Assignment 01: Enhanced Flashcard Application

Objective

Enhance the existing **Flashcard App** by introducing more **complex functionality** (e.g., additional screens, scoring, or input forms) to solidify your understanding of **stateless vs. stateful widgets**, **layouts**, and **basic state management** in Flutter.

Learning Outcomes (Mapped to CLOs)

CLO	Description
CLO-1	Apply Dart programming concepts to develop mobile application user interfaces.
CLO-2	Construct attractive front-end for mobile applications using Flutter.
CLO-4	Implement programs using Dart and Flutter for mobile applications.

Task Description

Building on your basic Flashcard App, **add new features** to make the user experience more dynamic and interactive. Below are some **options** (you may include multiple):

1. **Score Tracking (CLO-1, CLO-4)**
    - Each time the user reveals an answer and taps either “Correct” or “Incorrect,” update their score.
    - Display the score on the screen or in an overlay.
2. **Multiple Categories or Decks (CLO-1, CLO-4)**
    - Implement multiple flashcard decks (e.g., Flutter, Dart).
    - Use either multiple screens or a dropdown/side menu to let users switch between decks.
    - Demonstrate **basic navigation** to move from the home screen to a deck screen.
3. **User Created Flashcards (CLO-1, CLO-2, CLO-4)**
    - Add a form that allows the user to create a new question answer pair.
    - Store the new flashcards in your app’s current list (at least temporarily).
    - Consider basic form validation (e.g., no empty question or answer fields).
4. **Visual Enhancements (CLO-2)**
    - Improve the UI with advanced styling, custom themes, or animations (like a fancy flip animation).
    - Demonstrate usage of layout widgets (Row, Column, Stack, etc.) for more sophisticated designs.
5. **Timed Flashcards or Animated Transitions (CLO-4)**

- Implement a countdown timer that flips the card automatically after a specified time.
- Explore `AnimatedSwitcher`, `AnimationController`, or other animation widgets to make the flipping more eye-catching.

Technical Requirements

1. **State Management**: Use *stateful widgets* appropriately. Changes in score or flipping cards must refresh the UI via `setState` or another state management approach.
2. **Layouts**: Demonstrate knowledge of Flutter layout widgets (e.g., Row, Column, ListView, Stack) to create a user-friendly interface.
3. **Navigation** (If implementing multiple decks): Must use Flutter navigation (`Navigator.push`, `Navigator.pop`, or named routes) to switch between screens.
4. **Responsiveness**: The layout should be usable on different screen sizes (phones/tablets).

Deliverables

1. A **Flutter project** containing:
   - A main Dart file (`main.dart`) that configures the `MaterialApp`.
   - At least one additional Dart file for new screens or forms (if you implement multiscreen navigation).
   - Clear, well-structured code with **comments** explaining key sections.
2. A **short demonstration** (upload video and pictures on GitHub presentation) showing:
   - How the new features work.
   - How you handle state changes (score updates, flipping cards, data input, etc.).

Submission Guidelines

- **Due Date**: [Instructor to specify date]
- **Submission Format**:
  - A compressed folder containing your entire Flutter project, or a link to your Git repository.
  - Optionally, a short video/presentation file demonstrating the app's features in action.

Grading Rubric (Example)

Criteria	Marks	Description
Functionality	10%	- All required features (score, forms, multiple decks, etc.) implemented.
UI & UX (CLO-2)	10%	- Use of attractive layouts, themes, clear text, proper spacing, smooth navigation/animations if included.
Code Quality (CLO-1, CLO-4)	20%	- Code organization, naming conventions, meaningful comments, and best practices (e.g., minimal redundancy).
Presentation & GitHub	60%	- Clear demonstration of features, confident explanation of code and state management.

Conclusion

By completing this assignment, students will **reinforce** their understanding of:

- **CLO-1**: Writing clean, Dart-based UI logic.
- **CLO-2**: Crafting appealing, user-friendly Flutter interfaces.
- **CLO-4**: Effectively applying stateless and stateful widgets to manage app state and interactions.

Encourage creativity in UI design and additional features **have fun** while exploring how simple concepts (like showing a question/answer) can expand into a more **robust, real-world** application!