts.
- **AI Integration**: Together API for document analysis and parameter extraction.
- **Cloud Integration**: Cloudinary API for scalable image storage and retrieval.
- **System Advancements**: Enhanced querying with the Meta Llama model and real-time visualization for dynamic data insights.

Appendix B: Technical Specifications

- **Software**:
    - Streamlit: Framework for interactive web applications.
    - PyMuPDF: Extracting content from PDFs.
    - Pandas: Data manipulation and organization.
    - Plotly: Generating interactive visualizations (bar, pie charts).
    - Pillow: Processing and converting image formats.
    - Cloudinary API: Scalable image retrieval and storage.
    - Together API: AI-powered text extraction and analysis.
- **Hardware Requirements**:
    - RAM: Minimum 8GB.
    - CPU: 2.5 GHz or higher.

    - Storage: 50GB available space for file handling and processing.
- **Operating System**: Windows, macOS, or Linux.
- **Web Browser**: Chrome or Firefox recommended for Streamlit interface.

Appendix C: Hardware and Software Requirements

- **Hardware**:
    - Minimum RAM: 8GB.
    - CPU: Intel Core i5 or equivalent.
    - Storage: 50GB available space.
- **Software**:
    - Operating System: Windows, macOS, or Linux.
    - Python 3.8+ and libraries like Streamlit, PyMuPDF, Pillow, Pandas, Plotly, Together API.
    - Browser for UI interaction: Chrome or Firefox recommended.