

# IFIT STUDIO

ANUSHKA NARSIMA - MO0851749  
RURAMAI MUCHENGA - MO0912425  
JOELLA JOSE - MO0850468

PDE 2101  
ENGINEERING SOFTWARE DEVELOPMENT



# OVERVIEW

1

ABSTRACT

2

PROBLEM DEFINITION & BENEFITS

3

REQUIREMENTS SPECIFICATION

4

TECHNOLOGIES & DEVELOPMENT TOOLS

5

FLOWCHART

6

PROJECT MANAGEMENT DIAGRAMS

7

IMPLEMENTATION

8

CONCLUSION



# ABSTRACT

Our project introduces an innovative IoT-based fitness management system comprising a smartwatch and a companion mobile application.

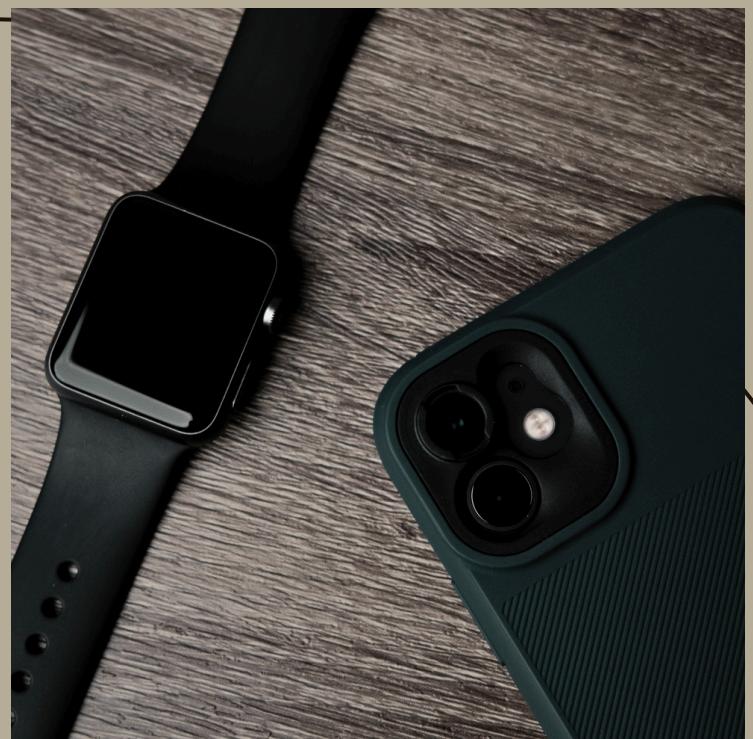
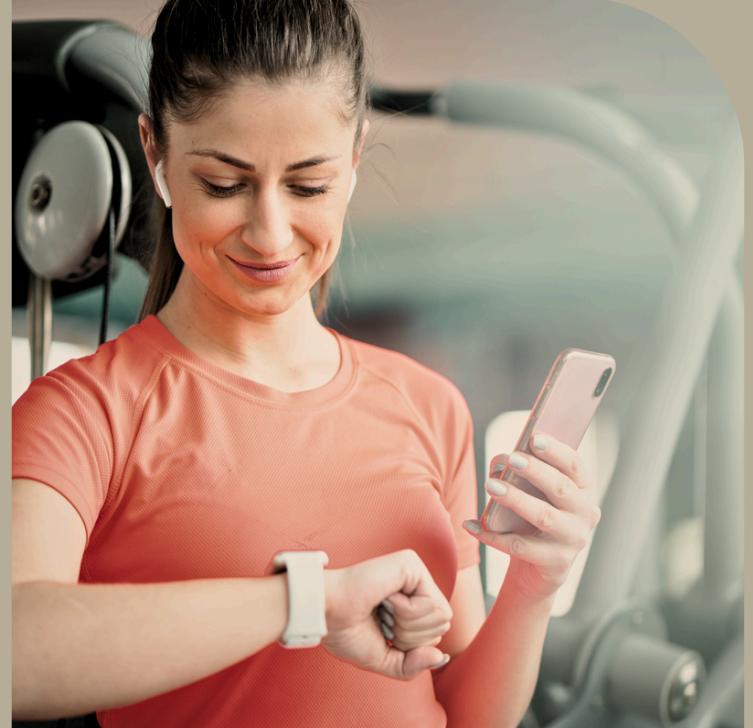
The system seamlessly integrates these components to enhance the fitness tracking experience and optimize gym operations.

# PROBLEM DEFINITION

In the realm of fitness management, a key challenge is seamlessly integrating equipment availability tracking and workout guidance.

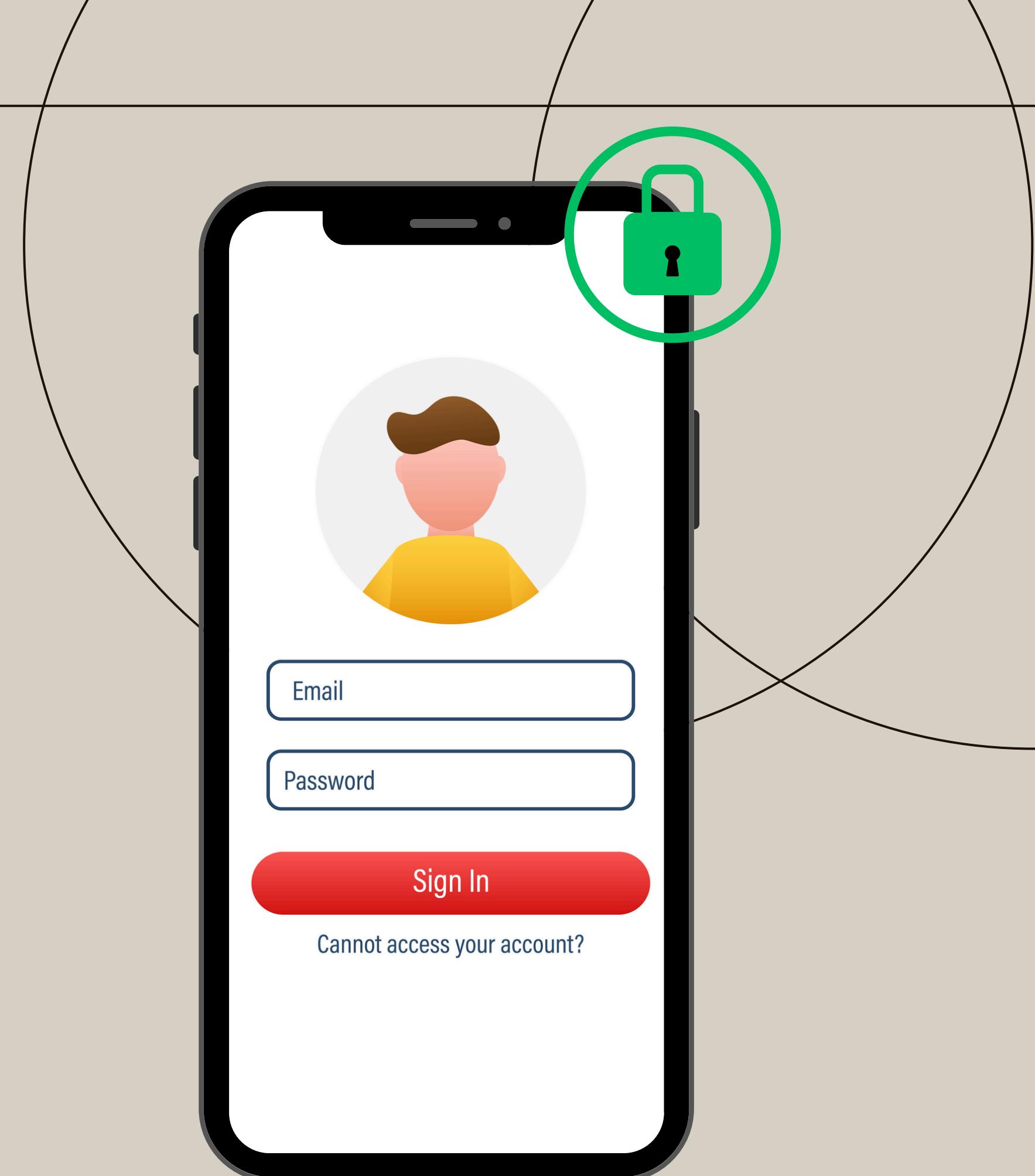
Based on research from Healthline and IEEE Xplore, we can effectively address these challenges by developing a comprehensive IoT-based fitness management system that displays present and upcoming workouts on a smartwatch.

This system aims to streamline gym operations, optimise equipment usage, and enhance the overall fitness experience for users.



# BENEFITS

- Exercise Tracking: Tracks exercise with minimal data for efficient monitoring and progress tracking.
- New workout addition Feature: No current products have this option, & it is essential since exercises are inexhaustive.
- Personalized Workout Guidance: The smartwatch displays present and upcoming exercises, providing personalized guidance tailored to users' fitness goals, keeping them motivated and on track.
- Improved Community Engagement: The mobile app's group chat fosters a sense of community, enabling users to share tips, motivation, and fitness-related discussions, promoting accountability.



## FUNCTIONAL REQUIREMENTS:

- › Workout tracking on the smartwatch
- › Plan Generation
- › Present and following scheduled exercises displayed based on the user's workout plan
- › User authentication on the mobile application
- › Group chat feature for community engagement

## NON-FUNCTIONAL REQUIREMENTS:

- › Reliability and accurate data tracking
- › User-friendly interfaces on smartwatch and mobile app
- › Low latency in displaying real-time data and notifications
- › Robust user authentication mechanisms
- › Safe keeping of sensitive data
- › Scalable to accommodate future updates and expanding user base

# REQUIREMENTS SPECIFICATION



# TECHNOLOGIES & DEVELOPMENT TOOLS

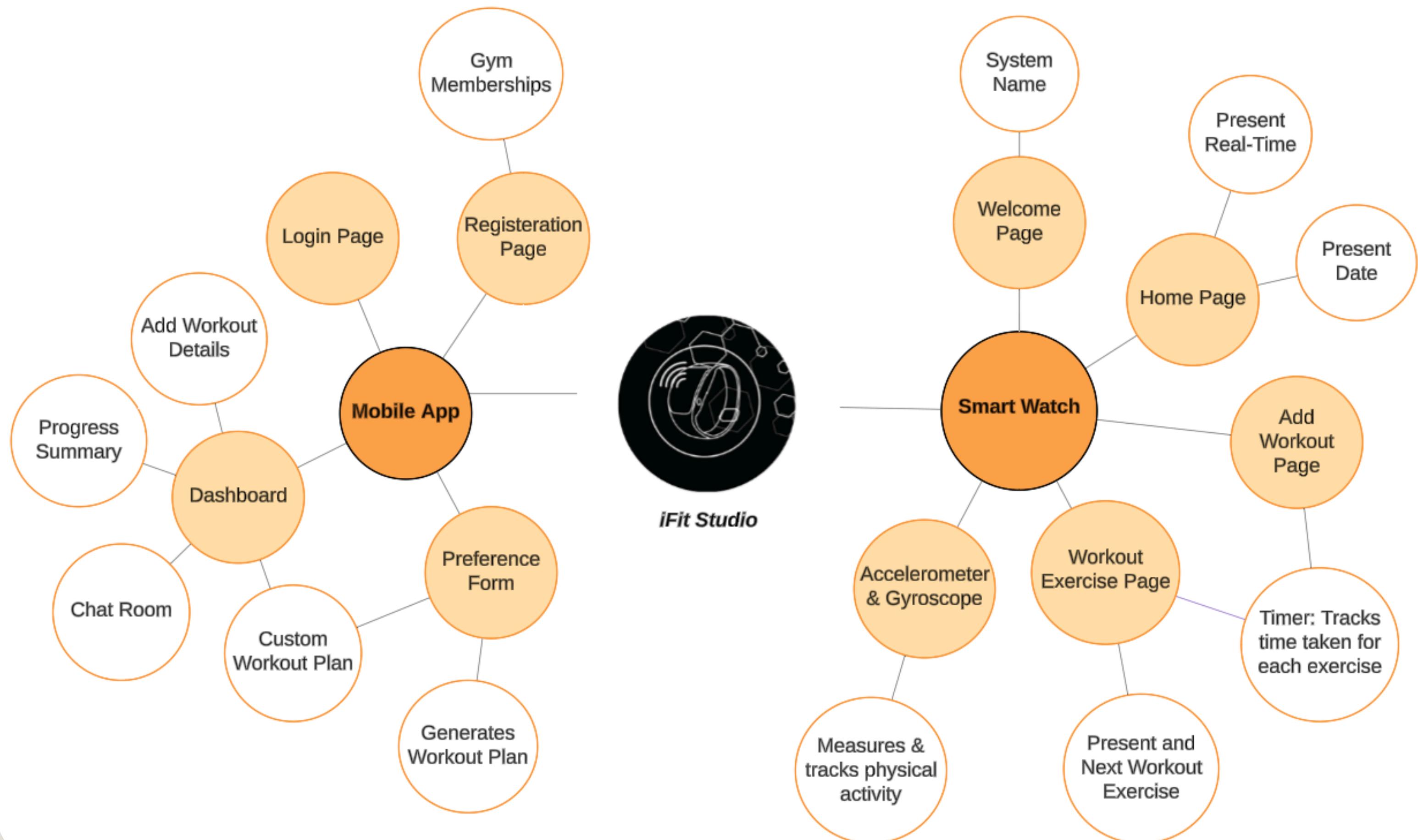
## SOFTWARE

- › Flutter
- › Firebase Firestore (NoSQL DB)
- › Arduino IDE
- › Python – Flask, Matplotlib, Pandas
- › Git/GitHub

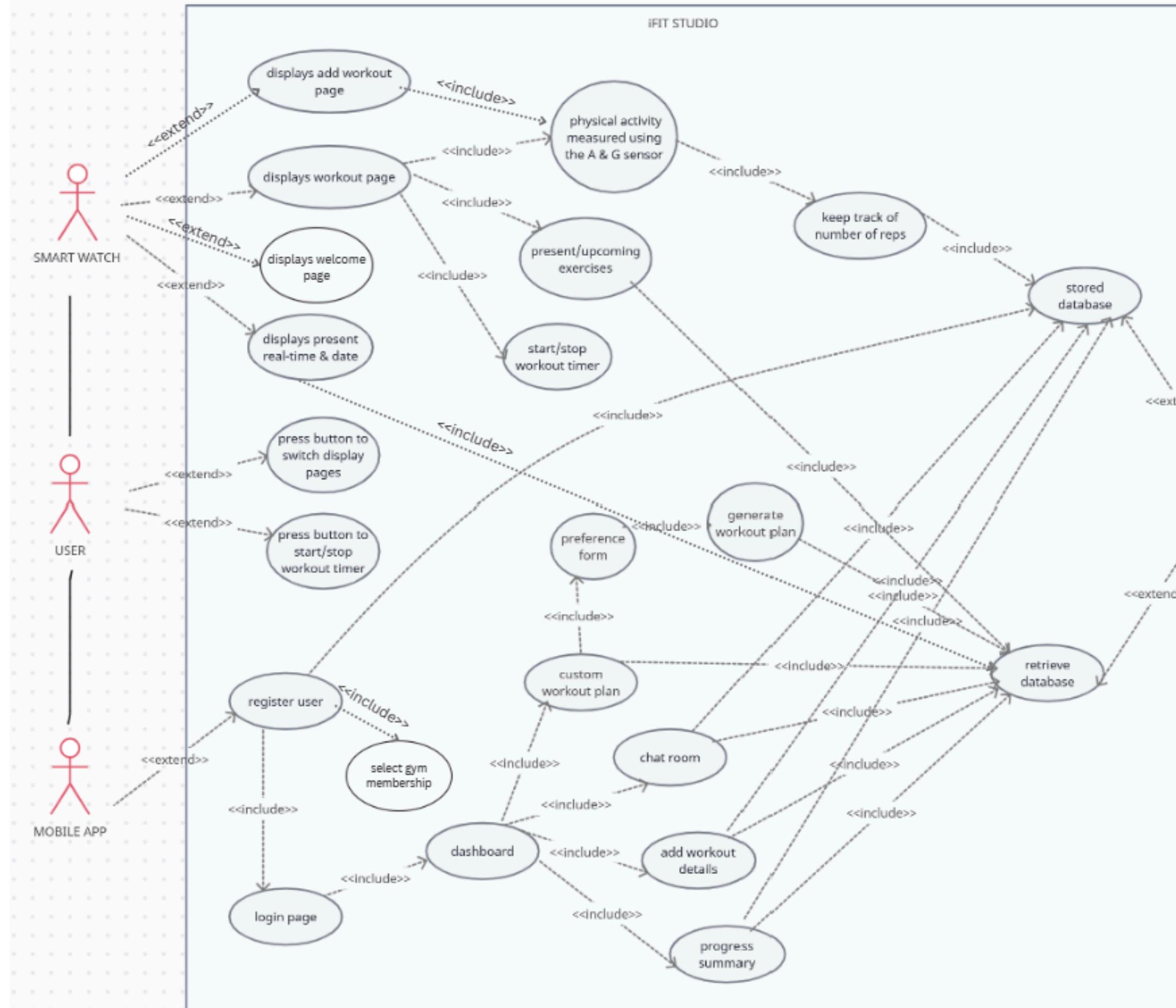
## HARDWARE

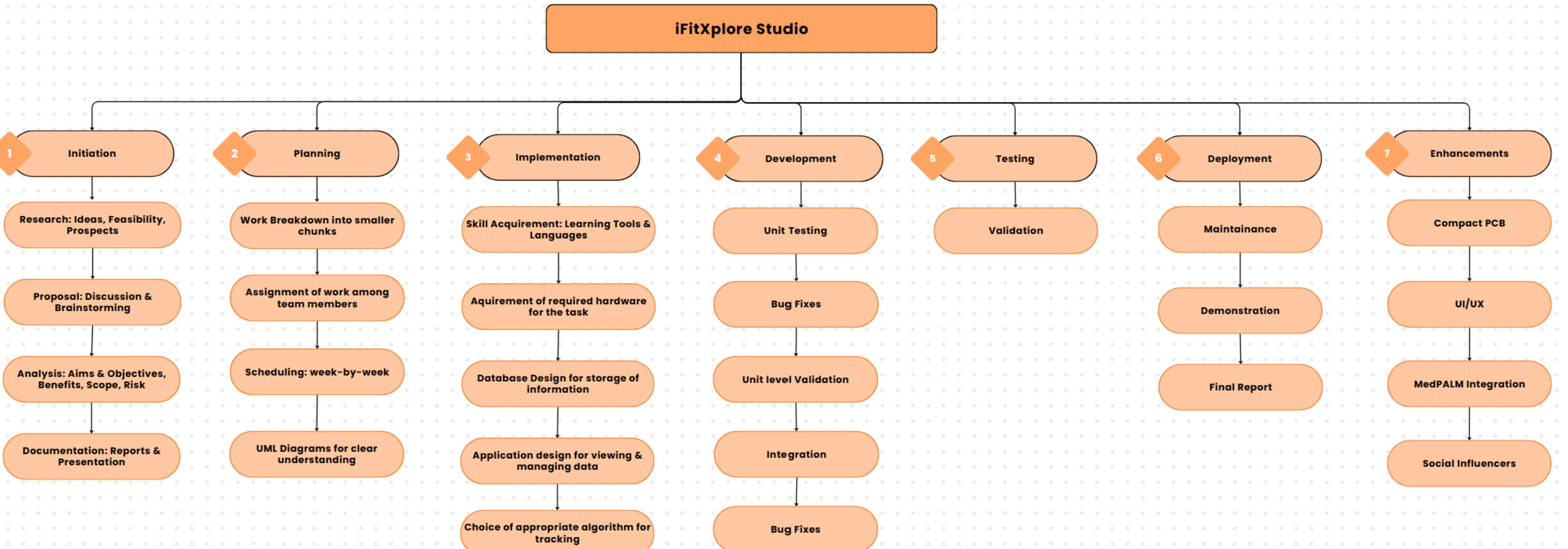
- › LCD 1.28inch waveshare
- › Accelerometer & Gyroscope MPU 6050
- › ESP 32 Board
- › Buttons
- › Power Supply
- › Jumper wires and a Breadboard

# FEATURES MINDMAP



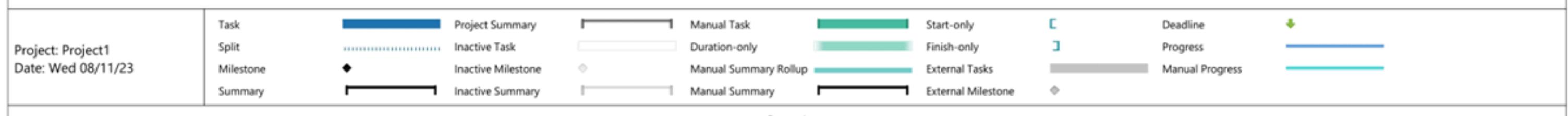
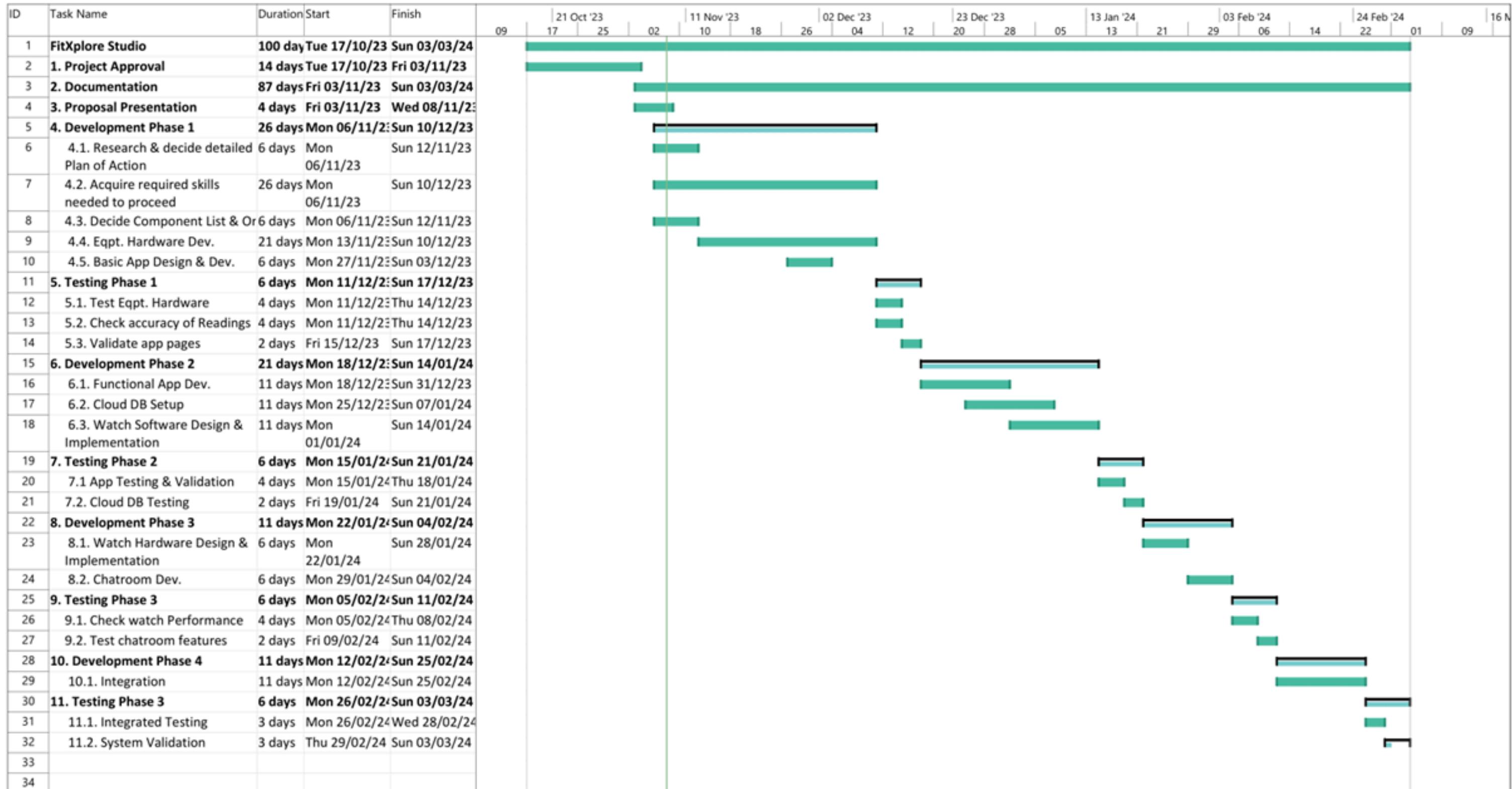
# USE CASE DIAGRAM





# WORK BREAKDOWN STRUCTURE

# GANTT CHART



# RACI MATRIX

Task / Role	Project Manager (Anushka)	Hardware Engineer (Joella)	Software Developer (Ruramai)	Module Tutor (Dr. Sumitra Kotipalli)		
Planning and Requirements	R	A	C	I		
Front end	A	I	R	I		
Back end	R	I	A	I		
Hardware Integration	A	R	I	I		
Documentation	A	R	A	I		

KEY:  
R (Responsible)  
A (Accountable)  
C (Consulted)  
I (Informed)

Tool used to clarify roles and responsibilities within a project



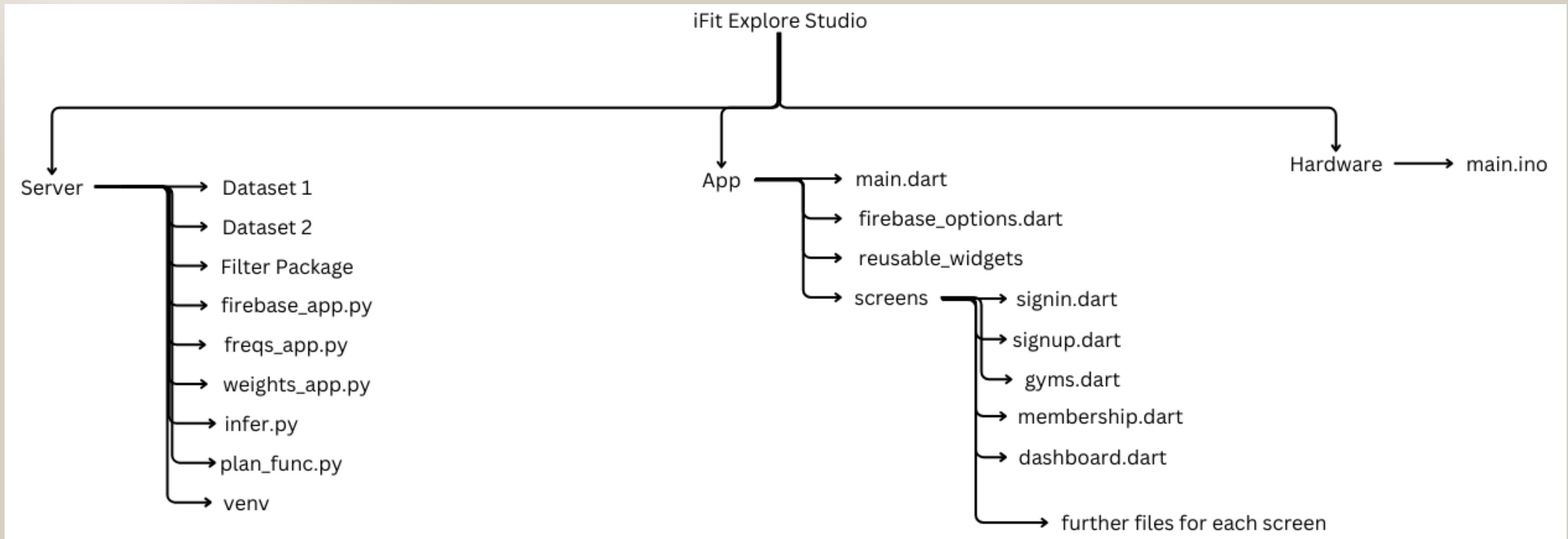
# RESEARCH & DEVELOPMENT METHODOLOGY

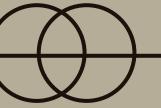
For the research methodology, we conducted exploratory research to understand IoT trends and challenges in the fitness sector. This involved observing industry trends, identifying challenges, and exploring opportunities. Applied research involved looking for areas of improvement in existing methods for the task. This research guided our decision-making throughout the project lifecycle, ensuring that we were well-informed about the landscape.

For the development methodology, we adopted the agile approach, specifically the Scrum framework, due to the duration & collaboration involved. This method suited our project's multiple small components, allowing for easier debugging and integration. The project was divided into sprints with regular meetings to update progress, plan actions, and address issues. This fast-paced and iterative approach facilitated rapid implementation, ensuring the production of a quality product.



# IMPLEMENTATION OVERVIEW





# IMPLEMENTATION: APP

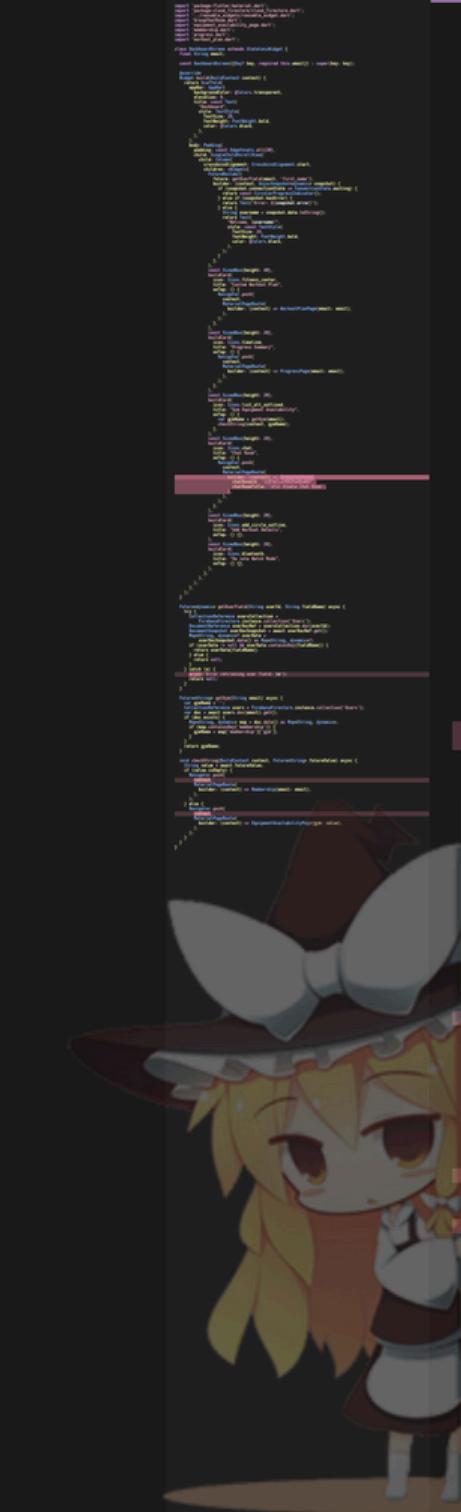
lib > screens > dashboard\_screen.dart > ...

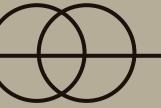
You, 2 weeks ago | 3 authors (Anushka-N12 and others)

You, 2 weeks ago • chatApp feature now working

```
1 import 'package:flutter/material.dart';
2 import 'package:cloud_firestore/cloud_firestore.dart';
3 import '../reusable_widgets/reusable_widget.dart';
4 import 'GroupChatRoom.dart';
5 import 'equipment_availability_page.dart';
6 import 'membership.dart';
7 import 'progress.dart';
8 import 'workout_plan.dart';

12 class DashboardScreen extends StatelessWidget {
13     final String email;
14
15     const DashboardScreen({Key? key, required this.email}) : super(key: key);
16
17     @override
18     Widget build(BuildContext context) {
19         return Scaffold(
20             appBar: AppBar(
21                 backgroundColor: Colors.transparent,
22                 elevation: 0,
23                 title: const Text(
24                     "Dashboard",
25                     style: TextStyle(
26                         fontSize: 28,
27                         fontWeight: FontWeight.bold,
28                         color: Colors.black,
29                     ), // TextStyle
30                     ), // Text
31             ), // AppBar
32             body: Padding(
```





# IMPLEMENTATION: APP

The screenshot shows a dark-themed code editor with the following file structure in the Explorer panel:

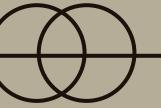
- IFITAPP
- assets
- build
- ios
- lib (56 files)
- reusable\_widgets
- reusable\_widget.dart
- screens (3 files)
- AddWorkoutPage.dart
- ChatRoomPage.dart
- dashboard\_screen.dart
- equipment\_availability\_page.dart
- FirebaseFunctions.dart
- FirebaseService.dart
- generate\_form.dart (2 files)
- group\_chat\_service.dart
- GroupChatRoom.dart (1 file)
- home\_screen.dart
- Membership\_options\_screen.dart
- membership.dart
- progress.dart (M)
- reset\_password.dart
- signin\_screen.dart
- signup\_screen.dart
- workout\_plan.dart
- utils
- firebase\_options.dart

The active file is `dashboard_screen.dart`, which contains the following Dart code:

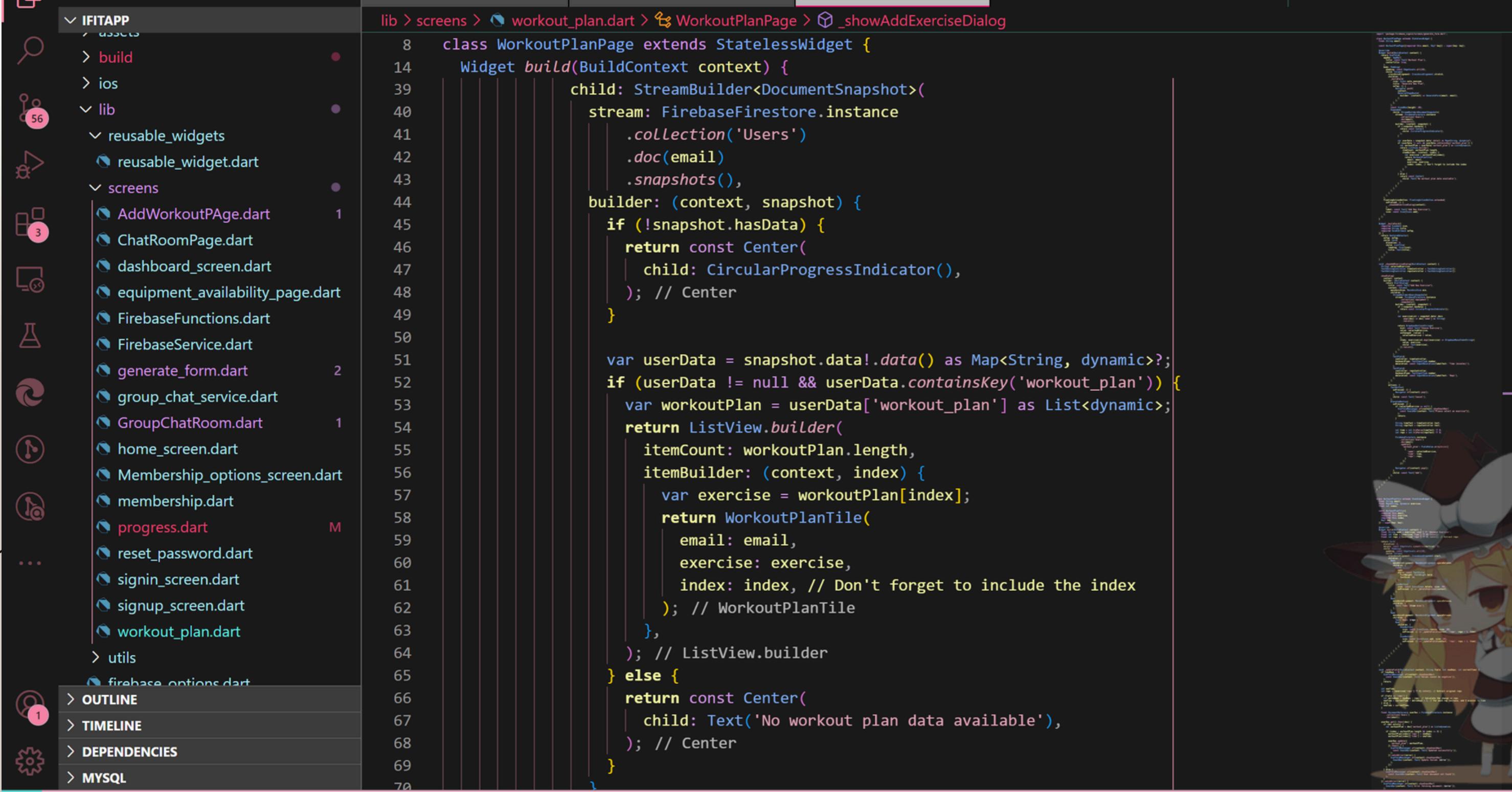
```
class DashboardScreen extends StatelessWidget {  
  Future<dynamic> getUserField(String userId, String fieldName) async {  
    try {  
      CollectionReference usersCollection =  
        FirebaseFirestore.instance.collection('Users');  
      DocumentReference userDocRef = usersCollection.doc(userId);  
      DocumentSnapshot userDocSnapshot = await userDocRef.get();  
      Map<String, dynamic>? userData =  
        userDocSnapshot.data() as Map<String, dynamic>?  
      if (userData != null && userData.containsKey(fieldName)) {  
        return userData[fieldName];  
      } else {  
        return null;  
      }  
    } catch (e) {  
      print('Error retrieving user field: $e');  
      return null;  
    }  
  }  
  
  Future<String> getGym(String email) async {  
    var gymName = '';  
    CollectionReference users = FirebaseFirestore.instance.collection('Users');  
    var doc = await users.doc(email).get();  
    if (doc.exists) {  
      Map<String, dynamic> map = doc.data() as Map<String, dynamic>;  
      if (map.containsKey('membership')) {  
        gymName = map['membership']['gym'];  
      }  
    }  
    return gymName;  
  }  
}
```

The status bar at the bottom shows the following information:

- main\* (active tab)
- outline
- timeline
- dependencies
- MySQL
- 1 main.dart
- 71 lines
- 0 errors
- Connect
- You, 2 weeks ago
- Ln 1, Col 1
- Spaces: 2
- UTF-8
- CRLF
- { } Dart
- Chrome (web-javascript)



# IMPLEMENTATION: APP

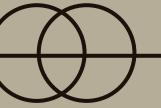


The screenshot shows a code editor with a dark theme. On the left is a file tree for a project named 'IFITAPP'. The 'screens' folder contains several files: AddWorkoutPAge.dart, ChatRoomPage.dart, dashboard\_screen.dart, equipment\_availability\_page.dart, FirebaseFunctions.dart, FirebaseService.dart, generate\_form.dart, group\_chat\_service.dart, GroupChatRoom.dart, home\_screen.dart, Membership\_options\_screen.dart, membership.dart, progress.dart, reset\_password.dart, signin\_screen.dart, signup\_screen.dart, and workout\_plan.dart. The 'workout\_plan.dart' file is open in the editor.

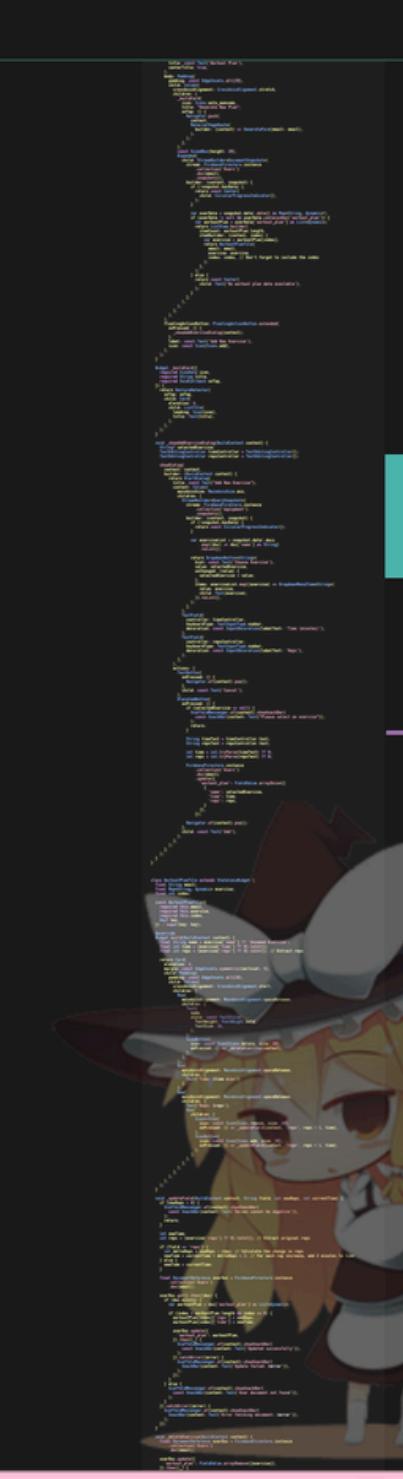
The code in 'workout\_plan.dart' is as follows:

```
lib > screens > workout_plan.dart > WorkoutPlanPage > _showAddExerciseDialog
8   class WorkoutPlanPage extends StatelessWidget {
14     Widget build(BuildContext context) {
39       return StreamBuilder<DocumentSnapshot>(
40         stream: FirebaseFirestore.instance
41           .collection('Users')
42           .doc(email)
43           .snapshots(),
44         builder: (context, snapshot) {
45           if (!snapshot.hasData) {
46             return const Center(
47               child: CircularProgressIndicator(),
48             ); // Center
49           }
50
51           var userData = snapshot.data!.data() as Map<String, dynamic>?;
52           if (userData != null && userData.containsKey('workout_plan')) {
53             var workoutPlan = userData['workout_plan'] as List<dynamic>;
54             return ListView.builder(
55               itemCount: workoutPlan.length,
56               itemBuilder: (context, index) {
57                 var exercise = workoutPlan[index];
58                 return WorkoutPlanTile(
59                   email: email,
60                   exercise: exercise,
61                   index: index, // Don't forget to include the index
62                 ); // WorkoutPlanTile
63               },
64             ); // ListView.builder
65           } else {
66             return const Center(
67               child: Text('No workout plan data available'),
68             ); // Center
69           }
70         }
71       );
72     }
73   }
```

The background of the slide features a faint watermark of a cartoon character with a bow and a sword.



# IMPLEMENTATION: APP



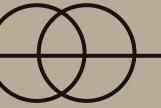
The image shows a mobile application interface with a dark theme. On the left is a sidebar with various icons and navigation links:

- IFITAPP
- assets
- build
- ios
- lib (56 files)
- reusable\_widgets
- reusable\_widget.dart
- screens (3 files)
- AddWorkoutPAge.dart
- ChatRoomPage.dart
- dashboard\_screen.dart
- equipment\_availability\_page.dart
- FirebaseFunctions.dart
- FirebaseService.dart
- generate\_form.dart (2 files)
- group\_chat\_service.dart
- GroupChatRoom.dart (1 file)
- home\_screen.dart
- Membership\_options\_screen.dart
- membership.dart
- progress.dart (M)
- reset\_password.dart
- signin\_screen.dart
- signup\_screen.dart
- workout\_plan.dart
- utils
- firebase\_options.dart

Below the sidebar are four buttons: OUTLINE, TIMELINE, DEPENDENCIES, and MYSQL.

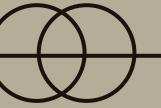
The main area displays a Dart code editor with the following code:

```
lib > screens > workout_plan.dart > WorkoutPlanPage > _showAddExerciseDialog
8   class WorkoutPlanPage extends StatelessWidget {
103  void _showAddExerciseDialog(BuildContext context) {
108    showDialog(
109      context: context,
110      builder: (BuildContext context) {
111        return AlertDialog(
112          title: const Text("Add New Exercise"),
113          content: Column(
114            mainAxisAlignment: MainAxisAlignment.min,
115            children: [
116              StreamBuilder<QuerySnapshot>(
117                stream: FirebaseFirestore.instance
118                  .collection('equipment')
119                  .snapshots(),
120                builder: (context, snapshot) {
121                  if (!snapshot.hasData) {
122                    return const CircularProgressIndicator();
123                  }
124
125                  var exerciseList = snapshot.data!.docs
126                    .map((doc) => doc['name'] as String)
127                    .toList();
128
129                  return DropdownButton<String>(
130                    hint: const Text('Choose Exercise'),
131                    value: selectedExercise,
132                    onChanged: (value) {
133                      selectedExercise = value;
134                    },
135                    items: exerciseList.map((exercise) => DropdownMenuItem<String>(
136                      value: exercise,
137                      child: Text(exercise),
138                    ).toList(), // DropdownMenuItem
139                  ); // DropdownButton
140                }
141              );
142            ],
143          ),
144        );
145      }
146    );
147  }
148}
```



# IMPLEMENTATION: APP

```
lib > screens > workout_plan.dart > WorkoutPlanPage > build
200
201
202 class WorkoutPlanTile extends StatelessWidget {
203   final String email;
204   final Map<String, dynamic> exercise;
205   final int index;
206
207   const WorkoutPlanTile({
208     required this.email,
209     required this.exercise,
210     required this.index,
211     Key? key,
212   }) : super(key: key);
213
214   @override
215   Widget build(BuildContext context) {
216     final String name = exercise['name'] ?? 'Unnamed Exercise';
217     final int time = (exercise['time'] ?? 0).toInt();
218     final int reps = (exercise['reps'] ?? 0).toInt(); // Extract reps
219
220     return Card(
221       elevation: 2,
222       margin: const EdgeInsets.symmetric(vertical: 5),
223       child: Padding(
224         padding: const EdgeInsets.all(10),
225         child: Column(
226           crossAxisAlignment: CrossAxisAlignment.start,
227           children: [
228             Row(
229               mainAxisAlignment: MainAxisAlignment.spaceBetween,
230               children: [
231                 Text(
232                   name,
```

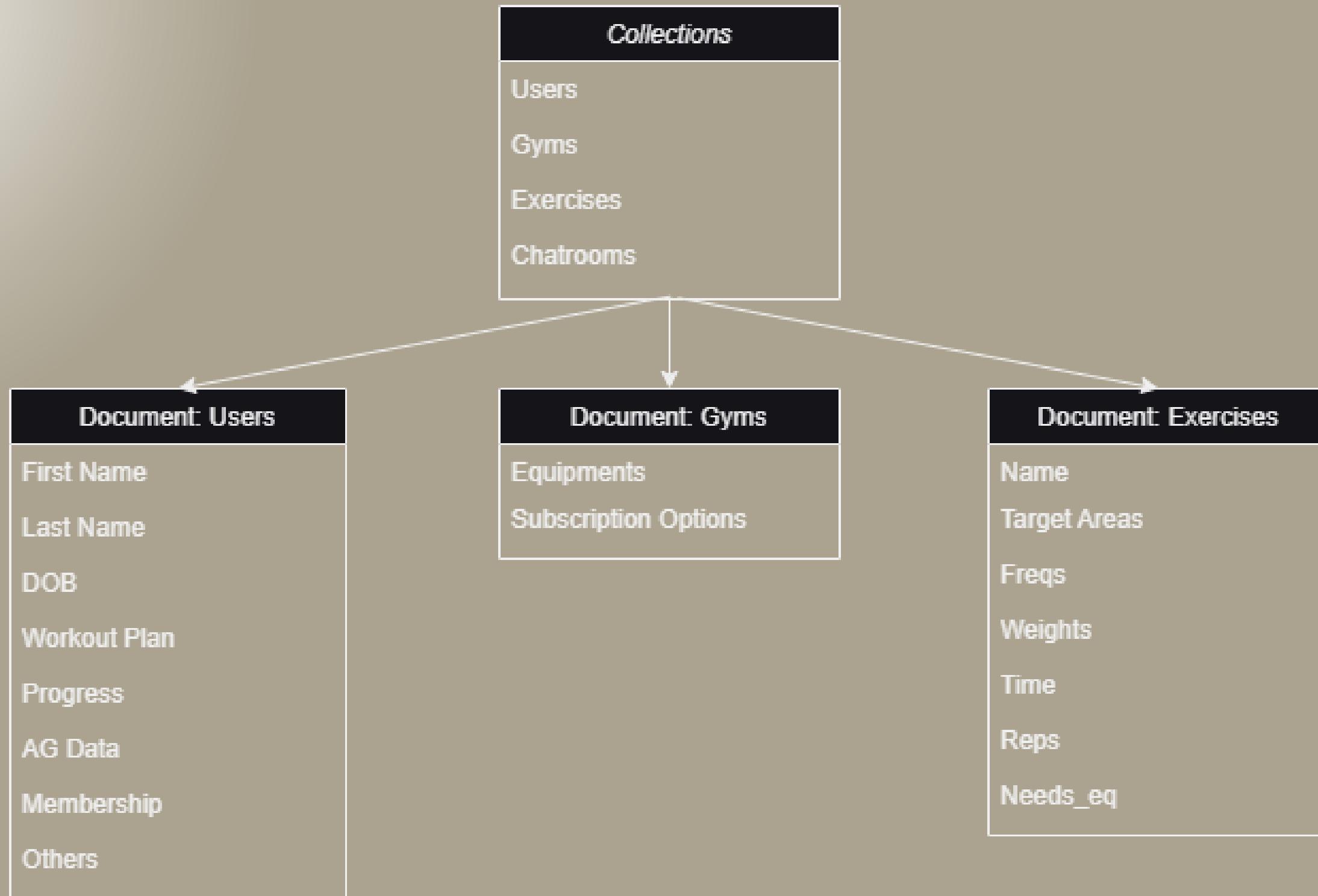
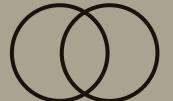


# IMPLEMENTATION: APP

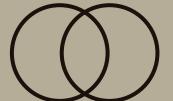
lib > screens > GroupChatRoom.dart > \_GroupChatRoomState > build

```
2 import 'package:firebase_auth/firebase_auth.dart';
3 import 'package:flutter/material.dart';
4 import 'package:cloud_firestore/cloud_firestore.dart';
5
6 You, last week | 1 author (You)
7 class GroupChatRoom extends StatefulWidget {
8   final String chatRoomId;
9   final String chatRoomTitle;
10
11   const GroupChatRoom({Key? key, required this.chatRoomId, required this.chatRoomTitle});
12
13   @override
14   _GroupChatRoomState createState() => _GroupChatRoomState();
15 }
16
17 You, last week | 1 author (You)
18 class _GroupChatRoomState extends State<GroupChatRoom> {
19   final TextEditingController _messageController = TextEditingController();
20   final ScrollController _scrollController = ScrollController();
21
22   @override
23   Widget build(BuildContext context) {
24     return Scaffold(
25       appBar: AppBar(
26         title: Text(widget.chatRoomTitle),
27       ), // AppBar
28       body: Column(
29         children: [
30           Expanded(
31             child: StreamBuilder<QuerySnapshot>(
32               stream: FirebaseFirestore.instance
33                 .collection('Gyms')
34                 .doc('iFit Studio')
35                 .collection('chatroom')
36                 .doc(widget.chatRoomId)
37                 .collection('messages')
38                 .snapshots(),
39               builder: (context, snapshot) {
40                 if (snapshot.connectionState == ConnectionState.waiting) {
41                   return Center(child: CircularProgressIndicator());
42                 } else if (snapshot.hasError) {
43                   return Center(child: Text('Error: ${snapshot.error}'));
44                 } else {
45                   final messages = snapshot.data!.docs;
46                   return ListView.builder(
47                     controller: _scrollController,
48                     itemCount: messages.length,
49                     itemBuilder: (context, index) {
50                       final message = messages[index];
51                       final sender = message['sender'];
52                       final text = message['text'];
53                       final timestamp = message['timestamp'];
54
55                       return MessageBubble(sender: sender!, text: text!, timestamp: timestamp);
56                     },
57                   );
58                 }
59               }
60             )
61           ),
62         ],
63       ),
64     );
65   }
66 }
67
68 void sendMessage(String message) {
69   FirebaseFirestore.instance
70     .collection('Gyms')
71     .doc('iFit Studio')
72     .collection('chatroom')
73     .doc(widget.chatRoomId)
74     .collection('messages')
75     .add({
76       'text': message,
77       'sender': FirebaseAuth.instance.currentUser?.uid,
78       'timestamp': FieldValue.serverTimestamp(),
79     });
80 }
81
82 @override
83 void dispose() {
84   _messageController.dispose();
85   _scrollController.dispose();
86   super.dispose();
87 }
88
89 You, last week | 1 author (You)
90 class MessageBubble extends StatelessWidget {
91   final String sender;
92   final String text;
93
94   const MessageBubble({Key? key, required this.sender, required this.text})
95   : super(key: key);
```

# IMPLEMENTATION: DATA



# IMPLEMENTATION: SERVER



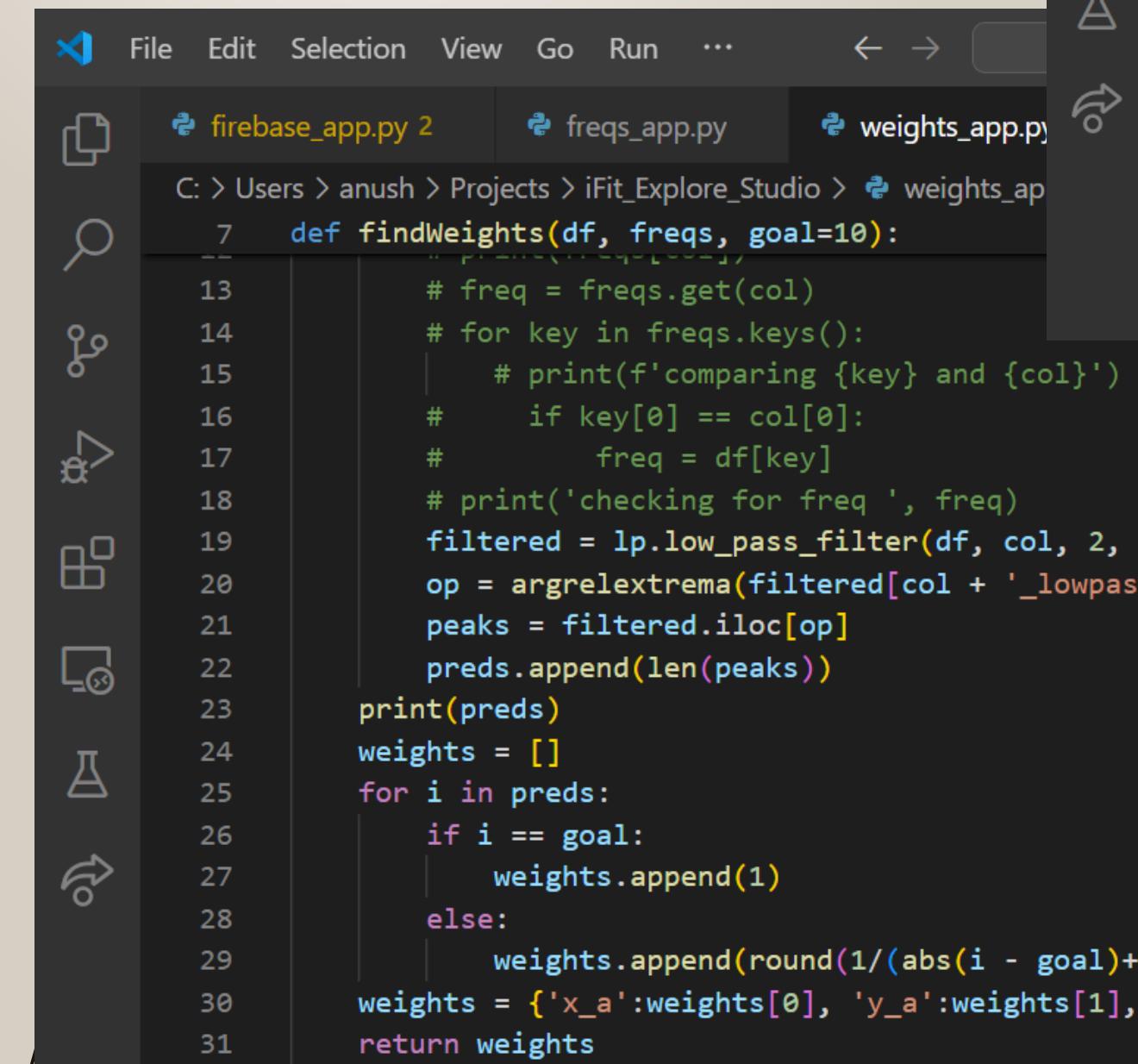
The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with various icons. The main area displays a Python script named `freqs_app.py`. The code implements a function `findFreqs` that processes a DataFrame `df` to find frequency peaks. It uses `pandas`, `numpy`, `LowPassFilter` from `features.data_trans`, and `argrelextrema` from `scipy.signal`. The code includes comments explaining the steps: calculating frequencies for each column, applying a low-pass filter, and identifying peaks using `argrelextrema`.

```
File Edit Selection View Go Run ... ← → Search

firebase_app.py 2 freqs_app.py X

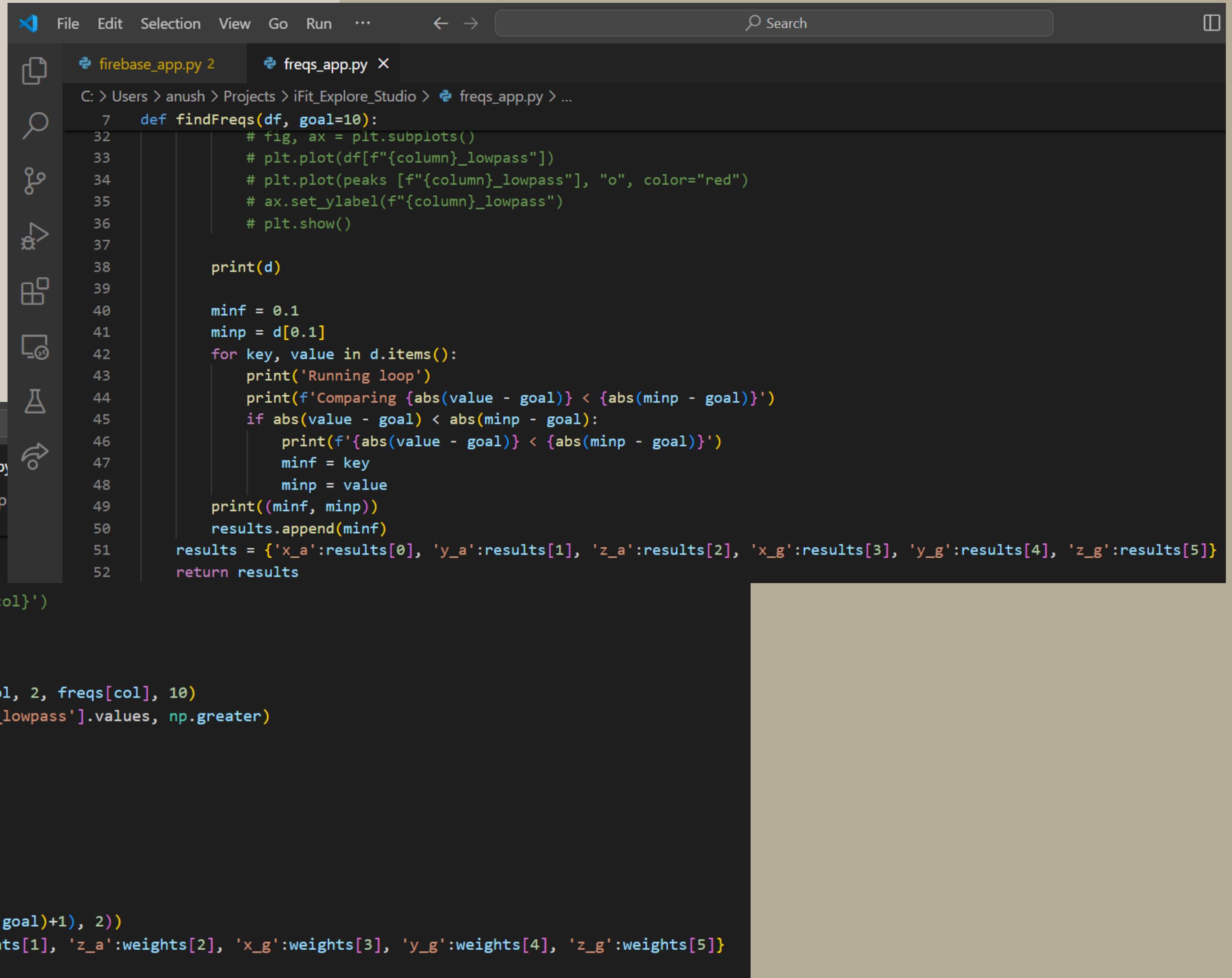
C: > Users > anush > Projects > iFit_Explore_Studio > freqs_app.py > ...
1 import pandas as pd
2 import numpy as np
3 from features.data_trans import LowPassFilter
4 from scipy.signal import argrelextrema
5 import matplotlib.pyplot as plt
6
7 def findFreqs(df, goal=10):
8     results = []
9     for i in df.columns:
10         column = i
11
12         d = {0.05:0, 0.1: 0, 0.15:0, 0.2:0, 0.25:0, 0.3:0, 0.35:0, 0.4:0, 0.5:0, 0.6:0, 0.7:0, 0.8:0, 0.9:0}
13         print(f'Calculating for {i}')
14         for key in d.keys():
15             freq = key
16             print(f'Doing freq {freq}')
17
18             lp = LowPassFilter()
19             filtered = lp.low_pass_filter(df, column, 2, freq, 10)
20
21             op = argrelextrema(filtered[column + '_lowpass'].values, np.greater)
22             peaks = filtered.iloc[op]
23             d[freq] = len(peaks)
24             print(d[freq])
25
26             # fig, ax = plt.subplots()
27             # plt.plot(df[f'{column}'])
28             # plt.plot(peaks[f'{column}"], "o", color="red")
29             # ax.set_xlabel(f'{column}')
```

# IMPLEMENTATION: SERVER



```
File Edit Selection View Go Run ... ← →
firebase_app.py 2 freqs_app.py
weights_app.py

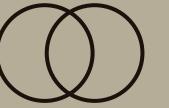
C: > Users > anush > Projects > iFit_Explore_Studio > freqs_app.py > ...
7 def findWeights(df, freqs, goal=10):
8     """
9         # freq = freqs.get(col)
10        # for key in freqs.keys():
11            # print(f'comparing {key} and {col}')
12            # if key[0] == col[0]:
13                # freq = df[key]
14                # print('checking for freq ', freq)
15                filtered = lp.low_pass_filter(df, col, 2, freqs[col], 10)
16                op = argrelextrema(filtered[col + '_lowpass'].values, np.greater)
17                peaks = filtered.iloc[op]
18                preds.append(len(peaks))
19
20                print(preds)
21                weights = []
22                for i in preds:
23                    if i == goal:
24                        weights.append(1)
25                    else:
26                        weights.append(round(1/(abs(i - goal)+1), 2))
27
28                weights = {'x_a':weights[0], 'y_a':weights[1], 'z_a':weights[2], 'x_g':weights[3], 'y_g':weights[4], 'z_g':weights[5]}
29
30                return weights
31
```



```
File Edit Selection View Go Run ... ← →
Search
firebase_app.py 2 freqs_app.py X

C: > Users > anush > Projects > iFit_Explore_Studio > freqs_app.py > ...
7 def findFreqs(df, goal=10):
8     """
9         # fig, ax = plt.subplots()
10        # plt.plot(df[f'{column}_lowpass'])
11        # plt.plot(peaks [f'{column}_lowpass'], "o", color="red")
12        # ax.set_ylabel(f'{column}_lowpass')
13        # plt.show()
14
15        print(d)
16
17        minf = 0.1
18        minp = d[0.1]
19        for key, value in d.items():
20            print('Running loop')
21            print(f'Comparing {abs(value - goal)} < {abs(minp - goal)}')
22            if abs(value - goal) < abs(minp - goal):
23                print(f'{abs(value - goal)} < {abs(minp - goal)}')
24                minf = key
25                minp = value
26                print(minf, minp)
27                results.append(minf)
28
29        results = {'x_a':results[0], 'y_a':results[1], 'z_a':results[2], 'x_g':results[3], 'y_g':results[4], 'z_g':results[5]}
30
31        return results
32
```

# IMPLEMENTATION: SERVER



Screenshot of a Jupyter Notebook interface showing the implementation of a server-side motion detection project.

The notebook is titled "Dave-metamotion-project". The left sidebar shows the file structure:

- File
- Edit
- Selection
- View
- Go
- Run
- ...

The code editor displays the contents of `freqs.py`:

```
def findFreqs(df, goal):
    freq = key
    print(f'Doing freq {freq}')

    lp = LowPassFilter()
    filtered = lp.low_pass_filter(df, column, 5, freq, 10)

    op = argrelextrema(filtered[column + '_lowpass'].values)
    peaks = filtered.iloc[op]
    d[freq] = len(peaks)

    fig, ax = plt.subplots()
    plt.plot(df[f'{column}'])
    # plt.plot(peaks [f'{column}"], "o", color="red")
    ax.set_ylabel(f'{column}')
    plt.show()

    fig, ax = plt.subplots()
    plt.plot(df[f'{column}_lowpass'])
    plt.plot(peaks [f'{column}_lowpass"], "o", color="red")
    ax.set_ylabel(f'{column}_lowpass')
    plt.show()
    print(d)

    minf = 0.2
    minp = d[0.2]
    for key, value in d.items():
        print('Running loop')
        print(f'Comparing {abs(value - goal)} < {abs(minp - goal)}
```

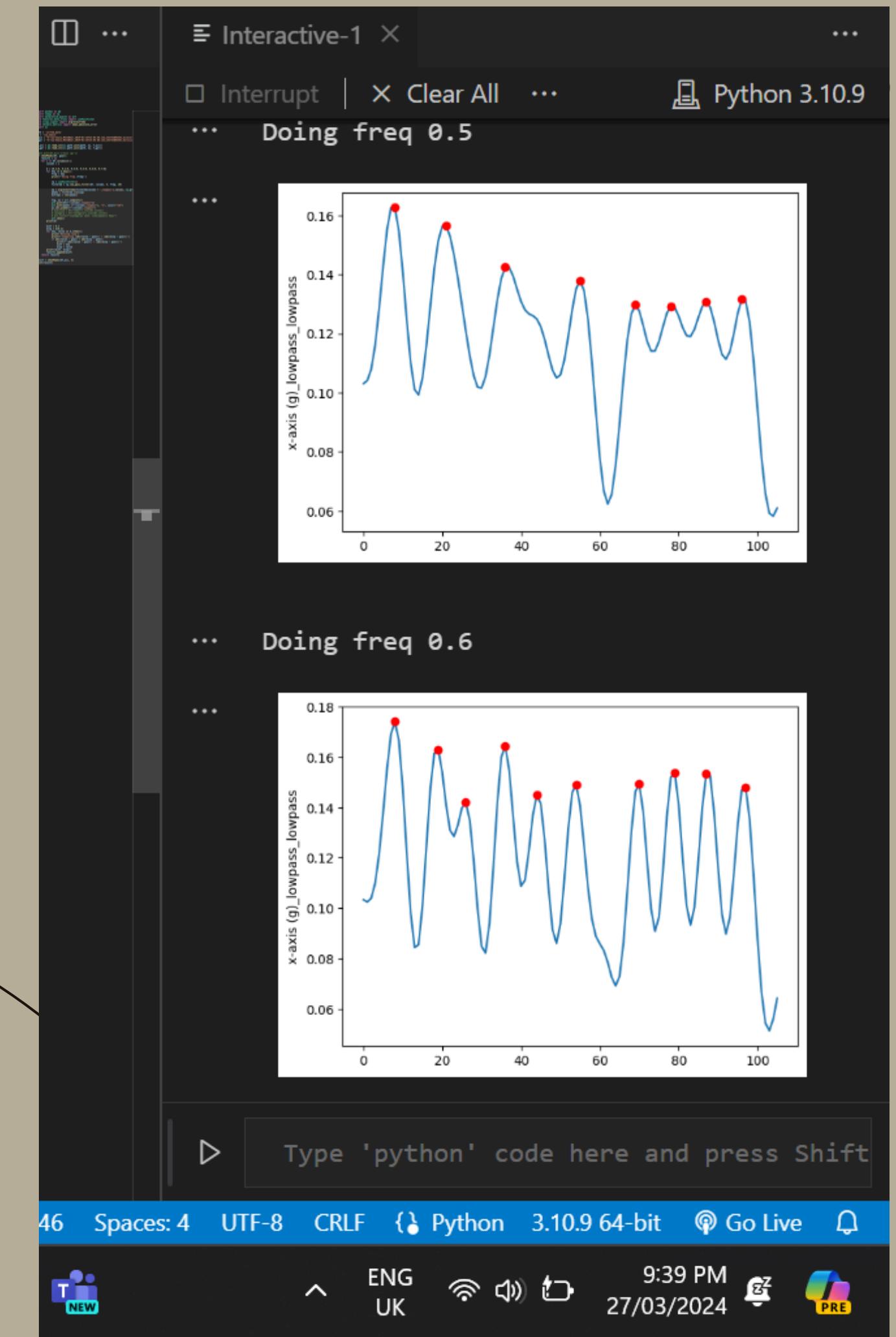
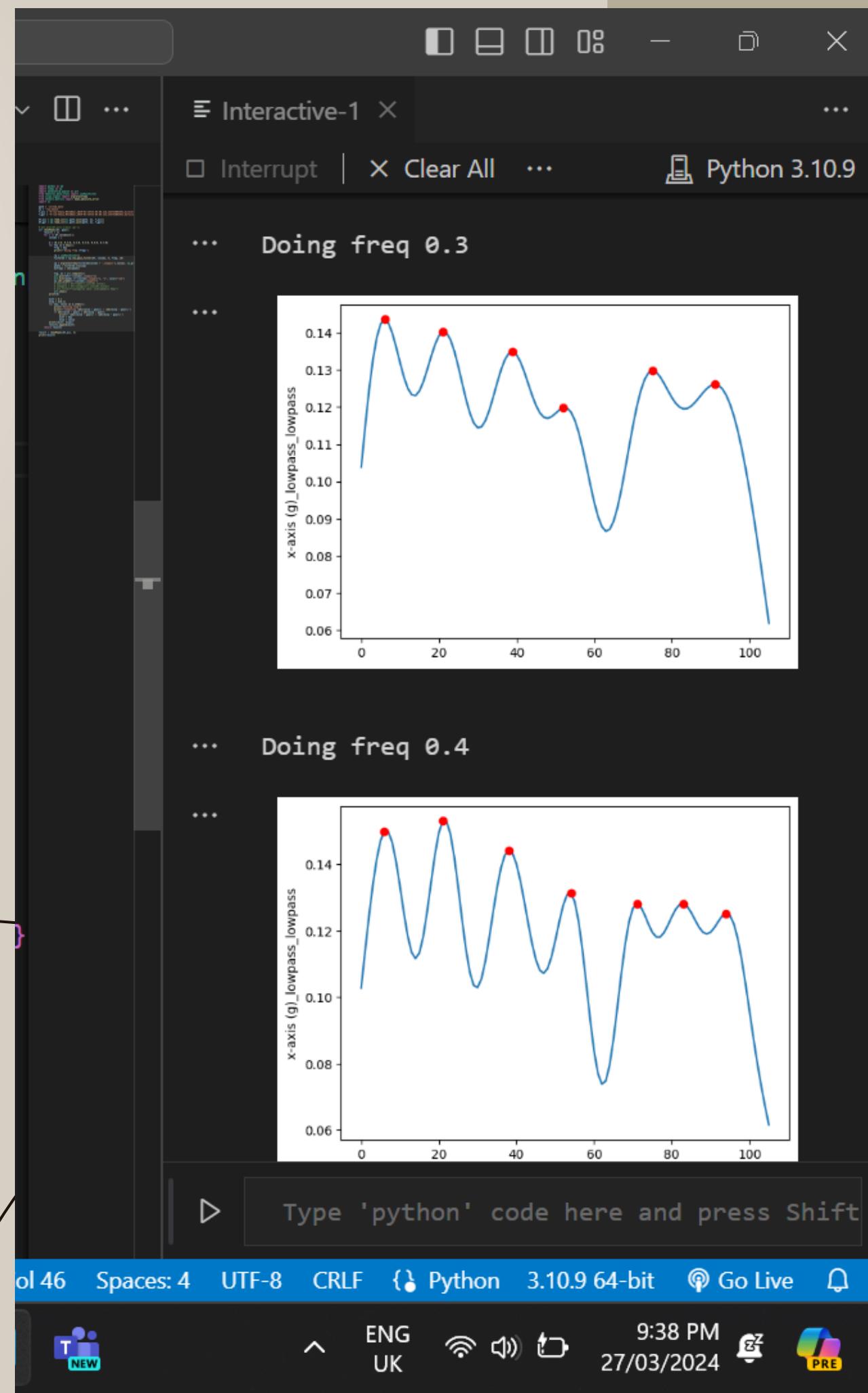
The right pane contains two plots:

- A plot titled "Interactive - freqs.py" showing raw data (blue line) and detected peaks (red dots). The y-axis is labeled "x-axis (deg/s)" and ranges from -40 to 40. The x-axis ranges from 0 to 200.
- A plot titled "Interactive - freqs.py" showing low-pass filtered data (blue line) and detected peaks (red dots). The y-axis is labeled "x-axis (deg/s)\_lowpass" and ranges from -40 to 40. The x-axis ranges from 0 to 200.

The bottom status bar indicates:

- Spaces: 4
- Cell 4 of 4
- Go Live

# IMPLEMENTATION SERVER



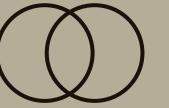
# IMPLEMENTATION: SERVER



The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations, search, navigation, and user settings. The main area displays a Python script named `firebase_app.py`. The code defines a `pred()` function that processes sensor data and a workout plan to determine the next exercise. It uses `infer.findReps` to find repetitions based on gyroscope frequencies and weights. It then compares the sum of acceleration weights with gyroscope weights to decide the result. The code iterates through a `plan` list, checking if the current exercise matches the previous one. If it does, it increments a counter and marks it as a change. It also updates the `c_workout` variable. If the current exercise is the last one in the plan and matches the previous one, it sets the workout as finished. Finally, it adds the current date to a progress array and updates the Firestore database with the new progress data.

```
File Edit Selection View Go Run ... ⏪ ⏩ Search C: > Users > anush > Projects > iFit_Explore_Studio > firebase_app.py > plan 157 def pred(): 183     gyr_freqs = {'x': freqs['x_g'], 'y': freqs['y_g'], 'z': freqs['z_g']} 184     gyr_weights = {'x': weights['x_g'], 'y': weights['y_g'], 'z': weights['z_g']} 185     df_gyr = df[df.columns[3:6]].rename(columns={'x_g':'x', 'y_g':'y', 'z_g': 'z'}) 186     gyr_result = infer.findReps(df_gyr, gyr_freqs, gyr_weights) 187     print(acc_result, gyr_result) 188 189     if sum(acc_weights) > sum(gyr_weights): 190         reps = acc_result 191     else: 192         reps = gyr_result 193 194     changed = False 195     for i in range(len(plan)): 196         if plan[i]['name'] == c_workout and i < len(plan)-1: 197             num = i+1 198             changed = True 199             c_workout = plan[num]['name'] 200         elif plan[i]['name'] == c_workout and i == len(plan)-1: 201             c_workout = 'Finished' 202             changed = True 203     if not changed: 204         c_workout = plan[0]['name'] 205 206     # Add to progress 207     current_date = datetime.now().strftime("%Y-%m-%d") 208     data = {'name': eqname, 'reps': reps, 'time': doc['time']*reps} 209     db.collection('Users').document(email).update({f"progress.{current_date}": firestore.ArrayUnion(data)}) 210
```

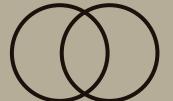
# IMPLEMENTATION: SERVER



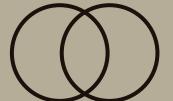
The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar with icons for file operations, search, navigation, and other developer tools. The main area displays a Python script named `infer.py`. The code defines a function `findReps` that takes three parameters: `df`, `freqs`, and `weights`. The function uses nested if statements and absolute value comparisons to determine the result based on the values in `weights` and `reps`. The code editor includes a status bar at the bottom with various settings and a purple footer bar.

```
8 def findReps(df, freqs, weights):
35     if weights['x'] == weights['y'] == 1:
36         if reps[0] == reps[1]:
37             result = reps[0]
38         else:
39             if abs(reps[2] - reps[0]) < abs(reps[2] - reps[1]):
40                 result = reps[0]
41             else:
42                 result = reps[1]
43     elif weights['x'] == weights['z'] == 1:
44         if reps[0] == reps[2]:
45             result = reps[0]
46         else:
47             if abs(reps[1] - reps[0]) < abs(reps[1] - reps[2]):
48                 result = reps[0]
49             else:
50                 result = reps[2]
51     elif weights['y'] == weights['z'] == 1:
52         if reps[1] == reps[2]:
53             result = reps[1]
54         else:
55             if abs(reps[0] - reps[1]) < abs(reps[0] - reps[2]):
56                 result = reps[1]
57             else:
58                 result = reps[2]
59     elif weights['x'] == 1:
60         if abs(reps[0]-reps[1]) > 1 and abs(reps[0]-reps[2]) > 1:
61             result = reps[0] + 1
62         elif abs(reps[0]-reps[1]) < 1 and abs(reps[0]-reps[2]) < 1:
```

# IMPLEMENTATION: SERVER



# IMPLEMENTATION: WATCH



main | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32 Wrover Module

main | Arduino IDE 2.3.2

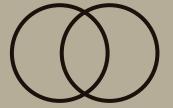
File Edit Sketch Tools Help

ESP32 Wrover Module

main.ino

```
1 #include <WiFi.h>
2 #include <Wire.h>
3 #include <WiFiMulti.h>
4 #include <HTTPClient.h>
5 #include <SPI.h>
6 #include <TFT_eSPI.h>      // Hardware-specific library
7 #include <Adafruit_MPU6050.h>
8 #include <Adafruit_Sensor.h>
9 #include <stdio.h>
10
11 WiFiMulti wifiMulti;
12 TFT_eSPI tft = TFT_eSPI(); // Invoke custom library
13 Adafruit_MPU6050 mpu;
14
15 // Variables for timer and button state
16 unsigned long startTime = 0;
17 unsigned long elapsedTime = 0;
18 bool timerRunning = false;
19 bool buttonPressed = true;
20 float data[6];
21 char cdata[50];
22 unsigned int count_ag = 1;
23 // Define button pin
24 #define BUTTON_PIN_AG 15
25
26 //const int buttonPin = 2; // the GND pin the button is attached to
27
28 void setup() {
29   Serial.begin(115200);
30   WiFi.begin("ESP32_Wrover", "1234567890");
31
32   if (!tft.begin(TFT_CS, TFT_DC)) {
33     Serial.println("TFT Failed to Initialize!");
34     while (true);
35   }
36
37   mpu.begin();
38
39   // Set the I2C address of the MPU6050
40   mpu.setI2CAddress(0x68);
41
42   // Set the gyroscope range to +/- 250 degrees/second
43   mpu.setGyroRange(GYRO_RANGE_250DPS);
44
45   // Set the accelerometer range to +/- 16g
46   mpu.setAccelRange(ACCEL_RANGE_16G);
47
48   // Set the gyroscope full scale to +/- 18 degrees/second
49   mpu.setGyroFullScale(GYRO_FS_18DPS);
50
51   // Set the accelerometer full scale to +/- 2g
52   mpu.setAccelFullScale(ACCEL_FS_2G);
53
54   // Set the gyroscope low pass filter to 100Hz
55   mpu.setGyroLPF(GYRO_LPF_100HZ);
56
57   // Set the accelerometer low pass filter to 100Hz
58   mpu.setAccelLPF(ACCEL_LPF_100HZ);
59
60   // Set the gyroscope sample rate to 100Hz
61   mpu.setGyroSR(GYRO_SR_100HZ);
62
63   // Set the accelerometer sample rate to 100Hz
64   mpu.setAccelSR(ACCEL_SR_100HZ);
65
66   // Set the gyroscope data rate to 100Hz
67   mpu.setGyroDR(GYRO_DR_100HZ);
68
69   // Set the accelerometer data rate to 100Hz
70   mpu.setAccelDR(ACCEL_DR_100HZ);
71
72   // Set the gyroscope data rate to 100Hz
73   mpu.setGyroDR(GYRO_DR_100HZ);
74
75   // Set the accelerometer data rate to 100Hz
76   mpu.setAccelDR(ACCEL_DR_100HZ);
77
78   // Set the gyroscope data rate to 100Hz
79   mpu.setGyroDR(GYRO_DR_100HZ);
80
81   // Set the accelerometer data rate to 100Hz
82   mpu.setAccelDR(ACCEL_DR_100HZ);
83
84   // Set the gyroscope data rate to 100Hz
85   mpu.setGyroDR(GYRO_DR_100HZ);
86
87   // Set the accelerometer data rate to 100Hz
88   mpu.setAccelDR(ACCEL_DR_100HZ);
89
90   // Set the gyroscope data rate to 100Hz
91   mpu.setGyroDR(GYRO_DR_100HZ);
92
93   // Set the accelerometer data rate to 100Hz
94   mpu.setAccelDR(ACCEL_DR_100HZ);
95
96   // Set the gyroscope data rate to 100Hz
97   mpu.setGyroDR(GYRO_DR_100HZ);
98
99   // Set the accelerometer data rate to 100Hz
100  mpu.setAccelDR(ACCEL_DR_100HZ);
101
102  // Set the gyroscope data rate to 100Hz
103  mpu.setGyroDR(GYRO_DR_100HZ);
104
105  // Set the accelerometer data rate to 100Hz
106  mpu.setAccelDR(ACCEL_DR_100HZ);
107
108  Serial.println("");
109  tft.init();
110  tft.fillScreen(TFT_BLACK);
111  delay(100);
112
113  void nextState() {
114    currentState++; // move to the next state
115    if (currentState > 5) { // if the current state exceeds the maximum state, reset to 1
116      currentState = 1;
117    }
118    if (currentState == 3){
119      if((wifiMulti.run() == WL_CONNECTED)) {
120        HTTPClient http;
121        Serial.print("[HTTP] begin...\n");
122        String urlString = "http://192.168.1.6:5000/cworkout";
123        http.begin(urlString.c_str()); //HTTP
124        Serial.print("[HTTP] GET...\n");
125        int httpCode = http.GET();
126        if(httpCode > 0) {
127          Serial.printf("[HTTP] GET... code: %d\n", httpCode);
128          if(httpCode == HTTP_CODE_OK) {
129            String payload = http.getString();
130            Serial.println(payload.substring(2,5));
131            //Serial.println("HTTP Response: " + payload);
132          }
133        }
134      }
135    }
136  }
137
138  void loop() {
139    if (buttonPressed) {
140      nextState();
141    }
142    buttonPressed = false;
143
144    // Read the gyroscope data
145    mpu.getImuData(data);
146
147    // Print the gyroscope data to the serial monitor
148    Serial.print("Gyroscope Data: ");
149    for (int i = 0; i < 6; i++) {
150      Serial.print(data[i]);
151      Serial.print(" ");
152    }
153    Serial.println();
154
155    // Print the accelerometer data to the serial monitor
156    Serial.print("Accelerometer Data: ");
157    for (int i = 0; i < 3; i++) {
158      Serial.print(data[i]);
159      Serial.print(" ");
160    }
161    Serial.println();
162
163    // Print the temperature data to the serial monitor
164    Serial.print("Temperature: ");
165    Serial.print(mpu.getTemp());
166    Serial.println(" C");
167
168    // Print the gyroscope range to the serial monitor
169    Serial.print("Gyroscope Range: ");
170    Serial.print(mpu.getGyroRange());
171    Serial.println(" DPS");
172
173    // Print the gyroscope full scale to the serial monitor
174    Serial.print("Gyroscope Full Scale: ");
175    Serial.print(mpu.getGyroFullScale());
176    Serial.println(" DPS");
177
178    // Print the gyroscope low pass filter to the serial monitor
179    Serial.print("Gyroscope Low Pass Filter: ");
180    Serial.print(mpu.getGyroLPF());
181    Serial.println(" Hz");
182
183    // Print the gyroscope sample rate to the serial monitor
184    Serial.print("Gyroscope Sample Rate: ");
185    Serial.print(mpu.getGyroSR());
186    Serial.println(" Hz");
187
188    // Print the gyroscope data rate to the serial monitor
189    Serial.print("Gyroscope Data Rate: ");
190    Serial.print(mpu.getGyroDR());
191    Serial.println(" Hz");
192
193    // Print the gyroscope data rate to the serial monitor
194    Serial.print("Gyroscope DR: ");
195    Serial.print(mpu.getGyroDR());
196    Serial.println(" Hz");
197
198    // Print the gyroscope data rate to the serial monitor
199    Serial.print("Gyroscope DR: ");
200    Serial.print(mpu.getGyroDR());
201    Serial.println(" Hz");
202
203    // Print the gyroscope data rate to the serial monitor
204    Serial.print("Gyroscope DR: ");
205    Serial.print(mpu.getGyroDR());
206    Serial.println(" Hz");
207
208    // Print the gyroscope data rate to the serial monitor
209    Serial.print("Gyroscope DR: ");
210    Serial.print(mpu.getGyroDR());
211    Serial.println(" Hz");
212
213    // Print the gyroscope data rate to the serial monitor
214    Serial.print("Gyroscope DR: ");
215    Serial.print(mpu.getGyroDR());
216    Serial.println(" Hz");
217
218    // Print the gyroscope data rate to the serial monitor
219    Serial.print("Gyroscope DR: ");
220    Serial.print(mpu.getGyroDR());
221    Serial.println(" Hz");
222
223    // Print the gyroscope data rate to the serial monitor
224    Serial.print("Gyroscope DR: ");
225    Serial.print(mpu.getGyroDR());
226    Serial.println(" Hz");
227
228    // Print the gyroscope data rate to the serial monitor
229    Serial.print("Gyroscope DR: ");
230    Serial.print(mpu.getGyroDR());
231    Serial.println(" Hz");
232
233    // Print the gyroscope data rate to the serial monitor
234    Serial.print("Gyroscope DR: ");
235    Serial.print(mpu.getGyroDR());
236    Serial.println(" Hz");
237
238    // Print the gyroscope data rate to the serial monitor
239    Serial.print("Gyroscope DR: ");
240    Serial.print(mpu.getGyroDR());
241    Serial.println(" Hz");
242
243    // Print the gyroscope data rate to the serial monitor
244    Serial.print("Gyroscope DR: ");
245    Serial.print(mpu.getGyroDR());
246    Serial.println(" Hz");
247
248    // Print the gyroscope data rate to the serial monitor
249    Serial.print("Gyroscope DR: ");
250    Serial.print(mpu.getGyroDR());
251    Serial.println(" Hz");
252
253    // Print the gyroscope data rate to the serial monitor
254    Serial.print("Gyroscope DR: ");
255    Serial.print(mpu.getGyroDR());
256    Serial.println(" Hz");
257
258    // Print the gyroscope data rate to the serial monitor
259    Serial.print("Gyroscope DR: ");
260    Serial.print(mpu.getGyroDR());
261    Serial.println(" Hz");
262
263    // Print the gyroscope data rate to the serial monitor
264    Serial.print("Gyroscope DR: ");
265    Serial.print(mpu.getGyroDR());
266    Serial.println(" Hz");
267
268    // Print the gyroscope data rate to the serial monitor
269    Serial.print("Gyroscope DR: ");
270    Serial.print(mpu.getGyroDR());
271    Serial.println(" Hz");
272
273    // Print the gyroscope data rate to the serial monitor
274    Serial.print("Gyroscope DR: ");
275    Serial.print(mpu.getGyroDR());
276    Serial.println(" Hz");
277
278    // Print the gyroscope data rate to the serial monitor
279    Serial.print("Gyroscope DR: ");
280    Serial.print(mpu.getGyroDR());
281    Serial.println(" Hz");
282
283    // Print the gyroscope data rate to the serial monitor
284    Serial.print("Gyroscope DR: ");
285    Serial.print(mpu.getGyroDR());
286    Serial.println(" Hz");
287
288    // Print the gyroscope data rate to the serial monitor
289    Serial.print("Gyroscope DR: ");
290    Serial.print(mpu.getGyroDR());
291    Serial.println(" Hz");
292
293    // Print the gyroscope data rate to the serial monitor
294    Serial.print("Gyroscope DR: ");
295    Serial.print(mpu.getGyroDR());
296    Serial.println(" Hz");
297
298    // Print the gyroscope data rate to the serial monitor
299    Serial.print("Gyroscope DR: ");
300    Serial.print(mpu.getGyroDR());
301    Serial.println(" Hz");
302
303    // Print the gyroscope data rate to the serial monitor
304    Serial.print("Gyroscope DR: ");
305    Serial.print(mpu.getGyroDR());
306    Serial.println(" Hz");
307
308    // Print the gyroscope data rate to the serial monitor
309    Serial.print("Gyroscope DR: ");
310    Serial.print(mpu.getGyroDR());
311    Serial.println(" Hz");
312
313    // Print the gyroscope data rate to the serial monitor
314    Serial.print("Gyroscope DR: ");
315    Serial.print(mpu.getGyroDR());
316    Serial.println(" Hz");
317
318    // Print the gyroscope data rate to the serial monitor
319    Serial.print("Gyroscope DR: ");
320    Serial.print(mpu.getGyroDR());
321    Serial.println(" Hz");
322
323    // Print the gyroscope data rate to the serial monitor
324    Serial.print("Gyroscope DR: ");
325    Serial.print(mpu.getGyroDR());
326    Serial.println(" Hz");
327
328    // Print the gyroscope data rate to the serial monitor
329    Serial.print("Gyroscope DR: ");
330    Serial.print(mpu.getGyroDR());
331    Serial.println(" Hz");
332
333    // Print the gyroscope data rate to the serial monitor
334    Serial.print("Gyroscope DR: ");
335    Serial.print(mpu.getGyroDR());
336    Serial.println(" Hz");
337
338    // Print the gyroscope data rate to the serial monitor
339    Serial.print("Gyroscope DR: ");
340    Serial.print(mpu.getGyroDR());
341    Serial.println(" Hz");
342
343    // Print the gyroscope data rate to the serial monitor
344    Serial.print("Gyroscope DR: ");
345    Serial.print(mpu.getGyroDR());
346    Serial.println(" Hz");
347
348    // Print the gyroscope data rate to the serial monitor
349    Serial.print("Gyroscope DR: ");
350    Serial.print(mpu.getGyroDR());
351    Serial.println(" Hz");
352
353    // Print the gyroscope data rate to the serial monitor
354    Serial.print("Gyroscope DR: ");
355    Serial.print(mpu.getGyroDR());
356    Serial.println(" Hz");
357
358    // Print the gyroscope data rate to the serial monitor
359    Serial.print("Gyroscope DR: ");
360    Serial.print(mpu.getGyroDR());
361    Serial.println(" Hz");
362
363    // Print the gyroscope data rate to the serial monitor
364    Serial.print("Gyroscope DR: ");
365    Serial.print(mpu.getGyroDR());
366    Serial.println(" Hz");
367
368    // Print the gyroscope data rate to the serial monitor
369    Serial.print("Gyroscope DR: ");
370    Serial.print(mpu.getGyroDR());
371    Serial.println(" Hz");
372
373    // Print the gyroscope data rate to the serial monitor
374    Serial.print("Gyroscope DR: ");
375    Serial.print(mpu.getGyroDR());
376    Serial.println(" Hz");
377
378    // Print the gyroscope data rate to the serial monitor
379    Serial.print("Gyroscope DR: ");
380    Serial.print(mpu.getGyroDR());
381    Serial.println(" Hz");
382
383    // Print the gyroscope data rate to the serial monitor
384    Serial.print("Gyroscope DR: ");
385    Serial.print(mpu.getGyroDR());
386    Serial.println(" Hz");
387
388    // Print the gyroscope data rate to the serial monitor
389    Serial.print("Gyroscope DR: ");
390    Serial.print(mpu.getGyroDR());
391    Serial.println(" Hz");
392
393    // Print the gyroscope data rate to the serial monitor
394    Serial.print("Gyroscope DR: ");
395    Serial.print(mpu.getGyroDR());
396    Serial.println(" Hz");
397
398    // Print the gyroscope data rate to the serial monitor
399    Serial.print("Gyroscope DR: ");
400    Serial.print(mpu.getGyroDR());
401    Serial.println(" Hz");
402
403    // Print the gyroscope data rate to the serial monitor
404    Serial.print("Gyroscope DR: ");
405    Serial.print(mpu.getGyroDR());
406    Serial.println(" Hz");
407
408    // Print the gyroscope data rate to the serial monitor
409    Serial.print("Gyroscope DR: ");
410    Serial.print(mpu.getGyroDR());
411    Serial.println(" Hz");
412
413    // Print the gyroscope data rate to the serial monitor
414    Serial.print("Gyroscope DR: ");
415    Serial.print(mpu.getGyroDR());
416    Serial.println(" Hz");
417
418    // Print the gyroscope data rate to the serial monitor
419    Serial.print("Gyroscope DR: ");
420    Serial.print(mpu.getGyroDR());
421    Serial.println(" Hz");
422
423    // Print the gyroscope data rate to the serial monitor
424    Serial.print("Gyroscope DR: ");
425    Serial.print(mpu.getGyroDR());
426    Serial.println(" Hz");
427
428    // Print the gyroscope data rate to the serial monitor
429    Serial.print("Gyroscope DR: ");
430    Serial.print(mpu.getGyroDR());
431    Serial.println(" Hz");
432
433    // Print the gyroscope data rate to the serial monitor
434    Serial.print("Gyroscope DR: ");
435    Serial.print(mpu.getGyroDR());
436    Serial.println(" Hz");
437
438    // Print the gyroscope data rate to the serial monitor
439    Serial.print("Gyroscope DR: ");
440    Serial.print(mpu.getGyroDR());
441    Serial.println(" Hz");
442
443    // Print the gyroscope data rate to the serial monitor
444    Serial.print("Gyroscope DR: ");
445    Serial.print(mpu.getGyroDR());
446    Serial.println(" Hz");
447
448    // Print the gyroscope data rate to the serial monitor
449    Serial.print("Gyroscope DR: ");
450    Serial.print(mpu.getGyroDR());
451    Serial.println(" Hz");
452
453    // Print the gyroscope data rate to the serial monitor
454    Serial.print("Gyroscope DR: ");
455    Serial.print(mpu.getGyroDR());
456    Serial.println(" Hz");
457
458    // Print the gyroscope data rate to the serial monitor
459    Serial.print("Gyroscope DR: ");
460    Serial.print(mpu.getGyroDR());
461    Serial.println(" Hz");
462
463    // Print the gyroscope data rate to the serial monitor
464    Serial.print("Gyroscope DR: ");
465    Serial.print(mpu.getGyroDR());
466    Serial.println(" Hz");
467
468    // Print the gyroscope data rate to the serial monitor
469    Serial.print("Gyroscope DR: ");
470    Serial.print(mpu.getGyroDR());
471    Serial.println(" Hz");
472
473    // Print the gyroscope data rate to the serial monitor
474    Serial.print("Gyroscope DR: ");
475    Serial.print(mpu.getGyroDR());
476    Serial.println(" Hz");
477
478    // Print the gyroscope data rate to the serial monitor
479    Serial.print("Gyroscope DR: ");
480    Serial.print(mpu.getGyroDR());
481    Serial.println(" Hz");
482
483    // Print the gyroscope data rate to the serial monitor
484    Serial.print("Gyroscope DR: ");
485    Serial.print(mpu.getGyroDR());
486    Serial.println(" Hz");
487
488    // Print the gyroscope data rate to the serial monitor
489    Serial.print("Gyroscope DR: ");
490    Serial.print(mpu.getGyroDR());
491    Serial.println(" Hz");
492
493    // Print the gyroscope data rate to the serial monitor
494    Serial.print("Gyroscope DR: ");
495    Serial.print(mpu.getGyroDR());
496    Serial.println(" Hz");
497
498    // Print the gyroscope data rate to the serial monitor
499    Serial.print("Gyroscope DR: ");
500    Serial.print(mpu.getGyroDR());
501    Serial.println(" Hz");
502
503    // Print the gyroscope data rate to the serial monitor
504    Serial.print("Gyroscope DR: ");
505    Serial.print(mpu.getGyroDR());
506    Serial.println(" Hz");
507
508    // Print the gyroscope data rate to the serial monitor
509    Serial.print("Gyroscope DR: ");
510    Serial.print(mpu.getGyroDR());
511    Serial.println(" Hz");
512
513    // Print the gyroscope data rate to the serial monitor
514    Serial.print("Gyroscope DR: ");
515    Serial.print(mpu.getGyroDR());
516    Serial.println(" Hz");
517
518    // Print the gyroscope data rate to the serial monitor
519    Serial.print("Gyroscope DR: ");
520    Serial.print(mpu.getGyroDR());
521    Serial.println(" Hz");
522
523    // Print the gyroscope data rate to the serial monitor
524    Serial.print("Gyroscope DR: ");
525    Serial.print(mpu.getGyroDR());
526    Serial.println(" Hz");
527
528    // Print the gyroscope data rate to the serial monitor
529    Serial.print("Gyroscope DR: ");
530    Serial.print(mpu.getGyroDR());
531    Serial.println(" Hz");
532
533    // Print the gyroscope data rate to the serial monitor
534    Serial.print("Gyroscope DR: ");
535    Serial.print(mpu.getGyroDR());
536    Serial.println(" Hz");
537
538    // Print the gyroscope data rate to the serial monitor
539    Serial.print("Gyroscope DR: ");
540    Serial.print(mpu.getGyroDR());
541    Serial.println(" Hz");
542
543    // Print the gyroscope data rate to the serial monitor
544    Serial.print("Gyroscope DR: ");
545    Serial.print(mpu.getGyroDR());
546    Serial.println(" Hz");
547
548    // Print the gyroscope data rate to the serial monitor
549    Serial.print("Gyroscope DR: ");
550    Serial.print(mpu.getGyroDR());
551    Serial.println(" Hz");
552
553    // Print the gyroscope data rate to the serial monitor
554    Serial.print("Gyroscope DR: ");
555    Serial.print(mpu.getGyroDR());
556    Serial.println(" Hz");
557
558    // Print the gyroscope data rate to the serial monitor
559    Serial.print("Gyroscope DR: ");
560    Serial.print(mpu.getGyroDR());
561    Serial.println(" Hz");
562
563    // Print the gyroscope data rate to the serial monitor
564    Serial.print("Gyroscope DR: ");
565    Serial.print(mpu.getGyroDR());
566    Serial.println(" Hz");
567
568    // Print the gyroscope data rate to the serial monitor
569    Serial.print("Gyroscope DR: ");
570    Serial.print(mpu.getGyroDR());
571    Serial.println(" Hz");
572
573    // Print the gyroscope data rate to the serial monitor
574    Serial.print("Gyroscope DR: ");
575    Serial.print(mpu.getGyroDR());
576    Serial.println(" Hz");
577
578    // Print the gyroscope data rate to the serial monitor
579    Serial.print("Gyroscope DR: ");
580    Serial.print(mpu.getGyroDR());
581    Serial.println(" Hz");
582
583    // Print the gyroscope data rate to the serial monitor
584    Serial.print("Gyroscope DR: ");
585    Serial.print(mpu.getGyroDR());
586    Serial.println(" Hz");
587
588    // Print the gyroscope data rate to the serial monitor
589    Serial.print("Gyroscope DR: ");
590    Serial.print(mpu.getGyroDR());
591    Serial.println(" Hz");
592
593    // Print the gyroscope data rate to the serial monitor
594    Serial.print("Gyroscope DR: ");
595    Serial.print(mpu.getGyroDR());
596    Serial.println(" Hz");
597
598    // Print the gyroscope data rate to the serial monitor
599    Serial.print("Gyroscope DR: ");
600    Serial.print(mpu.getGyroDR());
601    Serial.println(" Hz");
602
603    // Print the gyroscope data rate to the serial monitor
604    Serial.print("Gyroscope DR: ");
605    Serial.print(mpu.getGyroDR());
606    Serial.println(" Hz");
607
608    // Print the gyroscope data rate to the serial monitor
609    Serial.print("Gyroscope DR: ");
610    Serial.print(mpu.getGyroDR());
611    Serial.println(" Hz");
612
613    // Print the gyroscope data rate to the serial monitor
614    Serial.print("Gyroscope DR: ");
615    Serial.print(mpu.getGyroDR());
616    Serial.println(" Hz");
617
618    // Print the gyroscope data rate to the serial monitor
619    Serial.print("Gyroscope DR: ");
620    Serial.print(mpu.getGyroDR());
621    Serial.println(" Hz");
622
623    // Print the gyroscope data rate to the serial monitor
624    Serial.print("Gyroscope DR: ");
625    Serial.print(mpu.getGyroDR());
626    Serial.println(" Hz");
627
628    // Print the gyroscope data rate to the serial monitor
629    Serial.print("Gyroscope DR: ");
630    Serial.print(mpu.getGyroDR());
631    Serial.println(" Hz");
632
633    // Print the gyroscope data rate to the serial monitor
634    Serial.print("Gyroscope DR: ");
635    Serial.print(mpu.getGyroDR());
636    Serial.println(" Hz");
637
638    // Print the gyroscope data rate to the serial monitor
639    Serial.print("Gyroscope DR: ");
640    Serial.print(mpu.getGyroDR());
641    Serial.println(" Hz");
642
643    // Print the gyroscope data rate to the serial monitor
644    Serial.print("Gyroscope DR: ");
645    Serial.print(mpu.getGyroDR());
646    Serial.println(" Hz");
647
648    // Print the gyroscope data rate to the serial monitor
649    Serial.print("Gyroscope DR: ");
650    Serial.print(mpu.getGyroDR());
651    Serial.println(" Hz");
652
653    // Print the gyroscope data rate to the serial monitor
654    Serial.print("Gyroscope DR: ");
655    Serial.print(mpu.getGyroDR());
656    Serial.println(" Hz");
657
658    // Print the gyroscope data rate to the serial monitor
659    Serial.print("Gyroscope DR: ");
660    Serial.print(mpu.getGyroDR());
661    Serial.println(" Hz");
662
663    // Print the gyroscope data rate to the serial monitor
664    Serial.print("Gyroscope DR: ");
665    Serial.print(mpu.getGyroDR());
666    Serial.println(" Hz
```

# IMPLEMENTATION: WATCH



```
main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
✓ → 🛡️ ⚙️ ESP32 Wrover Module ▾
main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
✓ → 🛡️ ⚙️ ESP32 Wrover Module ▾
main.ino
157
158 void loop(){
159     if (digitalRead(buttonPin) == HIGH) { // check if the
160         nextState(); // if pressed, move to the next state
161     } //button 1
162     switch (currentState) {
163         case 1:
164             Serial.println("State 1: Perform action 1");
165             tft.fillRect(0, 0, tft.width(), tft.height(), TFT_BLACK);
166             tft.drawCircle(120, 120, 118, TFT_BLUE); // Dr
167             tft.drawLine(120, 0, 120, 12, TFT_GREEN); // D
168             tft.drawLine(120, 228, 120, 240, TFT_GREEN);
169             tft.drawLine(0, 120, 12, 120, TFT_GREEN);
170             tft.drawLine(228, 120, 240, 120, TFT_GREEN);
171             tft.setTextColor(TFT_WHITE);
172             tft.setCursor(70, 60, 4);
173             tft.println("iFit Studio");
174             tft.setCursor(60, 100, 4);
175             tft.println("Welcome!");
176             delay(1000);
177             nextState();
178             break;
179         case 2:
180             if((wifiMulti.run() == WL_CONNECTED)) {
181                 HTTPClient http;
182                 Serial.print("[HTTP] begin...\n");
183                 http.begin("http://192.168.0.104:5000/time"); //HTTP
184                 Serial.print("[HTTP] GET...\n");
185                 int httpCode = http.GET();
186                 if(httpCode > 0) {
187                     Serial.printf("[HTTP] GET... code: %d\n", httpCode);
188                     if(httpCode == HTTP_CODE_OK) {
189                         String payload = http.getString();
190                         tft.fillRect(0, 0, tft.width(), tft.height(), TFT_BLACK);
191                         tft.drawRect(0, 0, tft.width(), tft.height(), TFT_GREEN);
192                         tft.setTextColor(TFT_WHITE);
193                         tft.setCursor(80, 60, 4);
194                         tft.println(payload.substring(2,7)+"\n");
195                         Serial.println(payload.substring(2,7));
196                         tft.setCursor(40, 120, 4);
197                         tft.println(payload.substring(14,28));
198                         Serial.println(payload.substring(14,28));
199                     } else {
200                         Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
201                     } //else OK
202                 } //0
203                 http.end();
204             }
205         }
206     }
207 }
```



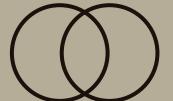
# IMPLEMENTATION: WATCH

The screenshot shows the Arduino IDE interface with two tabs open:

- main | Arduino IDE 2.3.2**: This tab is inactive.
- main | Arduino IDE 2.3.2**: This tab is active, showing the file `main.ino`. The code is as follows:

```
case 3:
    tft.fillRect(0, 0, tft.width(), tft.height(), TFT_GREEN);
    if (digitalRead(BUTTON_PIN_A) == LOW && !buttonPressed) {
        buttonPressed = true;
        timerRunning = !timerRunning;
        if (timerRunning) {
            startTime = millis();
        } else {
            if((wifiMulti.run() == WL_CONNECTED)) {
                HttpClient http;
                Serial.print("Sending");
                Serial.print("\n");
                Serial.print("[HTTP] begin...\n");
                http.begin("http://192.168.0.104:5000/pred");
                Serial.print("[HTTP] GET...\n");
                int httpCode = http.GET();
                if(httpCode > 0) {
                    Serial.printf("[HTTP] GET... code: %d\n", httpCode);
                    if(httpCode == HTTP_CODE_OK) {
                        String payload = http.getString();
                        Serial.println(payload);
                    } //OK
                } else {
                    Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
                } //else
            }
        }
    }
    http.end();
} else{Serial.println("Failed to begin communication");}//else connect
}//timerRunning
} else if (digitalRead(BUTTON_PIN_A) == HIGH && buttonPressed) {
    buttonPressed = false;
}//HIGH
if (timerRunning) {
    elapsedTime = millis() - startTime;
    Serial.println(count_ag);
    Serial.println("Recording data");
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);
    data[0] = a.acceleration.x;
    Serial.println(a.acceleration.x);
    data[1] = a.acceleration.y;
    data[2] = a.acceleration.z;
    data[3] = a.gyro.x;
    data[4] = a.gyro.y;
    data[5] = a.gyro.z;
    sprintf(cdata, "[%f,%f,%f,%f,%f,%f]", data[0], data[1], data[2], data[3], data[4], data[5]);
    Serial.println(cdata);
    Serial.println("Trying to connect to wifi now");
    if((wifiMulti.run() == WL_CONNECTED)) {
        HttpClient http;
        Serial.print("Sending");
    }
}
```

# IMPLEMENTATION: WATCH



main | Arduino IDE 2.3.2

File Edit Sketch Tools Help

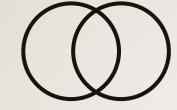
ESP32 Wrover Module

```
main.ino
250
251     data[0] = a.gy;
252     sprintf(cdata, "[%f,%f,%f,%f,%f,%f]", data[0], data[1], data[2], data[3], data[4], data[5]);
253     Serial.println(cdata);
254     Serial.println("Trying to connect to wifi now");
255     if((wifiMulti.run() == WL_CONNECTED)) {
256         HTTPClient http;
257         Serial.print("Sending");
258         Serial.print("\n");
259         Serial.print("[HTTP] begin...\n");
260         String urlString = "http://192.168.0.104:5000/sendArgs?ag_data=" + String(cdata);
261         http.begin(urlString.c_str());
262         Serial.print("[HTTP] GET...\n");
263         int httpCode = http.GET();
264         if(httpCode > 0) {
265             Serial.printf("[HTTP] GET... code: %d\n", httpCode);
266             if(httpCode == HTTP_CODE_OK) {
267                 String payload = http.getString();
268                 Serial.println(payload);
269             } else {
270                 Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
271             }
272             http.end();
273         } else{Serial.println("Failed to begin communication");}//else connect
274         char timeString[16];
275         sprintf(timeString, "%02d:%02d", minutes, seconds);
```

Help

ESP32 Wrover Module

```
case 4:
    tft.fillRect(0, 0, tft.width(), tft.height(), TFT_GREEN);
    if (digitalRead(BUTTON_PIN_A) == LOW && !buttonPressed) {
        buttonPressed = true;
        timerRunning = !timerRunning;
        if (timerRunning) {
            startTime = millis();
        } else {}
    } else if (digitalRead(BUTTON_PIN_A) == HIGH && buttonPressed) {
        buttonPressed = false;
    }//elseHIGH
    if (timerRunning) {
        elapsedTime = millis() - startTime;
        Serial.println(count_ag);
        Serial.println("Recording data");
        sensors_event_t a, g, temp;
        mpu.getEvent(&a, &g, &temp);
        data[0] = a.acceleration.x;
        Serial.println(a.acceleration.x);
        data[1] = a.acceleration.y;
        data[2] = a.acceleration.z;
        data[3] = a.gyro.x;
        data[4] = a.gyro.y;
        data[5] = a.gyro.z;
```



# DEMO VIDEO

[https://drive.google.com/file/d/12qZoK2mq690vt8tCjXizODcbNkykJB8g/view?usp=drive\\_link](https://drive.google.com/file/d/12qZoK2mq690vt8tCjXizODcbNkykJB8g/view?usp=drive_link)

# FUTURE PROSPECTS

This project opens up opportunities for further enhancements and refinements. We can continuously improve the system's functionality and user experience by leveraging data analytics and user feedback. Additionally, exploring partnerships and expanding the system's capabilities to support diverse workout routines and fitness goals will contribute to its long-term success.



5 main future prospects we would like to highlight are:

- Advancements in Hardware
- Influencer Workout led content
- UI/UX
- MedPALM Integration
- Compact PCB

# THANK YOU

The development and implementation of the iFit project have resulted in a comprehensive IoT-based fitness user management system that integrates a smartwatch and a companion mobile application.



# REFERENCES

1. Z. Liu, X. Liu and K. Li (2020). "Deeper Exercise Monitoring for Smart Gym using Fused RFID and CV Data". Available at: <https://ieeexplore.ieee.org/document/9155360>
2. Galetsi, P., Katsaliaki, K. and Kumar, S. (2022). "Exploring benefits and ethical challenges in the rise of mHealth (mobile healthcare) technology for the common good: An analysis of mobile applications for health specialists". Available at: <https://doi.org/10.1016/j.technovation.2022.102598>
3. Dinesh Kumar, A., Bhargav, K., Rayal, R. and Saraswathi, M. (2020). "Smart Gym Management System. International Journal of Scientific Research & Engineering Trends". Available at: [https://ijsret.com/wp-content/uploads/2020/05/IJSRET\\_V6\\_issue3\\_493.pdf](https://ijsret.com/wp-content/uploads/2020/05/IJSRET_V6_issue3_493.pdf)
4. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013). "Internet of things (IoT): A Vision, Architectural Elements, and Future Directions". Available at:<https://www.sciencedirect.com/science/article/pii/S0167739X13000241>
5. Bhatia M, Sood SK (2018). "An intelligent framework for workouts in gymnasium: M-health perspective". Available at: <https://www.sciencedirect.com/science/article/pii/S004579061732236X>
6. Thompson, Walter R. Ph.D., FACSM - "Worldwide Survey of Fitness Trends for 2022". Available at: [https://journals.lww.com/acsm-healthfitness/fulltext/2022/01000/worldwide\\_survey\\_of\\_fitness\\_trends\\_for\\_2022.6.aspx](https://journals.lww.com/acsm-healthfitness/fulltext/2022/01000/worldwide_survey_of_fitness_trends_for_2022.6.aspx)
7. Binbin Yong, Zijian Xu, Xin Wang, Libin Cheng, Xue Li, Xiang Wu, Qingguo Zhou - "IoT-based intelligent fitness system". Available at: <https://www.sciencedirect.com/science/article/pii/S0743731517301570>
8. Lyons, Elizabeth J. Ph.D., M.P.H.; Swartz, Maria C. Ph.D., M.P.H., RD. (2017). "Motivational Dynamics of Wearable Activity Monitors". Available at: [https://journals.lww.com/acsm-healthfitness/fulltext/2017/09000/motivational\\_dynamics\\_of\\_wearable\\_activity.8.aspx+ more](https://journals.lww.com/acsm-healthfitness/fulltext/2017/09000/motivational_dynamics_of_wearable_activity.8.aspx+ more)