```
name: dopamine_detox_coach
description: AI-powered Dopamine Detox Coach - MVP
publish_to: "none"
version: 0.1.0+1
environment:
  sdk: ">=2.19.0 <4.0.0"

dependencies:
  flutter:
    sdk: flutter

  # Firebase
  firebase_core: ^2.10.0
  firebase_auth: ^4.4.0
  cloud_firestore: ^4.6.0

  # Local DB
  sqflite: ^2.2.8
  path_provider: ^2.0.14
  path: ^1.8.3

  # Ads
  google_mobile_ads: ^3.2.0

  # State management & UI
  provider: ^6.0.5
  intl: ^0.18.0
  uuid: ^3.0.7
```

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true
```

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:firebase_core/firebase_core.dart';
import 'services/firebase_service.dart';
import 'services/sqlite_service.dart';
import 'services/admob_service.dart';
import 'screens/coach_screen.dart';
import 'screens/tasks_screen.dart';
import 'screens/progress_screen.dart';
import 'models/task.dart';
import 'package:google_mobile_ads/google_mobile_ads.dart';

// TODO: add your generated firebase_options.dart via FlutterFire CLI.
import 'firebase_options.dart';

void main() async {
```

```
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
 await AdMobService.initialize();

 final sqliteService = await SQLiteService.getInstance();
 runApp(MyApp(sqliteService: sqliteService));
}

class MyApp extends StatelessWidget {
 final SQLiteService sqliteService;
 const MyApp({super.key, required this.sqliteService});

 @override
 Widget build(BuildContext context) {
  return MultiProvider(
   providers: [
    Provider<SQLiteService>.value(value: sqliteService),
    ChangeNotifierProvider<FirebaseService>(create: (_) => FirebaseService()),
   ],
   child: MaterialApp(
    debugShowCheckedModeBanner: false,
    title: 'Dopamine Detox Coach',
    theme: ThemeData(
     scaffoldBackgroundColor: const Color(0xFFF6F7F9),
     primaryColor: const Color(0xFF2B6CB0),
     colorScheme: ColorScheme.fromSwatch().copyWith(secondary: const
Color(0xFF9FB3C8)),
     textTheme: const TextTheme(bodyLarge: TextStyle(color: Colors.black87)),
```

```
      ),
      home: const MainShell(),
    ),
   );
 }
}

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

 @override
 State<MainShell> createState() => _MainShellState();
}

class _MainShellState extends State<MainShell> {
 int _index = 0;
 final _pages = [
  const CoachScreen(),
  const TasksScreen(),
  const ProgressScreen(),
 ];

 @override
 void dispose() {
  // Ensure Ads are disposed inside AdMobService if needed.
  super.dispose();
 }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: _pages[_index],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _index,
        onTap: (i) => setState(() => _index = i),
        selectedItemColor: const Color(0xFF2B6CB0),
        unselectedItemColor: Colors.grey,
        items: const [
          BottomNavigationBarItem(icon: Icon(Icons.support_agent), label: 'Coach'),
          BottomNavigationBarItem(icon: Icon(Icons.checklist), label: 'Tasks'),
          BottomNavigationBarItem(icon: Icon(Icons.show_chart), label: 'Progress'),
        ],
      ),
    );
  }
}
```

```
class DetoxTask {
  String id;
  String title;
  String timeOfDay; // morning/afternoon/night
```

```
String description;
bool completed;
int durationMinutes;

DetoxTask({
  required this.id,
  required this.title,
  required this.timeOfDay,
  required this.description,
  this.completed = false,
  this.durationMinutes = 5,
});

factory DetoxTask.fromMap(Map<String, dynamic> m) => DetoxTask(
    id: m['id'],
    title: m['title'],
    timeOfDay: m['timeOfDay'],
    description: m['description'],
    completed: m['completed'] == 1,
    durationMinutes: m['durationMinutes'] ?? 5,
   );

Map<String, dynamic> toMap() => {
    'id': id,
    'title': title,
    'timeOfDay': timeOfDay,
    'description': description,
    'completed': completed ? 1 : 0,
```

```
      'durationMinutes': durationMinutes,
    };
}
```

```
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import '../models/task.dart';

class SQLiteService {
 static SQLiteService? _instance;
 static Database? _db;

 SQLiteService._();

 static Future<SQLiteService> getInstance() async {
  if (_instance != null) return _instance!;
  instance = SQLiteService.();
  await _instance!._init();
  return _instance!;
 }

 Future<void> _init() async {
  final path = join(await getDatabasesPath(), 'detox.db');
  _db = await openDatabase(path, version: 1, onCreate: (db, v) async {
   await db.execute('''
    CREATE TABLE tasks (
     id TEXT PRIMARY KEY,
```

```
      title TEXT,
      timeOfDay TEXT,
      description TEXT,
      completed INTEGER,
      durationMinutes INTEGER
    );
  ''');

  await db.execute('''
    CREATE TABLE streaks (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      date TEXT,
      streakCount INTEGER
    );
  ''');
 });
}

Future<void> insertTask(DetoxTask t) async {
 await _db!.insert('tasks', t.toMap(), conflictAlgorithm: ConflictAlgorithm.replace);
}

Future<List<DetoxTask>> getAllTasks() async {
 final rows = await _db!.query('tasks');
 return rows.map((r) => DetoxTask.fromMap(r)).toList();
}

Future<void> updateTask(DetoxTask t) async {
```

```
  await _db!.update('tasks', t.toMap(), where: 'id = ?', whereArgs: [t.id]);
 }

 Future<void> deleteTask(String id) async {
  await _db!.delete('tasks', where: 'id = ?', whereArgs: [id]);
 }

 // Simple streak functions
 Future<void> addStreak(String date, int count) async {
  await _db!.insert('streaks', {'date': date, 'streakCount': count});
 }

 Future<int> getTotalStreak() async {
  final rows = await _db!.query('streaks');
  if (rows.isEmpty) return 0;
  return rows.length; // simplistic
 }
}
```

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import '../models/task.dart';

class FirebaseService extends ChangeNotifier {
 final FirebaseAuth _auth = FirebaseAuth.instance;
```

```
final FirebaseFirestore _fs = FirebaseFirestore.instance;
User? get currentUser => _auth.currentUser;

FirebaseService() {
  auth.authStateChanges().listen(() => notifyListeners());
}

Future<UserCredential> signInAnonymously() async {
  final cred = await _auth.signInAnonymously();
  return cred;
}

Future<void> uploadTask(DetoxTask t) async {
  if (currentUser == null) return;
  final ref = _fs.collection('users').doc(currentUser!.uid).collection('tasks').doc(t.id);
  await ref.set(t.toMap());
}

Future<List<DetoxTask>> fetchRemoteTasks() async {
  if (currentUser == null) return [];
  final snap = await _fs.collection('users').doc(currentUser!.uid).collection('tasks').get();
  return snap.docs.map((d) => DetoxTask.fromMap(d.data())).toList();
}

// simple analytics event
Future<void> logEvent(String name, Map<String, dynamic> params) async {
  // Firestore doesn't do analytics - but you may use Firebase Analytics package.
  // Here we store events optionally
```

```
  if (currentUser == null) return;
  await _fs.collection('users').doc(currentUser!.uid).collection('events').add({
    'name': name,
    'params': params,
    'ts': FieldValue.serverTimestamp(),
  });
 }
}
```

```
import 'package:google_mobile_ads/google_mobile_ads.dart';

class AdMobService {
 static Future<void> initialize() async {
  await MobileAds.instance.initialize();
 }

 // Use test ad unit ids for development, replace with real ones on release.
 static String get bannerAdUnitId => 'ca-app-pub-3940256099942544/6300978111';
 static String get interstitialAdUnitId => 'ca-app-pub-3940256099942544/1033173712';
 static String get rewardedAdUnitId => 'ca-app-pub-3940256099942544/5224354917';

 static BannerAd createBannerAd() {
  return BannerAd(
   adUnitId: bannerAdUnitId,
   size: AdSize.banner,
   request: const AdRequest(),
   listener: BannerAdListener(
    onAdLoaded: (ad) => debugPrint('Banner loaded'),
    onAdFailedToLoad: (ad, err) {
```

```
      debugPrint('Banner failed: $err');
      ad.dispose();
    },
   ),
  )..load();
}

static Future<InterstitialAd?> loadInterstitial() {
  return InterstitialAd.load(
   adUnitId: interstitialAdUnitId,
   request: const AdRequest(),
   adLoadCallback: InterstitialAdLoadCallback(
    onAdLoaded: (ad) => debugPrint('Interstitial loaded'),
    onAdFailedToLoad: (err) {
     debugPrint('Interstitial failed: $err');
    },
   ),
  );
}

static Future<RewardedAd?> loadRewardedAd() {
  final completer = Completer<RewardedAd?>();
  RewardedAd.load(
   adUnitId: rewardedAdUnitId,
   request: const AdRequest(),
   rewardedAdLoadCallback: RewardedAdLoadCallback(
    onAdLoaded: (ad) => completer.complete(ad),
    onAdFailedToLoad: (err) {
```

```
        debugPrint('Rewarded failed: $err');
        completer.complete(null);
      },
    ),
  );
  return completer.future;
 }
}
```

```
import 'dart:math';

/// Dummy rule-based AI coach for MVP. Expandable to call your backend / LLM later.
class AiCoach {
 static List<String> morningTips = [
  "Start with a 5-minute journaling session — write three things you're grateful for.",
  "Try a short walk outside. Natural light helps reset your focus.",
  "Avoid social apps for the first 30 minutes after waking to keep dopamine low."
 ];

 static List<String> focusTips = [
  "Set a 25-minute focus timer and take a 5-minute break after.",
  "Put your phone in another room to reduce temptation.",
  "Use a small reward after finishing a session: a cup of tea, a 2-minute stretch."
 ];

 static List<String> nightTips = [
  "Switch to low-stimulation activities like reading or breathing exercises.",
  "Turn off push notifications after 9 PM for a calmer wind-down.",
  "Try a 10-minute gratitude journal to reduce bedtime rumination."
```

```
  ];

  static String getTip(String context) {
    final r = Random();
    switch (context) {
      case 'morning':
        return morningTips[r.nextInt(morningTips.length)];
      case 'focus':
        return focusTips[r.nextInt(focusTips.length)];
      case 'night':
        return nightTips[r.nextInt(nightTips.length)];
      default:
        final pool = [...morningTips, ...focusTips, ...nightTips];
        return pool[r.nextInt(pool.length)];
    }
  }

  // Simple rule-based response to user messages
  static String respondTo(String userMessage) {
    final msg = userMessage.toLowerCase();
    if (msg.contains('help') || msg.contains('can't') || msg.contains('struggle')) {
      return "I hear you. Try a short 5-minute break and a breathing exercise. Want me to
set a 5-min focus timer?";
    }
    if (msg.contains('focus') || msg.contains('timer')) {
      return "Let's do a 25-minute Pomodoro. Ready?";
    }
    return "Nice! Keep it simple: small wins add up. Want a quick tip?";
```

```
  }
}
```

```
import 'package:flutter/material.dart';
import '../services/ai_coach.dart';
import '../services/admob_service.dart';
import 'focus_mode_screen.dart';
import 'urge_tracker_screen.dart';
import 'package:google_mobile_ads/google_mobile_ads.dart';

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

  @override
  State<CoachScreen> createState() => _CoachScreenState();
}

class _CoachScreenState extends State<CoachScreen> {
  String _tip = AiCoach.getTip('morning');
  BannerAd? _banner;

  @override
  void initState() {
    super.initState();
    _banner = AdMobService.createBannerAd()..load();
  }

  @override
  void dispose() {
```

```
  _banner?.dispose();
  super.dispose();
 }

 void _refreshTip() {
  setState(() {
   _tip = AiCoach.getTip('focus');
  });
 }

 @override
 Widget build(BuildContext context) {
  final adWidget = _banner != null
     ? SizedBox(
       height: 50,
       child: AdWidget(ad: _banner!),
      )
     : const SizedBox.shrink();

  return SafeArea(
   child: Column(
    children: [
     Padding(
       padding: const EdgeInsets.all(18.0),
       child: Row(
        children: [
          CircleAvatar(radius: 28, backgroundColor: const Color(0xFF2B6CB0), child:
const Icon(Icons.self_improvement, color: Colors.white)),
```

```
          const SizedBox(width: 12),
          Expanded(
           child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: const [Text('Detox Coach', style: TextStyle(fontSize: 18, fontWeight: FontWeight.w600)), SizedBox(height: 4), Text('Your gentle guide', style: TextStyle(color: Colors.black54))],
           ),
          ),
          IconButton(onPressed: _refreshTip, icon: const Icon(Icons.refresh))
         ],
        ),
       ),
       Card(
        margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
        child: Padding(
         padding: const EdgeInsets.all(16.0),
         child: Row(children: [
          const Icon(Icons.lightbulb_outline, size: 36, color: Color(0xFF2B6CB0)),
          const SizedBox(width: 12),
          Expanded(child: Text(_tip, style: const TextStyle(fontSize: 16))),
         ]),
        ),
       ),
       Padding(
        padding: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 10),
        child: Row(children: [
         Expanded(
```

```
            child: ElevatedButton.icon(
              onPressed: () => Navigator.push(context, MaterialPageRoute(builder: (_) => const FocusModeScreen())),
              icon: const Icon(Icons.timer),
              label: const Text('Start Focus Mode'),
              style: ElevatedButton.styleFrom(backgroundColor: const Color(0xFF2B6CB0)),
            ),
          ),
          const SizedBox(width: 12),
          ElevatedButton(
            onPressed: () => Navigator.push(context, MaterialPageRoute(builder: (_) => const UrgeTrackerScreen())),
            child: const Icon(Icons.favorite),
            style: ElevatedButton.styleFrom(backgroundColor: const Color(0xFF9FB3C8)),
          )
        ]),
      ),
      const SizedBox(height: 12),
      const Text('Quick prompts', style: TextStyle(fontWeight: FontWeight.w600)),
      const SizedBox(height: 6),
      Wrap(
        spacing: 8,
        children: [
          ActionChip(label: const Text('5-min reset'), onPressed: () => setState(() => _tip = AiCoach.getTip('focus'))),
          ActionChip(label: const Text('Morning routine'), onPressed: () => setState(() => _tip = AiCoach.getTip('morning'))),
          ActionChip(label: const Text('Wind down'), onPressed: () => setState(() => _tip = AiCoach.getTip('night'))),
```

```
          ],
        ),
        const Spacer(),
        adWidget,
      ],
    ),
  );
 }
}
```

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:uuid/uuid.dart';
import '../models/task.dart';
import '../services/sqlite_service.dart';
import '../widgets/task_card.dart';

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

 @override
 State<TasksScreen> createState() => _TasksScreenState();
}

class _TasksScreenState extends State<TasksScreen> {
 List<DetoxTask> tasks = [];
 bool loading = true;

 @override
```

```
 void didChangeDependencies() {
  super.didChangeDependencies();
  _loadTasks();
 }

 Future<void> _loadTasks() async {
  final sqlite = Provider.of<SQLiteService>(context, listen: false);
  final rows = await sqlite.getAllTasks();
  if (rows.isEmpty) {
   // seed default tasks
   await _seedDefaultTasks(sqlite);
  }
  final r2 = await sqlite.getAllTasks();
  setState(() {
   tasks = r2;
   loading = false;
  });
 }

 Future<void> _seedDefaultTasks(SQLiteService sqlite) async {
  final u = Uuid();
  final seeds = [
   DetoxTask(id: u.v4(), title: '5-min journaling', timeOfDay: 'morning', description: 'Write 3 things you are grateful for'),
   DetoxTask(id: u.v4(), title: 'No social 30 mins', timeOfDay: 'afternoon', description: 'Avoid social apps after lunch'),
   DetoxTask(id: u.v4(), title: 'Wind down 15 mins', timeOfDay: 'night', description: 'Read a book, no screens'),
  ];
```

```
  for (var s in seeds) await sqlite.insertTask(s);
 }

 Future<void> _toggleComplete(DetoxTask t) async {
  final sqlite = Provider.of<SQLiteService>(context, listen: false);
  t.completed = !t.completed;
  await sqlite.updateTask(t);
  _loadTasks();
 }

 Future<void> _addTask() async {
  final sqlite = Provider.of<SQLiteService>(context, listen: false);
  final u = Uuid();
  final newTask = DetoxTask(id: u.v4(), title: 'Quick 5-min', timeOfDay: 'afternoon',
description: 'Small reset', durationMinutes: 5);
  await sqlite.insertTask(newTask);
  _loadTasks();
 }

 @override
 Widget build(BuildContext context) {
  return SafeArea(
   child: loading
     ? const Center(child: CircularProgressIndicator())
     : Column(
       children: [
        Padding(
         padding: const EdgeInsets.all(16.0),
```

```
  child: Row(children: const [
   Text('Detox Tasks', style: TextStyle(fontSize: 20, fontWeight: FontWeight.w600)),
   Spacer(),
  ]),
),
Expanded(
   child: ListView.builder(
     itemCount: tasks.length,
     itemBuilder: (context, i) {
      final t = tasks[i];
      return TaskCard(
       task: t,
       onToggle: () => _toggleComplete(t),
       onDelete: () async {
        final sqlite = Provider.of<SQLiteService>(context, listen: false);
        await sqlite.deleteTask(t.id);
        _loadTasks();
       },
      );
     })),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 12),
  child: Row(
   children: [
    Expanded(
     child: ElevatedButton.icon(
      onPressed: _addTask,
      icon: const Icon(Icons.add),
```

```
                label: const Text('Add Quick Task'),
              ),
            ),
          ],
        ),
      )
    ],
  ),
 );
}
}
```

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

class TaskCard extends StatelessWidget {
 final DetoxTask task;
 final VoidCallback onToggle;
 final VoidCallback onDelete;
 const TaskCard({super.key, required this.task, required this.onToggle, required this.onDelete});

 @override
 Widget build(BuildContext context) {
  return Card(
   margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
   shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
   child: ListTile(
    leading: Checkbox(value: task.completed, onChanged: (_) => onToggle()),
```

```
      title: Text(task.title, style: TextStyle(decoration: task.completed ?
TextDecoration.lineThrough : null)),
      subtitle: Text('${task.timeOfDay} • ${task.durationMinutes} min'),
      trailing: IconButton(onPressed: onDelete, icon: const Icon(Icons.delete_outline)),
    ),
   );
 }
}
```

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../services/sqlite_service.dart';

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

 @override
 State<ProgressScreen> createState() => _ProgressScreenState();
}

class _ProgressScreenState extends State<ProgressScreen> {
 int streak = 0;

 @override
 void didChangeDependencies() {
  super.didChangeDependencies();
  _load();
 }
```

```
 Future<void> _load() async {
  final sqlite = Provider.of<SQLiteService>(context, listen: false);
  final s = await sqlite.getTotalStreak();
  setState(() => streak = s);
 }

 @override
 Widget build(BuildContext context) {
  return SafeArea(
   child: Column(children: [
    Padding(padding: const EdgeInsets.all(16.0), child: Row(children: const [Text('Progress', style: TextStyle(fontSize: 20, fontWeight: FontWeight.w600)), Spacer()])),
    Card(
     margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
     child: Padding(
      padding: const EdgeInsets.all(20.0),
      child: Row(children: [
       Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
        const Text('Current Streak', style: TextStyle(fontSize: 16)),
        const SizedBox(height: 8),
        Text('$streak days', style: const TextStyle(fontSize: 28, fontWeight: FontWeight.bold)),
       ]),
       const Spacer(),
       const Icon(Icons.emoji_events, size: 48, color: Color(0xFF2B6CB0)),
      ]),
     ),
    ),
```

```
      const SizedBox(height: 12),
      const Padding(padding: EdgeInsets.symmetric(horizontal: 16.0), child: Text('Daily summary & simple analytics (MVP).')),
      const Spacer(),
    ]),
  );
 }
}
```

```
import 'dart:async';
import 'package:flutter/material.dart';
import '../services/admob_service.dart';
import 'package:google_mobile_ads/google_mobile_ads.dart';

class FocusModeScreen extends StatefulWidget {
 const FocusModeScreen({super.key});
 @override
 State<FocusModeScreen> createState() => _FocusModeScreenState();
}

class _FocusModeScreenState extends State<FocusModeScreen> {
 Timer? _timer;
 int _seconds = 25 * 60;
 bool running = false;
 InterstitialAd? _interstitial;

 @override
 void initState() {
  super.initState();
```

```
  InterstitialAd.load(
    adUnitId: AdMobService.interstitialAdUnitId,
    request: const AdRequest(),
    adLoadCallback: InterstitialAdLoadCallback(
      onAdLoaded: (ad) {
        _interstitial = ad;
      },
      onAdFailedToLoad: (err) => debugPrint('Interstitial load failed: $err'),
    ),
  );
}

void _start() {
  if (running) return;
  setState(() => running = true);
  _timer = Timer.periodic(const Duration(seconds: 1), (t) {
    if (_seconds <= 0) {
      t.cancel();
      setState(() {
        running = false;
      });
      _showInterstitial();
      return;
    }
    setState(() => _seconds--);
  });
}
```

```
void _pause() {
  _timer?.cancel();
  setState(() => running = false);
}

void _reset() {
  _timer?.cancel();
  setState(() {
    running = false;
    _seconds = 25 * 60;
  });
}

void _showInterstitial() {
  _interstitial?.show();
  _interstitial?.dispose();
  _interstitial = null;
}

String _format(int s) {
  final m = (s / 60).floor();
  final sec = s % 60;
  return '${m.toString().padLeft(2, '0')}:${sec.toString().padLeft(2, '0')}';
}

@override
void dispose() {
  _timer?.cancel();
```

```
  _interstitial?.dispose();
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: const Text('Focus Mode'), backgroundColor: const Color(0xFF2B6CB0)),
   body: SafeArea(
    child: Column(
     children: [
      const SizedBox(height: 24),
      Text(_format(_seconds), style: const TextStyle(fontSize: 56, fontWeight: FontWeight.bold)),
      const SizedBox(height: 12),
      Text(running ? 'Focus session in progress' : 'Ready to start your focus session', style: const TextStyle(color: Colors.black54)),
      const Spacer(),
      Padding(
       padding: const EdgeInsets.symmetric(horizontal: 20.0),
       child: Row(children: [
        Expanded(child: ElevatedButton(onPressed: _start, child: const Text('Start'))),
        const SizedBox(width: 12),
        ElevatedButton(onPressed: _pause, child: const Text('Pause')),
        const SizedBox(width: 12),
        ElevatedButton(onPressed: _reset, child: const Text('Reset')),
       ]),
      ),
```

```
          const SizedBox(height: 24)
        ],
      ),
    ),
  );
 }
}
```

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

class UrgeTrackerScreen extends StatefulWidget {
 const UrgeTrackerScreen({super.key});
 @override
 State<UrgeTrackerScreen> createState() => _UrgeTrackerScreenState();
}

class _UrgeTrackerScreenState extends State<UrgeTrackerScreen> {
 List<Map<String, dynamic>> urges = [];

 void _addUrge(int intensity) {
  urges.add({'ts': DateTime.now().toIso8601String(), 'intensity': intensity});
  setState(() {});
 }

 @override
 Widget build(BuildContext context) {
  final tip = AiCoach.getTip('focus');
  return Scaffold(
```

```
    appBar: AppBar(title: const Text('Urge Tracker'), backgroundColor: const Color(0xFF2B6CB0)),
    body: SafeArea(
      child: Column(children: [
        Padding(
          padding: const EdgeInsets.all(16.0),
          child: Text('How strong is your urge to check your phone?', style: TextStyle(fontSize: 18, color: Colors.grey[800])),
        ),
        Slider(
          min: 0,
          max: 10,
          divisions: 10,
          value: 5,
          onChanged: (v) {},
          onChangeEnd: (v) => _addUrge(v.round()),
        ),
        const SizedBox(height: 12),
        Text('Quick tip: $tip', style: const TextStyle(fontStyle: FontStyle.italic)),
        const SizedBox(height: 12),
        const Divider(),
        const Padding(padding: EdgeInsets.all(8.0), child: Text('Urge log')),
        Expanded(
          child: ListView.builder(
            itemCount: urges.length,
            itemBuilder: (context, i) {
              final u = urges[i];
              return ListTile(
                title: Text('Intensity: ${u['intensity']}'),
```

```
              subtitle: Text(u['ts']),
            );
          },
        ),
      )
    ]),
  ),
 );
 }
}
```