**PDF Data Extraction Web Application**

This documentation provides an overview of the PDF Data Extraction Web Application, including the approach taken, design choices, challenges faced, and instructions for running the application.

**Approach**

The approach for building this web application involved utilizing the Flask framework in Python. The application allows users to upload a PDF file, extract key-value pairs from the PDF, and display the extracted data on a web page. The PyPDF2 library was chosen for parsing and extracting information from the PDF files.

The implementation consisted of the following key steps:

1. Setting up a Flask application with appropriate routes and templates.

2. Designing an HTML template to present the extracted data in a well-organized table format.

3. Implementing a file upload route to handle the uploading and saving of PDF files to a specified folder.

4. Utilizing PyPDF2 library to extract key-value pairs from the uploaded PDF file.

5. Storing the extracted data in a dictionary and passing it to the template for rendering on the web page.

6. Incorporating functionality to save the extracted data in a CSV file.

7. Handling error cases, such as invalid file uploads or missing data, to ensure a smooth user experience.

**Design Choices**

Several design choices were made to enhance the functionality and usability of the web application:

1. Flask Framework: Flask was chosen due to its simplicity and flexibility, providing efficient routing and template rendering capabilities.

2. HTML Template: The HTML template was designed with a clean and professional look. The extracted data is displayed in a table format, making it easy to read and comprehend.

3. PyPDF2 Library: PyPDF2 was selected for its comprehensive PDF parsing and extraction features, enabling the application to handle a wide range of PDF file structures.

4. Upload Folder: An "uploads" folder was created within the project directory to centrally store uploaded PDF files and generated CSV files, ensuring better file management.

**Challenges**

During the development process, several challenges were encountered:

1. PDF Parsing: Parsing PDF files can be complex, with variations in file structures. Handling different scenarios and ensuring accurate extraction required careful consideration.

2. Table Extraction: Extracting tabular data from PDF files poses challenges due to formatting and structural variations. Assumptions were made about table structures, which may not be universally applicable.

3. Error Handling: Comprehensive error handling was implemented to address cases such as invalid uploads and missing data. However, additional edge cases may still need to be considered for robustness.

**Instructions to Run the Application**

To run the application, follow these steps:

1. Clone the repository to your local machine

2. Navigate to the project directory

3. Install the required dependencies using pip:

4. Start the Flask application:

5. Access the application in your web browser at `http://localhost:5000`.6. Upload a PDF file using the provided form.

7. The extracted data will be displayed on the web page.

8. The extracted data will also be saved in a CSV file located in the `uploads` folder.

Note: Ensure that Python 3.7 or above is installed on your machine.

**Conclusion**

The PDF Data Extraction Web Application provides a user-friendly way to extract key-value pairs from PDF files and display them on a web page. By leveraging Flask, PyPDF2, and CSV libraries, the application offers a simple and intuitive solution for PDF data extraction.