Badilla, Francis Nichol F.
BSCPE-3

Engr. Julian N. Semblante
December 12, 2025

Project Overview

Purpose

The project is a **C# console-based Journal Management System** designed to help users specifically students to record, store, view, and analyze their daily emotional experiences inside or outside the campus. It uses **text-file handling** to ensure that all journal entries are permanently saved.

Features

This application supports:

- Adding new journal entries
- Viewing all entries
- Editing or deleting entries
- Generating **weekly** and **monthly summaries**
- Searching entries
- Parsing and saving **emotion**, **rating**, **reflection**, **emotional response**, and **response**

Target Audience

Students who want a simple digital tool for emotional tracking and journaling. We are targeting the Guidance Center in the University where it will be managed.

Technology Stack

- **Language:** C#
- **Framework:** .NET (Console Application)
- **Libraries:**
  - System
  - System.IO (file handling)
  - System.Linq (sorting, filtering)

Requirements

Software Requirements

- .NET SDK
- Any C# IDE (Visual Studio, VS Code, Rider)

Installation Steps

1. Clone or download the project folder.
2. Open the folder in Visual Studio.
3. Press **F5** to run.

System Requirements

- Any Windows/macOS laptop capable of running .NET.

File Handling Overview

File Types and Purpose

- The program uses **.txt** files to store all journal entries.
- Each line represents **one entry** formatted like:

Date|Mood|Rating|Category|Reflection|EmotionalResponse: X | Response: Y

File Operations Used

✔️ **ReadAllLines** – loading all entries
✔️ **WriteAllLines** – overwriting data after edit/delete
✔️ **AppendAllText** – adding a new entry
✔️ **File.Exists / Create** – checking or creating database file

Error Handling

- Try/catch blocks when reading/writing files
- Validation for empty inputs
- Handling missing fields (e.g., missing Response or EmotionResponse)

Code Structure

Main Program Structure

Classes:

Class	Purpose

Program	Entry point, displays the main menu
JournalManager	Handles file reading, writing, updating, deleting
JournalEntry	Represents a single journal entry object

Key Methods in JournalManager

Method	Description
AddEntry()	Adds a new journal entry and saves it to file
ViewEntries()	Displays entries and parses EmotionalResponse + Response
EditEntry()	Modify an existing entry
DeleteEntry()	Removes an entry from the file
WeeklySummary()	Shows last 7 days
MonthlySummary()	Shows last 30 days

Modularity and Reusability

- Parsing logic separated
- Data stored in reusable JournalEntry objects
- Sorting handled through **LINQ**

User Interface (Console)

Design and Usability

- Simple menu-based UI
- Input prompts guide the user clearly
- Table-style display for entries

Input/Output

Input: user types values for mood, rating, reflection, etc.
Output: formatted tables, summaries, and lists.

Error Messages

- Invalid date
- Invalid option
- Missing file
- Missing EmotionResponse or Response

Challenges and Solutions

Challenges Encountered

Parsing EmotionalResponse and Response correctly	Rewrote the parsing logic using Split() and fallback defaults
Response disappearing after restarting	Fixed formatting and ensured append logic includes Response
LINQ sorting errors	Added proper using System.Linq
Ensuring persistence across edits	Used WriteAllLines instead of partial writes

Testing

Test Cases

Test	Description	Result
Add Entry	Input full journal data	PASS
Missing Response	Checks fallback display	PASS
Edit Entry	Modify reflection only	PASS
Delete Entry	File updates properly	PASS
Weekly Summary	Last 7 days show correctly	PASS
Monthly Summary	Last 30 days parsed correctly	PASS

Limitations

- No JSON or database support
- No GUI yet
- No encryption, text file is readable

Future Enhancements

Planned Features

- Full GUI (WinForms/WPF)
- SQL Database / SQLite
- Export to PDF or CSV
- Cloud sync
- Emotion trend graphs

Conclusion

Reflection

This project demonstrated how to use **C# file handling** to build a real working journaling system. You implemented data persistence, parsing, summaries, and a fully functional menu interface.