Flutter Mini Project — Student Cards App

A complete Flutter app: student data, a reusable card widget, a scrolling list and tap handling — the series finale.

Introduction

The Dart series ended with a console project (D.11). This finale turns the same idea into a **real phone app**: a Student Cards App that shows five students as designed cards in a scrolling list — and responds when you tap one. It uses everything: data as a list of maps (D.7), a stateless card widget (F.3), Row layout inside each card (F.4), and a stateful page that remembers your selection (F.3 again). Build it in the same three steps every real app follows: **data** → **widget** → **screen**. *(Roman Urdu: Data, phir card, phir poori screen — asli app isi tartib se banti hai.)*

Key points

- **Step 1 — data first.** The students are a `List` of `Map`s (D.7's shape): each map one student, with `name`, `className` and `marks`. Screens are temporary; data shape is the foundation. Changing a name in this list changes the app everywhere — one source of truth.
- **Step 2 — one card widget.** `StudentCard` is a **StatelessWidget** (F.3's decision question: does the card itself change? No — it just shows what it is given). It receives its student through named constructor parameters and draws a `Card` with a `ListTile`: avatar letter, name, class, and the marks in gold on the right.

  (Roman Urdu: Aik card design karo, phir wohi design har student ke liye khud chal jata hai.)
- **Step 3 — the list builds itself.** `ListView.builder` takes `itemCount` and an `itemBuilder` that returns the card for each index. Five students or five hundred — the code is the same, and the list scrolls. Writing five separate cards by hand is the beginner way; builders are the app way.
- **Taps are callbacks.** The card's `onTap` is a function handed down from the page (functions are values — D.6). When a card is tapped, the **page** — a StatefulWidget — calls `setState`, shows a banner with the selected name, and pops a `SnackBar` message at the bottom.
- Notice what did *not* happen: the card never learned about banners or snackbars, and the page never learned how a card is drawn. Each piece has one job. That separation is what lets apps grow past five screens without collapsing.

Code examples

Example 1 — Step 1: the data (students)

```
final List<Map<String, Object>> students = [
  {'name': 'Ahmed', 'className': 'BS CS - 5th semester', 'marks': 92},
  {'name': 'Sara', 'className': 'BS CS - 5th semester', 'marks': 85},
  {'name': 'Bilal', 'className': 'BS CS - 3rd semester', 'marks': 78},
  {'name': 'Zara', 'className': 'BS CS - 7th semester', 'marks': 95},
  {'name': 'Usman', 'className': 'BS CS - 3rd semester', 'marks': 66},
];
```

What it does: Five maps in one list — the D.11 project's data, promoted from console variables to app data. In a real project this list lives in its own file (say `students.dart`) and the screens import it. Values are read back with a cast: `s['name'] as String`, because a Map's values are typed broadly.

What you will see: nothing yet — data alone draws nothing, exactly like the Dart project's Steps 1–2. Hold on to it; Steps 2 and 3 turn it into a screen.

Example 2 — Step 2: the StudentCard widget

```
class StudentCard extends StatelessWidget {
  final String name;
  final String className;
  final int marks;
  final VoidCallback onTap;

  const StudentCard({
    super.key,
    required this.name,
    required this.className,
    required this.marks,
    required this.onTap,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.symmetric(horizontal: 12, vertical: 6),
      child: ListTile(
        leading: CircleAvatar(child: Text(name[0])),
        title: Text(name),
        subtitle: Text(className),
        trailing: Text(
          '$marks',
          style: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.bold,
            color: Colors.amber.shade800,
          ),
        ),
        onTap: onTap,
      ),
    );
  }
}
```

What it does: One reusable card. The four `final` fields are its settings; `required` means no card can be built without them. `ListTile` is Flutter's ready-made row: `leading` (the avatar circle showing the name's first letter — `name[0]`, D.7's index), `title`, `subtitle`, and `trailing` (the gold marks). The whole tile is tappable through `onTap`.

What you will see: still nothing alone — but you now own the brick. Five copies of this brick, each fed a different student, are the app. Designing the brick *before* the wall is the professional habit this project exists to teach.

Example 3 — Step 3: the full app — list, tap, banner

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

// Step 1 ki students list aur Step 2 ka StudentCard
// isi file mein, is hisse se upar rakhein.

void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Student Cards',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.teal),
      ),
      home: const StudentsPage(),
    );
  }
}

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

  @override
  State<StudentsPage> createState() => _StudentsPageState();
}

class _StudentsPageState extends State<StudentsPage> {
  String? selectedName;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Student Cards')),
      body: Column(
        children: [
          if (selectedName != null)
            Container(
              width: double.infinity,
```

```
              padding: const EdgeInsets.all(12),
              color: Colors.teal.shade50,
              child: Text('Selected: $selectedName'),
            ),
          Expanded(
            child: ListView.builder(
              itemCount: students.length,
              itemBuilder: (context, index) {
                final s = students[index];
                return StudentCard(
                  name: s['name'] as String,
                  className: s['className'] as String,
                  marks: s['marks'] as int,
                  onTap: () {
                    setState(() {
                      selectedName = s['name'] as String;
                    });
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text('Selected: ${s['name']}')),
                    );
                  },
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

What it does: The page is stateful because one thing changes — `selectedName` (a `String?`, honestly nullable: at the start, nobody is selected — D.9 in production). The `if (selectedName != null)` line shows the banner *only* after a selection exists. `ListView.builder` feeds each map into a `StudentCard`; the tap handler updates the state and shows the SnackBar. The five layers — MaterialApp → Scaffold → Column → ListView → StudentCard — are F.1's widget tree, grown up.

What you will see: a "Student Cards" title bar and five cards, one per student, each with an initial circle, name, class line and gold marks. Scroll — the list moves. Tap Ahmed's card: a pale banner appears at the top reading "Selected: Ahmed", and a dark SnackBar slides up at the bottom with the same message. Tap Sara: both update. You have built a complete, working Flutter app — data, design, list and interaction — from the language you started in Lesson D.1.

Quick recap

- Real apps are built in steps: data → reusable widget → screen that assembles them.
- `ListView.builder` turns any list into a scrolling list of widgets — the count can grow, the code does not.
- The card is stateless (shows), the page is stateful (remembers) — state lives where the change lives.
- Taps travel upward as callbacks: the widget reports, the page decides.

Common mistakes

1. **Writing each card by hand five times** — it works for five students and collapses at fifty. If you catch yourself copying a widget and changing only the data, stop: that is a builder asking to be born.
2. **Putting the selection state inside the card** — then every card remembers its own selection and the page cannot show one banner. State belongs to the *smallest piece that needs it* — here, the page that owns the banner.
3. **Forgetting the cast when reading a Map** — `s['name']` from a `Map<String, Object>` is an `Object`, and the card wants a `String`. The `as String` cast is not decoration; without it, Dart's type system (D.3) rightly refuses to run.

Practice

1. Add a sixth student to the data list — no other change — and run the app. The new card must simply appear. That feeling is why builders exist.
2. Grow the app: give each map a fourth key, `'grade'`, show it under the marks in the card (hint: make the trailing widget a small Column), and change the banner to show the selected student's grade as well.

Practice quiz

1. What does ListView.builder take, and why is it the app way?

- Five hand-written cards
- itemCount and an itemBuilder that returns the card for each index, so 5 or 500 students need the same code, and the list scrolls
- Only a list of names, no widgets
- A database connection

Answer: itemCount and an itemBuilder that returns the card for each index, so 5 or 500 students need the same code, and the list scrolls

2. Why is StudentCard a StatelessWidget?

- Because it is small
- Because the card itself never changes, it just shows what it is given
- Because stateless widgets are faster to type

- Because it has no constructor

Answer: Because the card itself never changes, it just shows what it is given

3. When a student card is tapped, what happens?

- The card itself changes its own state
- The tap travels up as a callback to the page, which calls setState, shows a banner with the selected name, and pops a SnackBar
- Nothing, taps are ignored
- The app restarts

Answer: The tap travels up as a callback to the page, which calls setState, shows a banner with the selected name, and pops a SnackBar

---

Watch the video lesson: https://youtu.be/ZKz50SVWa74

Read online: https://scopetreasure.com/notes/flutter-mini-project-student-cards · More free tools and notes at https://scopetreasure.com