Name:	Haseeb Ur Rehman
SAP:	55859
Section:	5-2
Subject:	Mobile Application Development

QUIZ – 3

```dart
import 'package:flutter/material.dart';

void main() => runApp(const SmartFlashApp());

class SmartFlashApp extends StatelessWidget {
  const SmartFlashApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Smart Flashcards',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorSchemeSeed: Colors.cyan,
        useMaterial3: true,
      ),
      home: const FlashHomeScreen(),
    );
  }
}

class Flashcard {
  final String id;
  final String question;
  final String answer;
  bool isLearned;

  Flashcard({
    required this.id,
    required this.question,
    required this.answer,
    this.isLearned = false,
  });
}

class FlashHomeScreen extends StatefulWidget {
  const FlashHomeScreen({super.key});

  @override
  State<FlashHomeScreen> createState() => _FlashHomeScreenState();
}

class _FlashHomeScreenState extends State<FlashHomeScreen> {
```

```
  final GlobalKey<AnimatedListState> _listKey =
GlobalKey<AnimatedListState>();
  List<Flashcard> _flashcards = [];
  int _newQuestionNumber = 1;

  @override
  void initState() {
    super.initState();
    _populateCards();
  }

  void _populateCards() {
    _flashcards = [
      Flashcard(
        id: '1',
        question: 'What is Flutter?',
        answer:
            'A UI toolkit from Google for building fast, cross-platform apps
with one codebase.',
      ),
      Flashcard(
        id: '2',
        question: 'Which programming language does Flutter use?',
        answer: 'It uses Dart, developed by Google.',
      ),
      Flashcard(
        id: '3',
        question: 'What is a StatelessWidget?',
        answer:
            'A widget that does not depend on mutable state — it's rebuilt
only when external data changes.',
      ),
      Flashcard(
        id: '4',
        question: 'What does the build() method do?',
        answer:
            'It describes how to display the widget in terms of other lower-
level widgets.',
      ),
      Flashcard(
        id: '5',
        question: 'What does Hot Reload do?',
        answer:
```

```
            'It lets developers instantly see code changes in the running app
without restarting it.',
      ),
    ];
    _newQuestionNumber = _flashcards.length + 1;
  }

  int get _learnedCount => _flashcards.where((f) => f.isLearned).length;

  Future<void> _onRefresh() async {
    await Future.delayed(const Duration(milliseconds: 900));
    setState(() {
      _populateCards();
    });
  }

  void _markAsLearned(int index) {
    final card = _flashcards[index];
    setState(() {
      card.isLearned = true;
    });

    _listKey.currentState?.removeItem(
      index,
      (context, animation) => _buildCardTile(card, animation, index),
      duration: const Duration(milliseconds: 400),
    );

    setState(() {
      _flashcards.removeAt(index);
    });

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text('Great job! Card marked as learned 🎉'),
        duration: Duration(seconds: 1),
        behavior: SnackBarBehavior.floating,
      ),
    );
  }

  void _addNewCard() {
    final newCard = Flashcard(
```

```
      id: DateTime.now().millisecondsSinceEpoch.toString(),
      question: 'New Question #$_newQuestionNumber',
      answer: 'Answer for question #$_newQuestionNumber. Add details here.',
    );

    setState(() {
      _flashcards.insert(0, newCard);
      _newQuestionNumber++;
    });
    _listKey.currentState?.insertItem(0);

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text('New card added! ✨'),
        duration: Duration(seconds: 1),
        behavior: SnackBarBehavior.floating,
      ),
    );
  }

  Widget _buildCardTile(Flashcard card, Animation<double> animation, int
index) {
    return SlideTransition(
      position: animation.drive(
        Tween(begin: const Offset(1, 0), end: Offset.zero)
            .chain(CurveTween(curve: Curves.easeOutBack)),
      ),
      child: Dismissible(
        key: Key(card.id),
        direction: DismissDirection.endToStart,
        onDismissed: (_) => _markAsLearned(index),
        background: Container(
          alignment: Alignment.centerRight,
          margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
          padding: const EdgeInsets.only(right: 20),
          decoration: BoxDecoration(
            color: Colors.orange,
            borderRadius: BorderRadius.circular(16),
          ),
          child: const Icon(Icons.check_circle_outline,
              color: Colors.white, size: 32),
        ),
        child: FlashcardTile(card: card),
```

```
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: RefreshIndicator(
      onRefresh: _onRefresh,
      color: Colors.cyan,
      child: CustomScrollView(
        slivers: [
          SliverAppBar(
            expandedHeight: 150,
            pinned: true,
            backgroundColor: Colors.cyan,
            flexibleSpace: FlexibleSpaceBar(
              title: Text(
                'Learned: $_learnedCount',
                style: const TextStyle(
                  fontWeight: FontWeight.w600,
                  fontSize: 18,
                ),
              ),
              background: Container(
                decoration: BoxDecoration(
                  gradient: LinearGradient(
                    colors: [Colors.cyan, Colors.teal.shade300],
                    begin: Alignment.topLeft,
                    end: Alignment.bottomRight,
                  ),
                ),
                child: Center(
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      const Icon(Icons.flash_on,
                          size: 44, color: Colors.white70),
                      const SizedBox(height: 8),
                      Text(
                        '${_flashcards.length} cards remaining',
                        style: const TextStyle(
                          color: Colors.white70,
```

```
                          fontSize: 14,
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            ),
          ),
          SliverToBoxAdapter(
            child: Padding(
              padding: const EdgeInsets.symmetric(vertical: 14),
              child: Text(
                '👉 Swipe left to mark as learned',
                textAlign: TextAlign.center,
                style: TextStyle(
                  color: Colors.grey.shade700,
                  fontStyle: FontStyle.italic,
                ),
              ),
            ),
          ),
          SliverPadding(
            padding: const EdgeInsets.only(bottom: 80),
            sliver: AnimatedList(
              key: _listKey,
              initialItemCount: _flashcards.length,
              itemBuilder: (context, index, animation) {
                return _buildCardTile(_flashcards[index], animation,
index);
              },
            ),
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton.extended(
      onPressed: _addNewCard,
      backgroundColor: Colors.cyan,
      icon: const Icon(Icons.add),
      label: const Text('Add Card'),
    ),
  );
```

```dart
  }
}

class FlashcardTile extends StatefulWidget {
  final Flashcard card;

  const FlashcardTile({super.key, required this.card});

  @override
  State<FlashcardTile> createState() => _FlashcardTileState();
}

class _FlashcardTileState extends State<FlashcardTile> {
  bool _showAnswer = false;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => setState(() => _showAnswer = !_showAnswer),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 350),
        curve: Curves.easeInOut,
        margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
        padding: const EdgeInsets.all(20),
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: _showAnswer
                ? [Colors.orange.shade400, Colors.orange.shade600]
                : [Colors.cyan.shade400, Colors.cyan.shade700],
            begin: Alignment.topLeft,
            end: Alignment.bottomRight,
          ),
          borderRadius: BorderRadius.circular(16),
          boxShadow: [
            BoxShadow(
              color: Colors.black.withOpacity(0.12),
              blurRadius: 8,
              offset: const Offset(0, 4),
            ),
          ],
        ),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
```

```
          children: [
            Row(
              children: [
                Icon(
                  _showAnswer ? Icons.fact_check : Icons.help,
                  color: Colors.white,
                  size: 26,
                ),
                const SizedBox(width: 10),
                Text(
                  _showAnswer ? 'Answer' : 'Question',
                  style: const TextStyle(
                    color: Colors.white70,
                    fontSize: 14,
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ],
            ),
            const SizedBox(height: 16),
            AnimatedCrossFade(
              duration: const Duration(milliseconds: 300),
              firstChild: Text(
                widget.card.question,
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                  height: 1.4,
                ),
              ),
              secondChild: Text(
                widget.card.answer,
                style: const TextStyle(
                  color: Colors.white,
                  fontSize: 16,
                  height: 1.5,
                ),
              ),
              crossFadeState: _showAnswer
                  ? CrossFadeState.showSecond
                  : CrossFadeState.showFirst,
            ),
```

```
            const SizedBox(height: 10),
            Text(
              _showAnswer ? 'Tap to see question' : 'Tap to reveal answer',
              style: const TextStyle(
                color: Colors.white60,
                fontSize: 12,
                fontStyle: FontStyle.italic,
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```