Dart Final Project - Overview & Guidelines

Project Concept

For your final project, you will develop a **menu-driven application** that applies all the concepts you've learned throughout the Dart course. This project will simulate a **real-world system**, such as a **Library Management System, Student Management System, Inventory System, etc.**

When the Dart program runs:

1. The system will **display a menu** with different numbered options.
2. The user will **select an option** by entering a number.
3. Based on the selection, the program will **perform the corresponding action** (e.g., adding, listing, updating, or deleting data).
4. (Optional) Before accessing the menu, the system can include **user login/signup functionality**.

Project Requirements

- **User Interaction:** The program should take **input** from the user and display an appropriate **response/output**.
- **Data Management:**
    - You can store data **temporarily in a list** (clears when the program ends).
    - OR store data **persistently in a file** (e.g., JSON file) so it remains even after the program restarts.
- **OOP Principles:** Implement **classes, objects, encapsulation, inheritance, abstraction, and polymorphism** to structure your code efficiently.
- **Error Handling:** Validate user inputs and handle **edge cases** (e.g., trying to delete a book that doesn't exist).

Example: Library Management System

Menu Example

Welcome to the Library Management System
1. Add Book
2. List Books
3. Delete Book
4. Update Book
5. Assign Book to a User
6. Exit
Enter your choice:

Feature Breakdown

✅ **Add Book** → The program asks for book details (title, author, genre) and saves it in a list or file.
✅ **List Books** → Displays all books in the system.
✅ **Delete Book** → Removes a book from the system.
✅ **Update Book** → Allows modification of book details.
✅ **Assign Book** → Assigns a book to a user and updates availability status.
✅ **Exit** → Closes the program.

Project Submission Guidelines

- Your program should be **error-free and fully functional**.
- Submit the **Dart source code** with clear comments explaining the logic.
- If using file storage, submit the **JSON/text file** along with your code.
- (Optional) Add a **README file** explaining how to run your program.

Final Tips

- **Break down the project into smaller parts** and implement them step by step.
- **Test your program** with different inputs before submission.
- **Keep your code clean and well-structured** using proper function names and class structures.

💡 **Be creative!** You can choose **any system** (library, student management, store inventory, etc.). The goal is to **apply all your Dart knowledge** in a practical project! 🚀