Python Personal Finance Manager

1. Project Overview

This project is a comprehensive command-line application designed to help users track, manage, and analyze their personal financial expenses. It demonstrates mastery of fundamental Python concepts, including Object-Oriented Programming (OOP), robust file handling for data persistence, modular code organization, and essential error handling.

- **Version:** Standard Version (Expense Tracking, Reporting, and Analysis)
- **Deliverables Met:** Object-Oriented Design, CSV Data Persistence, Modular Code Structure, Interactive CLI, Error Handling, and GitHub Repository structure.

2. Technical Requirements Met

Requirement	Implementation Details
OOP Design	Implemented the `Expense` class (`src/expense.py`) to encapsulate expense attributes (amount, category, date, description) and utility methods (`__str__`, `to_list`).
File Handling/Persistence	Implemented CSV read/write operations in `src/file_manager.py` using the built-in `csv` module to store and retrieve data from `data/expenses.csv`.
Error Handling	Implemented comprehensive input validation in `src/utils.py` (e.g., ensuring amount is numerical, date is `YYYY-MM-DD`, and category is valid). Includes `try...except` blocks in `file_manager.py` to handle `IOError` and file corruption.
Code Structure	Code is organized into dedicated modules (`expense.py`, `file_manager.py`, `menu.py`, `reports.py`, `utils.py`) within the `src/` directory.
User Interface	Implemented an interactive command-line interface with a menu system in `src/menu.py`.
Report Generation	`src/reports.py` generates a detailed text file summary (comprehensive report) showing total expenses, category-wise breakdown, and percentages.

3. Setup and Installation Guide

Prerequisites

- Python 3.6+
- pip (Python package installer)
- git (for cloning)

Step-by-Step Installation

1. **Clone the Repository:**

```
git clone https://github.com/VISWANADHVEERA/personal-finance-manager.git

cd personal-finance-manager
```

2. Verify Dependencies:The Standard Version uses only built-in Python modules (csv, os, datetime). The requirements.txt file is empty for this version.
3. Run the Application: Execute the main program file from the root directory:

```
python main.py
```

4. User Manual and Functionality

The application opens with the main menu, providing access to all features:

```
==========================================
    PERSONAL FINANCE MANAGER (STANDARD)
==========================================
MAIN MENU:
1. Add New Expense
2. View All Expenses
3. View Category-wise Summary
4. Generate Comprehensive Report
5. Search Expenses by Category/Description
6. Backup Data
7. Exit & Save
```

Option	Description	Evidence/Testing
1. Add New Expense	Prompts for validated input (Amount, Date, Category, Description). Uses utils.py for input integrity.	Test with non-numerical amount or wrong date format.
2. View All Expenses	Displays all expenses loaded from the CSV file.	Verify successful display of data added in Option 1.
3. View Category-wise Summary	Calculates and displays total spending per category and the percentage share of each category.	Compare calculated totals against expected sums.
4. Generate Report	Creates a detailed text report in the reports/ folder.	Check reports/ folder for the timestamped file.

5. Search Expenses	Allows filtering expenses by keywords in the Category or Description fields.	Search for a specific expense description.
6. Backup Data	Creates a timestamped duplicate of the `expenses.csv` file in the `data/` folder.	Check `data/` folder for the `expenses_backup_...csv` file.
7. Exit & Save	Saves all current in-memory expenses to `data/expenses.csv` and closes the application.	**Crucial:** Always use this to persist data.

5. Technical Architecture and Code Explanation

The project uses a clean, **modular architecture** with clear separation of concerns, making it easy to maintain and expand.

File/Directory	Role	Key Concepts Demonstrated
main.py	Application Entry Point	Initialization, calling the main menu loop.
src/expense.py	**Expense Class Definition**	**OOP**, Data Encapsulation.
src/file_manager.py	**CSV Operations**	**File Handling**, Data Persistence, Error Handling (IOError), Backup/Restore.
src/utils.py	**Input Validation**	**Error Handling**, Data Formatting (is_valid_date, get_valid_amount).
src/reports.py	**Analysis Functions**	Data Aggregation, Report Generation, File Output.
src/menu.py	**User Interface**	Program flow, User Interaction.
data/	Data storage (contains expenses.csv)	
reports/	Generated analysis reports	

6. Troubleshooting and Testing Evidence

Problem	Cause	Solution
StopIteration	Attempting to read a non-existent row in an empty CSV file.	Added try...except StopIteration block around next(reader) in file_manager.py.
Permission denied	The expenses.csv file was locked by the VS Code editor or another program.	**Action:** Close the expenses.csv file tab in VS Code.
src refspec main does not match	Local Git branch was named master, but remote push was to main.	**Action:** Used git branch -M main to rename the local branch before pushing.