Backend Design Doc

1 Database

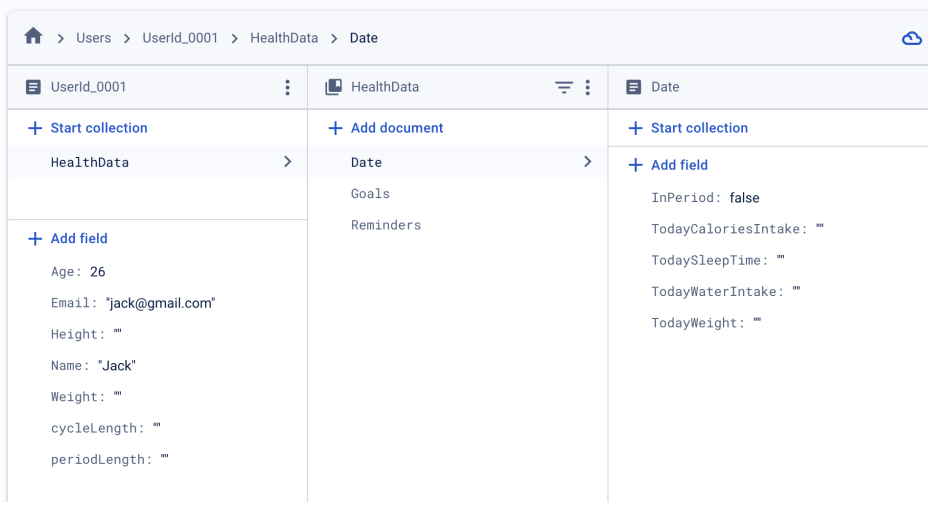
We have decided to select Firestore as our database for the PosiLife app because it offers more flexible queries and a granular data model. This choice aligns well with the specific data requirements of our application.

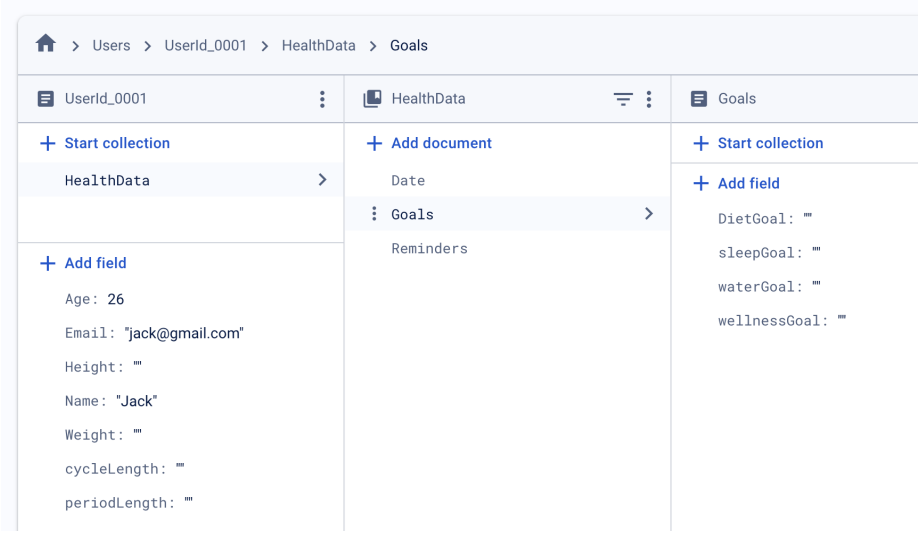
1.1 Design Firestore Structure

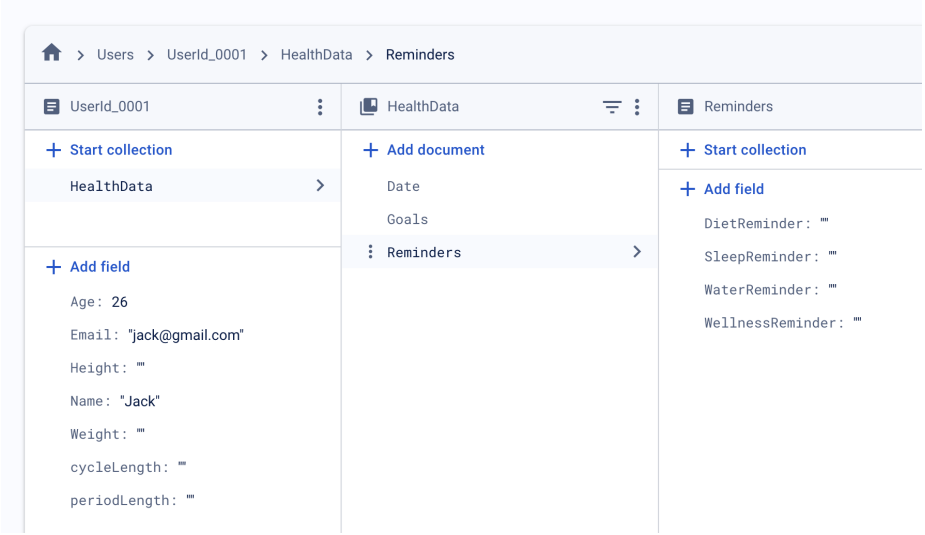
In Firestore, we need to design the data structure by creating **collections** and **documents**. User's data can be stored in the documents as fields.

- User Collection
    - Every user should be a single user document, and use User's UID to be the Document ID.
    - Each user document contains several fundamental fields such as: Name, Email, Weight, Height, Age, as well as specific fields like CycleLength and PeriodLength. Additionally, within the user document, there is a HealthData collection, which in turn contains a document labeled Date. This document is used to store all of the user's daily health data.
    - Since Firebase Authentication handles user authentication securely and stores passwords safely. So we will use Firebase Authentication to handle the email and password fields, instead of store it in the Firestore.

Here is the first version for our app:





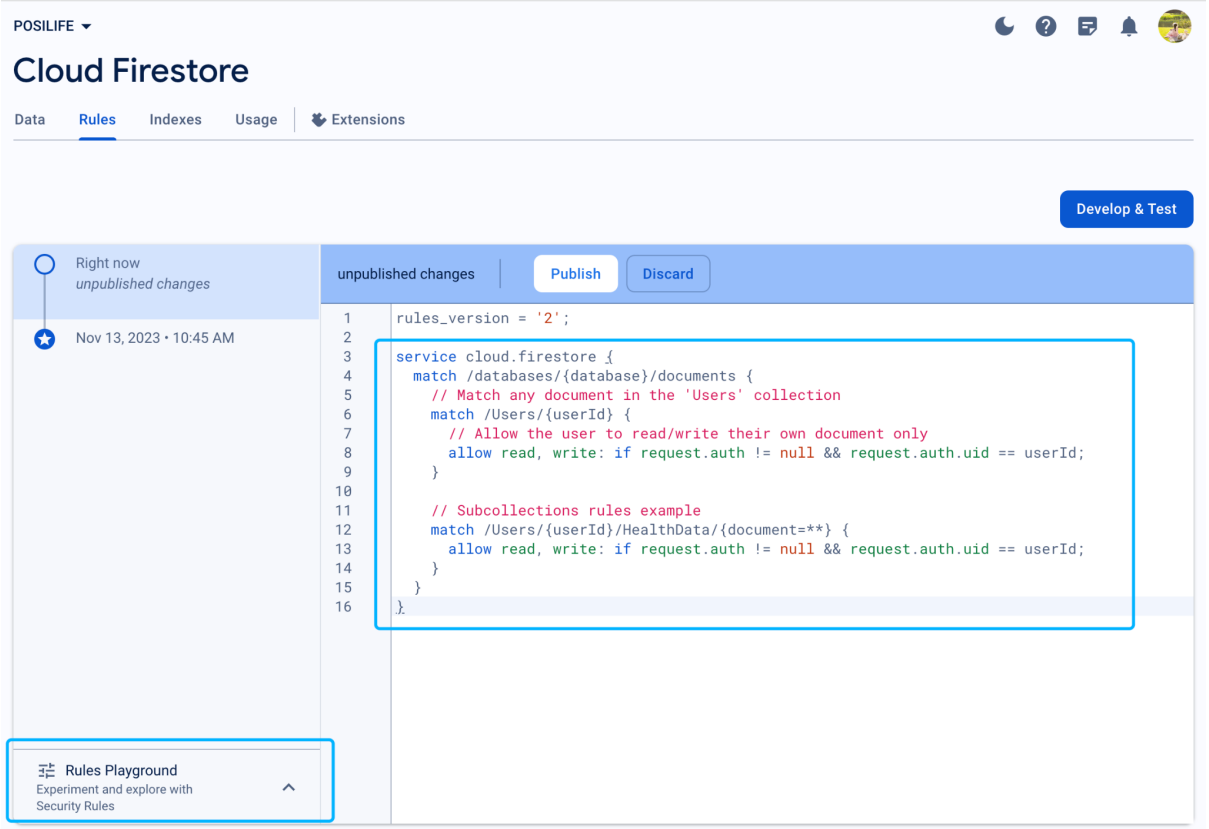


1.2 Implement Security Rules

To ensure the security of user data, we need to determine which data is public and which is private. Therefore, we need to set up Firestore Security Rules to control

access based on user authentication status and ensure that users can only access their own data.

Here is a sample version for our app:



We can test these security rules in the “Rules Playground” before we publish/deploy the rules to our Firestore database.

1.3 Connect with Our Flutter App

Use the **cloud_firestore** package within the app to read from and write to the database within flowing steps:

- Step1: Add the Dependency to “pubspec.yaml” file.
- Step2: Import the package to the Dart file where we want to access Firestore.
- Step3: Create an instance of “FirebaseFirestore” to interact with the Firestore database
- Step4: Read from Firestore by retrieving the document from a collection and “get” data.

- Step5: Write to Firestore by creating or updating documents in a collection and “set” data.
- Step6: Listen for real-time updates by “listen” the documents in a collection.

2 Data Processing Detail

2.1 HomeScreen

*这里这个sweet angle是不是要换成从database获取用户的user name？我先写在这边，用不用都行哈。

```
-            children: [
-              Padding(
-                padding: const EdgeInsets.only(top: 20.0),
-                child: Text(
-                  'Hello, Sweet Angel! 🌻',
-                  style: TextStyle(
```

1）The backend logic for displaying the User’s name on HomeScreen.

- Logic:

  Access the Firestore database, navigate to the User collection, then to the specific user document by the user's ID, and retrieve the user's name.

- Function: fetchUserNameFromFirestore()

  Input:

  userId (string): This is the unique identifier for the user's document in the Users collection.

  Output:

  userName (String) - The name of the user fetched from the database.

- Example:

```
// fetch User Name
import 'package:cloud_firestore/cloud_firestore.dart';

class _HomeScreenState extends State<HomeScreen> {
String userName = ''; // Initial empty string for user's name

@override
void initState() {
super.initState();
fetchUserNameFromFirestore('UserId_0001'); // Replace with the actual user ID
}

Future<void> fetchUserNameFromFirestore(String userId) async {
FirebaseFirestore firestore = FirebaseFirestore.instance;
DocumentSnapshot userDoc = await firestore.collection('Users').doc(userId).get
();

if (userDoc.exists && userDoc.data().containsKey('Name')) {
  setState(() {
    userName = userDoc.data()['Name']; // Set the user's name to the state varia
ble
  });
} else {
  // Handle the case where the user document or name does not exist
  setState(() {
    userName = 'Sweet Angel'; // Fallback to a default name
  });
}
@override
Widget build(BuildContext context) {
return Scaffold(
// ... existing code ...
body: SafeArea(
child: Column(
children: [
Padding(
padding: const EdgeInsets.only(top: 20.0),
child: Text(
'Hello, $userName! 🌻', // Use the userName from the state
style: TextStyle(
// ... existing style ...
```

2.2 Water Section

2.2.1 WaterIntakeScreen

（1）The backend logic for displaying the value of water intake.

- Logic:

Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodayWaterIntake’ field. Similarly, access the ‘Goals’ document to retrieve the ‘WaterGoal’.

- Function: fetchWaterIntakeAndGoal()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple that contains Today's water intake and water Goal.

（2）The backend logic for displaying water intake percentage base on user’s water intake process

- Logic:

  Compare today's water intake against the goal to determine the percentage. If the intake is equal to or greater than the goal, display the completion message; otherwise, display the percentage.

- Function: determineDisplayContent()

  Inputs:

  todayIntake (double): Today's water intake from Firestore.

  goalIntake (double): The water intake goal from Firestore.

  Output:

  waterIntakeProcessRes(String): to display under the pie chart

- Example:

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

class _WaterIntakeScreenState extends State<WaterIntakeScreen> {
  int todayWaterIntake = 0;
  int waterIntakeGoal = 0;

  @override
  void initState() {
    super.initState();
    fetchWaterIntakeAndGoal('UserId_0001'); // Replace with the actual user ID
  }

  Future<void> fetchWaterIntakeAndGoal(String userId) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String todayDateString = getTodayDateString(); // Implement this function ba
sed on your date format

    // Fetch Today's Water Intake
    DocumentSnapshot todayDoc = await firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc(todayDateString)
      .get();

    // Fetch Water Goal
    DocumentSnapshot goalDoc = await firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc('Goals')
      .get();

    if (todayDoc.exists && goalDoc.exists) {
      setState(() {
        todayWaterIntake = int.parse(todayDoc.data()['TodayWaterIntake']);
        waterIntakeGoal = int.parse(goalDoc.data()['WaterGoal']);
      });
    }
  }

  String determineDisplayContent(int todayIntake, int goalIntake) {
    if (todayIntake >= goalIntake) {
      return 'You Completed your goal! 🎉';
    } else {
      double percent = (todayIntake / goalIntake) * 100;
      return '${percent.toStringAsFixed(0)}% of your goal!';
    }
  }

  @override
  Widget build(BuildContext context) {
    // ... existing code ...
    return Padding(
      // ... existing padding ...
      child: CircularPercentIndicator(
        // ... existing properties ...
        percent: todayWaterIntake / waterIntakeGoal.clamp(1, double.infinity),
// Avoid division by zero
        center: Text(
          determineDisplayContent(todayWaterIntake, waterIntakeGoal),
          // ... existing style ...
        ),
        // ... existing properties ...
      ),
    );
    // ... existing code ...
  }
}
```

（3）The backend logic for handling unit change

- Logic:

  Detect the toggle button selection. If the user selects 'ml', the water intake should be displayed in milliliters. If the user selects 'oz', the water intake should be converted from milliliters to ounces and displayed accordingly.

- Function: mlToOz()

  Input:

  currentIntake (double): The current water intake in ml.

  isOunces (bool): A boolean to indicate if the output should be in ounces.

  Output:

  waterIntakeAmount(double): The water intake in the selected unit (either ml or oz).

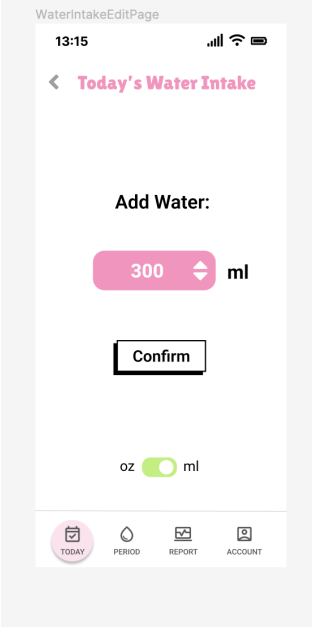
- Example:（完整的waterIntakeScreen在sampleCode.pdf）

```
  double mlToOz(int mlValue) {
    return mlValue / 29.5735;
  }

  String getFormattedValue(int value, bool isOunces) {
    double convertedValue = isOunces ? mlToOz(value) : value.toDouble();
    return isOunces ? '${convertedValue.toStringAsFixed(1)} oz' : '${convertedVa
lue.round()} ml';
  }
```

```
@override
Widget build(BuildContext context) {
  // ... existing code ...

  // ToggleButtons widget or similar toggle UI component
  ToggleButtons(
    children: [Text('ml'), Text('oz')],
    isSelected: _selections,
    onPressed: (int index) {
      setState(() {
        isOunces = index == 1;
        _selections[index] = true;
        _selections[1 - index] = false;
      });
    },
  ),
```

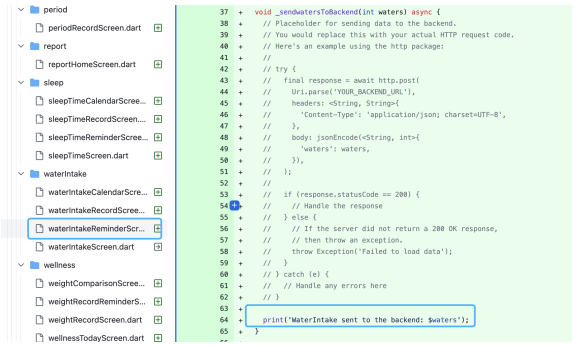
2.2.2 WaterIntakeRecordScreen

这个部分我有一点迷茫，就是我感觉是这个screen是这一页吗，如果是的话，我觉得是不是你写在remindScreen里面的update waterIntake 到 database的部分应该放在这里呀？



2.2.3 WaterIntakeRemindScreen

*这个部分我看你已经写了，但是我有个问题是这个reminder的逻辑是不是变了，不是按定时设置了对吗，就是按用户到达多少饮水量来提醒对嘛？



2.2.4 WaterIntakeCalendarScreen

（1）The backend logic for displaying daily status in water calendar by fetching water intake amount and goal from firestore

Step1 fetch user's water intake amount and goal（这个部分和前面有重复的）

- Logic:

  Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodayWaterIntake’ field. Similarly, access the ‘Goals’ document to retrieve the ‘WaterGoal’.

- Function: fetchWaterIntakeAndGoal()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple or Map containing Today's water intake and waterIntake Goal.

Step2 determine the status for each day

- Function: determineDayStatus(int sleepTime, int sleepGoal)

  Inputs:

  waterIntake (double): The amount of water intake recorded for a given day.

  WaterGoal (double): The user's water goal.

  Output:

  A status code or an object representing the sleep status (e.g., 'completed', 'incomplete', 'notRecorded').

```dart
//updating water calendar satus
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';

class WaterIntakeCalendarScreenState extends State<WaterIntakeCalendarScreen> {
  final String userId = 'actual_user_id'; // Replace with actual user ID logic
  Map<DateTime, String> waterStatusMap = {};

  @override
  void initState() {
    super.initState();
    fetchWaterIntakeAndGoal(userId, DateTime.now().month);
  }

  Future<void> fetchWaterIntakeAndGoal(String userId, int month) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    CollectionReference healthDataCollection = firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData');

    // Get water goal from 'Goals' document
    int waterGoal = 0;
    DocumentSnapshot goalDoc = await healthDataCollection.doc('Goals').get();
    if (goalDoc.exists) {
      waterGoal = goalDoc['WaterGoal'];
    }

    // Get all water intake data for the month
    QuerySnapshot monthSnapshot = await healthDataCollection
      .where('Month', isEqualTo: month.toString().padLeft(2, '0'))
      .get();

    Map<DateTime, String> newWaterStatusMap = {};
    for (var doc in monthSnapshot.docs) {
      int day = int.parse(doc.id.substring(6, 8)); // Assuming the format 'yyyyM
Mdd'
      DateTime date = DateTime(DateTime.now().year, month, day);
      int waterIntake = doc['TodayWaterIntake'];
      String status = determineDayStatus(waterIntake, waterGoal);
      newWaterStatusMap[date] = status;
    }

    setState(() {
      waterStatusMap = newWaterStatusMap;
    });
  }

  String determineDayStatus(int waterIntake, int waterGoal) {
    if (waterIntake == 0) {
      return 'notRecorded'; // No intake recorded
    } else if (waterIntake >= waterGoal) {
      return 'completed'; // Goal met or exceeded
    } else {
      return 'incomplete'; // Intake recorded but goal not met
    }
  }

  // ... Rest of the widget code ...

}
```

2.3 Diet Section

2.3.1 DietIntakeScreen

（1）The backend logic for displaying the value of today's diet intake.

- Logic:

  Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodayCalorieIntake’ field. Similarly, access the ‘Goals’ document to retrieve the ‘DietGoal’.

- Function: fetchDietIntakeAndGoal()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple that contains Today's Diet intake and diet Goal.

* (1)&(2)&(3) 的example,我感觉看是在一个screen就都放在一起了，在（3）

（2）The backend logic for displaying diet intake percentage base on user’s diet intake process

- Logic:

  Compare today's diet intake against the goal to determine the percentage. If the intake is equal to or greater than the goal, display the completion message; otherwise, display the percentage.

- Function: determineDisplayContent()

  Inputs:

  todayIntake (double): Today's diet intake from Firestore.

  goalIntake (double): The diet intake goal from Firestore.

Output:

dietIntakeProcessRes(String): to display under the pie chart

（3）The backend logic for handling unit change

- Logic:

  Detects the toggle button selection. If the user selects 'KJ', the diet intake should be displayed in KJ. If the user selects 'KCal', the diet intake should be converted from ‘KJ’ to 'KCal' and displayed accordingly.

- Function: convertUnitsInDeit()

  Input:

  currentIntake (double): The current diet intake in milliliters.

  isKJ (bool): A boolean to indicate if the output should be in KJ.

  Output:

  deitIntakeAmount(double): The deit intake in the selected unit (either kcal or kj).

- Example:

```dart
//Sleep Time fetch and Percenttage or completion message
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';

class DietIntakeScreenState extends State<DietIntakeScreen> {
  bool isKJ = false; // State variable to track unit toggle
  int todayCaloriesIntake = 0; // Actual intake value, to be fetched from Firest
ore
  int dietGoal = 0; // Actual goal value, to be fetched from Firestore

  @override
  void initState() {
    super.initState();
    fetchDietDataAndGoal('UserId_0001'); // Replace with the actual user ID
  }

  Future<void> fetchDietDataAndGoal(String userId) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String todayDateString = getTodayDateString(); // Implement this function ba
sed on your date format

    // Fetch Today's Calorie Intake
    DocumentSnapshot todayDoc = await firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc(todayDateString)
      .get();

    // Fetch Diet Goal
    DocumentSnapshot goalDoc = await firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc('Goals')
      .get();

    if (todayDoc.exists && goalDoc.exists) {
      setState(() {
        todayCaloriesIntake = int.parse(todayDoc.data()['TodayCaloriesIntake']);
        dietGoal = int.parse(goalDoc.data()['DietGoal']);
      });
    }
  }

  double convertKcalToKJ(int kcal) {
    return kcal * 4.184; // 1 kcal = 4.184 kJ
  }

  String getFormattedIntake() {
    double intakeValue = isKJ ? convertKcalToKJ(todayCaloriesIntake) : todayCalo
riesIntake.toDouble();
    return isKJ ? '${intakeValue.toStringAsFixed(0)} kJ' : '${intakeValue.round
()} kcal';
  }

  String getFormattedGoal() {
    double goalValue = isKJ ? convertKcalToKJ(dietGoal) : dietGoal.toDouble();
    return isKJ ? '${goalValue.toStringAsFixed(0)} kJ' : '${goalValue.round()} k
cal';
  }

  @override
  Widget build(BuildContext context) {
    // ... existing code ...

    // ToggleButton widget or similar UI component for kcal to kJ conversion
    // ...

    // Display the calorie intake and goal in the selected unit
    Text('Your calorie intake today: ${getFormattedIntake()}'),
    Text('Your calorie intake goal: ${getFormattedGoal()}'),

    // ... rest of your build method code ...
  }

  String getTodayDateString() {
    // Implement this function based on your date format
    DateTime now = DateTime.now();
    return "${now.year}-${now.month}-${now.day}";
  }
}
```

2.3.2 DietIntakeRecordScreen

（1）The backend logic for updating sleep time to firestore

- Logic:

  When the user confirms their diet intake, this function will be called to update the corresponding field in the Firestore database.

- Function:confirmCalorieIntake()

  Inputs:

  userId(String): The user’s unique identifier

  date(String): The string representing the date for the record to be updated.

  diet(int): The amount of diet intake modified by user and need to be updated

  Output:

  update the user's diet intake record for the given date in Firestore.

- Example:

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

class DietIntakeRecordScreen extends StatefulWidget {
  // Your existing widget code...
}

class _DietIntakeRecordScreenState extends State<DietIntakeRecordScreen> {
  // Your existing state code...

  void _confirmCalorieIntake() async {
    // Get the calorie intake from the text field and convert it if necessary
    int calorieIntake = int.parse(_calorieAmountController.text.trim());
    if (!isKcal) {
      // If the switch is on KJ, convert it to Kcal
      calorieIntake = (calorieIntake / 4.184).round(); // 1KJ = 0.239005736 Kcal
    }

    // Get the current date as a string
    String todayDateString = getTodayDateString(); // Implement this based on yo
ur date format

    // Update Firestore with the new calorie intake
    await FirebaseFirestore.instance
        .collection('Users')
        .doc('actual_user_id') // Replace with the actual user ID
        .collection('HealthData')
        .doc(todayDateString) // Use the appropriate document id for the current
day
        .update({
          'TodayCaloriesIntake': calorieIntake.toString(),
        })
        .then((_) => print('Calorie intake updated'))
        .catchError((error) => print('Failed to update calorie intake: $erro
r'));

    Navigator.of(context).pop();
  }

  // Rest of the code...

  String getTodayDateString() {
    // Implement this function based on your date format
    DateTime now = DateTime.now();
    return "${now.year}-${now.month.toString().padLeft(2, '0')}-${now.day.toStri
ng().padLeft(2, '0')}";
  }
}
```

2.3.3 DietIntakeReminderScreen

（1）The backend logic for updating Reminder to firestore

- Logic:

  When the user confirms their desired diet reminder time, this function will update the ‘DietReminder’ field in the Firestore "Reminders" document for the user.

- Function: _sendDietRinderToBackend()

  Inputs:

  userId(String): The user’s unique identifier

  reminderTime(TimeOfDay): The time user wants to be reminded to control diet

  Output:

  update the user's diet reminder time in Firestore.

- Example:

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

class DietIntakeReminderScreen extends StatefulWidget {
  @override
  _DietIntakeReminderScreenState createState() => _DietIntakeReminderScreenState
();
}

class _DietIntakeReminderScreenState extends State<DietIntakeReminderScreen> {
  final TextEditingController _calorieController = TextEditingController();
  // Other state variables...

  void _sendDietRinderToBackend(int calories) async {
    String userId = 'actual_user_id'; // TODO: Replace with actual logic to retr
ieve user ID
    FirebaseFirestore firestore = FirebaseFirestore.instance;

    // Check if the calorie value is for 'Kcal' or convert if it's 'KJ'
    if (!isKcal) {
      calories = (calories / 4.184).round(); // Convert KJ to Kcal if needed
    }

    // Update the 'DietReminder' field in Firestore for the user's document
    await firestore
        .collection('Users')
        .doc(userId)
        .collection('HealthData')
        .doc('Reminders')
        .set({
          'DietReminder': calories, // Assuming you want to store the value as a
n integer
        })
        .then((_) {
          // Handle the success - maybe navigate back or show a confirmation mes
sage
          print('Diet reminder updated');
        })
        .catchError((error) {
          // Handle any errors here
          print('Failed to update diet reminder: $error');
        });
  }

  @override
  Widget build(BuildContext context) {
    // Your existing build method...
    ElevatedButton(
      onPressed: () {
        int? calories = int.tryParse(_calorieController.text);
        if (calories != null) {
          _sendCaloriesToBackend(calories);
          // Clear the controller after setting the reminder
          _calorieController.clear();
        } else {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text('Please enter a valid number for calorie
s.')),
          );
        }
      },
      child: Text('Confirm'),
      // Your existing button style...
    ),
    // Rest of your build method...
  }
}
```

2.3.4 DietIntakeCalendarScreen

（1）The backend logic for displaying daily status in Diet calendar by fetching diet intake and goal from firestore

Step1 fetch user's diet Intake and goal

- Logic:

  Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodayDietIntake’ field. Similarly, access the ‘Goals’ document to retrieve the ‘DietGoal’.

- Function: fetchDietIntakeAndGoal()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple that contains Today's calorie intake and diet Goal.

Step2 determine the status for each day

- Function: determineDayStatus(int sleepTime, int sleepGoal)

  Inputs:

  dietIntake(int): The amount of diet intake recorded for a given day.

  diertGoal (int): The user's diet goal.

  Output:

  A status code or an object representing the diet status (e.g., 'completed', 'incomplete', 'notRecorded').

- Example:

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

class DietIntakeCalendarScreenState extends State<DietIntakeCalendarScreen> {
  final String userId = 'actual_user_id'; // Replace with actual user ID logic
  Map<DateTime, String> DietStatusMap = {};

  @override
  void initState() {
    super.initState();
    fetchDietIntakeAndGoal(userId, DateTime.now().month);
  }

  Future<void> fetchWaterIntakeAndGoal(String userId, int month) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    CollectionReference healthDataCollection = firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData');

    // Get diet goal from 'Goals' document
    int dietGoal = 0;
    DocumentSnapshot goalDoc = await healthDataCollection.doc('Goals').get();
    if (goalDoc.exists) {
      dietGoal = goalDoc['DietGoal'];
    }

    // Get all water intake data for the month
    QuerySnapshot monthSnapshot = await healthDataCollection
      .where('Month', isEqualTo: month.toString().padLeft(2, '0'))
      .get();

    Map<DateTime, String> newDietStatusMap = {};
    for (var doc in monthSnapshot.docs) {
      int day = int.parse(doc.id.substring(6, 8)); // Assuming the format 'yyyyM
Mdd'
      DateTime date = DateTime(DateTime.now().year, month, day);
      int dietIntake = doc['TodayDietIntake'];
      String status = determineDayStatus(dietIntake, dietGoal);
      newDietStatusMap[date] = status;
    }

    setState(() {
      dietStatusMap = newDietStatusMap;
    });
  }

  String determineDayStatus(int dietIntake, int dietGoal) {
    if (dietIntake == 0) {
      return 'notRecorded'; // No intake recorded
    } else if (dietIntake <= dietGoal) {
      return 'completed'; // Goal met or exceeded
    } else {
      return 'incomplete'; // Intake recorded but goal not met
    }
  }

  // ... Rest of the widget code ...

}
```

2.4 Sleep Section

2.4.1 SleepTimeScreen

（1）The backend logic for fetching sleep time and goal from firestore

- Logic:

  Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodaySleepTime’ field. Similarly, access the ‘Goals’ document to retrieve the ‘SleepGoal’.

- Function: fetchSleepTimeAndGoal()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple or Map containing Today's Sleep Time and Sleep Goal.

* 关于fetch这部分我感觉其实可以单独写一个helper class，因为后面calendar里也要用，相当于4个section都要重复写一下，但我感觉不写也没事，我感觉卤蛋也不会仔细看源代码……

如果把获取当天的睡眠时间以及睡眠时长目标单独拿出来，那就可以把：未记录，记录未达标，记录并达标写成0，1，2 三种状态；然后这三个状态在SleepTimeScreen和SleepTimeCalendarScreen都能用。我现在是每个screen单独写了的，你看看哪种更合适？

* (1) 和（2）的example 都写在（2）里面了

（2）The backend logic for displaying sleep process percentage or completion message

- Logic:

  Compare today's sleep time against the goal to determine the percentage. If the sleep time is equal to or greater than the goal, display the completion message; otherwise, display the percentage.

- Function: determineDisplayContent(int todaySleepTime, int goalSleepTime)

Inputs:

todaySleepTime (int): Today's sleep time from Firestore.

goalSleepTime (int): The sleep time goal from Firestore.

Output:

sleepTimeProcessRes(String): to display under the pie chart

- Example:

```dart
//Sleep Time fetch and Percenttage or completion message

import 'package:cloud_firestore/cloud_firestore.dart';

class _WaterIntakeScreenState extends State<WaterIntakeScreen> {
  int todayWaterIntake = 0;
  int waterIntakeGoal = 0;

  @override
  void initState() {
    super.initState();
    fetchWaterIntakeAndGoal('UserId_0001'); // Replace with the actual user ID
  }

  Future<void> fetchWaterIntakeAndGoal(String userId) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String todayDateString = getTodayDateString(); // Implement this function ba
sed on your date format

    // Fetch Today's Water Intake
    DocumentSnapshot todayDoc = await firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc(todayDateString)
      .get();

    // Fetch Water Goal
    DocumentSnapshot goalDoc = await firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc('Goals')
      .get();

    if (todayDoc.exists && goalDoc.exists) {
      setState(() {
        todayWaterIntake = int.parse(todayDoc.data()['TodayWaterIntake']);
        waterIntakeGoal = int.parse(goalDoc.data()['WaterGoal']);
      });
    }
  }

  String determineDisplayContent(int todayIntake, int goalIntake) {
    if (todayIntake >= goalIntake) {
      return 'You Completed your goal! 🎉';
    } else {
      double percent = (todayIntake / goalIntake) * 100;
      return '${percent.toStringAsFixed(0)}% of your goal!';
    }
  }

  @override
  Widget build(BuildContext context) {
    // ... existing code ...
    return Padding(
      // ... existing padding ...
      child: CircularPercentIndicator(
        // ... existing properties ...
        percent: todayWaterIntake / waterIntakeGoal.clamp(1, double.infinity),
// Avoid division by zero
        center: Text(
          determineDisplayContent(todayWaterIntake, waterIntakeGoal),
          // ... existing style ...
        ),
        // ... existing properties ...
      ),
    );
    // ... existing code ...
  }
}
```

2.4.2 SleepTimeRecordScreen

（1）The backend logic for updating sleep time to firestore

- Logic:

  When the user confirms their sleep time, this function will be called to update the corresponding field in the Firestore database.

- Function:updateSleepTimeInFirestore()

  Inputs:

  userId(String): The user’s unique identifier

  date(String): The string representing the date for the record to be updated.

  sleepTime(String): The number of hours modified by user and need to be updated

  Output:

  update the user's sleep time record for the given date in Firestore.

- Example:

```dart
//updating sleep time to firestore

import 'package:cloud_firestore/cloud_firestore.dart';

class _SleepTimeRecordScreenState extends State<SleepTimeRecordScreen> {
  int quantity = 0; // Initialize with 0 as per the requirement

  void incrementQuantity() {
    setState(() {
      quantity++;
    });
  }

  void decrementQuantity() {
    if (quantity > 0) {
      setState(() {
        quantity--;
      });
    }
  }

  Future<void> updateSleepTimeInFirestore(String userId, String dateString, int
sleepTime) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;

    // Get the reference to the user's document for today's date
    DocumentReference dateDocRef = firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc(dateString);

    // Update the document with the new sleep time
    return dateDocRef
      .set({'TodaySleepTime': sleepTime.toString()}, SetOptions(merge: true))
      .then((_) => print("Sleep time updated"))
      .catchError((error) => print("Failed to update sleep time: $error"));
  }

  @override
  Widget build(BuildContext context) {
    // ... existing widget code ...

    // The button to add SleepTime intake record will go here
    GestureDetector(
      onTap: () {
        // TODO: Replace 'UserId_0001' and 'todayDateString' with actual user ID
and date string
        updateSleepTimeInFirestore('UserId_0001', getTodayDateString(), quantit
y).then((_) {
          // After updating, pop back to the previous screen (presumably the Tod
ay-Sleep page)
          Navigator.of(context).pop();
        });
      },
    );

    // ... remaining widget code ...
  }

  String getTodayDateString() {
    // Implement this function to match the date format used in Firestore docume
nt IDs
    DateTime now = DateTime.now();
    return "${now.year.toString().padLeft(4, '0')}-${now.month.toString().padLef
t(2, '0')}-${now.day.toString().padLeft(2, '0')}";
  }
}
```

2.4.3 SleepTimeReminderScreen

（1）The backend logic for updating Reminder to firestore

- Logic:

  When the user confirms their desired sleep reminder time, this function will update the ‘sleepReminder’ field in the Firestore "Reminders" document for the user.

- Function:updateSleepReminderInFirestore()

  Inputs:

  userId(String): The user’s unique identifier

  reminderTime(TimeOfDay): The time user wants to be reminded to sleep

  Output:

  update the user's sleep reminder time in Firestore.

- Example:

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

class SleepTimeReminderScreen extends StatefulWidget {
  @override
  _SleepTimeReminderScreenState createState() => _SleepTimeReminderScreenState
();
}

class _SleepTimeReminderScreenState extends State<SleepTimeReminderScreen> {
  TimeOfDay selectedTime = TimeOfDay.now();

  Future<void> updateSleepReminderTime(String userId, TimeOfDay reminderTime) as
ync {
    FirebaseFirestore firestore = FirebaseFirestore.instance;

    // Get the reference to the 'Reminders' document for the user
    DocumentReference remindersDocRef = firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData')
      .doc('Reminders');

    // Convert TimeOfDay to a format that can be stored in Firestore
    Map<String, dynamic> reminderData = {
      'SleepReminder': '${reminderTime.hour.toString().padLeft(2, '0')}:${remind
erTime.minute.toString().padLeft(2, '0')}',
    };

    // Update the 'sleepReminder' field in the document
    return remindersDocRef
      .set(reminderData, SetOptions(merge: true))
      .then((_) => print("Sleep reminder time updated"))
      .catchError((error) => print("Failed to update sleep reminder time: $erro
r"));
  }

  @override
  Widget build(BuildContext context) {
    // ... existing widget code ...

    // Confirm button
    Center(
      child: ElevatedButton(
        onPressed: () {
          // TODO: Replace 'UserId_0001' with actual user ID
          updateSleepReminderTime('UserId_0001', selectedTime).then((_) {
            // After updating, pop back to the previous screen (presumably the T
oday-Sleep page)
            Navigator.of(context).pop();
          });
        },
        child: Text('Confirm'),
        // ... existing button style ...
      ),
    ),

    // ... remaining widget code ...
  }
}
```

2.4.4 SleepTimeCalendarScreen

（1）The backend logic for displaying daily status in sleep calendar by fetching sleep time and goal from firestore

Step1 fetch user's sleep time and goal（这个部分和前面有重复的）

- Logic:

  Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodaySleepTime’ field. Similarly, access the ‘Goals’ document to retrieve the ‘SleepGoal’.

- Function: fetchSleepTimeAndGoal()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple or Map containing Today's Sleep Time and Sleep Goal.

Step2 determine the status for each day

- Function: determineDayStatus(int sleepTime, int sleepGoal)

  Inputs:

  sleepTime (int): The amount of sleep recorded for a given day.

  sleepGoal (int): The user's sleep goal.

  Output:

  A status code or an object representing the sleep status (e.g., 'completed', 'incomplete', 'notRecorded').

- Example:

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

class SleepTimeCalendarScreenState extends State<SleepTimeCalendarScreen> {
  final String userId = 'actual_user_id'; // Replace with actual user ID logic
  Map<DateTime, String> sleepStatusMap = {};

  @override
  void initState() {
    super.initState();
    fetchSleepDataAndGoal(userId, DateTime.now().month);
  }

  Future<void> fetchSleepDataAndGoal(String userId, int month) async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    CollectionReference healthDataCollection = firestore
      .collection('Users')
      .doc(userId)
      .collection('HealthData');

    // Get sleep goal from 'Goals' document
    int sleepGoal = 0;
    DocumentSnapshot goalDoc = await healthDataCollection.doc('Goals').get();
    if (goalDoc.exists) {
      sleepGoal = goalDoc['SleepGoal'];
    }

    // Get all sleep data for the month
    QuerySnapshot monthSnapshot = await healthDataCollection
      .where('Month', isEqualTo: month.toString().padLeft(2, '0'))
      .get();

    Map<DateTime, String> newSleepStatusMap = {};
    for (var doc in monthSnapshot.docs) {
      int day = int.parse(doc.id.substring(6, 8)); // Assuming the format 'yyyyM
Mdd'
      DateTime date = DateTime(DateTime.now().year, month, day);
      int sleepTime = doc['TodaySleepTime'];
      String status = determineDayStatus(sleepTime, sleepGoal);
      newSleepStatusMap[date] = status;
    }

    setState(() {
      sleepStatusMap = newSleepStatusMap;
    });
  }

  String determineDayStatus(int sleepTime, int sleepGoal) {
    if (sleepTime == 0) {
      return 'notRecorded';
    } else if (sleepTime >= sleepGoal) {
      return 'completed';
    } else {
      return 'incomplete';
    }
  }

  // ... Rest of your widget code ...
}
```

2.5 Wellness Section

2.5.1 WellnessTodayScreen

（1）The backend logic for fetching today’s weight from firestore

- Logic:

  Access the Firestore database, navigate to the ‘Users’ collection, then to the specific user document, inside the ‘HealthData’ subcollection get the ‘Date’ document for today's date, and fetch the ‘TodayWeight’ field.

- Function: fetchWeightAndHeight()

  Input:

  userId (string): This is the unique identifier for the user's document in the ‘Users’ collection.

  Output:

  Tuple that contains Today's weight and User’s height .

（2）The backend logic for unit conversion if needed

- Logic:

  Detect the toggle button selection. If the user selects 'kg', the weight should be displayed in kg. If the user selects 'lb', the weight should be converted from kg to lb and displayed accordingly.

- Function1: kgToLb()

  Input:

  currentWeight (double): The current weight in kg.

  isKg (bool): A boolean to indicate if the output should be in kg.

  Output:

  currentWeight(double): The weight in the selected unit (either kg or lb).

- Function2: getFormattedValue()

  Input:

currentWeight (double): The current weight in kg.

isKg (bool): A boolean to indicate if the output should be in kg.

Output:

display the weight in correct version

- Example:

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

class _WellnessTodayScreenState extends State<WellnessTodayScreen> {
  // ... Your existing variables ...
 bool isKg = true; // State variable to track unit toggle
  List<bool> _selections = [true, false]; // Initial selection is set to 'kg'
double currentWeight = 0.0; // This will hold the weight fetched from Firestore
  double height = 0.0; // Height in meters

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

//Today weight and Height fetch
  void fetchCurrentWeightAndHeight() async {
  // Fetch the user's document from Firestore
  var userDocument = await FirebaseFirestore.instance.collection('Users').doc('u
ser_id').get();
  var healthDataDocument = await FirebaseFirestore.instance
      .collection('Users')
      .doc('user_id')
      .collection('HealthData')
      .doc('Date')
      .get();

  if (userDocument.exists && healthDataDocument.exists) {
    setState(() {
      // Assuming height is stored in centimeters and weight in kilograms
      // in the user's Firestore document
      height = userDocument.data()['Height'] / 100; // Convert cm to meters for
BMI calculation
      currentWeight = double.tryParse(healthDataDocument.data()['TodayWeight'].t
oString()) ?? 0.0;
    });
  }
}
  // toggle unit goes here
  double kgToLb(int kgValue) {
    return kgValue * 2.2046;
  }

  // display in correct unit
  String getFormattedValue(int value, bool isKg) {
    double convertedValue = isKg ?  value.toDouble() : kgToLb(value);
    return isKg ? '${convertedValue.round()} kg' : '${convertedValue.toStringAsF
ixed(1)} lb';
  }

  @override
  Widget build(BuildContext context) {
    // ... existing code ...

    // ToggleButtons widget or similar toggle UI component
    ToggleButtons(
      children: [Text('kg'), Text('lb')],
      isSelected: _selections,
      onPressed: (int index) {
        setState(() {
          isLb = index == 1;
          _selections[index] = true;
          _selections[1 - index] = false;
        });
      },
    ),

    // Display the weight and goal in the selected unit
    Text('Your weight today: ${getFormattedValue(currentWeight, isLb)}'),
  double calculateBMI(double weight, double height) {
    return weight / (height * height);
  }

  double calculateBMR(double weight) {
    // This is a simplified BMR calculation. Please use the correct formula that
applies to your case.
    return 10 * weight + 6.25 * height * 100 - 5 * age + -161;
  }

  // ... Rest of your widget code ...

  @override
  Widget build(BuildContext context) {
    // When displaying weight
    String weightText = isKg ? '${currentWeight.toStringAsFixed(1)} kg' : '${(cu
rrentWeight * 2.20462).toStringAsFixed(1)} lbs';
    // Calculate BMI and BMR
    double bmi = calculateBMI(currentWeight, height);
    double bmr = calculateBMR(currentWeight);

    // Rest of your build method
    // ...
  }
}
```

2.5.2 WeightRecordScreen

*这个页面有个问题是在wellnessToday页面是默认isKg是true的，但是这个页面默认isKg是false

（1）**The backend logic for updating today weight to firestore**

- Logic:

  When the user confirms their weight, this function will be called to update the corresponding field in the Firestore database.

- Function: _confirmWeight()

  Inputs:

  userId(String): The user’s unique identifier

  date(String): The string representing the date for the record to be updated.

  weight(number): The weight modified by user and need to be updated

  Output:

  update the user's weight record for the given date in Firestore.

- Example:

```dart
void _confirmWeight() async {
    // Convert string to double for weight
    double weight = double.tryParse(_weightAmountController.text) ?? 0.0;
    if (!isKg) {
      // Convert to Kg if the input was in lbs (1 lb = 0.453592 kg)
      weight *= 0.453592;
    }

    // Replace with the user's actual ID
    String userId = FirebaseAuth.instance.currentUser?.uid ?? '';

    // Get today's date as a string
    String today = DateTime.now().toString().substring(0, 10);

    // Update the user's weight in Firestore
    await FirebaseFirestore.instance
        .collection('Users')
        .doc(userId)
        .collection('HealthData')
        .doc(today)
        .set({'TodayWeight': weight}, SetOptions(merge: true));

    // Confirmation message or navigation
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Weight updated successfully!')),
    );

    // Clear the text field
    _weightAmountController.clear();
  }
```

2.5.3 WeightRecordReminderScreen

（1）The backend logic for updating Reminder to firestore

- Logic:

  When the user confirms their desired wellness record reminder time, this function will update the ‘wellnessReminder’ field in the Firestore "Reminders" document for the user.

- Function:updateWellnessReminderInFirestore()

  Inputs:

  userId(String): The user’s unique identifier

  reminderTime(TimeOfDay): The time user wants to be reminded to record weight

  Output:

  update the user's wellness reminder time in Firestore.

- Example:

```
  void updateWellnessReminderInFirestore() async {

    String reminderTime = formatTimeOfDay(selectedTime);
    String userId = FirebaseAuth.instance.currentUser?.uid ?? ''; // Replace wit
h the actual user ID

    // Update the reminder time in Firestore
    await FirebaseFirestore.instance
        .collection('Users')
        .doc(userId)
        .collection('Reminders')
        .doc('weightReminder') // Assuming you have a document for weight remind
ers
        .set({'reminderTime': reminderTime}, SetOptions(merge: true));
  }
```

2.5.4 WeightComparisonScreen

（1）The backend logic for comparing Today's weight and BMI with last time's data

- Logic:

  The WeightComparisonScreen fetches and compares the current and last recorded weights and BMI values for the user from Firestore. It calculates the changes in weight and BMI since the last record.

- Function:compareWithLastRecord()

  Inputs:

  userId(String): The user's unique identifier

  Output:

  'currentWeight' (double): The most recent weight record.

  'lastWeight' (double): The weight record before the most recent one.

  'changeInWeight' (double): The difference between currentWeight and lastWeight.

  'currentBMI' (double): The BMI calculated from the most recent weight record.

  'lastBMI' (double): The BMI calculated from the weight record before the most recent one.

  'changeInBMI' (double): The difference between currentBMI and lastBMI.

  Display 'currentWeight', 'changeInWeight', 'changeInBMI' on the screen

- Example:

```
    // Fetch today's weight and the last recorded weight
    var weightCollection = FirebaseFirestore.instance.collection('Users').doc(us
erId).collection('WeightRecords');
    var weightDocs = await weightCollection.orderBy('Date', descending: true).li
mit(2).get();

    if (weightDocs.docs.length > 0) {
      currentWeight = weightDocs.docs.first.data()['Weight'];
      currentBMI = calculateBMI(currentWeight, height);

      if (weightDocs.docs.length > 1) {
        lastWeight = weightDocs.docs[1].data()['Weight'];
        lastBMI = calculateBMI(lastWeight, height);
      }

      // Calculate the change in weight and BMI
      if (lastWeight != null && lastBMI != null) {
        changeInWeight = currentWeight! - lastWeight!;
        changeInBMI = currentBMI! - lastBMI!;
      }

      setState(() {});
    }
  }
```