README.md:

health_buddy

A new Flutter project.

Getting Started

This project is a starting point for a Flutter application.

A few resources to get you started if this is your first Flutter project:

- [Lab: Write your first Flutter app](https://docs.flutter.dev/get-started/codelab)
- [Cookbook: Useful Flutter samples](https://docs.flutter.dev/cookbook)

For help getting started with Flutter development, view the
[online documentation](https://docs.flutter.dev/), which offers tutorials,
samples, guidance on mobile development, and a full API reference.

Pubspec.yaml :

```
name: health_buddy
description: "A new Flutter project."
# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as versionCode.
# Read more about Android versioning at https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as
CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReferenc
e/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1

environment:
  sdk: ^3.5.3

# Dependencies specify other packages that your package needs in order to work.
```

```
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.8
  get: ^4.6.6
  csv: ^6.0.0
  lottie: ^3.1.3
  animated_bottom_navigation_bar: ^1.3.3
  http: ^1.2.2
  shared_preferences: ^2.3.3
  sqflite: ^2.4.0
  path: ^1.9.0
  carousel_slider: ^5.0.0

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^4.0.0


# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - assets/images/doctor-5216835_640.webp
    - assets/images/logo.jpg
    - assets/images/home_lottie.json
    - assets/csv/food_list_csv.csv
    - assets/images/greeting_img1.png

  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies, see
```

```
  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/to/font-from-package
```

build.gradle :

```
plugins {
    id "com.android.application"
    id "kotlin-android"
    // The Flutter Gradle Plugin must be applied after the Android and Kotlin Gradle plugins.
    id "dev.flutter.flutter-gradle-plugin"
}

android {
    namespace = "com.example.health_buddy"
    compileSdk = flutter.compileSdkVersion
    ndkVersion = flutter.ndkVersion

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = JavaVersion.VERSION_1_8
    }

    defaultConfig {
        // TODO: Specify your own unique Application ID
```

```
(https://developer.android.com/studio/build/application-id.html).
        applicationId = "com.example.health_buddy"
        // You can update the following values to match your application needs.
        // For more information, see: https://flutter.dev/to/review-gradle-config.
        minSdk = flutter.minSdkVersion
        targetSdk = flutter.targetSdkVersion
        versionCode = flutter.versionCode
        versionName = flutter.versionName
    }

    buildTypes {
        release {
            // TODO: Add your own signing config for the release build.
            // Signing with the debug keys for now, so `flutter run --release` works.
            signingConfig = signingConfigs.debug
        }
    }
}

flutter {
    source = "../.."
}
```

main.dart:

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:health_buddy/Pages/splash_screen.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Health Buddy',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
```

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

chatbot_dart:

```
import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:health_buddy/constants/app_color.dart';

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

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

  @override
  State<ChatBot> createState() => _ChatBotState();
}

class _ChatBotState extends State<ChatBot> with SingleTickerProviderStateMixin {
  final List<Map<String, String>> _messages = [];
  final TextEditingController _textController = TextEditingController();
  bool _isListening = false;
  bool _isTyping = false;

  late AnimationController _animationController;

  @override
  void initState() {
    super.initState();
    _animationController = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 1000),
    )..repeat();

    _loadMessagesFromPreferences(); // Load saved messages on initialization
  }

  @override
  void dispose() {
    _animationController.dispose();
    super.dispose();
```

```
}

Future<void> _loadMessagesFromPreferences() async {
  final prefs = await SharedPreferences.getInstance();
  final savedMessages = prefs.getString('chat_messages');

  if (savedMessages != null) {
    final decodedMessages = jsonDecode(savedMessages) as List;
    setState(() {
      _messages
          .addAll(decodedMessages.map((e) => Map<String, String>.from(e)));
    });
  }
}

Future<void> _saveMessagesToPreferences() async {
  final prefs = await SharedPreferences.getInstance();
  final encodedMessages = jsonEncode(_messages);
  await prefs.setString('chat_messages', encodedMessages);
}

Future<void> _sendMessage(String input) async {
  if (input.trim().isEmpty) return;

  String? text =
      "You are my nutritionist, guide me according to the text given below: \n + ${input.trim()}";

  setState(() {
    _messages.add({'text': input, 'sender': "user"});
    _isTyping = true;
  });

  _textController.clear();
  await _saveMessagesToPreferences(); // Save messages after user input

  const String togetherApiKey =
      "81ae561ed7fe920dd7d30a96b82a793feab87f4e3c1cb138d6283f3df5e1e3a8";

  try {
    final response = await http.post(
      Uri.parse('https://api.together.xyz/v1/chat/completions'),
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer $togetherApiKey',
      },
      body: jsonEncode({
        'model': 'meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo',
        'messages': [
```

```
          {'role': 'system', 'content': 'You are a helpful gym mentor.'},
          {'role': 'user', 'content': text},
        ],
      }),
    );

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      final assistantMessage = data['choices'][0]['message']['content'];

      setState(() {
        _messages.add({'text': assistantMessage, 'sender': 'assistant'});
      });

      await _saveMessagesToPreferences(); // Save messages after bot response
    } else {
      throw Exception('Failed to fetch response');
    }
  } catch (error) {
    setState(() {
      _messages.add({
        'text': "I'm sorry, I couldn't process your request at the moment.",
        "sender": "assistant"
      });
    });
  } finally {
    setState(() {
      _isTyping = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.black,
    appBar: AppBar(
      backgroundColor: AppColors.cardColor,
      title: const Text(
        'Diet Assistant',
        style: TextStyle(color: Colors.white),
      ),
      centerTitle: true,
    ),
    body: Padding(
      padding: const EdgeInsets.symmetric(vertical: 60.0),
      child: Column(
        children: [
```

```
// Chat display
Expanded(
  child: ListView.builder(
    padding: const EdgeInsets.all(16.0),
    itemCount: _messages.length + (_isTyping ? 1 : 0),
    itemBuilder: (context, index) {
      if (index == _messages.length && _isTyping) {
        // Typing indicator
        return Align(
          alignment: Alignment.centerLeft,
          child: Container(
            margin: const EdgeInsets.symmetric(vertical: 8.0),
            padding: const EdgeInsets.all(12.0),
            decoration: BoxDecoration(
              color: Colors.grey[850],
              borderRadius: BorderRadius.circular(16),
            ),
            child: Row(
              mainAxisSize: MainAxisSize.min,
              children: List.generate(3, (dotIndex) {
                return AnimatedBuilder(
                  animation: _animationController,
                  builder: (context, child) {
                    return Opacity(
                      opacity: (dotIndex ==
                              (_animationController.value * 3)
                                  .floor() %
                                  3)
                          ? 1
                          : 0.3,
                      child: const Text(
                        '.',
                        style: TextStyle(
                            color: Colors.white, fontSize: 18),
                      ),
                    );
                  },
                );
              }),
            ),
          ),
        );
      }

      final message = _messages[index];
      final isUser = message['sender'] == 'user';

      return Align(
```

```
        alignment:
            isUser ? Alignment.centerRight : Alignment.centerLeft,
        child: Container(
          margin: const EdgeInsets.symmetric(vertical: 8.0),
          padding: const EdgeInsets.all(12.0),
          constraints: BoxConstraints(
            maxWidth: MediaQuery.of(context).size.width * 0.75,
          ),
          decoration: BoxDecoration(
            color: isUser ? Colors.blueGrey[800] : Colors.grey[850],
            borderRadius: BorderRadius.only(
              topLeft: const Radius.circular(16),
              topRight: const Radius.circular(16),
              bottomLeft: Radius.circular(isUser ? 16 : 0),
              bottomRight: Radius.circular(isUser ? 0 : 16),
            ),
          ),
          child: Text(
            message['text']!,
            style: const TextStyle(color: Colors.white),
          ),
        ),
      );
    },
  ),
),
// Input field at the bottom
Container(
  padding:
      const EdgeInsets.symmetric(horizontal: 8.0, vertical: 4.0),
  decoration: BoxDecoration(
    color: Colors.transparent,
    boxShadow: [
      BoxShadow(
        color: Colors.black.withOpacity(0.3),
        offset: const Offset(0, -2),
        blurRadius: 6,
      ),
    ],
  ),
  child: Row(
    crossAxisAlignment: CrossAxisAlignment.center,
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Expanded(
        child: TextField(
          controller: _textController,
          style: const TextStyle(color: Colors.white),
```

```
                decoration: InputDecoration(
                  hintText: 'Type your message...',
                  hintStyle: TextStyle(color: Colors.grey[700]),
                  filled: true,
                  fillColor: AppColors.cardColor,
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(25.0),
                    borderSide: BorderSide.none,
                  ),
                ),
              ),
            ),
            const SizedBox(width: 8),
            IconButton(
              icon: const Icon(Icons.send, color: Colors.black),
              style: ButtonStyle(
                shape: WidgetStatePropertyAll(
                  RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                ),
                backgroundColor:
                    const WidgetStatePropertyAll(Colors.blueAccent),
                padding: const WidgetStatePropertyAll(
                  EdgeInsets.all(13),
                ),
              ),
              onPressed: () => _sendMessage(_textController.text),
            ),
          ],
        ),
      ),
    ),
  ],
 ),
),
   );
  }
}
```

foodList_controller.dart:

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:health_buddy/Controllers/home_controller.dart';
import 'package:health_buddy/Modals/food_modal.dart';
```

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

class FoodListController extends GetxController {
  List<List<FoodModal>> userFoodLists = [];
  List<String> listTitles = [];

  List<FoodModal> foodList = [];
  List<bool> selectedItems = [];


  HomeController homeController = Get.find<HomeController>();

  @override
  void onInit() {
    foodList = homeController.foodList;
    userFoodLists = homeController.userCustomFoodLists;
    listTitles = homeController.userCustomFoodListTitles;

    selectedItems = List<bool>.filled(foodList.length, false);

    super.onInit();
  }

  void getFoodItemList() async {
    foodList = Get.find<HomeController>().foodList;
    selectedItems = List<bool>.filled(foodList.length, false);
    update();
  }

  void createFoodList(String title) {
    List<FoodModal> selectedFoods = [];
    for (int i = 0; i < foodList.length; i++) {
      if (selectedItems[i]) {
        selectedFoods.add(foodList[i]);
      }
    }
    if (selectedFoods.isNotEmpty) {
      userFoodLists.add(selectedFoods);
      listTitles.add(title);
      update();
    }
    saveData();
    update();
  }

  void refineFoodListByTempreature(double tempreature) {
    int tempRange;
    if (tempreature < 36.5) {
```

```
  tempRange = 1;
  Get.snackbar(
    "Temperature Alert",
    "Your body temperature is Low.",
    snackPosition: SnackPosition.TOP,
    backgroundColor: Colors.grey.shade800,
    colorText: Colors.white,
    margin: const EdgeInsets.symmetric(horizontal: 15, vertical: 10),
    borderRadius: 10,
    borderColor: Colors.cyan,
    borderWidth: 2,
    icon: Icon(Icons.thermostat, color: Colors.cyan, size: 24),
    padding: const EdgeInsets.all(16),
    animationDuration: const Duration(milliseconds: 300),
    barBlur: 15,
    isDismissible: true,
    duration: const Duration(seconds: 5),


  );
 }
 else if (tempreature > 37.5){
  tempRange = 2;
  Get.snackbar(
    "Temperature Alert",
    "Your body temperature is High",
    snackPosition: SnackPosition.TOP,
    backgroundColor: Colors.grey.shade900,
    colorText: Colors.white,
    margin: const EdgeInsets.symmetric(horizontal: 15, vertical: 10),
    borderRadius: 10,
    borderColor: Colors.cyan,
    borderWidth: 2,
    icon: Icon(Icons.thermostat, color: Colors.cyan, size: 24),
    padding: const EdgeInsets.all(16),
    animationDuration: const Duration(milliseconds: 300),
    barBlur: 15,
    isDismissible: true,
    duration: const Duration(seconds: 5),
  );
 }
 else{
  tempRange = 0;
  Get.snackbar(
    "Temperature Alert",
    "Your body temperature is Normal.",
    snackPosition: SnackPosition.TOP,
    backgroundColor: Colors.grey.shade800,
```

```
      colorText: Colors.white,
      margin: const EdgeInsets.symmetric(horizontal: 15, vertical: 10),
      borderRadius: 10,
      borderColor: Colors.cyan,
      borderWidth: 2,
      icon: Icon(Icons.thermostat, color: Colors.cyan, size: 24),
      padding: const EdgeInsets.all(16),
      animationDuration: const Duration(milliseconds: 300),
      barBlur: 15,
      isDismissible: true,
      duration: const Duration(seconds: 5),

    );
  }
  List<FoodModal> refinedList = [];
  if(tempRange == 1){
   refinedList =  foodList.where((food) => food.tempRate == 1).toList();
  }
  else if( tempRange == 2){
   refinedList = foodList.where((food) => food.tempRate == 2).toList();
  }
  else{
   refinedList = foodList;
  }

  update();

 }

 void saveData() async {
  SharedPreferences sharedPreferences = await SharedPreferences.getInstance();
  sharedPreferences.setStringList("userCustomFoodListTitles", listTitles);

  List<String> userFoodListsJson = userFoodLists.map((list) {
   return jsonEncode(list.map((food) => food.toJson()).toList());
  }).toList();

  await sharedPreferences.setStringList(
     "userCustomFoodList", userFoodListsJson);
 }

 void updateFoodList(String listname) {}
}
```