To-Do List Project in Python

Project Overview

This project documents a Python-based to-do list application featuring task addition, deletion, completion marking, and persistent storage using text files. The focus is on beginner-to-intermediate Python learners interested in using lists, loops, file handling, and optionally GUI frameworks.

Detailed Table of Contents

1. Introduction

Begin the document by introducing the concept of a to-do list—its importance for productivity and time management. Talk about Python's suitability for rapid prototyping—even for beginners. Expand on why persistent storage (text file) is useful

2. Problem Statement and Objective

Clearly describe the aims:

- **Build a command-line to-do list for students and busy professionals.**
- **Implement features: Add, delete, mark complete, save/load tasks.**
- **Encourage practice of coding basics (lists, loops, file I/O).**

3. Technical Stack and Requirements

Go into detail about:

- **Python version used (mention how to install it).**
- **Necessary libraries (standard Python only.**
- **Operating system compatibility.**

4. Project Architecture

Provide a diagram or text explanation of program flow:

- **Main loop**
- **User menu**
- **Functions for each task**
- **File structure ("tasks.txt")**

5. Code: Step-by-Step Development

Expand each main part of the code, with extensive explanations, comments, and reasoning:

- **Initialize lists**
- **Write each function in detail (add, delete, mark complete)**
- **Explain file handling practices**
- **Present full code, broken into logical parts, with explanations for each**

6. Sample Output and Screenshots

Provide multiple sample runs:

- **Adding tasks**
- **Viewing and marking tasks as complete**
- **Deleting items**
- **Show sample contents of "tasks.txt"**

7. Features Explained

Detail each feature:

- **Task addition: Input validation, duplicate prevention**
- **Editing tasks: How to modify text**
- **Task completion: Mark and show status visually in text**
- **Persistent storage: Explain read/write, error handling for missing files**

8. Enhancements and Advanced Concepts

Introduce ways to expand the project for higher marks and word count:

- **GUI implementation , showing code and screenshots.**
- **User authentication (simple password protection).**
- **Adding deadlines, priorities, categories.**
- **Exporting lists to CSV or PDF.**
- **Code optimizations, exception handling.**

9. Testing and Debugging

Detail your testing approach:

- **List of edge cases (empty task, invalid input, file errors)**
- **Example buggy runs and resolutions**
- **Screenshot of debugging in IDE**

10. User Guide

Write full step-by-step instructions for users:

- **How to run the app**
- **How to add, delete, mark complete**
- **File format references**

11. Conclusion

Summarize key learnings from the project:

- **Coding skills developed**
- **Practical file management experience**
- **How users can further expand this project**