PyVault – Personal Knowledge Manager (CLI)

Student Name: Daniel Thomas
Module: Software Development / Programming with Python
Date: [Insert Date]

1. Introduction

This document describes the design and implementation plan for PyVault, a command-line personal knowledge manager written in Python. The aim of this project is to consolidate and reinforce core Python concepts such as functions, object-oriented programming, modules, file handling, and optionally concurrency, by applying them to a realistic, software-engineering style project.

PyVault is intended to be a foundational Python project that demonstrates clean code structure, modular design, and practical use of data modeling and persistence.

2. Project Overview

2.1 Project Name

PyVault — Personal Knowledge Manager (CLI)

2.2 High-Level Concept

PyVault is a terminal-based system for managing different types of knowledge entries, such as notes, tasks, and bookmarks. These entries are stored locally using either JSON files or an SQLite database.

Core features include:
• Adding, viewing, and searching entries
• Categories and tags
• Persistent storage
• CLI interface
• Optional concurrency for background indexing

3. Initial Project Structure

pyvault/
pyproject.toml
README.md
vault/
__init__.py
cli.py
models.py
storage.py
manager.py
utils.py
indexing.py
data/
vault.json

4. Milestones

4.1 Milestone 1 — Project Setup

Goal: Create the project structure and environment.
Outcome: Skeleton project ready.

4.2 Milestone 2 — Define the Data Model
Goal: Plan the classes you'll use to represent entries.
Outcome: In-memory data modeling complete.

4.3 Milestone 3 — Storage System Design
Goal: Design how data is saved and loaded.
Outcome: Clear storage architecture plan.

4.4 Milestone 4 — Manager Layer
Goal: Design the core logic interacting with storage and models.
Outcome: All operations planned.

4.5 Milestone 5 — CLI System
Goal: Design user interaction through CLI.
Outcome: Full CLI command spec.

4.6 Milestone 6 — Formatting & Presentation
Goal: Decide terminal output formatting.
Outcome: UX and styling decisions made.

4.7 Milestone 7 — Optional Concurrency
Goal: Add background indexing or auto-save.
Outcome: Advanced concurrency plan.

4.8 Milestone 8 — Documentation
Goal: Make PyVault professional.
Outcome: GitHub-ready documentation.

5. Expected Deliverables (Design Phase Only)
• README.md with project specification
• Project folder structure
• pyproject.toml metadata
• Written design notes
• Implementation order plan

6. Next Steps
1. Finalise data model
2. Confirm storage backend
3. Begin implementation