```
--- projectToText.py ---
import os
from fpdf import FPDF

# Directory where the project files are located
directory = r"D:\HARRISBURG\Harrisburg Master's Fifth Term Late Summer\CISC 699\DiscordBotProject_CISC699"
output_pdf_path = os.path.join(directory, "projectToText.pdf")

# Lists for files and folders to ignore
filesToIgnore = ['ignore_this.py', 'Tests_URLs.txt', 'UseCases.txt', 'Read.md', '*.pdf']  # Example file names to ignore
foldersToIgnore = ['ignore_folder', '.git', '__pycache__', 'PersonelTest', 'MockTesting', 'ExportedFiles']  # Folders to ignore

# Function to retrieve all text from files, ignoring specific folders and files
def extract_project_text(directory, ignore_files=None, ignore_folders=None):
    if ignore_files is None:
        ignore_files = []
    if ignore_folders is None:
        ignore_folders = []

    project_text = ""
    for root, dirs, files in os.walk(directory):
        # Ignore specific folders
        dirs[:] = [d for d in dirs if d not in ignore_folders]
```

```
        for file in files:
            # Skip ignored files
            if file in ignore_files:
                continue
            # Only considering relevant file types
            if file.endswith('.py'):
                file_path = os.path.join(root, file)
                try:
                    with open(file_path, 'r', encoding='utf-8') as f:
                        project_text += f"--- {file} ---\n"
                        project_text += f.read() + "\n\n"
                except Exception as e:
                    print(f"Could not read file {file_path}: {e}")

    return project_text

# Function to generate a PDF with the extracted text
def create_pdf(text, output_path):
    pdf = FPDF()
    pdf.set_auto_page_break(auto=True, margin=15)
    pdf.add_page()
    pdf.set_font("Arial", size=12)

    # Ensure proper encoding handling
    for line in text.split("\n"):
        # Convert the text to UTF-8 and handle unsupported characters
        try:
```

```
            pdf.multi_cell(0, 10, line.encode('latin1', 'replace').decode('latin1'))
        except UnicodeEncodeError:
            # Handle any other encoding issues
            pdf.multi_cell(0, 10, line.encode('ascii', 'replace').decode('ascii'))

    pdf.output(output_path)

# Function to create PDFs for specific folders
def create_folder_specific_pdfs(directory, ignore_files=None, ignore_folders=None):
    if ignore_files is None:
        ignore_files = []
    if ignore_folders is None:
        ignore_folders = []

    # Create PDFs for each folder in the project
    for folder in os.listdir(directory):
        folder_path = os.path.join(directory, folder)
        if os.path.isdir(folder_path) and folder not in ignore_folders:
            folder_text = extract_project_text(folder_path, ignore_files, ignore_folders)
            if folder_text:
                folder_pdf_path = os.path.join(folder_path, f"All_files_in_{folder}_folder_toText.pdf")
                create_pdf(folder_text, folder_pdf_path)
                print(f"PDF created for folder {folder} at: {folder_pdf_path}")

# Extract project text and create the main project PDF
project_text = extract_project_text(directory, filesToIgnore, foldersToIgnore)
if project_text:
```

```
    create_pdf(project_text, output_pdf_path)
    print(f"Main PDF created with all project's text at: {output_pdf_path}")
else:
    print("No project text found.")

# Create PDFs for each specific folder
create_folder_specific_pdfs(directory, filesToIgnore, foldersToIgnore)



--- project_structure.py ---
import os

def list_files_and_folders(directory, output_file):
    with open(output_file, 'w') as f:
        for root, dirs, files in os.walk(directory):
            # Ignore .git and __pycache__ folders
            dirs[:] = [d for d in dirs if d not in ['.git', '__pycache__']]

            f.write(f"Directory: {root}\n")
            for dir_name in dirs:
                f.write(f"  Folder: {dir_name}\n")
            for file_name in files:
                f.write(f"  File: {file_name}\n")

# Update the directory path to your project folder
project_directory = "D:/HARRISBURG/Harrisburg Master's Fifth Term Late Summer/CISC
699/DiscordBotProject_CISC699"
```

```
output_file = os.path.join(project_directory, "other/project_structure.txt")


# Call the function to list files and save output to .txt

list_files_and_folders(project_directory, output_file)


print(f"File structure saved to {output_file}")
```