FLUTTER : STUDENT CONNECT

For app source code visit :

https://github.com/carry2408/Flutter/tree/master

About the Student Connect App

Welcome to Student Connect, your personal mobile command center.

Designed for students, it centralizes essential daily tools in one place.

The app features a sleek, custom-built dark space theme for a unique look.

Instantly check a 3-day weather forecast to plan your week ahead.

Manage your assignments with a full-featured, persistent to-do list.

You can create, edit, and delete tasks that save locally on your device.

Easily send inquiries directly to an administrator using the contact form.

Navigate effortlessly through all features using dashboard shortcuts.

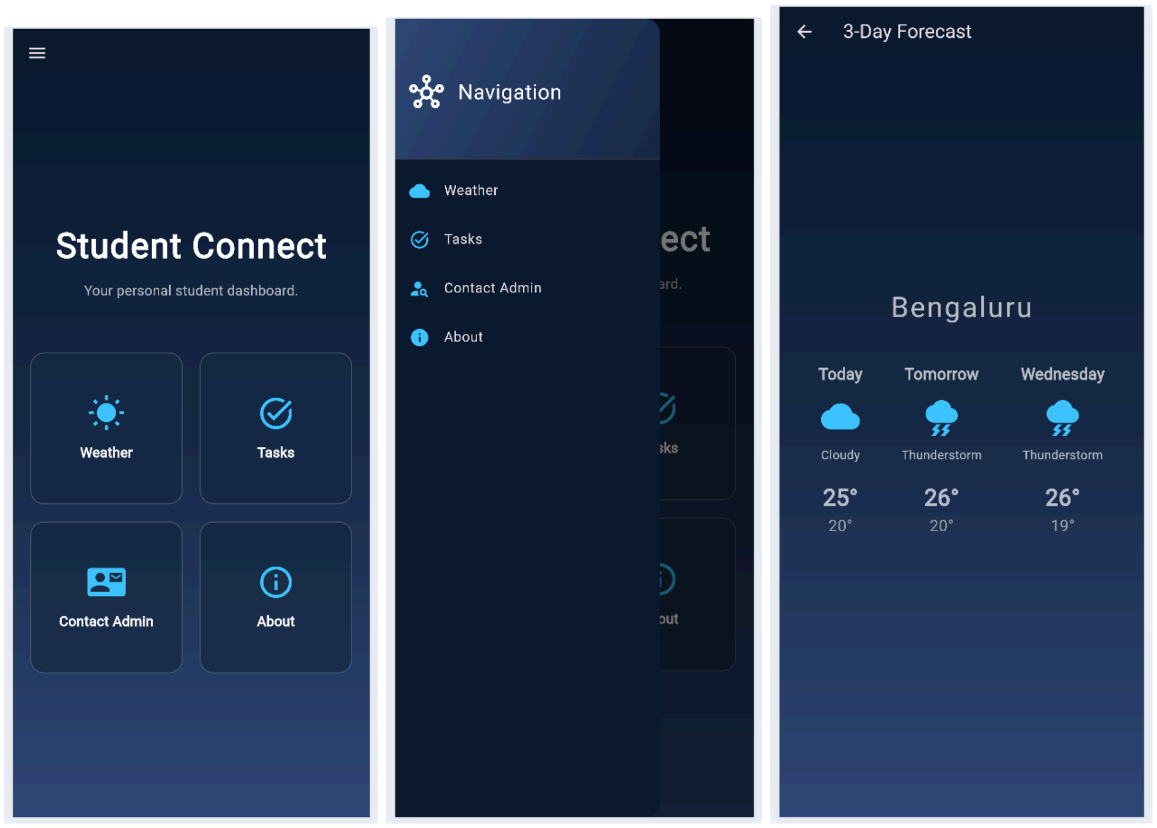
A classic slide-out drawer is also available for alternative navigation.

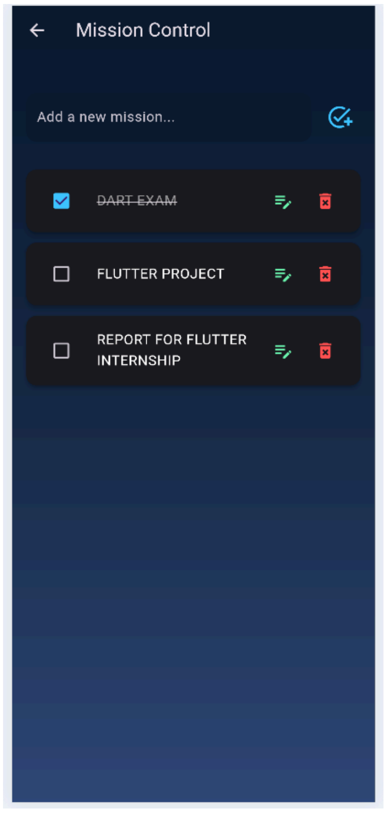
It's the all-in-one utility to keep your student life organized.

STRUCTURE OF STUDENT CONNECT APP

```
student_connect/
│
├── lib/
│   │
│   ├── main.dart
│   │
│   ├── models/
│   │   └── task_model.dart
│   │
│   └── screens/
│       ├── home_screen.dart
│       ├── weather_screen.dart
│       ├── tasks_screen.dart
│       ├── profile_screen.dart
│       └── about_screen.dart
│
└── pubspec.yaml
```

PHOTO OF APP


Student Connect
Your personal student dashboard.
Weather
Tasks
Contact Admin
About
Navigation
Weather
Tasks
Contact Admin
About
3-Day Forecast
Bengaluru
Today
Tomorrow
Wednesday
Cloudy
Thunderstorm
Thunderstorm
25°
20°
26°
20°
26°
19°



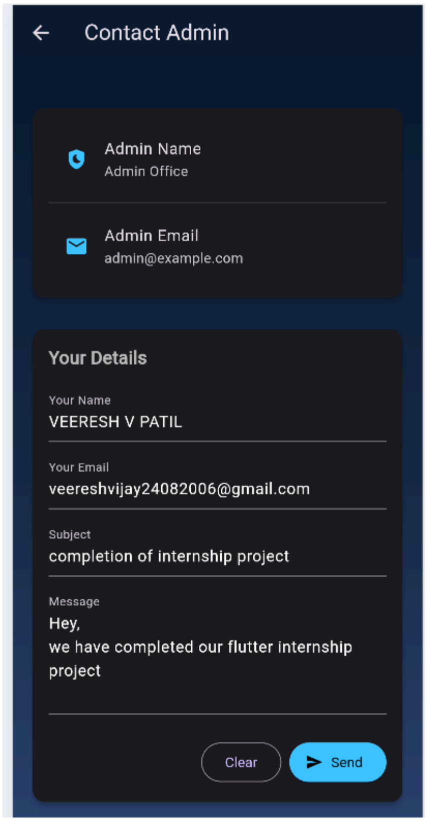
Mission Control

Add a new mission...

~~DART EXAM~~

FLUTTER PROJECT

REPORT FOR FLUTTER INTERNSHIP



Contact Admin

Admin Name
Admin Office

Admin Email
admin@example.com

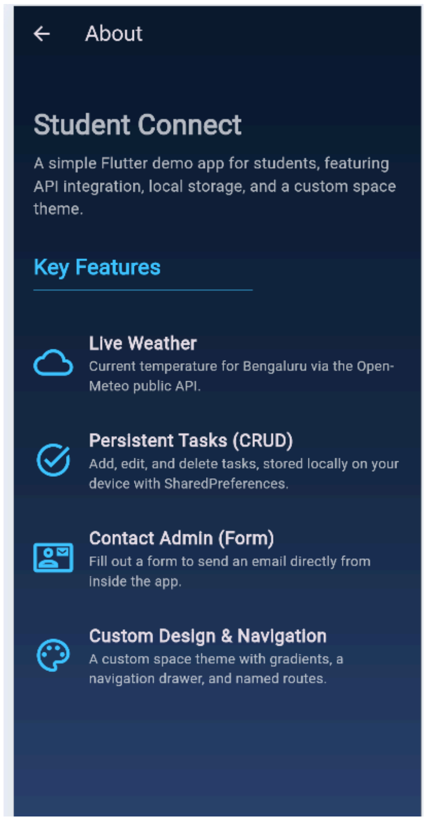
Your Details

Your Name
VEERESH V PATIL

Your Email
veereshvijay24082006@gmail.com

Subject
completion of internship project

Message
Hey,
we have completed our flutter internship project



About

Student Connect

A simple Flutter demo app for students, featuring API integration, local storage, and a custom space theme.

Key Features

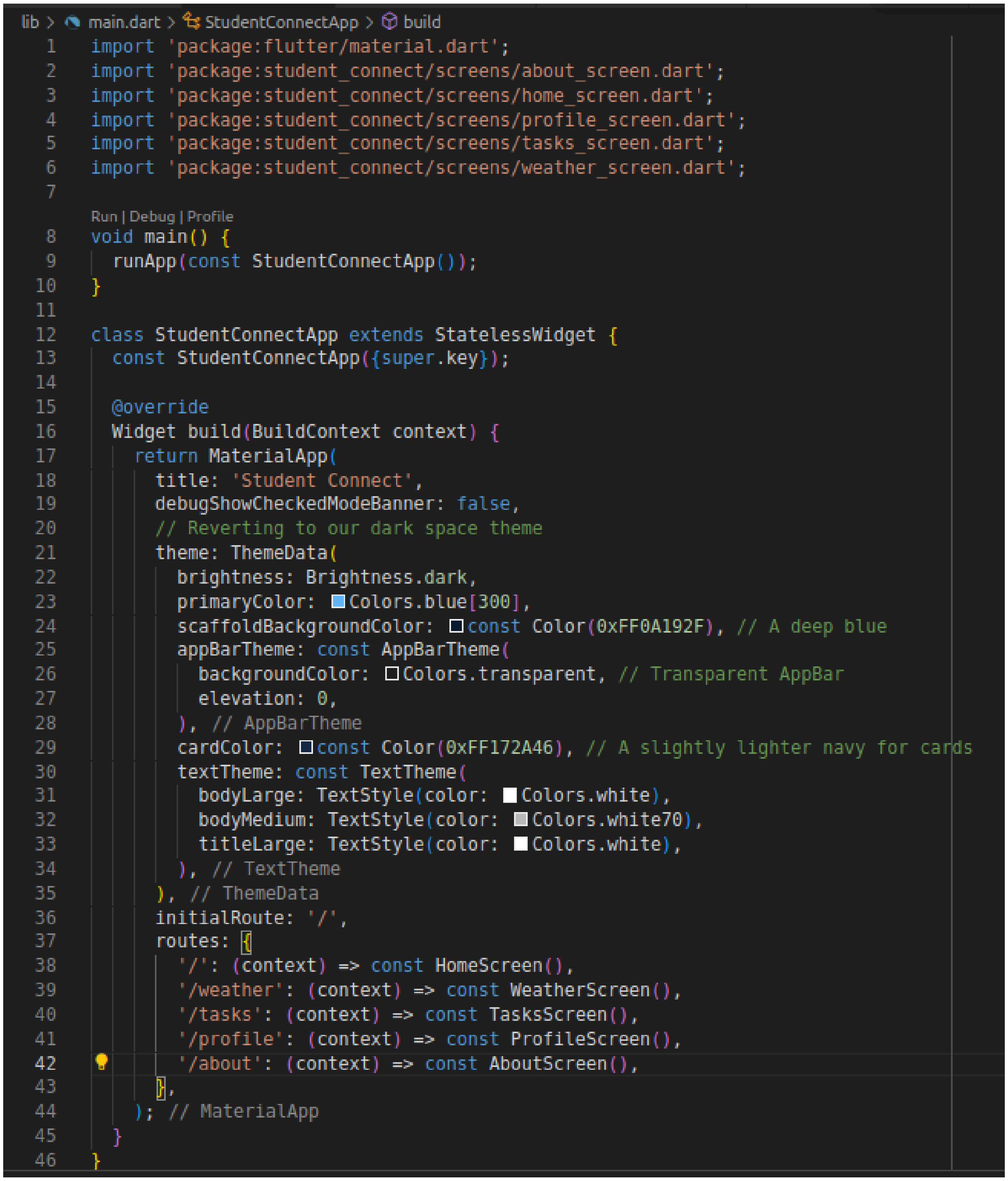
Live Weather
Current temperature for Bengaluru via the Open-Meteo public API.

Persistent Tasks (CRUD)
Add, edit, and delete tasks, stored locally on your device with SharedPreferences.

Contact Admin (Form)
Fill out a form to send an email directly from inside the app.

Custom Design & Navigation
A custom space theme with gradients, a navigation drawer, and named routes.

Main.dart


lib > main.dart > StudentConnectApp > build

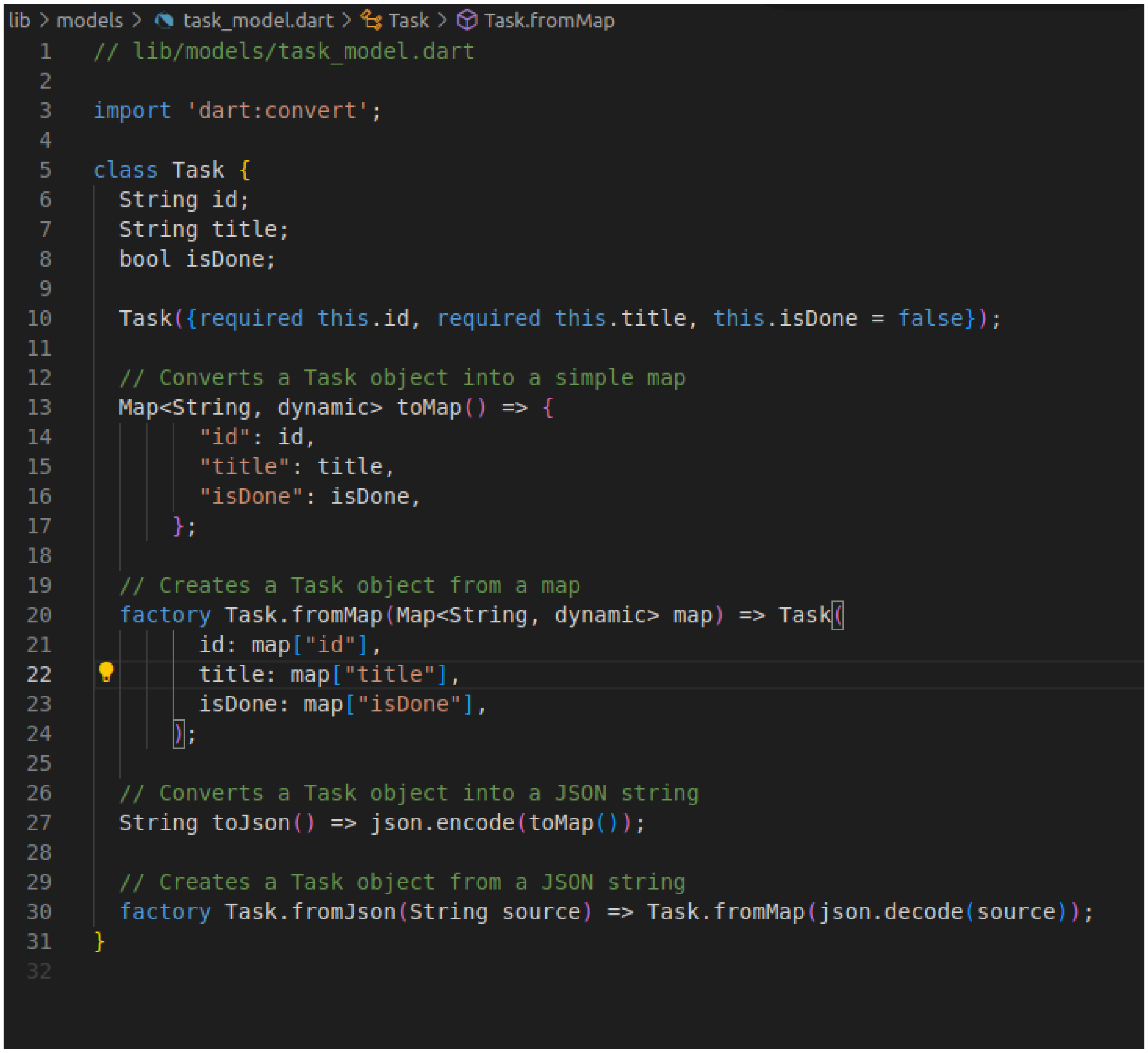
```dart
import 'package:flutter/material.dart';
import 'package:student_connect/screens/about_screen.dart';
import 'package:student_connect/screens/home_screen.dart';
import 'package:student_connect/screens/profile_screen.dart';
import 'package:student_connect/screens/tasks_screen.dart';
import 'package:student_connect/screens/weather_screen.dart';

Run | Debug | Profile
void main() {
  runApp(const StudentConnectApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Student Connect',
      debugShowCheckedModeBanner: false,
      // Reverting to our dark space theme
      theme: ThemeData(
        brightness: Brightness.dark,
        primaryColor: Colors.blue[300],
        scaffoldBackgroundColor: const Color(0xFF0A192F), // A deep blue
        appBarTheme: const AppBarTheme(
          backgroundColor: Colors.transparent, // Transparent AppBar
          elevation: 0,
        ), // AppBarTheme
        cardColor: const Color(0xFF172A46), // A slightly lighter navy for cards
        textTheme: const TextTheme(
          bodyLarge: TextStyle(color: Colors.white),
          bodyMedium: TextStyle(color: Colors.white70),
          titleLarge: TextStyle(color: Colors.white),
        ), // TextTheme
      ), // ThemeData
      initialRoute: '/',
      routes: {
        '/': (context) => const HomeScreen(),
        '/weather': (context) => const WeatherScreen(),
        '/tasks': (context) => const TasksScreen(),
        '/profile': (context) => const ProfileScreen(),
        '/about': (context) => const AboutScreen(),
      },
    ); // MaterialApp
  }
}
```

Task_model.dart


lib > models > task_model.dart > Task > Task.fromMap

```dart
// lib/models/task_model.dart

import 'dart:convert';

class Task {
  String id;
  String title;
  bool isDone;

  Task({required this.id, required this.title, this.isDone = false});

  // Converts a Task object into a simple map
  Map<String, dynamic> toMap() => {
        "id": id,
        "title": title,
        "isDone": isDone,
      };

  // Creates a Task object from a map
  factory Task.fromMap(Map<String, dynamic> map) => Task(
        id: map["id"],
        title: map["title"],
        isDone: map["isDone"],
      );

  // Converts a Task object into a JSON string
  String toJson() => json.encode(toMap());

  // Creates a Task object from a JSON string
  factory Task.fromJson(String source) => Task.fromMap(json.decode(source));
}
```

3. Home_screen.dart

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

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

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     drawer: _buildDrawer(context), // The navigation drawer
     appBar: AppBar(
         // The AppBar is transparent, so the gradient shows through
```

```
        ),
     extendBodyBehindAppBar: true,
     body: Container(
       width: double.infinity,
       height: double.infinity,
       // The gradient background
       decoration: const BoxDecoration(
         gradient: LinearGradient(
           colors: [Color(0xFF0A192F), Color(0xFF172A46),
Color(0xFF304878)],
           begin: Alignment.topCenter,
           end: Alignment.bottomCenter,
         ),
       ),
       child: SafeArea(
         child: Padding(
           padding: const EdgeInsets.symmetric(horizontal: 20.0),
           child: Column(
             mainAxisAlignment: MainAxisAlignment.center,
             children: [
               // App Name Title
               const Text(
                 'Student Connect',
                 style: TextStyle(
                   fontSize: 40,
                   fontWeight: FontWeight.bold,
                   color: Colors.white,
                   letterSpacing: 1.2,
                 ),
               ),
               const SizedBox(height: 12),
               const Text(
                 'Your personal student dashboard.',
                 style: TextStyle(
                   fontSize: 16,
                   color: Colors.white70,
                 ),
               ),
               const SizedBox(height: 60),
```

```dart
              // Grid of shortcut icons
              GridView.count(
                crossAxisCount: 2,
                shrinkWrap: true,
                physics: const NeverScrollableScrollPhysics(),
                crossAxisSpacing: 20,
                mainAxisSpacing: 20,
                children: [
                  _buildShortcutCard(
                    context: context,
                    icon: Icons.wb_sunny,
                    label: 'Weather',
                    routeName: '/weather',
                  ),
                  _buildShortcutCard(
                    context: context,
                    icon: Icons.task_alt,
                    label: 'Tasks',
                    routeName: '/tasks',
                  ),
                  _buildShortcutCard(
                    context: context,
                    icon: Icons.contact_mail,
                    label: 'Contact Admin',
                    routeName: '/profile',
                  ),
                  _buildShortcutCard(
                    context: context,
                    icon: Icons.info_outline,
                    label: 'About',
                    routeName: '/about',
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    ),
  );
```

```
}

// Helper widget for the styled shortcut cards
Widget _buildShortcutCard({
  required BuildContext context,
  required IconData icon,
  required String label,
  required String routeName,
}) {
  return InkWell(
    onTap: () => Navigator.pushNamed(context, routeName),
    borderRadius: BorderRadius.circular(16),
    child: Container(
      decoration: BoxDecoration(
        color: Theme.of(context).cardColor.withOpacity(0.8),
        borderRadius: BorderRadius.circular(16),
        border: Border.all(color: Colors.white24, width: 1),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(icon, size: 44, color: Colors.lightBlueAccent),
          const SizedBox(height: 12),
          Text(
            label,
            textAlign: TextAlign.center,
            style: const TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.w600,
              color: Colors.white,
            ),
          ),
        ],
      ),
    ),
  );
}

// The drawer from our original space theme
Widget _buildDrawer(BuildContext context) {
   return Drawer(
     backgroundColor: const Color(0xFF0A192F), // Match the dark theme
     child: ListView(
       padding: EdgeInsets.zero,
       children: <Widget>[
         const DrawerHeader(
           decoration: BoxDecoration(
             gradient: LinearGradient(
               colors: [Color(0xFF304878), Color(0xFF172A46)],
               begin: Alignment.topLeft,
               end: Alignment.bottomRight,
             ),
           ),
           child: Row(
             children: [
               Icon(Icons.hub_outlined, color: Colors.white, size: 40),
               SizedBox(width: 16),
               Text('Navigation',
                   style: TextStyle(color: Colors.white, fontSize: 24)),
             ],
           ),
         ),
         ListTile(
             leading:
                 const Icon(Icons.wb_cloudy, color:
Colors.lightBlueAccent),
             title: const Text('Weather'),
             onTap: () => Navigator.pushNamed(context, '/weather')),
         ListTile(
             leading:
                 const Icon(Icons.task_alt, color:
Colors.lightBlueAccent),
             title: const Text('Tasks'),
             onTap: () => Navigator.pushNamed(context, '/tasks')),
         ListTile(
             leading: const Icon(Icons.person_search,
                 color: Colors.lightBlueAccent),
             title: const Text('Contact Admin'),
             onTap: () => Navigator.pushNamed(context, '/profile')),
         ListTile(
              leading: const Icon(Icons.info, color:
Colors.lightBlueAccent),
              title: const Text('About'),
              onTap: () => Navigator.pushNamed(context, '/about')),
        ],
      ),
    );
  }
}
```

Profile_screen.dart

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

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

 @override
 State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
 // A key to identify and validate our form
 final _formKey = GlobalKey<FormState>();

 // Controllers to manage the text in each field
 final _nameController = TextEditingController();
 final _emailController = TextEditingController();
 final _subjectController = TextEditingController();
 final _messageController = TextEditingController();

 // The admin's email address
 final String _adminEmail = 'admin@example.com';

 @override
 void dispose() {
   // Clean up the controllers when the widget is removed
```

```dart
  _nameController.dispose();
  _emailController.dispose();
  _subjectController.dispose();
  _messageController.dispose();
  super.dispose();
}

// --- Functionality ---

void _clearForm() {
  _nameController.clear();
  _emailController.clear();
  _subjectController.clear();
  _messageController.clear();
}

Future<void> _sendEmail() async {
  // Validate the form first
  if (_formKey.currentState!.validate()) {
    final Email email = Email(
      // The body of the email includes the user's details
      body: '''
        Message:
        ${_messageController.text}

        ---
        From: ${_nameController.text}
        Email: ${_emailController.text}
      ''',
      subject: _subjectController.text,
      recipients: [_adminEmail], // Send to the admin
      isHTML: false,
    );

    try {
      await FlutterEmailSender.send(email);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Email client opened.')),
      );
    } catch (error) {
       ScaffoldMessenger.of(context).showSnackBar(
         SnackBar(content: Text('Could not open email client: $error')),
       );
     }
   }
 }

 // --- UI ---

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('Contact Admin'),
       backgroundColor: Colors.transparent,
       elevation: 0,
     ),
     extendBodyBehindAppBar: true,
     body: Container(
       width: double.infinity,
       height: double.infinity,
       decoration: const BoxDecoration(
         gradient: LinearGradient(
           colors: [Color(0xFF0A192F), Color(0xFF172A46),
Color(0xFF304878)],
           begin: Alignment.topCenter,
           end: Alignment.bottomCenter,
         ),
       ),
       // Use a ListView to prevent overflow on small screens
       child: SingleChildScrollView(
         padding: const EdgeInsets.fromLTRB(16, 100, 16, 16),
         child: Column(
           children: [
             // Admin Details Card
             _buildAdminDetailsCard(),
             const SizedBox(height: 24),
             // User Details Form Card
             _buildUserDetailsForm(),
           ],
        ),
      ),
    ),
  );
}

Widget _buildAdminDetailsCard() {
  return Card(
    elevation: 4,
    shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
    child: const Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        children: [
          ListTile(
            leading: Icon(Icons.shield_moon, color:
Colors.lightBlueAccent),
            title: Text('Admin Name'),
            subtitle: Text('Admin Office'),
          ),
          Divider(),
          ListTile(
            leading: Icon(Icons.email, color: Colors.lightBlueAccent),
            title: Text('Admin Email'),
            subtitle: Text('admin@example.com'),
          ),
        ],
      ),
    ),
  );
}

Widget _buildUserDetailsForm() {
  return Card(
    elevation: 4,
    shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
```

```dart
       child: Form(
         key: _formKey,
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.stretch,
           children: [
             const Text('Your Details',
                   style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold)),
             const SizedBox(height: 16),
             TextFormField(
               controller: _nameController,
               decoration: const InputDecoration(labelText: 'Your Name'),
               validator: (v) => v!.isEmpty ? 'Please enter your name' :
null,
             ),
             const SizedBox(height: 12),
             TextFormField(
               controller: _emailController,
               decoration: const InputDecoration(labelText: 'Your Email'),
               keyboardType: TextInputType.emailAddress,
               validator: (v) => v!.isEmpty ? 'Please enter your email' :
null,
             ),
             const SizedBox(height: 12),
             TextFormField(
               controller: _subjectController,
               decoration: const InputDecoration(labelText: 'Subject'),
               validator: (v) => v!.isEmpty ? 'Please enter a subject' :
null,
             ),
             const SizedBox(height: 12),
             TextFormField(
               controller: _messageController,
               decoration: const InputDecoration(labelText: 'Message'),
               maxLines: 4,
               validator: (v) => v!.isEmpty ? 'Please enter a message' :
null,
             ),
             const SizedBox(height: 24),
             Row(
               mainAxisAlignment: MainAxisAlignment.end,
               children: [
                 OutlinedButton(
                   onPressed: _clearForm,
                   child: const Text('Clear'),
                 ),
                 const SizedBox(width: 8),
                 ElevatedButton.icon(
                   onPressed: _sendEmail,
                   icon: const Icon(Icons.send),
                   label: const Text('Send'),
                   style: ElevatedButton.styleFrom(
                     backgroundColor: Colors.lightBlueAccent,
                     foregroundColor: Colors.black,
                   ),
                 ),
               ],
             ),
           ],
         ),
       ),
     ),
   );
 }
}
```

Task_screen.dart

```dart
// lib/screens/tasks_screen.dart

import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:student_connect/models/task_model.dart'; // Import our new
model

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

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

class _TasksScreenState extends State<TasksScreen> {
 final List<Task> _tasks = [];
 final TextEditingController _taskController = TextEditingController();
 static const String _tasksKey = 'tasks_list_key';

 @override
 void initState() {
   super.initState();
   _loadTasks(); // Load tasks when the app starts
 }

 // ---- CRUD Logic ----

 // READ from local storage
 Future<void> _loadTasks() async {
   final prefs = await SharedPreferences.getInstance();
   final List<String> taskListJson = prefs.getStringList(_tasksKey) ?? [];
   setState(() {
     _tasks.clear();
     _tasks.addAll(taskListJson.map((json) => Task.fromJson(json)));
   });
 }

 // Helper method to SAVE all tasks to local storage
```

```dart
Future<void> _saveTasks() async {
  final prefs = await SharedPreferences.getInstance();
  final List<String> taskListJson =
      _tasks.map((task) => task.toJson()).toList();
  await prefs.setStringList(_tasksKey, taskListJson);
}

// CREATE a new task
void _addTask(String title) {
  if (title.isNotEmpty) {
    final newTask = Task(id: DateTime.now().toString(), title: title);
    setState(() {
      _tasks.add(newTask);
    });
    _saveTasks();
    _taskController.clear();
    FocusScope.of(context).unfocus(); // Close keyboard
  }
}

// UPDATE an existing task's completion status
void _toggleTaskStatus(Task task) {
  setState(() {
    task.isDone = !task.isDone;
  });
  _saveTasks();
}

// UPDATE an existing task's title
void _editTaskTitle(Task task, String newTitle) {
  if (newTitle.isNotEmpty) {
    setState(() {
      task.title = newTitle;
    });
    _saveTasks();
  }
}

// DELETE a task
void _deleteTask(String id) {
   setState(() {
     _tasks.removeWhere((task) => task.id == id);
   });
   _saveTasks();
 }

 // ---- UI ----

 // Dialog for editing a task
 void _showEditTaskDialog(Task task) {
   final editController = TextEditingController(text: task.title);
   showDialog(
     context: context,
     builder: (context) => AlertDialog(
       backgroundColor: const Color(0xFF172A46),
       title: const Text('Edit Mission'),
       content: TextField(
         controller: editController,
         autofocus: true,
         decoration: const InputDecoration(labelText: 'New mission
title'),
       ),
       actions: [
         TextButton(
           child: const Text('Cancel'),
           onPressed: () => Navigator.pop(context),
         ),
         ElevatedButton(
           style: ElevatedButton.styleFrom(
               backgroundColor: Colors.lightBlueAccent),
           child: const Text('Save', style: TextStyle(color:
Colors.black)),
           onPressed: () {
             _editTaskTitle(task, editController.text);
             Navigator.pop(context);
           },
         ),
       ],
     ),
   );
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('Mission Control'),
       backgroundColor: Colors.transparent,
       elevation: 0,
     ),
     extendBodyBehindAppBar: true,
     body: Container(
       width: double.infinity,
       decoration: const BoxDecoration(
         gradient: LinearGradient(
           colors: [Color(0xFF0A192F), Color(0xFF172A46),
Color(0xFF304878)],
           begin: Alignment.topCenter,
           end: Alignment.bottomCenter,
         ),
       ),
       child: Column(
         children: [
           // Input field for adding new tasks
           _buildAddTaskUI(),
           // List of tasks
           Expanded(
             child: _tasks.isEmpty
                 ? const Center(child: Text("No missions assigned. Add
one!"))
                 : ListView.builder(
                     padding: const EdgeInsets.only(top: 10),
                     itemCount: _tasks.length,
                     itemBuilder: (context, index) {
                       final task = _tasks[index];
                       return _buildTaskCard(task);
                     },
                   ),
           ),
         ],
      ),
    ),
  );
}

// UI for the input field
Widget _buildAddTaskUI() {
  return Padding(
    padding: const EdgeInsets.fromLTRB(16, 100, 16, 16),
    child: Row(
      children: [
        Expanded(
          child: TextField(
            controller: _taskController,
            decoration: InputDecoration(
              hintText: 'Add a new mission...',
              filled: true,
              fillColor: const Color(0xFF0A192F).withOpacity(0.7),
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(12),
                borderSide: BorderSide.none,
              ),
            ),
            onSubmitted: (value) => _addTask(value),
          ),
        ),
        const SizedBox(width: 8),
        IconButton(
          icon: const Icon(Icons.add_task),
          iconSize: 30,
          color: Colors.lightBlueAccent,
          onPressed: () => _addTask(_taskController.text),
        )
      ],
    ),
  );
}

// UI for a single task card
Widget _buildTaskCard(Task task) {
   return Card(
     margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 6),
     elevation: 4,
     shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
     child: ListTile(
       contentPadding: const EdgeInsets.symmetric(horizontal: 16,
vertical: 8),
       leading: Checkbox(
         value: task.isDone,
         onChanged: (value) => _toggleTaskStatus(task),
         activeColor: Colors.lightBlueAccent,
       ),
       title: Text(
         task.title,
         style: TextStyle(
           decoration:
               task.isDone ? TextDecoration.lineThrough :
TextDecoration.none,
           color: task.isDone ? Colors.white54 : Colors.white,
         ),
       ),
       trailing: Row(
         mainAxisSize: MainAxisSize.min,
         children: [
           IconButton(
             icon: const Icon(Icons.edit_note, color: Colors.greenAccent),
             onPressed: () => _showEditTaskDialog(task),
           ),
           IconButton(
             icon: const Icon(Icons.delete_forever, color:
Colors.redAccent),
             onPressed: () => _deleteTask(task.id),
           ),
         ],
       ),
     ),
   );
 }
}
```

Profile_screen.dart

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

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

 @override
 State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
 // A key to identify and validate our form
 final _formKey = GlobalKey<FormState>();

 // Controllers to manage the text in each field
 final _nameController = TextEditingController();
 final _emailController = TextEditingController();
 final _subjectController = TextEditingController();
 final _messageController = TextEditingController();

 // The admin's email address
 final String _adminEmail = 'admin@example.com';

 @override
 void dispose() {
   // Clean up the controllers when the widget is removed
   _nameController.dispose();
   _emailController.dispose();
   _subjectController.dispose();
   _messageController.dispose();
   super.dispose();
 }

 // --- Functionality ---

 void _clearForm() {
   _nameController.clear();
   _emailController.clear();
```

```
  _subjectController.clear();
  _messageController.clear();
}

Future<void> _sendEmail() async {
  // Validate the form first
  if (_formKey.currentState!.validate()) {
    final Email email = Email(
      // The body of the email includes the user's details
      body: '''
        Message:
        ${_messageController.text}

        ---
        From: ${_nameController.text}
        Email: ${_emailController.text}
      ''',
      subject: _subjectController.text,
      recipients: [_adminEmail], // Send to the admin
      isHTML: false,
    );

    try {
      await FlutterEmailSender.send(email);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Email client opened.')),
      );
    } catch (error) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Could not open email client: $error')),
      );
    }
  }
}

// --- UI ---

@override
Widget build(BuildContext context) {
  return Scaffold(
     appBar: AppBar(
       title: const Text('Contact Admin'),
       backgroundColor: Colors.transparent,
       elevation: 0,
     ),
     extendBodyBehindAppBar: true,
     body: Container(
       width: double.infinity,
       height: double.infinity,
       decoration: const BoxDecoration(
         gradient: LinearGradient(
           colors: [Color(0xFF0A192F), Color(0xFF172A46),
Color(0xFF304878)],
           begin: Alignment.topCenter,
           end: Alignment.bottomCenter,
         ),
       ),
       // Use a ListView to prevent overflow on small screens
       child: SingleChildScrollView(
         padding: const EdgeInsets.fromLTRB(16, 100, 16, 16),
         child: Column(
           children: [
             // Admin Details Card
             _buildAdminDetailsCard(),
             const SizedBox(height: 24),
             // User Details Form Card
             _buildUserDetailsForm(),
           ],
         ),
       ),
     ),
   );
 }

 Widget _buildAdminDetailsCard() {
   return Card(
     elevation: 4,
     shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
     child: const Padding(
       padding: EdgeInsets.all(16.0),
       child: Column(
         children: [
           ListTile(
             leading: Icon(Icons.shield_moon, color:
Colors.lightBlueAccent),
             title: Text('Admin Name'),
             subtitle: Text('Admin Office'),
           ),
           Divider(),
           ListTile(
             leading: Icon(Icons.email, color: Colors.lightBlueAccent),
             title: Text('Admin Email'),
             subtitle: Text('admin@example.com'),
           ),
         ],
       ),
     ),
   );
 }

 Widget _buildUserDetailsForm() {
   return Card(
     elevation: 4,
     shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
     child: Padding(
       padding: const EdgeInsets.all(16.0),
       child: Form(
         key: _formKey,
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.stretch,
           children: [
             const Text('Your Details',
                 style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold)),
             const SizedBox(height: 16),
             TextFormField(
               controller: _nameController,
               decoration: const InputDecoration(labelText: 'Your Name'),
                validator: (v) => v!.isEmpty ? 'Please enter your name' :
null,
              ),
              const SizedBox(height: 12),
              TextFormField(
                controller: _emailController,
                decoration: const InputDecoration(labelText: 'Your Email'),
                keyboardType: TextInputType.emailAddress,
                validator: (v) => v!.isEmpty ? 'Please enter your email' :
null,
              ),
              const SizedBox(height: 12),
              TextFormField(
                controller: _subjectController,
                decoration: const InputDecoration(labelText: 'Subject'),
                validator: (v) => v!.isEmpty ? 'Please enter a subject' :
null,
              ),
              const SizedBox(height: 12),
              TextFormField(
                controller: _messageController,
                decoration: const InputDecoration(labelText: 'Message'),
                maxLines: 4,
                validator: (v) => v!.isEmpty ? 'Please enter a message' :
null,
              ),
              const SizedBox(height: 24),
              Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: [
                  OutlinedButton(
                    onPressed: _clearForm,
                    child: const Text('Clear'),
                  ),
                  const SizedBox(width: 8),
                  ElevatedButton.icon(
                    onPressed: _sendEmail,
                    icon: const Icon(Icons.send),
                    label: const Text('Send'),
                    style: ElevatedButton.styleFrom(
                     backgroundColor: Colors.lightBlueAccent,
                     foregroundColor: Colors.black,
                   ),
                 ),
               ],
             ),
           ],
         ),
       ),
     ),
   );
 }
}
```

weather _screen.dart

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:intl/intl.dart'; // For date formatting

// A simple class to hold the data for a single day's forecast
class DailyForecast {
 final DateTime date;
 final String weatherCondition;
 final IconData weatherIcon;
 final double maxTemp;
 final double minTemp;

 DailyForecast({
   required this.date,
   required this.weatherCondition,
   required this.weatherIcon,
   required this.maxTemp,
   required this.minTemp,
 });
}

class WeatherScreen extends StatefulWidget {
```

```dart
 const WeatherScreen({super.key});

 @override
 State<WeatherScreen> createState() => _WeatherScreenState();
}

class _WeatherScreenState extends State<WeatherScreen> {
 // A list to hold our 3-day forecast
 List<DailyForecast> _forecasts = [];
 bool _isLoading = true;
 String? _errorMessage;

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

 // Fetches the 3-day forecast from the Open-Meteo API
 Future<void> _fetchWeather() async {
   // UPDATED: Requesting daily data for the next 3 days
   final url = Uri.parse(

'https://api.open-meteo.com/v1/forecast?latitude=12.97&longitude=77.59&dai
ly=weathercode,temperature_2m_max,temperature_2m_min&forecast_days=3');

   try {
     final response = await http.get(url);
     if (response.statusCode == 200) {
       final data = json.decode(response.body)['daily'];

       // Clear previous forecast data
       final List<DailyForecast> fetchedForecasts = [];

       for (int i = 0; i < data['time'].length; i++) {
         final weatherCode = data['weathercode'][i];
         fetchedForecasts.add(
           DailyForecast(
             date: DateTime.parse(data['time'][i]),
             weatherCondition: _getWeatherCondition(weatherCode),
            weatherIcon: _getWeatherIcon(weatherCode),
            maxTemp: data['temperature_2m_max'][i],
            minTemp: data['temperature_2m_min'][i],
          ),
        );
      }

      setState(() {
        _forecasts = fetchedForecasts;
        _isLoading = false;
      });
    } else {
      throw Exception('Failed to load weather data');
    }
  } catch (e) {
    setState(() {
      _errorMessage = 'Could not fetch weather data.';
      _isLoading = false;
    });
  }
}

// Helper functions to interpret the weather code from the API
String _getWeatherCondition(int code) {
  // (This function can be expanded for more detail)
  switch (code) {
    case 0:
      return 'Clear Sky';
    case 1:
    case 2:
    case 3:
      return 'Cloudy';
    case 45:
    case 48:
      return 'Fog';
    case 61:
    case 63:
    case 65:
      return 'Rain';
    case 80:
```

```
    case 81:
    case 82:
      return 'Showers';
    case 95:
    case 96:
    case 99:
      return 'Thunderstorm';
    default:
      return 'Cloudy';
  }
}

IconData _getWeatherIcon(int code) {
  switch (code) {
    case 0:
      return Icons.wb_sunny;
    case 1:
    case 2:
    case 3:
      return Icons.cloud;
    case 45:
    case 48:
      return Icons.foggy;
    case 61:
    case 63:
    case 65:
      return Icons.water_drop;
    case 80:
    case 81:
    case 82:
      return Icons.shower;
    case 95:
    case 96:
    case 99:
      return Icons.thunderstorm;
    default:
      return Icons.cloud_outlined;
  }
}
```

```
 // Helper to get the day's name (Today, Tomorrow, or Weekday)
 String _getDayName(DateTime date, int index) {
   if (index == 0) return 'Today';
   if (index == 1) return 'Tomorrow';
   return DateFormat('EEEE').format(date); // e.g., "Wednesday"
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('3-Day Forecast'),
     ),
     extendBodyBehindAppBar: true,
     body: Container(
       width: double.infinity,
       height: double.infinity,
       decoration: const BoxDecoration(
         gradient: LinearGradient(
           colors: [Color(0xFF0A192F), Color(0xFF172A46),
Color(0xFF304878)],
           begin: Alignment.topCenter,
           end: Alignment.bottomCenter,
         ),
       ),
       child: Center(
         child: _isLoading
             ? const CircularProgressIndicator()
             : _errorMessage != null
                 ? Text(_errorMessage!,
                     style: const TextStyle(color: Colors.red, fontSize:
16))
                 : _buildForecastView(),
       ),
     ),
   );
 }

 // The main UI widget for displaying the 3-day forecast
 Widget _buildForecastView() {
   return Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
       const Text(
         'Bengaluru',
         style: TextStyle(
             fontSize: 32, fontWeight: FontWeight.w300, letterSpacing: 2),
       ),
       const SizedBox(height: 40),
       Row(
         mainAxisAlignment: MainAxisAlignment.spaceEvenly,
         children: List.generate(_forecasts.length, (index) {
           return _buildForecastCard(
             dayName: _getDayName(_forecasts[index].date, index),
             forecast: _forecasts[index],
           );
         }),
       ),
     ],
   );
 }

 // A card widget for a single day's forecast
 Widget _buildForecastCard(
     {required String dayName, required DailyForecast forecast}) {
   return Column(
     children: [
       Text(
         dayName,
         style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
       ),
       const SizedBox(height: 12),
       Icon(forecast.weatherIcon, size: 44, color:
Colors.lightBlueAccent),
       const SizedBox(height: 12),
       Text(
         forecast.weatherCondition,
         style: const TextStyle(color: Colors.white70),
       ),
       const SizedBox(height: 20),
       Text(
         '${forecast.maxTemp.round()}°',
         style: const TextStyle(fontSize: 26, fontWeight:
FontWeight.w600),
       ),
       Text(
         '${forecast.minTemp.round()}°',
         style: const TextStyle(fontSize: 18, color: Colors.white54),
       ),
     ],
   );
 }
}
```