# 📁Project Structure: PDF Outline Extractor

This document explains the purpose of each file in the repository submitted for **Round 1A** of the **Adobe India Hackathon 2025: Connecting the Dots**.

---

## 🔗`main.py`

- **Description:**\ The **core script** responsible for orchestrating the PDF outline extraction process.
- **Responsibilities:**
- Loads the input PDF.
- Extracts headings (Title, H1, H2, H3).
- Formats and writes the output JSON.
- Calls utility functions from `utils.py`.

---

## 🔗`utils.py`

- **Description:**\ A **helper module** that contains reusable functions used by `main.py`.
- **Responsibilities:**
- Text classification and heading level detection (e.g., Title vs H1-H3).
- PDF layout parsing (page-wise, font-based).
- Sorting and grouping lines for hierarchy inference.

---

## 🔗`requirements.txt`

- **Description:**\ Specifies all **Python dependencies** required to run the extractor.
- **Common Entries:**
- `PyMuPDF` (`fitz`) – for reading and analyzing PDF files.
- `numpy`, `json`, or any other package used in processing.

---

## 🔗`Dockerfile`

- **Description:**\ A **container configuration file** used to build and run the extractor in an isolated environment (e.g., for offline judging).
- **Features:**
- CPU-only setup.
- Installs Python and project dependencies.
- Copies all necessary code and files.
- Sets `main.py` as the entry point.

---

### 🔗 `sample.pdf`

- **Description:**\ A **sample input PDF** used for testing and verifying the outline extraction.
- **Contents:**\ Includes structured headings (Title, H1-H3) to demonstrate the tool's functionality.

---

### 🔗 `output.json`

- **Description:**\ The **JSON result** file generated after running `main.py` on `sample.pdf`.
- **Format:**

```
[
  {
    "text": "Introduction",
    "type": "H1",
    "page": 1
  },
  ...
]
```

- Follows the format defined in the Hackathon Round 1A prompt.

---

### 🔗 `README.md`

- **Description:**\ The **project readme file** that provides an overview of the extractor, setup instructions, usage examples, and output format.
- **Should Include:**
- How to run locally and with Docker.
- Sample command-line usage.
- Screenshot or snippet of JSON output.

---

### 🔍 `utils.cpython-39.pyc`

- **Description:**\ A **compiled Python bytecode file** automatically generated when `utils.py` is imported.
- **Recommendation:**\ Not necessary for version control. Add to `.gitignore`:

```
*.pyc
__pycache__/
```

---

## 🔗 Suggested `.gitignore`

To keep the repo clean:

```
*.pyc
__pycache__/
output.json
```

---