Project Report
Computer Science

TITLE: Note Manager/ To-Do-List

Submitted By:
Name: _AFIYA BEGUM__
Registration Number : 25BCE11165

Table of Contents:

Introduction

The Notes Manager is an easy-to-use console application created in Python for students to keep their notes and tasks organized. You will develop some core concepts in programming such as functions, classes, file handling, and storing data in JSON. Users can create notes, edit notes, delete notes that are not useful, search through their notes, and export their notes for storage. The project demonstrates how simple terminal-based programs can solve common problems people face during their everyday lives while also providing a basis for software development in the future.

Objectives

- Provide CRUD operations for notes.
- Allow categories/tags and simple prioritization.
- Enable search/filtering by title, content, category, priority, date.
- Persist notes in a local JSON file.
- Provide export/backup (JSON/CSV).
- Keep UI simple (console menu) and code modular for maintainability

System Requirements

Hardware:

- **Processor:**
  Any dual-core processor (Intel/AMD) or higher
  (e.g., Intel Core i3 or equivalent)
- **RAM:**
  Minimum **2 GB RAM**
  Recommended: **4 GB or higher**
- **Storage:**
  Minimum **100 MB** of free disk space
  (The project files and JSON data require very little storage)
- **Display:**
  Any standard display capable of running a terminal/command prompt

- **Input Devices:**
  Keyboard (mandatory)
  Mouse (optional – since it's a console app)

Software:

- Operating System: Windows/Linux/MacOS
- Programming Language: Python
- Database: JSON

Design Diagrams

Use Case Diagram

- Actor: User
- Actions: Create, View, Edit, Delete, Search Notes

Workflow Diagram

```
Start → Show Menu → User Selects Option → Perform
Action → Save → Return to Menu → Exit
```

Sequence Diagram (Create Note)

```
User → Menu → create_note() → save_notes() → JSON
File
```

Class/Component Diagram

(Not classes, but components)

- Input Handler
- Notes Logic Module
- JSON Storage Module

ER Diagram

```
NOTE (id, title, content, category, priority,
created_at, updated_at)
```

Design Decisions & Rationale

- JSON chosen for simplicity and readability.
- Functions used instead of classes for beginner-friendliness.
- Console UI chosen to avoid GUI complexity.
- UUID used for unique note identification.

Implementation Details

- Written entirely in Python using basic functions.
- Uses `json` module for storage.
- Menu-driven interface using loops and conditionals.
- Each feature implemented as a separate function.

Testing Approach

- Manual testing of each menu option.
- Test with multiple notes.
- Check JSON file updates.
- Verify edit/delete functions.
- Test edge cases (empty input, invalid ID).

Challenges Faced

- Managing unique IDs for notes.
- Handling file reading/writing errors.
- Ensuring notes persist after program closes.
- Designing simple and clean menu

Learnings & Key Takeaways

- Working with functions and modular design.
- Understanding JSON file handling.
- Implementing CRUD operations in Python.
- Basic error handling and user input validation.
- Experience with real-world project structure.

Future Enhancements

- Add password protection.
- Add categories and tags.
- Add reminder notifications.
- Convert console app into GUI or web app.
- Add data export/import feature.

References

- Python Official Documentation
- W3Schools Python Tutorials
- GeeksforGeeks Python File Handling
- StackOverflow Discussion