TEST

Question 1:

import 'dart:io';

/// Rounds a double value to two decimal places.

///

/// This function performs standard mathematical rounding (round half up).

/// For example, 10.125 becomes 10.13, and 10.124 becomes 10.12.

double \_roundToTwoDecimals(double value) {

return (value \* 100).round() / 100.0;

}

/// Calculates the amount each person should pay, including tip.

///

/// This function calculates the tip amount and the total bill with tip,

/// then divides by the number of people to find the per-person amount.

/// The final per-person amount is rounded to two decimal places.

///

/// Throws [ArgumentError] if `people` is not positive, `total` is negative,

/// or `tipPercent` is negative. These checks provide robust validation

/// within the function, although inputs are also validated before calling.

double calculatePerPerson(double total, double tipPercent, int people) {

if (people <= 0) {

throw ArgumentError('Number of people must be greater than zero.');

}

if (total < 0) {

throw ArgumentError('Total bill amount cannot be negative.');

}

if (tipPercent < 0) {

throw ArgumentError('Tip percentage cannot be negative.');

}

// Calculate tip amount

double tipAmount = total \* (tipPercent / 100.0);

// Calculate total amount including tip

double totalWithTip = total + tipAmount;

// Calculate amount per person

double perPersonAmount = totalWithTip / people;

// Round the final per-person amount to two decimal places as required.

return \_roundToTwoDecimals(perPersonAmount);

}

void main() {

double totalBill;

int numPeople;

double tipPercentage;

// --- Get Total Bill Amount from User ---

stdout.write('Enter the total bill amount: \$');

String? billInput = stdin.readLineSync();

if (billInput == null || billInput.isEmpty) {

print('Error: Bill amount cannot be empty.');

return;

}

try {

totalBill = double.parse(billInput);

if (totalBill < 0) {

print('Error: Bill amount cannot be negative. Please enter a positive number.');

return;

}

} catch (e) {

print('Error: Invalid bill amount. Please enter a valid number (e.g., 50.75).');

return;

}

// --- Get Number of People from User ---

stdout.write('Enter the number of people: ');

String? peopleInput = stdin.readLineSync();

if (peopleInput == null || peopleInput.isEmpty) {

print('Error: Number of people cannot be empty.');

return;

}

try {

numPeople = int.parse(peopleInput);

if (numPeople <= 0) {

print('Error: Number of people must be a positive whole number.');

return;

}

} catch (e) {

print('Error: Invalid number of people. Please enter a whole number (e.g., 3).');

return;

}

// --- Get Tip Percentage from User ---

stdout.write('Enter the tip percentage (e.g., 15 for 15%): ');

String? tipInput = stdin.readLineSync();

if (tipInput == null || tipInput.isEmpty) {

print('Error: Tip percentage cannot be empty.');

return;

}

try {

tipPercentage = double.parse(tipInput);

if (tipPercentage < 0) {

print('Error: Tip percentage cannot be negative. Please enter a non-negative number.');

return;

}

} catch (e) {

print('Error: Invalid tip percentage. Please enter a valid number (e.g., 15.0).');

return;

}

// --- Calculate Results ---

// Calculate tip amount and total including tip for display.

// These calculations are repeated inside calculatePerPerson for its internal logic,

// as per the strict interpretation of the function's signature and return type.

double tipAmount = \_roundToTwoDecimals(totalBill \* (tipPercentage / 100.0));

double totalWithTip = \_roundToTwoDecimals(totalBill + tipAmount);

// Calculate amount per person using the specified function.

double perPersonAmount;

try {

perPersonAmount = calculatePerPerson(totalBill, tipPercentage, numPeople);

} catch (e) {

// This catch block handles potential ArgumentErrors from calculatePerPerson

// which should ideally be prevented by the input validation above.

print('An unexpected error occurred during calculation: ${e.toString()}');

return;

}

// --- Display Results ---

print('\n--- Bill Details ---');

print('Total Bill: \$${totalBill.toStringAsFixed(2)}');

print('Tip Percentage: ${tipPercentage.toStringAsFixed(2)}%');

print('Number of People: $numPeople');

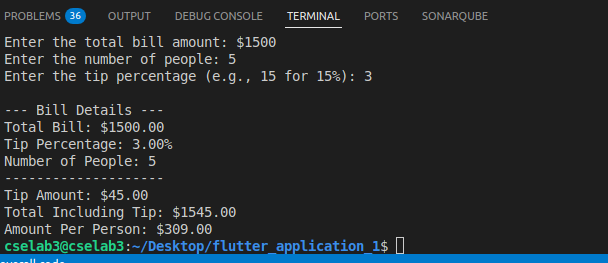
print('--------------------');

print('Tip Amount: \$${tipAmount.toStringAsFixed(2)}');

print('Total Including Tip: \$${totalWithTip.toStringAsFixed(2)}');

print('Amount Per Person: \$${perPersonAmount.toStringAsFixed(2)}');

}



QUESTION 2:

import 'dart:io';

/// Rounds a double value to two decimal places.

///

/// This function performs standard mathematical rounding (round half up).

/// For example, 10.125 becomes 10.13, and 10.124 becomes 10.12.

double \_roundToTwoDecimals(double value) {

return (value \* 100).round() / 100.0;

}

void main() {

// Create a list of student names.

final List<String> studentNames = [

'Alice Smith',

'Bob Johnson',

'Charlie Brown',

'Diana Miller',

'Eve Davis',

];

// Create a 2D list to store attendance.

// Each inner list contains 5 boolean values (true for present, false for absent)

// representing attendance for 5 days (e.g., Monday to Friday).

// The order of inner lists corresponds to the order of student names.

final List<List<bool>> attendanceData = [

// Alice Smith's attendance (Mon, Tue, Wed, Thu, Fri)

[true, true, false, true, true],

// Bob Johnson's attendance

[false, true, true, false, true],

// Charlie Brown's attendance

[true, false, true, true, false],

// Diana Miller's attendance

[true, true, true, true, true],

// Eve Davis's attendance

[false, false, false, true, false],

];

const int totalDaysInWeek = 5; // Fixed number of days for attendance tracking

print('--- Student Attendance Report ---');

// Iterate through each student to calculate and display their attendance.

for (int i = 0; i < studentNames.length; i++) {

final String studentName = studentNames[i];

final List<bool> studentAttendance = attendanceData[i];

// Count the number of days the student was present.

// The `where` method filters for 'true' values (present days), and `length` counts them.

final int presentDays = studentAttendance.where((isPresent) => isPresent).length;

// Calculate the attendance percentage.

// Ensure floating-point division by using 100.0.

final double attendancePercentage = (presentDays / totalDaysInWeek) \* 100.0;

// Round the attendance percentage to two decimal places.

final double roundedPercentage = \_roundToTwoDecimals(attendancePercentage);

// Display the student's name and their attendance percentage.

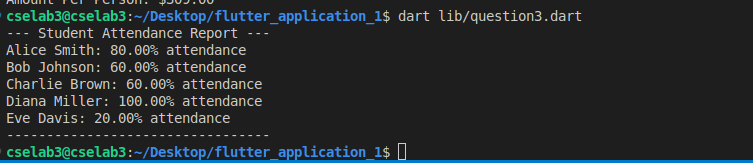
print(

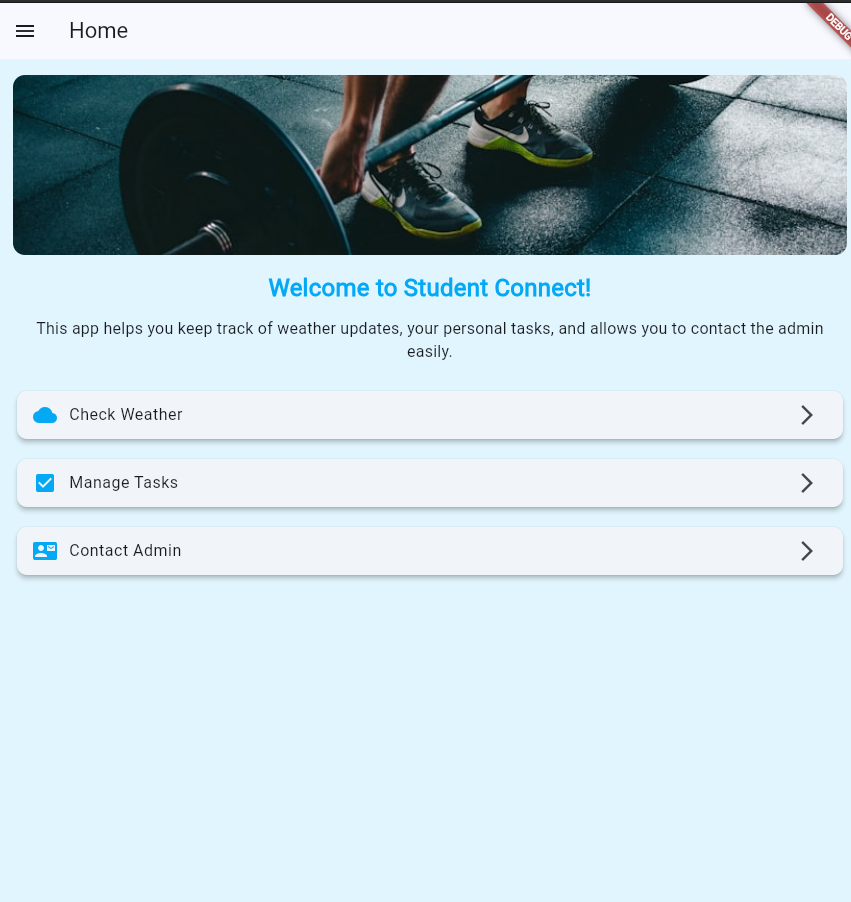
'${studentName}: ${roundedPercentage.toStringAsFixed(2)}% attendance');

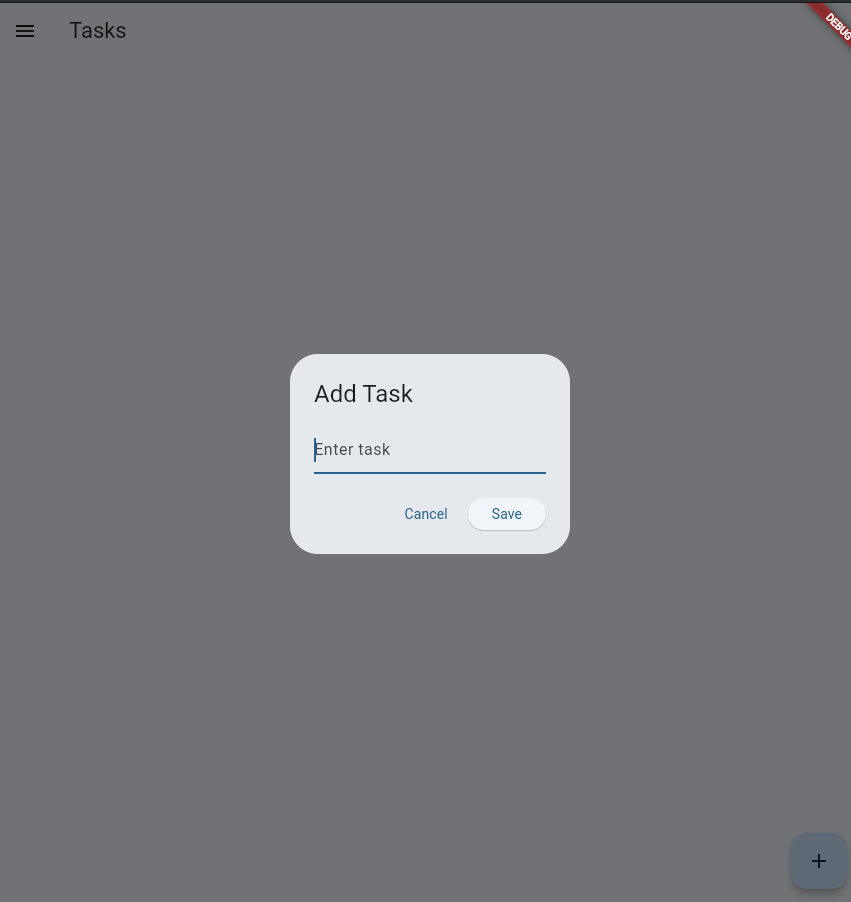
}

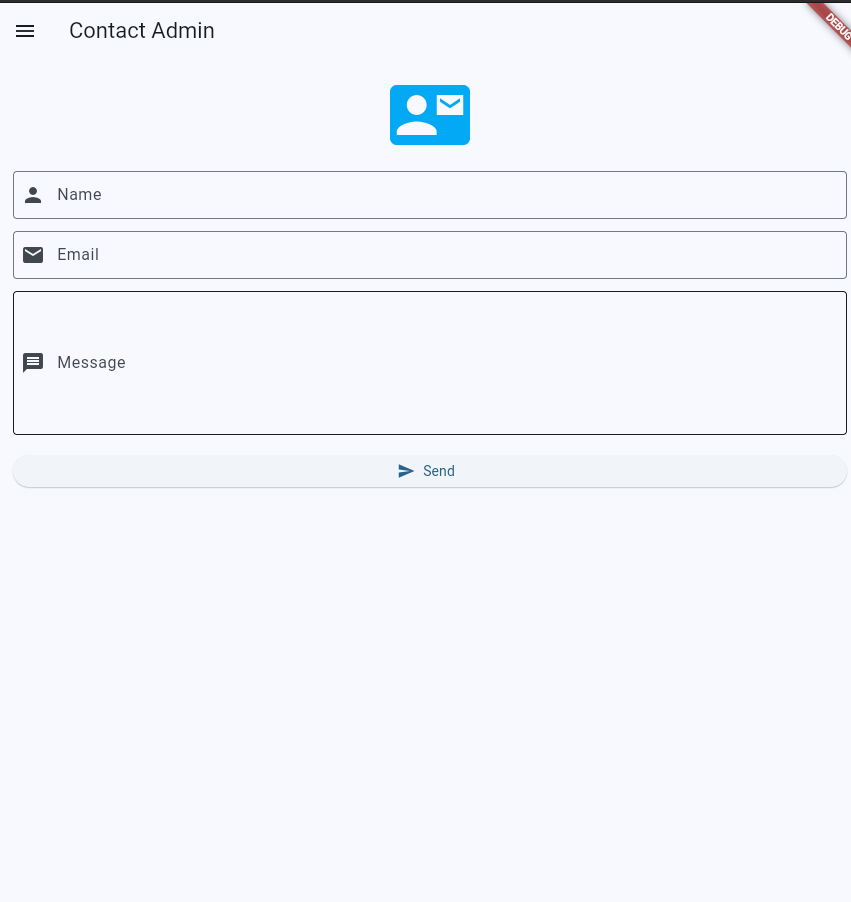
print('---------------------------------');

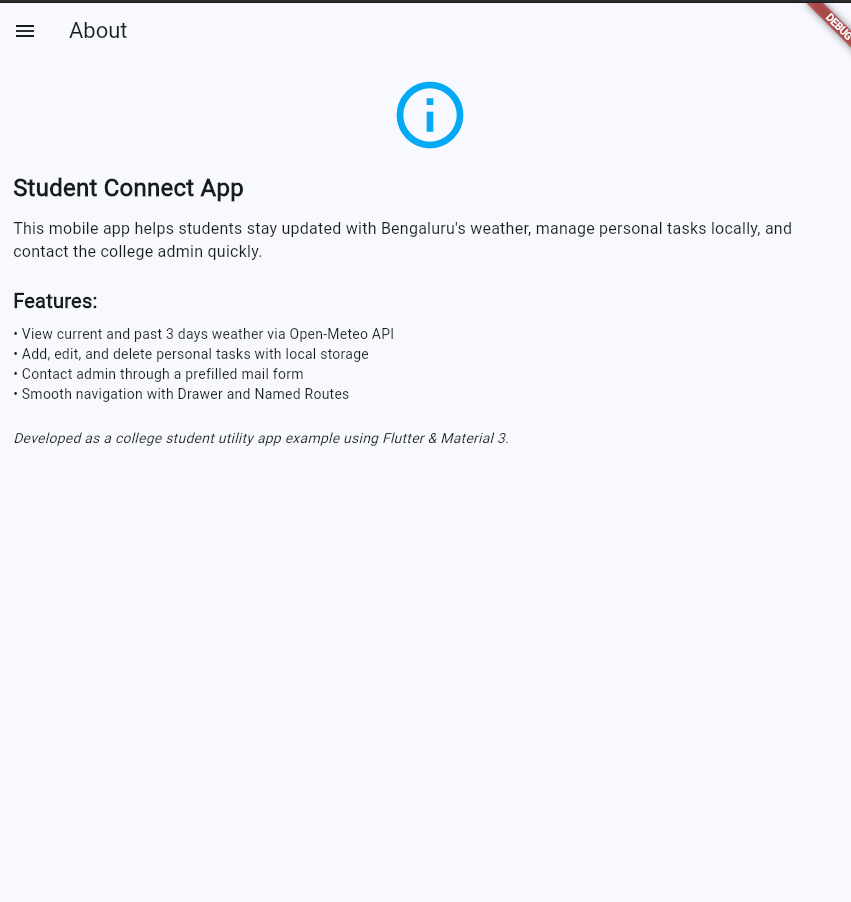
}











import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:url\_launcher/url\_launcher.dart';

void main() {

runApp(const StudentConnectApp());

}

class StudentConnectApp extends StatelessWidget {

const StudentConnectApp({super.key});

static const String homeRoute = '/home';

static const String weatherRoute = '/weather';

static const String tasksRoute = '/tasks';

static const String contactRoute = '/contact';

static const String aboutRoute = '/about';

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Student Connect',

theme: ThemeData(

colorSchemeSeed: Colors.lightBlue,

useMaterial3: true,

),

initialRoute: homeRoute,

routes: {

homeRoute: (context) => const HomeScreen(),

weatherRoute: (context) => const WeatherScreen(),

tasksRoute: (context) => const TasksScreen(),

contactRoute: (context) => const ContactAdminScreen(),

aboutRoute: (context) => const AboutScreen(),

},

);

}

}

class AppDrawer extends StatelessWidget {

const AppDrawer({super.key});

@override

Widget build(BuildContext context) {

return Drawer(

child: ListView(

children: [

const DrawerHeader(

decoration: BoxDecoration(

color: Colors.lightBlue,

),

child: Text(

'Student Connect',

style: TextStyle(

fontSize: 24,

color: Colors.white,

),

),

),

ListTile(

leading: const Icon(Icons.home),

title: const Text('Home'),

onTap: () {

Navigator.pushNamedAndRemoveUntil(

context, StudentConnectApp.homeRoute, (route) => false);

},

),

ListTile(

leading: const Icon(Icons.cloud),

title: const Text('Weather'),

onTap: () {

Navigator.pushNamedAndRemoveUntil(

context, StudentConnectApp.weatherRoute, (route) => false);

},

),

ListTile(

leading: const Icon(Icons.check\_box),

title: const Text('Tasks'),

onTap: () {

Navigator.pushNamedAndRemoveUntil(

context, StudentConnectApp.tasksRoute, (route) => false);

},

),

ListTile(

leading: const Icon(Icons.contact\_mail),

title: const Text('Contact Admin'),

onTap: () {

Navigator.pushNamedAndRemoveUntil(

context, StudentConnectApp.contactRoute, (route) => false);

},

),

ListTile(

leading: const Icon(Icons.info),

title: const Text('About'),

onTap: () {

Navigator.pushNamedAndRemoveUntil(

context, StudentConnectApp.aboutRoute, (route) => false);

},

),

],

),

);

}

}

// Home Screen with banner image and background color

class HomeScreen extends StatelessWidget {

const HomeScreen({super.key});

@override

Widget build(BuildContext context) {

return Scaffold(

drawer: const AppDrawer(),

appBar: AppBar(title: const Text('Home')),

body: Container(

color: Colors.lightBlue.shade50,

child: ListView(

padding: const EdgeInsets.all(16),

children: [

ClipRRect(

borderRadius: BorderRadius.circular(12),

child: Image.network(

'https://images.unsplash.com/photo-1517836357463-d25dfeac3438?auto=format&fit=crop&w=800&q=80', // Changed image

height: 180,

fit: BoxFit.cover,

),

),

const SizedBox(height: 16),

const Text(

'Welcome to Student Connect!',

style: TextStyle(

fontSize: 24,

fontWeight: FontWeight.bold,

color: Colors.lightBlue,

),

textAlign: TextAlign.center,

),

const SizedBox(height: 12),

const Text(

'This app helps you keep track of weather updates, your personal tasks, and allows you to contact the admin easily.',

style: TextStyle(fontSize: 16),

textAlign: TextAlign.center,

),

const SizedBox(height: 24),

Card(

elevation: 4,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(10)),

child: ListTile(

leading: const Icon(Icons.cloud, color: Colors.lightBlue),

title: const Text('Check Weather'),

trailing: const Icon(Icons.arrow\_forward\_ios),

onTap: () =>

Navigator.pushNamed(context, StudentConnectApp.weatherRoute),

),

),

const SizedBox(height: 12),

Card(

elevation: 4,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(10)),

child: ListTile(

leading: const Icon(Icons.check\_box, color: Colors.lightBlue),

title: const Text('Manage Tasks'),

trailing: const Icon(Icons.arrow\_forward\_ios),

onTap: () =>

Navigator.pushNamed(context, StudentConnectApp.tasksRoute),

),

),

const SizedBox(height: 12),

Card(

elevation: 4,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(10)),

child: ListTile(

leading: const Icon(Icons.contact\_mail, color: Colors.lightBlue),

title: const Text('Contact Admin'),

trailing: const Icon(Icons.arrow\_forward\_ios),

onTap: () =>

Navigator.pushNamed(context, StudentConnectApp.contactRoute),

),

),

],

),

),

);

}

}

// Weather Screen calling Open-Meteo API for Bengaluru (with past days)

class WeatherScreen extends StatefulWidget {

const WeatherScreen({super.key});

@override

State<WeatherScreen> createState() => \_WeatherScreenState();

}

class \_WeatherScreenState extends State<WeatherScreen> {

double? temperatureC;

List<Map<String, dynamic>> pastTemperatures = [];

Future<void> fetchWeather() async {

final currentUrl = Uri.parse(

'https://api.open-meteo.com/v1/forecast?latitude=12.9716&longitude=77.5946&current\_weather=true');

final pastUrl = Uri.parse(

'https://api.open-meteo.com/v1/forecast?latitude=12.9716&longitude=77.5946&past\_days=3&daily=temperature\_2m\_max,temperature\_2m\_min&timezone=auto');

try {

final currentResponse = await http.get(currentUrl);

final pastResponse = await http.get(pastUrl);

if (currentResponse.statusCode == 200 && pastResponse.statusCode == 200) {

final currentData = jsonDecode(currentResponse.body);

final pastData = jsonDecode(pastResponse.body);

setState(() {

temperatureC = currentData['current\_weather']['temperature'];

pastTemperatures = [];

List dates = pastData['daily']['time'];

List maxTemps = pastData['daily']['temperature\_2m\_max'];

List minTemps = pastData['daily']['temperature\_2m\_min'];

for (int i = 0; i < dates.length; i++) {

pastTemperatures.add({

'date': dates[i],

'max': maxTemps[i],

'min': minTemps[i],

});

}

});

} else {

temperatureC = null;

}

} catch (e) {

temperatureC = null;

}

}

@override

void initState() {

super.initState();

fetchWeather();

}

@override

Widget build(BuildContext context) {

return Scaffold(

drawer: const AppDrawer(),

appBar: AppBar(title: const Text('Weather')),

body: Center(

child: temperatureC == null

? const CircularProgressIndicator()

: ListView(

padding: const EdgeInsets.all(16),

children: [

const Icon(Icons.wb\_sunny, size: 80, color: Colors.orange),

const SizedBox(height: 16),

Text(

'Bengaluru Current Temperature',

style: Theme.of(context).textTheme.headlineSmall,

textAlign: TextAlign.center,

),

const SizedBox(height: 8),

Text(

'${temperatureC!.toStringAsFixed(1)} °C',

style: const TextStyle(fontSize: 48, fontWeight: FontWeight.bold),

textAlign: TextAlign.center,

),

const SizedBox(height: 24),

const Divider(),

const SizedBox(height: 8),

const Text(

'Past 3 Days Temperatures',

style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

textAlign: TextAlign.center,

),

const SizedBox(height: 8),

...pastTemperatures.map((day) {

return Card(

child: ListTile(

title: Text(day['date']),

subtitle: Text(

'Max: ${day['max']} °C, Min: ${day['min']} °C',

),

),

);

}).toList(),

const SizedBox(height: 20),

ElevatedButton.icon(

onPressed: fetchWeather,

icon: const Icon(Icons.refresh),

label: const Text('Refresh'),

),

],

),

),

);

}

}

// Tasks Screen with CRUD & SharedPreferences storage

class TasksScreen extends StatefulWidget {

const TasksScreen({super.key});

@override

State<TasksScreen> createState() => \_TasksScreenState();

}

class \_TasksScreenState extends State<TasksScreen> {

List<String> tasks = [];

final \_taskController = TextEditingController();

int? editingIndex;

@override

void initState() {

super.initState();

\_loadTasks();

}

Future<void> \_loadTasks() async {

final prefs = await SharedPreferences.getInstance();

final savedTasks = prefs.getStringList('tasks') ?? [];

setState(() {

tasks = savedTasks;

});

}

Future<void> \_saveTasks() async {

final prefs = await SharedPreferences.getInstance();

await prefs.setStringList('tasks', tasks);

}

void \_showTaskDialog({String? initialText, int? index}) {

if (initialText != null) {

\_taskController.text = initialText;

editingIndex = index;

} else {

\_taskController.clear();

editingIndex = null;

}

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text(editingIndex == null ? 'Add Task' : 'Edit Task'),

content: TextField(

controller: \_taskController,

decoration: const InputDecoration(

hintText: 'Enter task',

),

autofocus: true,

),

actions: [

TextButton(

onPressed: () {

Navigator.pop(context);

\_taskController.clear();

},

child: const Text('Cancel')),

ElevatedButton(

onPressed: () {

final text = \_taskController.text.trim();

if (text.isNotEmpty) {

setState(() {

if (editingIndex == null) {

tasks.add(text);

} else {

tasks[editingIndex!] = text;

}

});

\_saveTasks();

Navigator.pop(context);

\_taskController.clear();

}

},

child: const Text('Save')),

],

);

},

);

}

void \_deleteTask(int index) {

setState(() {

tasks.removeAt(index);

});

\_saveTasks();

}

@override

Widget build(BuildContext context) {

return Scaffold(

drawer: const AppDrawer(),

appBar: AppBar(title: const Text('Tasks')),

body: tasks.isEmpty

? const Center(

child: Text(

'No tasks added yet.\nTap + to add your first task.',

textAlign: TextAlign.center,

style: TextStyle(fontSize: 18),

),

)

: ListView.builder(

itemCount: tasks.length,

itemBuilder: (context, index) {

return Card(

margin:

const EdgeInsets.symmetric(horizontal: 12, vertical: 6),

child: ListTile(

leading: const Icon(Icons.task\_alt, color: Colors.lightBlue),

title: Text(tasks[index]),

trailing: Row(

mainAxisSize: MainAxisSize.min,

children: [

IconButton(

icon: const Icon(Icons.edit, color: Colors.orange),

onPressed: () => \_showTaskDialog(

initialText: tasks[index], index: index),

),

IconButton(

icon: const Icon(Icons.delete, color: Colors.red),

onPressed: () => \_deleteTask(index),

),

],

),

),

);

},

),

floatingActionButton: FloatingActionButton(

onPressed: () => \_showTaskDialog(),

tooltip: 'Add Task',

child: const Icon(Icons.add),

),

);

}

}

// Contact Admin screen with a form, input & mail launcher

class ContactAdminScreen extends StatefulWidget {

const ContactAdminScreen({super.key});

@override

State<ContactAdminScreen> createState() => \_ContactAdminScreenState();

}

class \_ContactAdminScreenState extends State<ContactAdminScreen> {

final \_formKey = GlobalKey<FormState>();

final \_nameController = TextEditingController();

final \_emailController = TextEditingController();

final \_messageController = TextEditingController();

Future<void> \_sendEmail() async {

final name = Uri.encodeComponent(\_nameController.text.trim());

final email = Uri.encodeComponent(\_emailController.text.trim());

final message = Uri.encodeComponent(\_messageController.text.trim());

final mailtoUrl =

'mailto:admin@college.edu?subject=Student%20Contact&body=Name:%20$name%0AEmail:%20$email%0A%0AMessage:%20$message';

if (await canLaunchUrl(Uri.parse(mailtoUrl))) {

await launchUrl(Uri.parse(mailtoUrl));

} else {

ScaffoldMessenger.of(context).showSnackBar(

const SnackBar(content: Text('Could not open mail app')),

);

}

}

@override

void dispose() {

\_nameController.dispose();

\_emailController.dispose();

\_messageController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

drawer: const AppDrawer(),

appBar: AppBar(title: const Text('Contact Admin')),

body: Padding(

padding: const EdgeInsets.all(16),

child: Form(

key: \_formKey,

child: ListView(

children: [

const Icon(Icons.contact\_mail,

size: 80, color: Colors.lightBlue),

const SizedBox(height: 16),

TextFormField(

controller: \_nameController,

decoration: const InputDecoration(

labelText: 'Name',

border: OutlineInputBorder(),

prefixIcon: Icon(Icons.person),

),

validator: (value) =>

value == null || value.isEmpty ? 'Enter your name' : null,

),

const SizedBox(height: 12),

TextFormField(

controller: \_emailController,

decoration: const InputDecoration(

labelText: 'Email',

border: OutlineInputBorder(),

prefixIcon: Icon(Icons.email),

),

keyboardType: TextInputType.emailAddress,

validator: (value) {

if (value == null || value.isEmpty) {

return 'Enter your email';

}

final emailRegex = RegExp(r'^[^@]+@[^@]+\.[^@]+');

if (!emailRegex.hasMatch(value)) {

return 'Enter valid email';

}

return null;

},

),

const SizedBox(height: 12),

TextFormField(

controller: \_messageController,

decoration: const InputDecoration(

labelText: 'Message',

border: OutlineInputBorder(),

prefixIcon: Icon(Icons.message),

),

maxLines: 5,

validator: (value) =>

value == null || value.isEmpty ? 'Enter your message' : null,

),

const SizedBox(height: 20),

ElevatedButton.icon(

icon: const Icon(Icons.send),

label: const Text('Send'),

onPressed: () {

if (\_formKey.currentState!.validate()) {

\_sendEmail();

}

},

),

],

),

),

),

);

}

}

// About Screen with app info

class AboutScreen extends StatelessWidget {

const AboutScreen({super.key});

@override

Widget build(BuildContext context) {

return Scaffold(

drawer: const AppDrawer(),

appBar: AppBar(title: const Text('About')),

body: Padding(

padding: const EdgeInsets.all(16),

child: ListView(

children: const [

Icon(Icons.info\_outline, size: 80, color: Colors.lightBlue),

SizedBox(height: 16),

Text(

'Student Connect App',

style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),

),

SizedBox(height: 12),

Text(

'This mobile app helps students stay updated with Bengaluru\'s weather, manage personal tasks locally, and contact the college admin quickly.',

style: TextStyle(fontSize: 16),

),

SizedBox(height: 24),

Text(

'Features:',

style: TextStyle(fontSize: 20, fontWeight: FontWeight.w600),

),

SizedBox(height: 8),

Text('• View current and past 3 days weather via Open-Meteo API'),

Text('• Add, edit, and delete personal tasks with local storage'),

Text('• Contact admin through a prefilled mail form'),

Text('• Smooth navigation with Drawer and Named Routes'),

SizedBox(height: 24),

Text(

'Developed as a college student utility app example using Flutter & Material 3.',

style: TextStyle(fontSize: 14, fontStyle: FontStyle.italic),

),

],

),

),

);

}

}