Project Documentation

```
/Users/A1064331/Desktop/pruebas/project2pdf/project2pdf_lib
??? README.md
??? dist
?   ??? project2pdf-0.1.0-py3-none-any.whl
?   ??? project2pdf-0.1.0.tar.gz
??? project2pdf.egg-info
?   ??? PKG-INFO
?   ??? SOURCES.txt
?   ??? dependency_links.txt
?   ??? entry_points.txt
?   ??? requires.txt
?   ??? top_level.txt
??? project2pdf_src
?   ??? __init__.py
?   ??? __pycache__
?   ??? pdf_generator.py
?   ??? utils.py
??? pyproject.toml
??? requirements.txt
??? scripts
?   ??? __init__.py
?   ??? generate_pdf.py
??? setup.py
??? tests
    ??? test_pdf_generator.py
    ??? test_utils.py

7 directories, 19 files
```

Project Structure: dist

```
/Users/A1064331/Desktop/pruebas/project2pdf/project2pdf_lib/dist
??? project2pdf-0.1.0-py3-none-any.whl
??? project2pdf-0.1.0.tar.gz

1 directory, 2 files
```

Project Structure: tests

```
/Users/A1064331/Desktop/pruebas/project2pdf/project2pdf_lib/tests
??? test_pdf_generator.py
??? test_utils.py

1 directory, 2 files
```

? tests / test_utils.py

? tests / test_pdf_generator.py

Project Structure: scripts

```
/Users/A1064331/Desktop/pruebas/project2pdf/project2pdf_lib/scripts
??? __init__.py
??? generate_pdf.py

1 directory, 2 files
```

? scripts / __init__.py

? scripts / generate_pdf.py

```
import os
from project2pdf.pdf_generator import ProjectPDFGenerator

def main():
     """Automatically detects projects in the root directory and generates PDFs without
requiring parameters."""
    root_dir = os.getcwd()  # Use the current directory as the root
    generator = ProjectPDFGenerator(root_dir)
    generator.generate_pdf()

if __name__ == "__main__":
    main()
```

Project Structure: project2pdf_src

```
/Users/A1064331/Desktop/pruebas/project2pdf/project2pdf_lib/project2pdf_src
??? __init__.py
??? __pycache__
?   ??? __init__.cpython-311.pyc
?   ??? pdf_generator.cpython-311.pyc
??? pdf_generator.py
??? utils.py

2 directories, 5 files
```

? project2pdf_src / pdf_generator.py

```
import os
import subprocess
from fpdf import FPDF

class ProjectPDFGenerator:
      """Class to generate a single PDF documentation for all projects in the root
directory."""
    def __init__(self, root_dir):
        self.root_dir = root_dir
        self.pdf = FPDF()
        self.pdf.set_auto_page_break(auto=True, margin=15)

    def get_project_structure(self, directory):
        """Runs the 'tree -L 2' command and returns the result as text."""
        try:
             result = subprocess.run(["tree", "-L", "2", directory], capture_output=True,
text=True)
            return result.stdout
        except FileNotFoundError:
               return "The 'tree' command is not available. Install it or use another
method."

    def add_section(self, title, content):
        """Adds a section with title and content to the PDF."""
        self.pdf.add_page()
        self.pdf.set_font("Arial", 'B', 12)
             self.pdf.cell(0,  10,  title.encode('latin-1',  'replace').decode('latin-1'),
ln=True, align='L')
        self.pdf.ln(5)
        self.pdf.set_font("Courier", '', 10)
                                self.pdf.multi_cell(0,    5,    content.encode('latin-1',
'replace').decode('latin-1'))

    def extract_text_from_file(self, file_path):
        """Reads the content of compatible text files."""
        try:
            with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
                return f.read()
        except Exception as e:
            return f"Error reading {file_path}: {str(e)}"

    def generate_pdf(self):
           """Generates a single PDF with the structure of all projects in the root
directory."""
                        projects   =   [d   for   d   in   os.listdir(self.root_dir)   if
os.path.isdir(os.path.join(self.root_dir, d)) and not d.startswith(".")]
        output_pdf = os.path.join(self.root_dir, "project_documentation.pdf")

        # First page: Root directory structure
        self.pdf.add_page()
        self.pdf.set_font("Arial", 'B', 16)
```

```
        self.pdf.cell(0, 10, "Project Documentation", ln=True, align='C')
        self.pdf.ln(10)
        self.pdf.set_font("Courier", '', 10)
                                                        self.pdf.multi_cell(0,       5,
self.get_project_structure(self.root_dir).encode('latin-1',
'replace').decode('latin-1'))

        # Iterate through each project and add details
        for project in projects:
            project_path = os.path.join(self.root_dir, project)

            self.pdf.add_page()
            self.pdf.set_font("Arial", 'B', 14)
            self.pdf.cell(0, 10, f"Project Structure: {project}", ln=True, align='C')
            self.pdf.ln(10)
            self.pdf.set_font("Courier", '', 10)
                                                          self.pdf.multi_cell(0,     5,
self.get_project_structure(project_path).encode('latin-1', 'replace').decode('latin-1'))

              valid_extensions = {".py", ".ipynb", ".tsx", ".js", "dockerfile", ".env",
".ignore", ".md"}

            for root, _, files in os.walk(project_path):
                for file in files:
                       if any(file.lower().endswith(ext) for ext in valid_extensions) or
file.lower() in {"dockerfile", ".env", ".gitignore"}:
                        file_path = os.path.join(root, file)
                        content = self.extract_text_from_file(file_path)
                        relative_path = os.path.relpath(file_path, project_path)
                        self.add_section(f"? {project} / {relative_path}", content)

        self.pdf.output(output_pdf)
        print(f"Single PDF generated at: {output_pdf}")

if __name__ == "__main__":
    root_dir = os.getcwd()  # Uses the current working directory as the root
    generator = ProjectPDFGenerator(root_dir)
    generator.generate_pdf()
```

? project2pdf_src / __init__.py

? project2pdf_src / utils.py

```
import os
import subprocess

def get_project_structure(root_dir):
    """Executes the 'tree -L 2' command and returns the result as text."""
    try:
            result = subprocess.run(["tree", "-L", "2", root_dir], capture_output=True,
text=True)
        return result.stdout
    except FileNotFoundError:
        return "The 'tree' command is not available. Install it or use another method."

def extract_text_from_file(file_path):
    """Reads the content of compatible text files."""
    try:
        with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
            return f.read()
    except Exception as e:
        return f"Error reading {file_path}: {str(e)}"
```

Project Structure: project2pdf.egg-info

```
/Users/A1064331/Desktop/pruebas/project2pdf/project2pdf_lib/project2pdf.egg-info
??? PKG-INFO
??? SOURCES.txt
??? dependency_links.txt
??? entry_points.txt
??? requires.txt
??? top_level.txt

1 directory, 6 files
```