1. Introduction

This project aims to create a web application that allows users to upload a PDF file, extract data from it, and filter the data based on specific criteria. The application is particularly useful for educators, administrators, or any users who need to analyse and manage data in PDF files without relying on manual extraction and filtering.

2. Project Overview

hy	jit	Karan	Mandloi
45	78	15	89
546	456	89	87
45	465	654	19

The web application will provide users with a simple interface to upload a PDF containing student data. The system will extract the relevant data from the PDF, allow users to apply filters (e.g., selecting students with marks greater than a specified value), and display the filtered results.

3. Prerequisites

3.1. Functional Requirements

- **User Interface**: A web page where users can upload a PDF file.
- **Data Extraction**: The ability to extract text and tabular data from the uploaded PDF.
- **Filtering**: Options to filter the extracted data based on user-specified criteria (e.g., marks greater than 50).
- **Result Display**: Display the filtered results on the web page.

3.2. Non-Functional Requirements

- **Performance**: The web application should handle PDF files up to a reasonable size without significant delays.
- **Usability**: The interface should be simple , requiring minimal user input.
- **Security**: The application should securely handle uploaded files and prevent unauthorized access or data leaks.

4. Technologies Used

4.1. Frontend

- **HTML/CSS**: For structuring and styling the web pages.

- **JavaScript**: For handling client-side interactions and form submissions.

4.2. Backend

- **Python (Flask/Django)**: To handle server-side logic, including PDF processing, data extraction, and filtering.
- **pdfplumber**: A Python library used to extract text from PDF files. ●**pandas**: A Python library for data manipulation and filtering.

4.3. Tools

- **VS Code**: For writing and editing the code.
- **Git**: For version control.

5. Workflow of the Project

5.1. User Uploads PDF

- The user navigates to the web application and uploads a PDF file containing e.g. student data.

5.2. PDF Processing

- The backend server (Flask/Django) receives the PDF file,uses pdfplumber to extract the text, and processes the text into a structured format using pandas.

5.3. Data Filtering

- The user specifies filtering criteria (e.g., students with marks greater than 50).
- The application filters the processed data using pandas based on the user's input.

5.4. Display and Download Results

- The filtered results are displayed on the web page in a tabular format.

6. Conclusion

This project demonstrates the development of a web application that simplifies the process of extracting and filtering data from PDF files. By leveraging Python's data processing libraries and a web framework like Flask or Django, the application provides a user-friendly platform for handling PDF data without requiring advanced technical knowledge.