Sure! Here's an overview of the code implementation for the Document Management System (DMS) from beginning to end:

1. File Parsing and Data Extraction:
   * Use a suitable library like PyPDF2 to parse the PDF file containing the recipe book.
   * Extract the text and images from the PDF file using the library functions.
   * Implement text processing techniques, such as tokenization and stemming, to analyze and extract relevant information like headings and content for each recipe.
   * Store the extracted data, including recipe names, headings, text, and image references, in a database or data structure for efficient retrieval.
2. User Interface Development:
   * Choose a suitable user interface framework, such as Flask or Django, to build the user interface for the DMS.
   * Design the interface to allow users to input their queries and display the search results.
   * Implement appropriate forms and input fields for query submission and result presentation.
3. Search Engine Implementation:
   * Design and implement a search engine component that can process user queries and retrieve relevant recipes from the stored data.
   * Utilize natural language processing techniques, such as keyword matching or similarity scoring, to compare the user query with the recipe names, headings, or content.
   * Retrieve the matching recipes based on the search results and present them to the user.
4. Image Extraction and Display:
   * Implement image extraction algorithms to extract the associated images for each recipe.
   * Store the image references or paths in the database alongside the recipe data.
   * Display the images along with the corresponding recipe information in the search results.
5. Integration and Testing:
   * Integrate the different components of the DMS, including the file parsing, data extraction, search engine, and user interface.
   * Conduct thorough testing to ensure the system functions as expected.
   * Perform unit tests for individual functions and integration tests to validate the interactions between components.
   * Test the search functionality with different queries to ensure accurate and relevant results.
6. Deployment and Documentation:
   * Prepare the DMS for deployment in the desired environment, considering factors like server configuration, security measures, and scalability requirements.
   * Create comprehensive documentation that includes installation instructions, configuration settings, and usage guidelines for the DMS.
   * Provide detailed documentation on the codebase, explaining the purpose of each module, function, and class.
7. User Training and Support:
   * Conduct training sessions to familiarize end-users with the DMS and its functionalities.
   * Provide user manuals or guides to assist users in utilizing the system effectively.
   * Establish a support mechanism to address user queries, provide assistance, and handle any technical issues that may arise.

It's important to note that the above steps provide a high-level overview, and the actual implementation details may vary based on your specific requirements, chosen technologies, and programming style. It's recommended to refer to relevant documentation, tutorials, and code examples for the libraries and frameworks you choose to work with.

Example code ->

import PyPDF2

def search\_item\_in\_pdf(item\_name, pdf\_path):

with open(pdf\_path, 'rb') as file:

pdf\_reader = PyPDF2.PdfReader(file)

total\_pages = len(pdf\_reader.pages)

item\_results = []

current\_item = None

current\_text = ""

current\_images = []

for page\_number, page in enumerate(pdf\_reader.pages, start=1):

page\_text = page.extract\_text().lower()

if item\_name.lower() in page\_text:

if current\_item is not None:

item\_results.append((current\_item, current\_text, current\_images))

current\_item = item\_name

current\_text = page\_text

current\_images = []

elif current\_item is not None:

current\_text += page\_text

current\_images.extend(page.extract\_images())

if current\_item is not None:

item\_results.append((current\_item, current\_text, current\_images))

return item\_results, total\_pages

# Example usage

item\_to\_search = "Chocolate Cake"

pdf\_file\_path = 'path\_to\_your\_pdf.pdf'

item\_results, total\_pages = search\_item\_in\_pdf(item\_to\_search, pdf\_file\_path)

if len(item\_results) > 0:

print(f"Results for item '{item\_to\_search}':")

for item, text, images in item\_results:

print(f"Item: {item}")

print("Text:")

print(text)

print("Images:")

for i, image in enumerate(images):

print(f"Image {i+1}:")

print("Width:", image.width)

print("Height:", image.height)

print("Image Data:", image.data)

print()

else:

print(f"Item '{item\_to\_search}' not found in the PDF.")