Middlesex University London

PDE2101
Engineering Software Development

“IFit Studio”

PROJECT FINAL REPORT

Module Tutor: Dr. Sumitra Kotipalli

Anushka Narsima - M00851749
Ruramai Muchenga - M00912425
Joella Jose - M00850468

Table Of Content

1. ABSTRACT

This report presents the development and implementation of an IoT-based fitness management system comprising a smartwatch and a companion mobile application. The smartwatch serves as a personal fitness assistant, helping users track their fitness routines by tracking activities. The companion mobile application complements the smartwatch by providing additional features and functionalities, such as enabling users to connect with a supportive fitness community, share achievements, and access guidance and advice from the community. Our system aims to enhance the overall user experience by providing a comprehensive fitness monitoring solution that uses minimal data, provides personalised guidance tailored to users' fitness goals, and fosters a supportive fitness community.

Keywords: IoT, smartwatch, mobile application, fitness management.

2. INTRODUCTION

2.1 PROBLEM DEFINITION

In the realm of fitness management, we identified several challenges, one of which is the need for more seamless integration between workout guidance and community engagement. Current solutions often need more cohesion, leading to fragmented user experiences. Users frequently struggle to receive personalised workout guidance and engage with a supportive fitness community. This disjointed experience can hinder motivation and progress towards fitness goals.

Hypothesis:

Based on research from Healthline and IEEE Xplore, we can effectively address these challenges by developing a comprehensive IoT-based fitness management system to optimise the overall fitness experience for users by streamlining workout guidance and enhancing community engagement.

2.2 BENEFITS

1. Exercise Tracking: The hardware finds patterns with minimal data for efficient monitoring and progress tracking.

2. Personalised Workout Guidance: The smartwatch displays present and upcoming exercises, providing users with personalised workout guidance tailored to their fitness goals. This feature helps users stay motivated and on track with their workout routines.

3. Improved Community Engagement: The group chat room in the mobile application fosters a sense of community among users. They can share tips, motivate each other, and discuss fitness-related topics, enhancing the overall fitness experience and promoting accountability.

2.3 AIMS & OBJECTIVES

This project aims to create an integrated IoT-based fitness management system to enhance user experience. Objectives include developing real-time exercise tracking, providing workout guidance on a smartwatch, ensuring user authentication and personalised workout plans, fostering community engagement through a group chat feature, optimising gym management efficiency, and utilising data analytics for insights into user activity.

2.4 BRIEF OVERVIEW OF REPORT LAYOUT

The report layout following the introduction consists of three main sections: Design Methodology, Development & Implementation, Testing & Evaluation, and Conclusion. The Design Methodology section discusses the project's requirements, research and development methodologies, system architecture, and ethical considerations. This is followed by the Development & Implementation section, which covers the technologies, schematic diagrams, and code snippets. The Testing & Evaluation section details testing approaches, analysis of results, and evidence supporting the project's objectives. Finally, the conclusion section summarises the work, discusses limitations, and suggests future developments.

3. DESIGN METHODOLOGY

3.1 REQUIREMENTS SPECIFICATION

3.1.1 FUNCTIONAL

- Starting from the app – the user can either log in or register. To log in, they enter their email ID and password.
- To register, they enter the above, along with their name, phone number & date of birth. They should then be given the option to choose a gym from the ones in the system’s database and choose a membership showing price & duration. Then, you can either select or skip this.
- After logging in, they land on the main homepage, which has four features: progress summary, equipment availability, workout plan, and chatroom.

- In the progress summary, the page must show a list of all the finished workouts in descending order. The user should be able to add a finished workout with time and reps manually.
- The workout plan page should start with a ‘Generate’ option, where the user will be redirected to a form & asked some questions related to their goals, based on which the customised plan will be generated by a decision tree.
- The current daily plan will be displayed below the button. Each item must show the workout name, time, & reps if applicable. The user should be able to edit the reps/time if they want to, so this number will come with ‘+’ & ‘-‘ buttons. To sum it up, the page should show the total time and number of exercises.
- The chatroom is a standard page that displays chats, with a bar below to type & send.
- Each page must have a navbar with options to go to the homepage and log out.
- The user is supposed to click button two on their watch at the beginning & end of each exercise so that the system records their progress correctly.
- The watch should show the current (time/reps) & next workout in the plan. When the plan for the day is over, it should show finished.
- After the workout page, it should allow recording a new workout. It starts tracking upon clicking the second button and stops upon clicking again. The user is expected to do 20 repetitions of this new workout if it is repetition-based.
- There must be an add workout option on the app's main page, where the user can view these recorded data and set the details for it to be added to the workout database. If repetition-based, this will run through an algorithm to analyse the recorded data for future repetition prediction.
- Personalised workout plans using robust algorithms are another key functionality, as the system must generate plans tailored to individual fitness goals, capabilities, and preferences. So, users must be able to input their timings and should be able to add or remove exercises from the plan.

3.1.2 NON-FUNCTIONAL

The system ensures reliability and accurate data tracking. It is user-friendly, with interfaces on both the smartwatch and mobile app. The system has low latency in displaying real-time data and notifications. It also prioritises security, with robust user authentication mechanisms and safekeeping of sensitive data. Finally, the system is scalable to accommodate future updates and expansions.

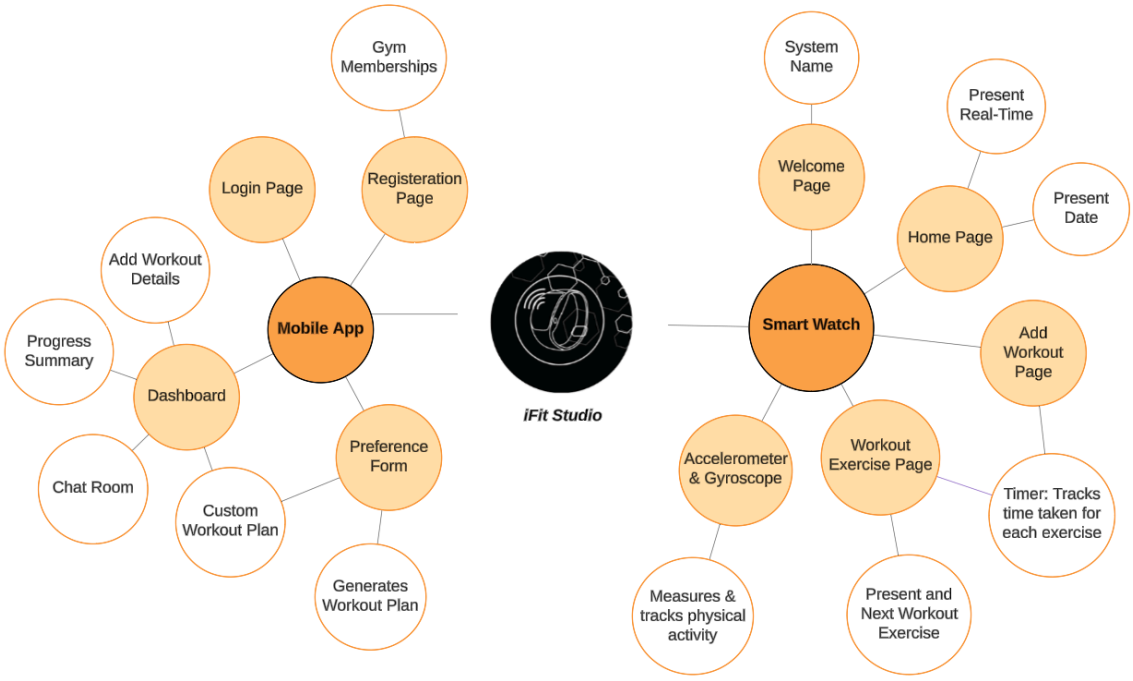
3.2 RESEARCH & DEVELOPMENT METHODOLOGY

A combination of exploratory, applied and quantitative research and iterative development methodologies have been employed for the final implementation of our project. Exploratory research has been conducted to observe trends, challenges, and opportunities in the IoT industry, mainly focusing on applications related to the fitness sector. Applied research was done to observe existing methods and identify areas of improvement. This allowed us to understand the landscape better and inform decision-making throughout the project lifecycle.

In terms of development methodology, the agile approach, specifically the Scrum framework, was adopted. This methodology is well-suited for our project as it involves multiple small components, allowing easier debugging and integration. The timeline was over several months & was a group project, so there was lots of collaboration & feedback involved. The project is divided into several sprints throughout the development period, with regular meetings to update progress, plan actions, retrospect, and address any issues. The agile approach's fast-paced and iterative nature facilitated rapid implementation and ensured the production of a quality product.

3.3 COMPREHENSIVE DESIGN OVERVIEW

3.3.1 MIND MAP FEATURES



3.3.2 SYSTEM ARCHITECTURE

The project's system architecture integrates a few hardware components for seamless fitness tracking and management. At its core is the ESP32 board with an inbuilt Wi-Fi module, serving as the central processing unit. Connected to the ESP32 is the LCD 1.28-inch Waveshare display for real-time data visualisation and the MPU6050 accelerometer and gyroscope for movement tracking.

Additionally, two buttons allow user interaction: one for navigating LCD display pages and another for starting/stopping workout timers. A power supply powers the system, and jumper wires and a breadboard are used to connect and prototype the hardware components.

3.3.3 INTERFACE DESIGN

The interface design aims to provide a user-friendly experience on the smartwatch and companion mobile application and is focused on simplicity, usability, and visual appeal. The LCD on the smartwatch presents essential fitness metrics such as movement tracking, the present and following workout exercises, and the addition of new workouts. Users can easily navigate the display pages and track their time for each workout using two distinct buttons.

The mobile application features a clean and intuitive layout, with separate sections for the registration page, login page, and dashboard, including options: custom workout plans with a preference form to generate the workout plan, progress summary, community chat room, and add workout details. Users can log in securely using their login credentials and access personalised workout recommendations based on their fitness goals. The interface is designed to provide a seamless experience across the smartwatch and mobile app, ensuring consistency with easy-to-understand icons and buttons for seamless navigation.

3.3.4 INTERACTION DESIGNS

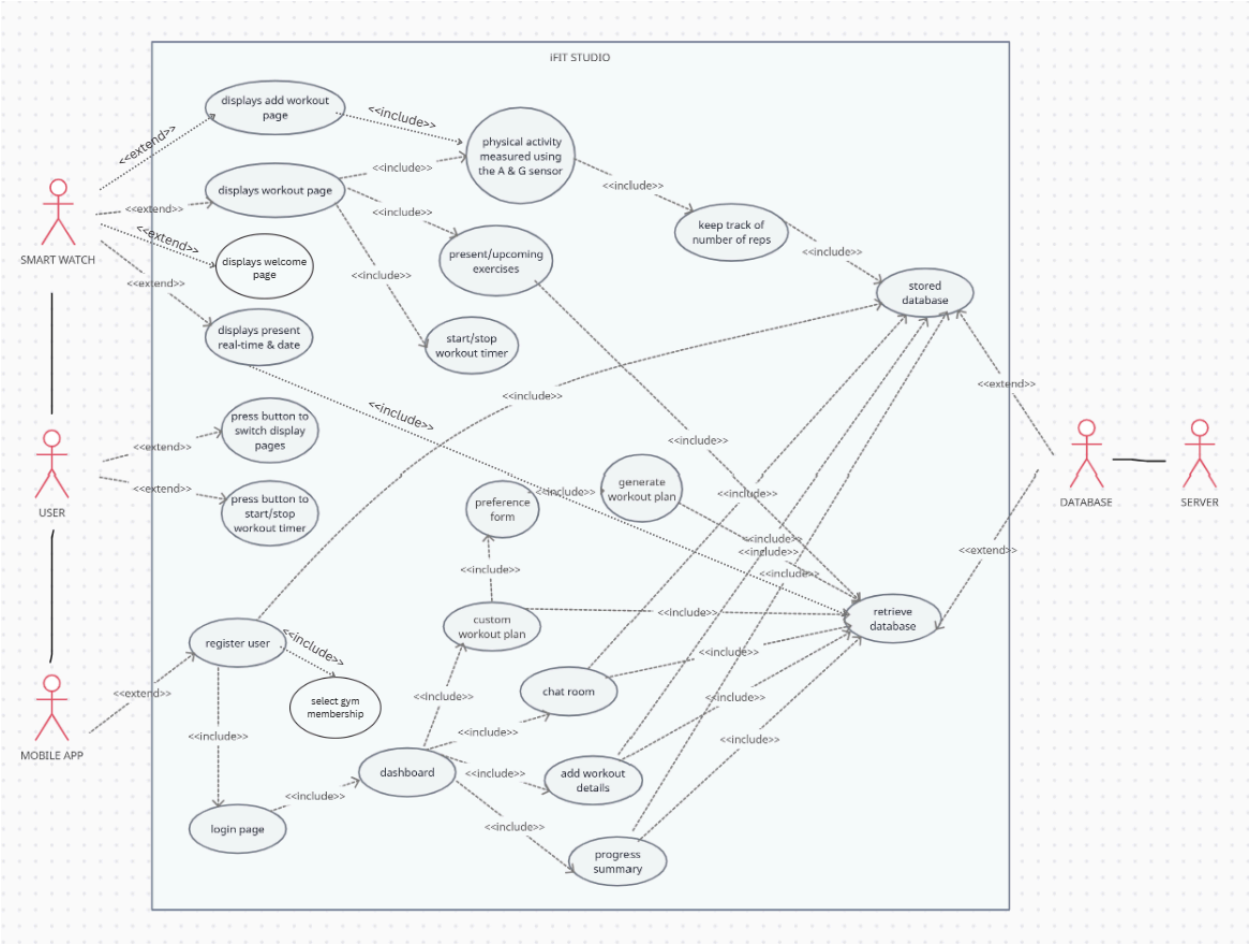
Here, it is focused on providing intuitive and efficient user interactions across all hardware components. On the smartwatch, users can interact with the interface using button inputs. They can navigate through display pages and start/stop workout timers.

The mobile application lets users interact with their workout plans, equipment availability status, and community chat. The interface allows easy and secure access to one's workout plans and encourages community engagement through the chat feature.

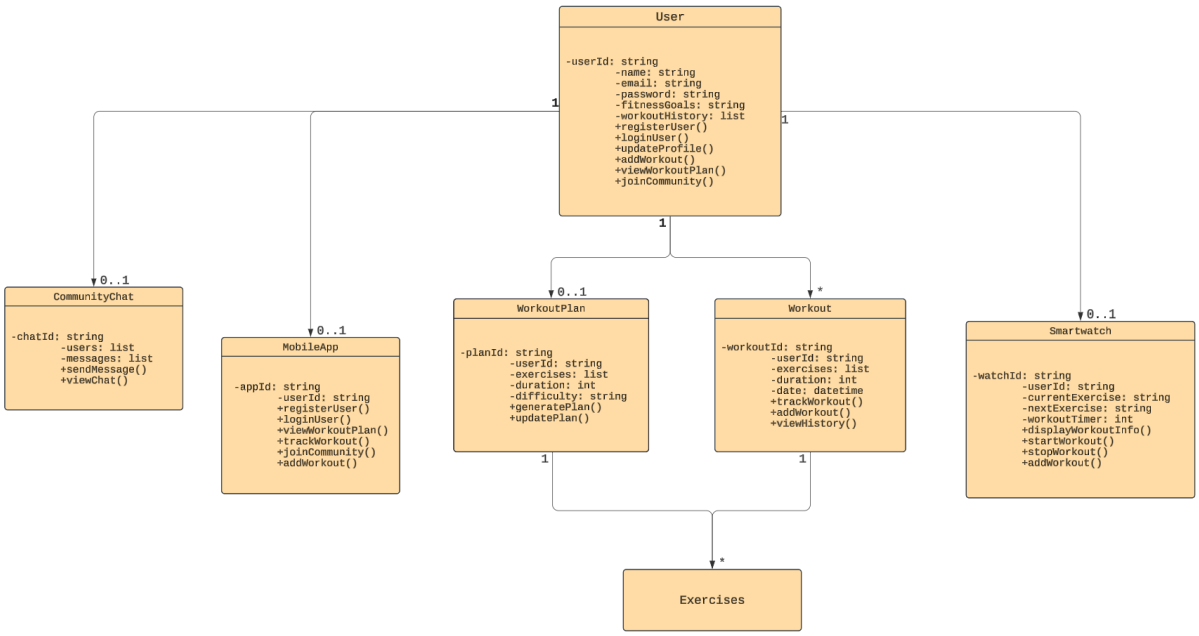
The interaction designs aim to provide a seamless and engaging user experience, facilitating efficient fitness tracking and management.

3.4 UML DIAGRAMS

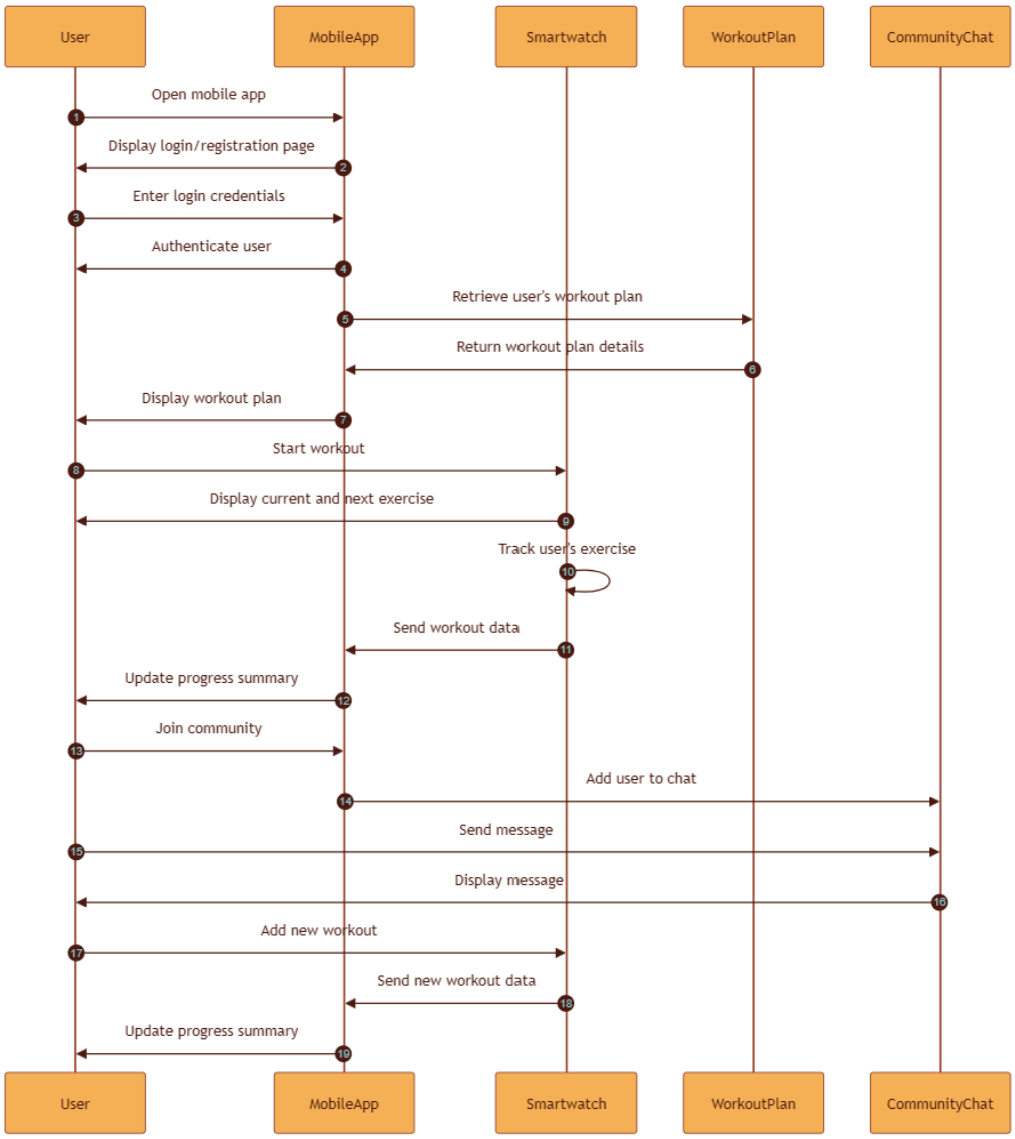
3.4.1 USE CASE



3.4.2 CLASS

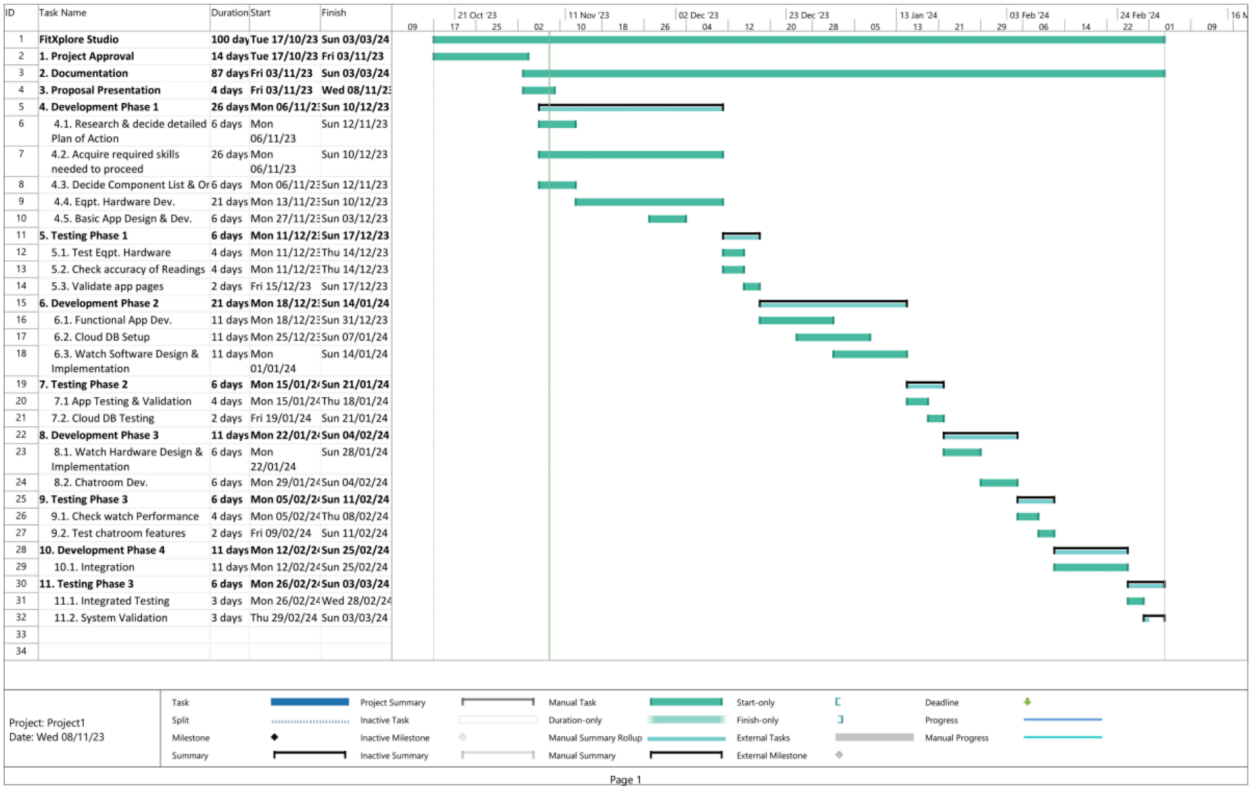

User
-userId: string
-name: string
-email: string
-password: string
-fitnessGoals: string
-workoutHistory: list
+registerUser()
+loginUser()
+updateProfile()
+addWorkout()
+viewWorkoutPlan()
+joinCommunity()
CommunityChat
-chatId: string
-users: list
-messages: list
+sendMessage()
+viewChat()
MobileApp
-appId: string
-userId: string
+registerUser()
+loginUser()
+viewWorkoutPlan()
+trackWorkout()
+joinCommunity()
+addWorkout()
WorkoutPlan
-planId: string
-userId: string
-exercises: list
-duration: int
-difficulty: string
+generatePlan()
+updatePlan()
Workout
-workoutId: string
-userId: string
-exercises: list
-duration: int
-date: datetime
+trackWorkout()
+addWorkout()
+viewHistory()
Smartwatch
-watchId: string
-userId: string
-currentExercise: string
-nextExercise: string
-workoutTimer: int
+displayWorkoutInfo()
+startWorkout()
+stopWorkout()
+addWorkout()
Exercises
1
1
1
0..1
0..1
0..1
*
1
1
*

3.4.3 SEQUENCE



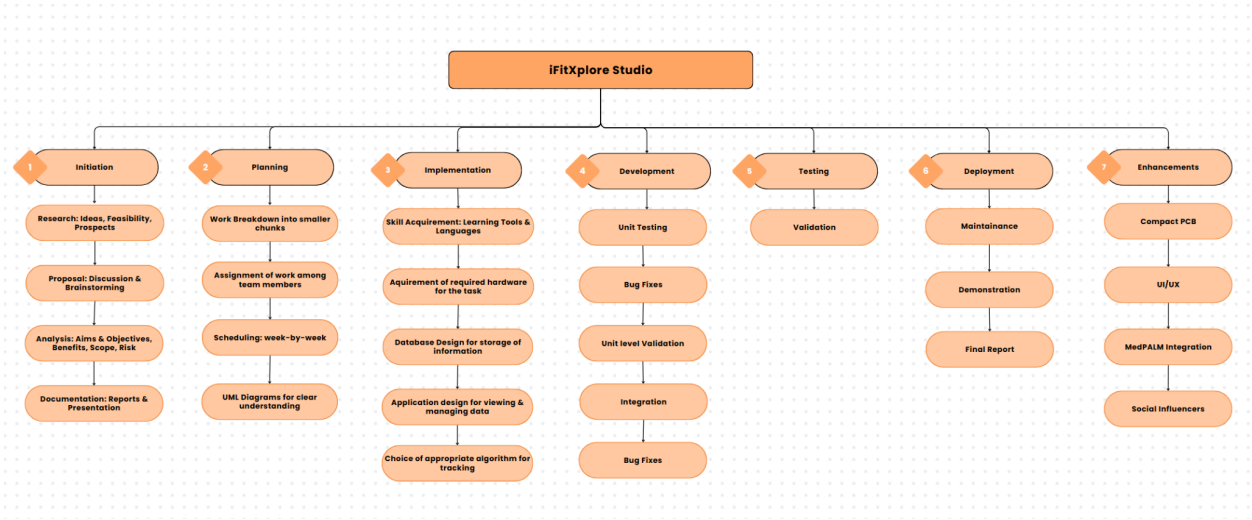
3.5 PROJECT MANAGEMENT DIAGRAMS

3.5.1 GANTT CHART



ID	Task Name	Duration	Start	Finish
1	**FitXplore Studio**	**100 day**	**Tue 17/10/23**	**Sun 03/03/24**
2	**1. Project Approval**	**14 days**	**Tue 17/10/23**	**Fri 03/11/23**
3	**2. Documentation**	**87 days**	**Fri 03/11/23**	**Sun 03/03/24**
4	**3. Proposal Presentation**	**4 days**	**Fri 03/11/23**	**Wed 08/11/2**
5	**4. Development Phase 1**	**26 days**	**Mon 06/11/2**	**Sun 10/12/23**
6	4.1. Research & decide detailed Plan of Action	6 days	Mon 06/11/23	Sun 12/11/23
7	4.2. Acquire required skills needed to proceed	26 days	Mon 06/11/23	Sun 10/12/23
8	4.3. Decide Component List & Or	6 days	Mon 06/11/2	Sun 12/11/23
9	4.4. Eqpt. Hardware Dev.	21 days	Mon 13/11/2	Sun 10/12/23
10	4.5. Basic App Design & Dev.	6 days	Mon 27/11/2	Sun 03/12/23
11	**5. Testing Phase 1**	**6 days**	**Mon 11/12/2**	**Sun 17/12/23**
12	5.1. Test Eqpt. Hardware	4 days	Mon 11/12/2	Thu 14/12/23
13	5.2. Check accuracy of Readings	4 days	Mon 11/12/2	Thu 14/12/23
14	5.3. Validate app pages	2 days	Fri 15/12/23	Sun 17/12/23
15	**6. Development Phase 2**	**21 days**	**Mon 18/12/2**	**Sun 14/01/24**
16	6.1. Functional App Dev.	11 days	Mon 18/12/2	Sun 31/12/23
17	6.2. Cloud DB Setup	11 days	Mon 25/12/2	Sun 07/01/24
18	6.3. Watch Software Design & Implementation	11 days	Mon 01/01/24	Sun 14/01/24
19	**7. Testing Phase 2**	**6 days**	**Mon 15/01/2**	**Sun 21/01/24**
20	7.1 App Testing & Validation	4 days	Mon 15/01/2	Thu 18/01/24
21	7.2. Cloud DB Testing	2 days	Fri 19/01/24	Sun 21/01/24
22	**8. Development Phase 3**	**11 days**	**Mon 22/01/2**	**Sun 04/02/24**
23	8.1. Watch Hardware Design & Implementation	6 days	Mon 22/01/24	Sun 28/01/24
24	8.2. Chatroom Dev.	6 days	Mon 29/01/2	Sun 04/02/24
25	**9. Testing Phase 3**	**6 days**	**Mon 05/02/2**	**Sun 11/02/24**
26	9.1. Check watch Performance	4 days	Mon 05/02/2	Thu 08/02/24
27	9.2. Test chatroom features	2 days	Fri 09/02/24	Sun 11/02/24
28	**10. Development Phase 4**	**11 days**	**Mon 12/02/2**	**Sun 25/02/24**
29	10.1. Integration	11 days	Mon 12/02/2	Sun 25/02/24
30	**11. Testing Phase 3**	**6 days**	**Mon 26/02/2**	**Sun 03/03/24**
31	11.1. Integrated Testing	3 days	Mon 26/02/2	Wed 28/02/2
32	11.2. System Validation	3 days	Thu 29/02/24	Sun 03/03/24
33				
34				

3.5.2 WORK BREAKDOWN STRUCTURE



3.5.3 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)

3.6 ETHICAL ISSUES

While "iFit Studio" project aims to enhance the fitness experience of gym-goers through IoT technology, it also raises significant ethical considerations. Research has shown that data management privacy, security, and accuracy issues are one of the most critical challenges when using digital information technology and apps (Yaacoub et al., 2020). As the system collects and stores a wealth of user data, including workout information, it is imperative to guarantee the utmost security for this sensitive information in compliance with data protection regulations.

Additionally, user consent is a fundamental ethical principle; individuals must be informed about what data is collected and how it will be used (Johnson, 2021).
Furthermore, the system's reliance on algorithms to generate personalised workout plans necessitates vigilance against biases based on user characteristics, including data accuracy. The hardware and sensors used in the system must provide precise and reliable data. Inaccuracies could not only lead to incorrect workouts but may also pose risks to users' health.

To address these ethical concerns, we established clear privacy data management features such as authorisation, password protection, and registration to enable navigation to the app. In addition, we can include trust-based features such as backing from credible sources (Galetsi, 2023). This approach will build user trust and foster the project's long-term success and positive impact on the fitness community.

Implementing robust encryption and authentication measures is crucial to protect user data from potential threats. Furthermore, it's essential to establish clear guidelines for data retention and allow users to control their information, ensuring compliance with data protection regulations like GDPR (Williams, 2019). Addressing these ethical issues can promote user trust and foster a safer environment for fitness management applications.

4. DEVELOPMENT & IMPLEMENTATION

4.1 TECHNOLOGIES & DEVELOPMENT TOOLS

4.1.1 HARDWARE

1. LCD 1.28-inch Waveshare

   We chose the 1.28-inch Waveshare LCD for its compact and round design, which fits our project's aesthetic and ergonomic requirements well. Its small form factor makes it suitable for wearable devices or compact setups. This display module provided visual feedback to the user during workouts, displaying information such as exercise instructions, time elapsed, and workout status. Its round shape adds a modern and sleek look to the overall design.

2. Accelerometer & Gyroscope MPU 6050

   The MPU6050 was selected for its combined accelerometer and gyroscope functionality, allowing us to track user movement accurately. This component monitors workout activities, providing real-time performance and motion analysis data. By integrating this sensor, we can measure the number of reps done for each workout.

3. ESP 32 Board

   The ESP32 board was chosen for its versatility and powerful capabilities, including Wi-Fi connectivity. As the central controller of our system, it manages data processing, communication with other devices, and interaction with the user interface. With its robust features and ample processing power, the ESP32 enables seamless integration of various hardware components. It supports the implementation of advanced IoT functionalities, such as cloud connectivity and remote monitoring.

4. Buttons

   The buttons served as simple yet effective input devices for user interaction. We incorporated them into our project to enable timer control and navigation through the interface. They allow users to start/stop workout timers and navigate the display pages conveniently.

5. Power Supply

A reliable power supply is essential for ensuring uninterrupted operation of the system. We selected a suitable power supply to provide stable voltage and current to all components, ensuring they function properly without voltage fluctuations or power disruptions. This ensures consistent performance and prevents unexpected shutdowns during workouts or data processing.

6. Jumper wires and a Breadboard

   Jumper wires and a breadboard were used for prototyping and testing the circuitry of our project. They provided a flexible and easy-to-use platform for making electrical connections between components, allowing for quick iteration and troubleshooting during development. The breadboard allowed components to be easily arranged and connected without soldering.

4.1.2 SOFTWARE

1. Flutter

   We used Flutter to create our mobile application for our project, enabling a seamless user experience across multiple platforms. The framework's flexibility allowed the implementation of key features such as sign-in, sign-up, and password reset functionalities, all integrated with Firebase Authentication to ensure secure user management (Brown, 2021). Custom workout plans were developed using Firestore, a real-time NoSQL database, allowing users to personalise their fitness routines and track their progress. The app also includes a feature for users to add new workouts, promoting flexibility and user-driven customisation (Jones, 2020). A chatroom feature was integrated, allowing users to interact with the fitness community, share advice, and motivate one another. This functionality was built with Firestore to ensure real-time communication and data synchronisation (Smith, 2019). Overall, Flutter's cross-platform capabilities and integration with Firebase's authentication and database features proved instrumental in developing a robust and feature-rich mobile application.

2. Firebase Firestore

   Firebase Firestore played a crucial role in our project. It provided the flexibility and real-time data synchronisation needed to manage various collections within the mobile application. The gym collection served as a central hub for all gyms, with a chatroom subcollection allowing members to communicate, share tips, and foster a sense of community. This subcollection was vital in creating an engaging environment where users could interact in real time.

   The user's collection stored essential user information, including personal details, authentication data, and workout progress. This structure allowed for efficient retrieval and update of user-specific information, facilitating personalised workout plans and tracking progress over time. Firestore's security rules ensured access to this sensitive data was restricted to authorised users, maintaining privacy and data integrity.

   In addition, the equipment collection provided a comprehensive list of all available gym equipment, including details such as equipment type, status, and location. This collection enabled the mobile app to offer users real-time insights into the equipment available at their gym, contributing to a more seamless workout planning experience.

3. Arduino IDE

   Arduino IDE (Integrated Development Environment) is a software tool for programming Arduino microcontrollers and compatible boards. It provides a simple and intuitive platform for writing, compiling, and uploading code to control hardware components. We have used this for its compatibility with the ESP32 microcontroller and MPU6050 accelerometer sensor. It offers ease of use, extensive library support, and a supportive community, making it ideal for programming and interfacing with hardware components in IoT applications.

4. Python - Flask, Matplotlib, Pandas

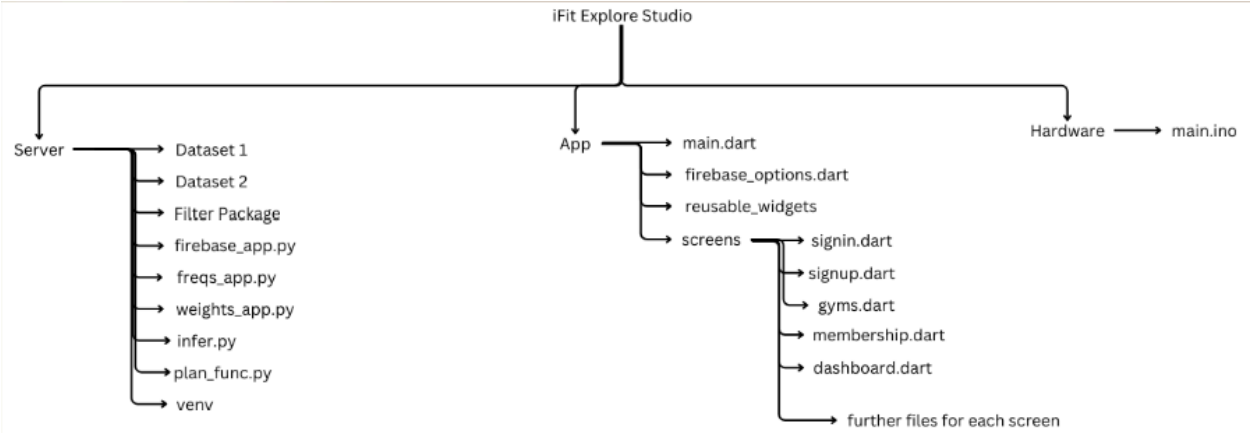
   The Python server was the heart of the system - it took care of most of the computation and served as a connector between the app and the hardware.

Various frameworks, like Pandas and Scipy, could be accessed for data processing. This decision accelerated the development process due to the team's previous experience with this language.

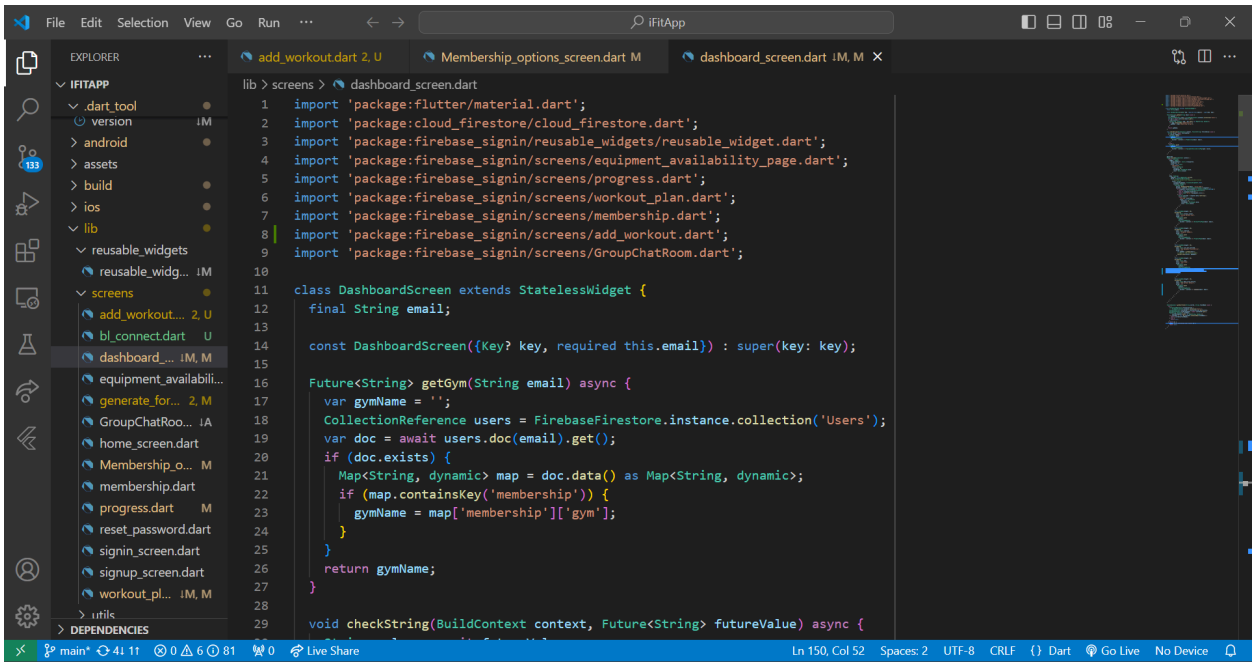
5. Git/GitHub
   Git provided a reliable platform where our team could post, review, and manage the codebase, enabling seamless contributions from multiple developers. With Git, we could track changes to the code, branch out for specific features, and merge updates without overwriting each other's work. This approach promoted efficient teamwork and minimised conflicts during integration.

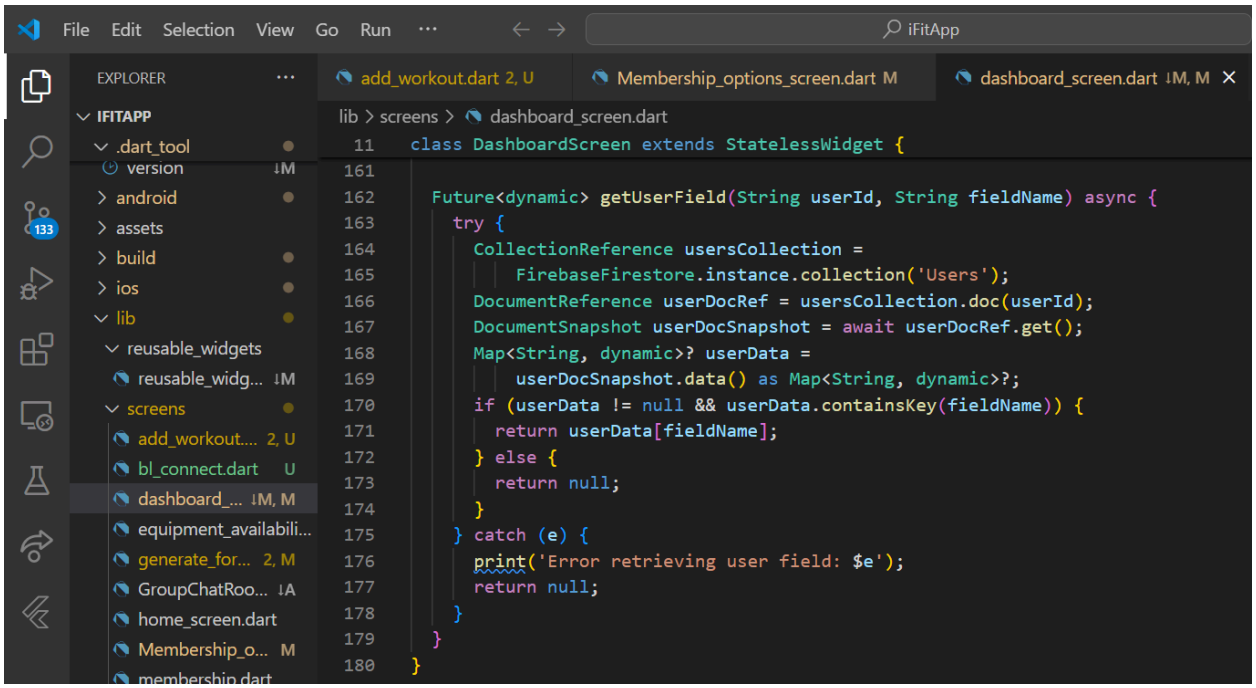
4.3 IMPLEMENTATION

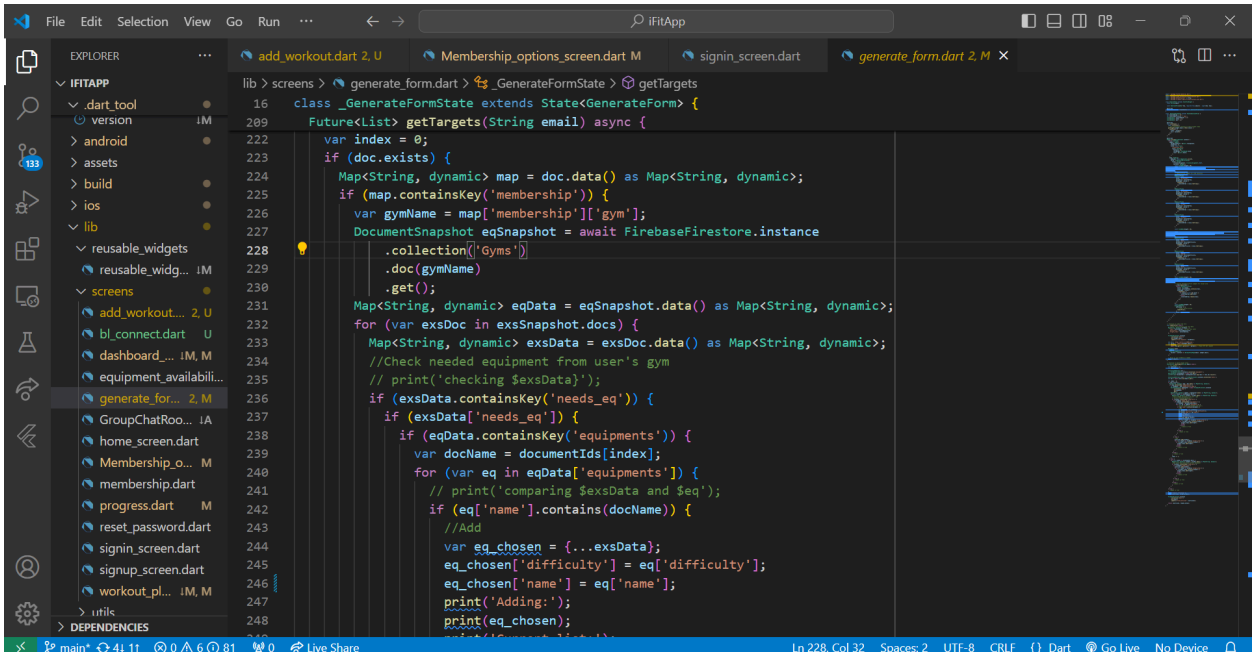
4.3.1 OVERVIEW

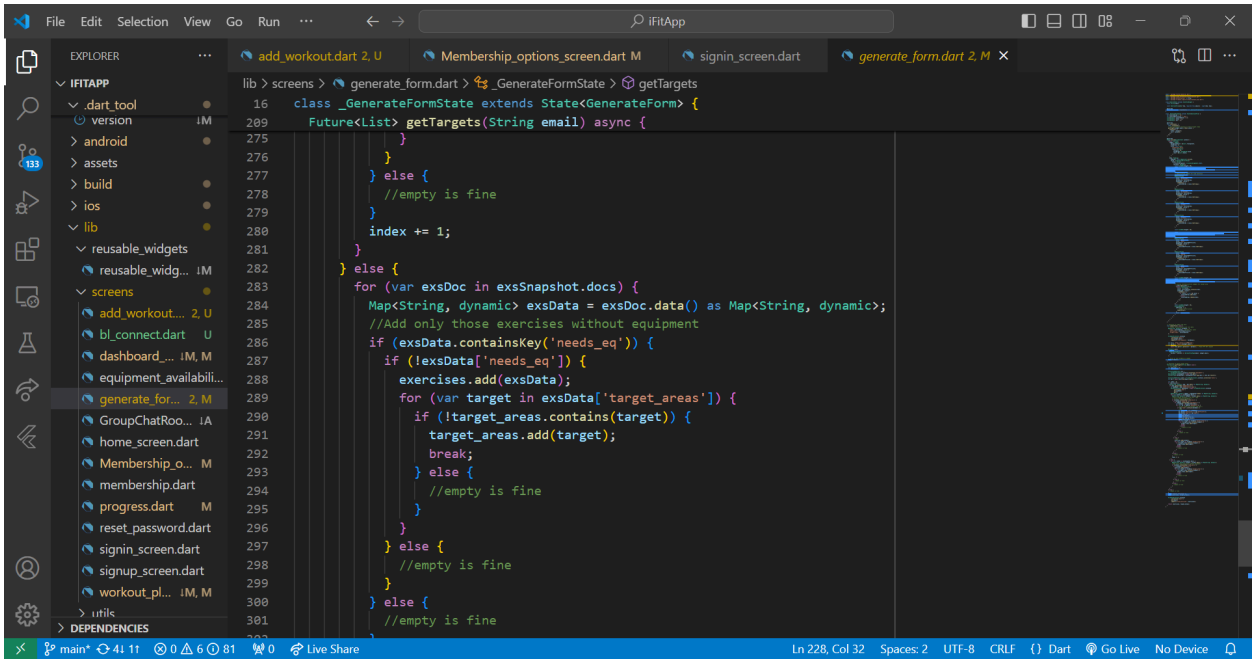


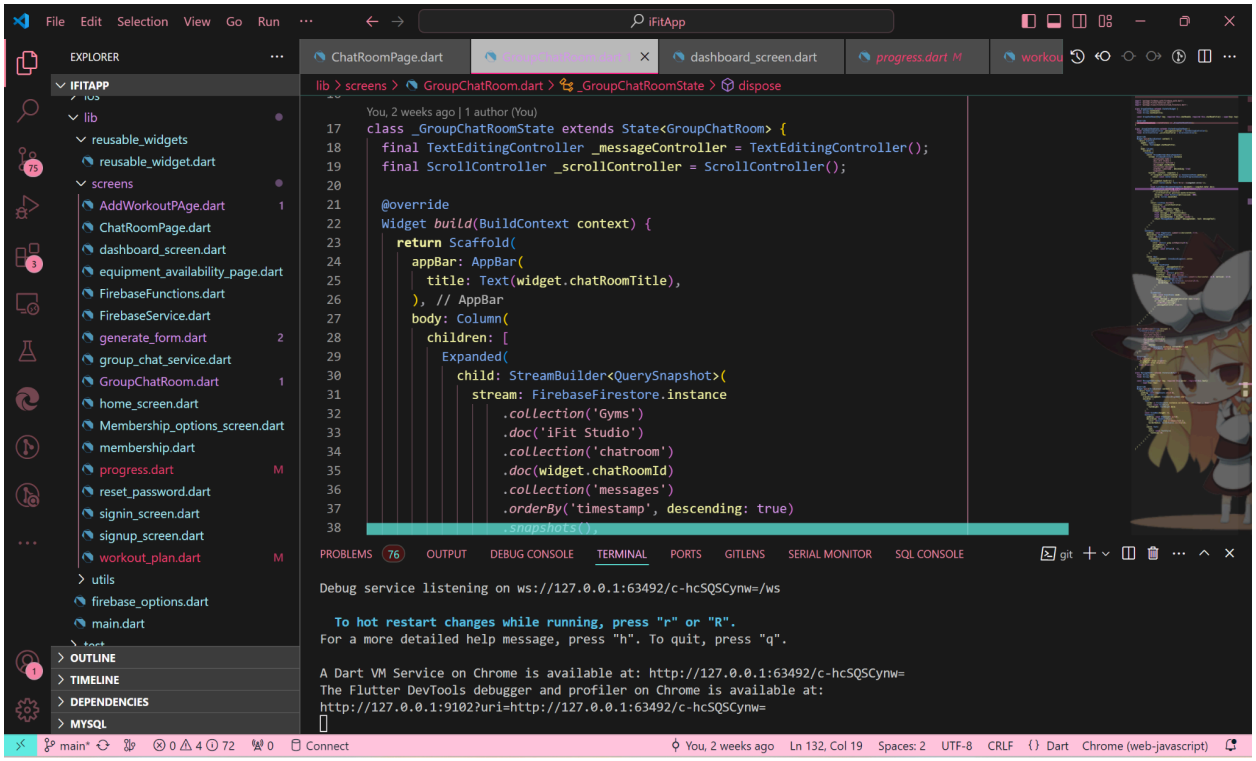
4.3.2 MOBILE APP



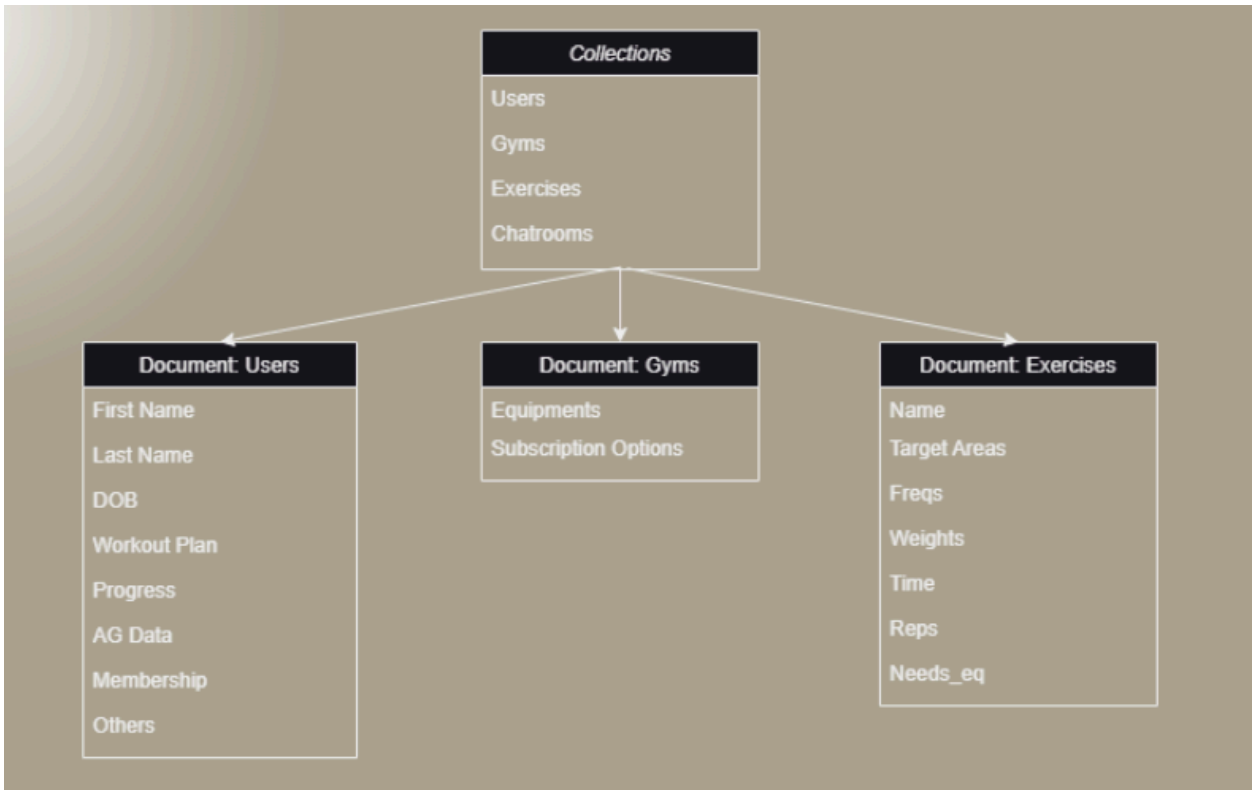



File Edit Selection View Go Run
iFitApp
EXPLORER
add_workout.dart 2, U
Membership_options_screen.dart M
signin_screen.dart
generate_form.dart 2, M
lib > screens > generate_form.dart > _GenerateFormState > getTargets
class _GenerateFormState extends State<GenerateForm> {
Future<List> getTargets(String email) async {
var index = 0;
if (doc.exists) {
Map<String, dynamic> map = doc.data() as Map<String, dynamic>;
if (map.containsKey('membership')) {
var gymName = map['membership']['gym'];
DocumentSnapshot eqSnapshot = await FirebaseFirestore.instance
.collection('Gyms')
.doc(gymName)
.get();
Map<String, dynamic> eqData = eqSnapshot.data() as Map<String, dynamic>;
for (var exsDoc in exsSnapshot.docs) {
Map<String, dynamic> exsData = exsDoc.data() as Map<String, dynamic>;
//Check needed equipment from user's gym
// print('checking $exsData}');
if (exsData.containsKey('needs_eq')) {
if (exsData['needs_eq']) {
if (eqData.containsKey('equipments')) {
var docName = documentIds[index];
for (var eq in eqData['equipments']) {
// print('comparing $exsData and $eq');
if (eq['name'].contains(docName)) {
//Add
var eq_chosen = {...exsData};
eq_chosen['difficulty'] = eq['difficulty'];
eq_chosen['name'] = eq['name'];
print('Adding:');
print(eq_chosen);
Ln 228, Col 32
Spaces: 2
UTF-8
CRLF
Dart
Go Live
No Device

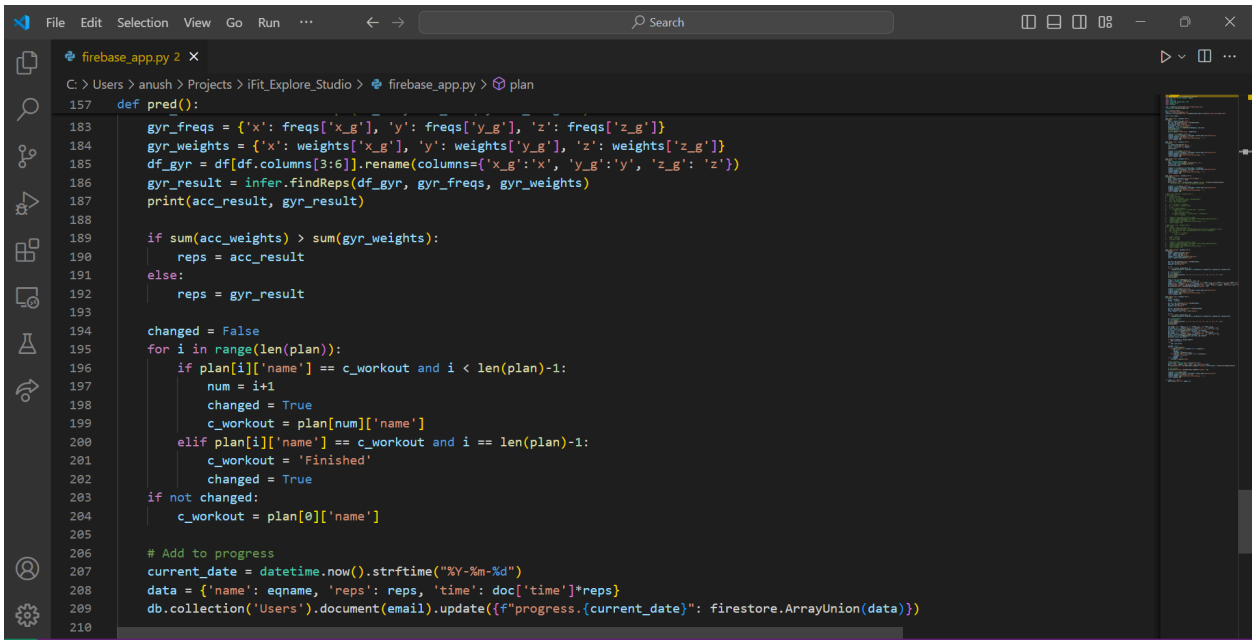

File Edit Selection View Go Run
iFitApp
EXPLORER
add_workout.dart 2, U
Membership_options_screen.dart M
signin_screen.dart
generate_form.dart 2, M
lib > screens > generate_form.dart > _GenerateFormState > getTargets
class _GenerateFormState extends State<GenerateForm> {
Future<List> getTargets(String email) async {
}
}
} else {
//empty is fine
}
index += 1;
}
} else {
for (var exsDoc in exsSnapshot.docs) {
Map<String, dynamic> exsData = exsDoc.data() as Map<String, dynamic>;
//Add only those exercises without equipment
if (exsData.containsKey('needs_eq')) {
if (!exsData['needs_eq']) {
exercises.add(exsData);
for (var target in exsData['target_areas']) {
if (!target_areas.contains(target)) {
target_areas.add(target);
break;
} else {
//empty is fine
}
}
} else {
//empty is fine
}
} else {
//empty is fine
Ln 228, Col 32
Spaces: 2
UTF-8
CRLF
Dart
Go Live
No Device


File Edit Selection View Go Run
iFitApp
EXPLORER
IFITAPP
lib
reusable_widgets
reusable_widget.dart
screens
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
workout_plan.dart
utils
firebase_options.dart
main.dart
OUTLINE
TIMELINE
DEPENDENCIES
MYSQL
ChatRoomPage.dart
dashboard_screen.dart
progress.dart M
lib > screens > GroupChatRoom.dart > _GroupChatRoomState > dispose
You, 2 weeks ago | 1 author (You)
17 class _GroupChatRoomState extends State<GroupChatRoom> {
18 final TextEditingController _messageController = TextEditingController();
19 final ScrollController _scrollController = ScrollController();
20
21 @override
22 Widget build(BuildContext context) {
23 return Scaffold(
24 appBar: AppBar(
25 title: Text(widget.chatRoomTitle),
26), // AppBar
27 body: Column(
28 children: [
29 Expanded(
30 child: StreamBuilder<QuerySnapshot>(
31 stream: FirebaseFirestore.instance
32 .collection('Gyms')
33 .doc('iFit Studio')
34 .collection('chatroom')
35 .doc(widget.chatRoomId)
36 .collection('messages')
37 .orderBy('timestamp', descending: true)
38 .snapshots(),
PROBLEMS 76 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SERIAL MONITOR SQL CONSOLE
Debug service listening on ws://127.0.0.1:63492/c-hcSQSCynw=/ws
To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".
A Dart VM Service on Chrome is available at: http://127.0.0.1:63492/c-hcSQSCynw=
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:9102?uri=http://127.0.0.1:63492/c-hcSQSCynw=
main* Connect
You, 2 weeks ago Ln 132, Col 19 Spaces: 2 UTF-8 CRLF Dart Chrome (web-javascript)

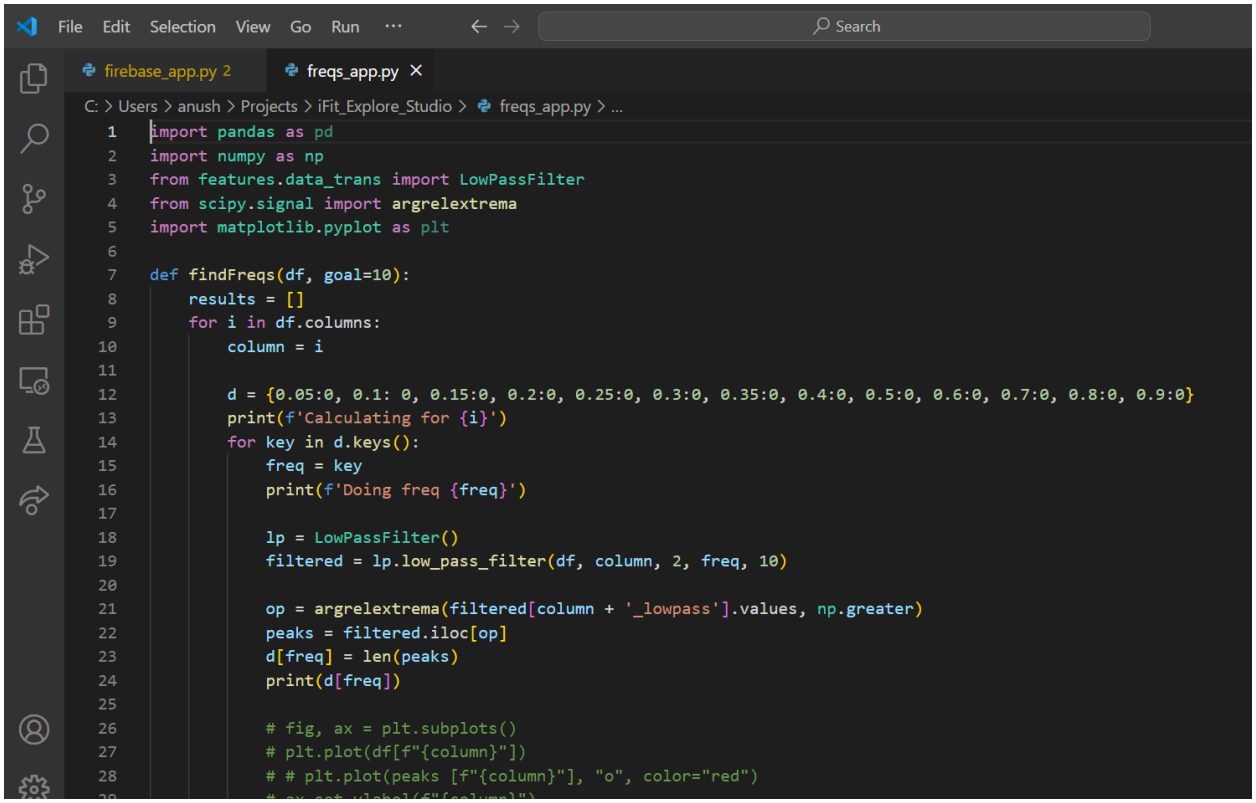
4.3.3 DATA


Collections
Users
Gyms
Exercises
Chatrooms
Document: Users
First Name
Last Name
DOB
Workout Plan
Progress
AG Data
Membership
Others
Document: Gyms
Equipments
Subscription Options
Document: Exercises
Name
Target Areas
Freqs
Weights
Time
Reps
Needs_eq

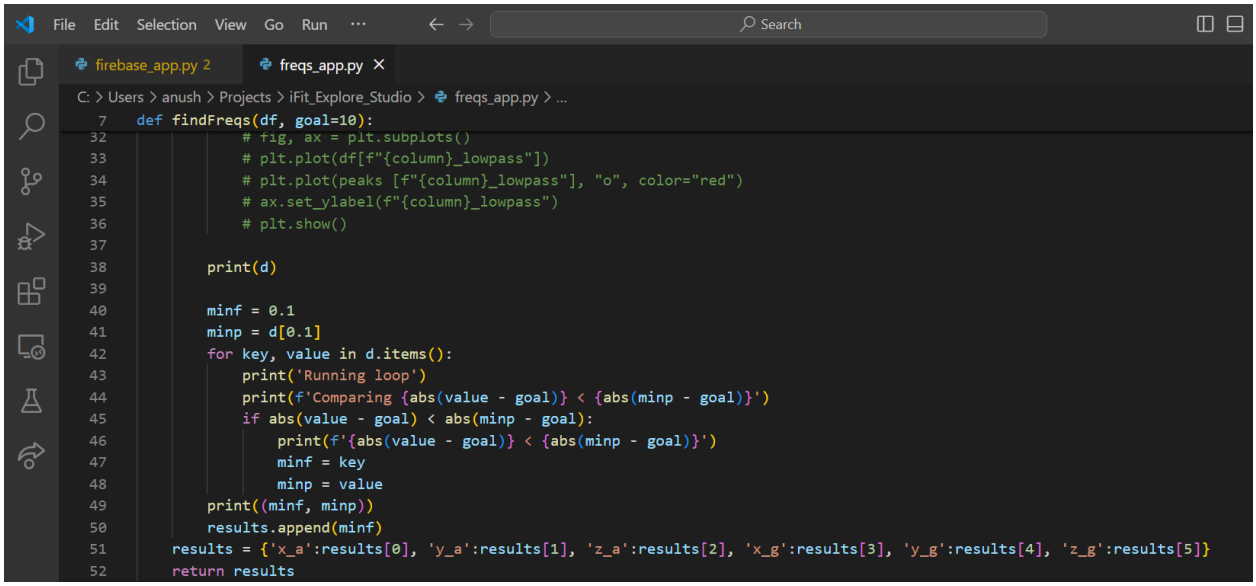
4.3.4 SERVER

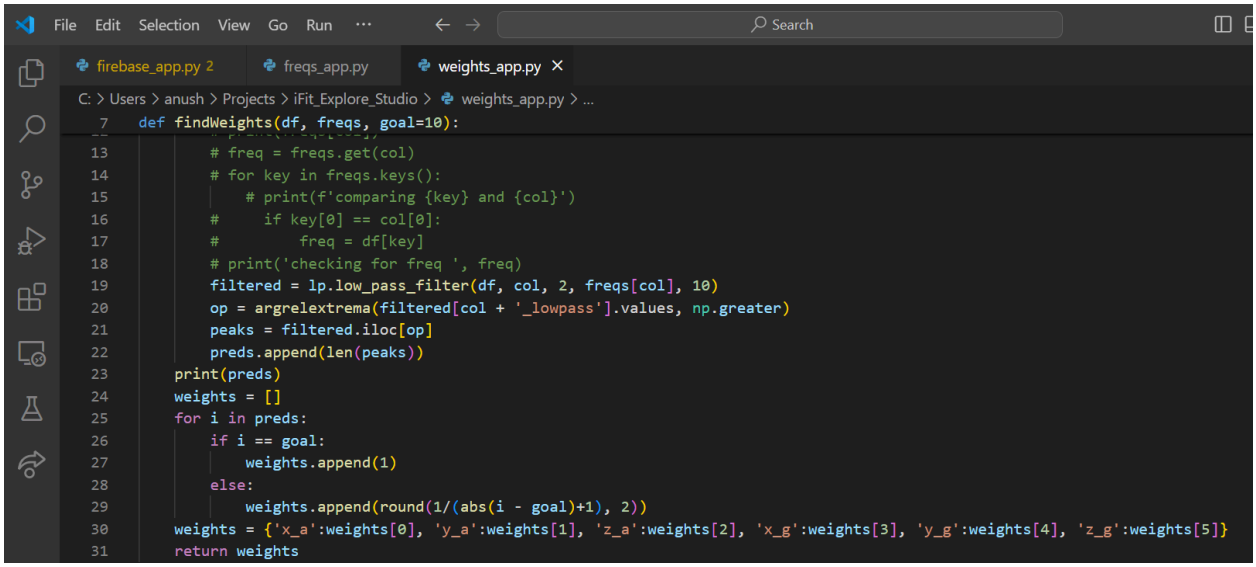


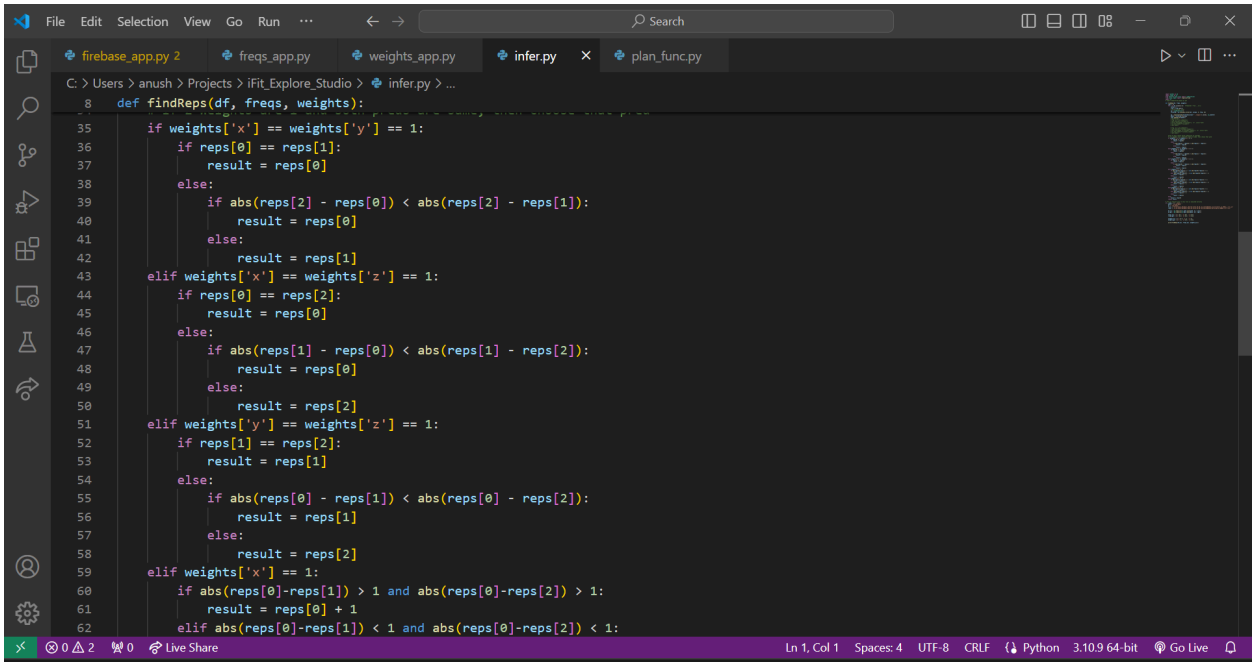
```
157  def pred():
183      gyr_freqs = {'x': freqs['x_g'], 'y': freqs['y_g'], 'z': freqs['z_g']}
184      gyr_weights = {'x': weights['x_g'], 'y': weights['y_g'], 'z': weights['z_g']}
185      df_gyr = df[df.columns[3:6]].rename(columns={'x_g':'x', 'y_g':'y', 'z_g': 'z'})
186      gyr_result = infer.findReps(df_gyr, 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
```

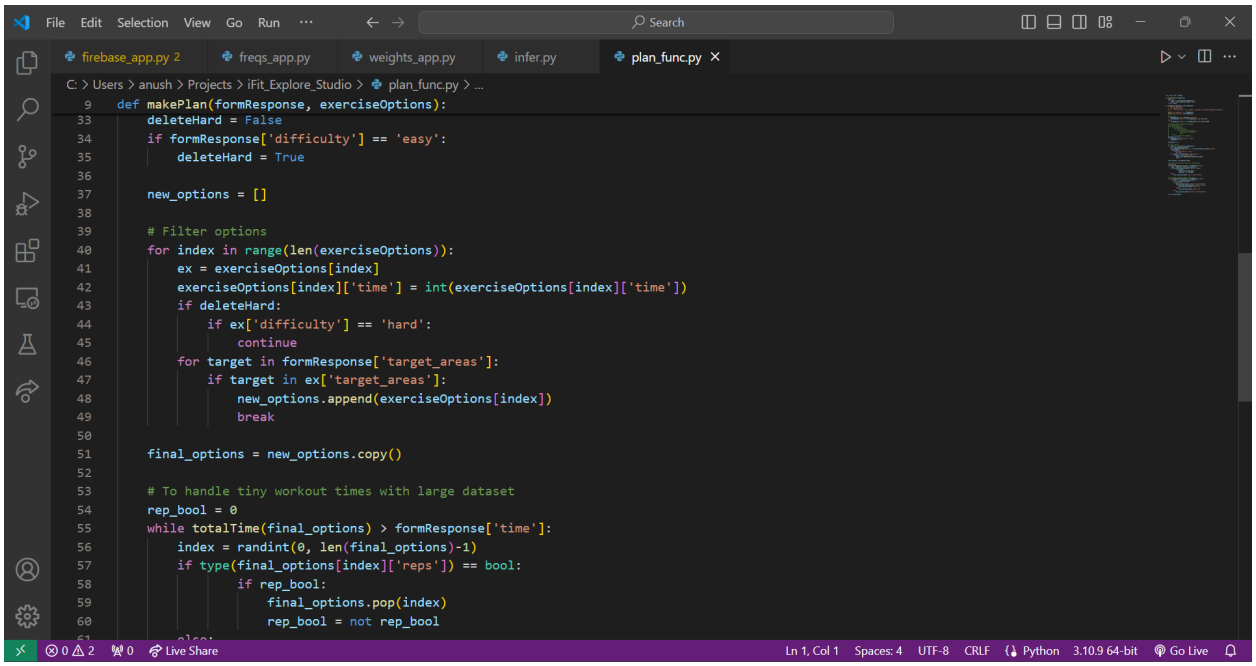


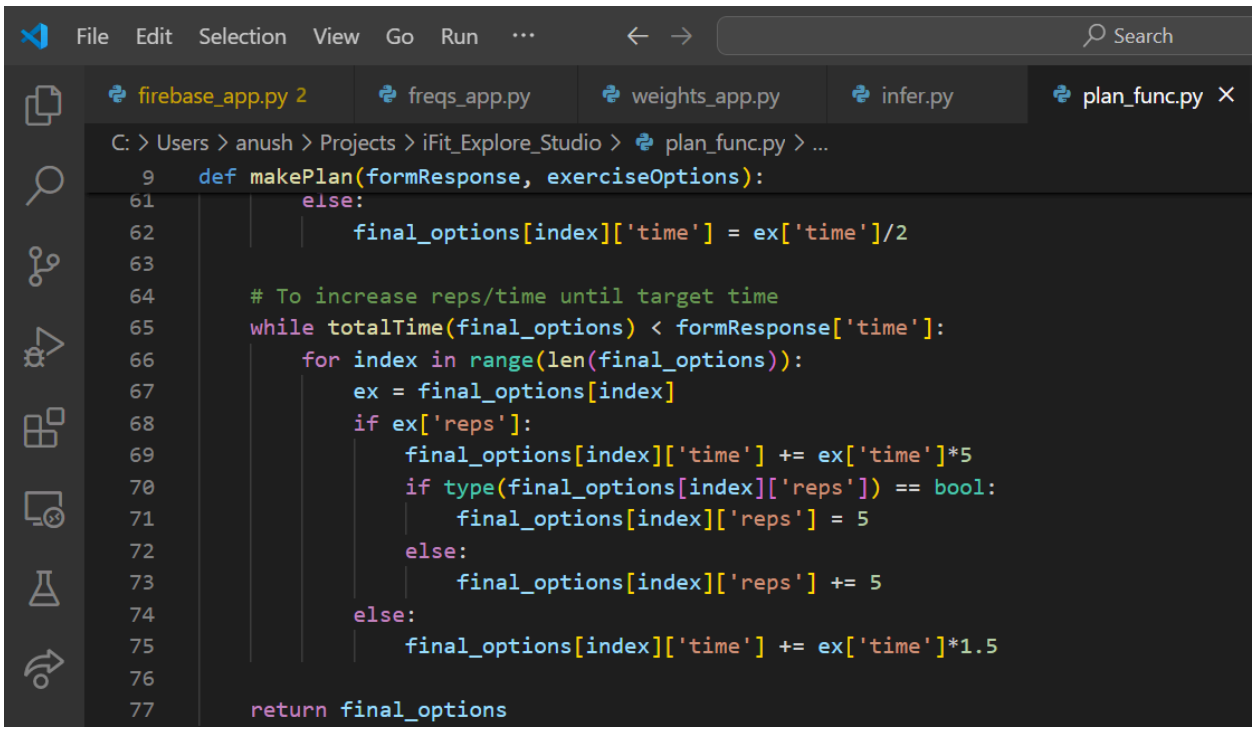
```
1   import pandas as pd
2   import numpy as np
3   from features.data_trans import LowPassFilter
4   from scipy.signal import argrelextrema
5   import matplotlib.pyplot as plt
6
7   def findFreqs(df, goal=10):
8       results = []
9       for i in df.columns:
10          column = i
11
12          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")
```


File Edit Selection View Go Run ···
Search
firebase_app.py 2
freqs_app.py
C: > Users > anush > Projects > iFit_Explore_Studio > freqs_app.py > ...
def findFreqs(df, goal=10):
# 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 = 0.1
minp = d[0.1]
for key, value in d.items():
print('Running loop')
print(f'Comparing {abs(value - goal)} < {abs(minp - goal)}')
if abs(value - goal) < abs(minp - goal):
print(f'{abs(value - goal)} < {abs(minp - goal)}')
minf = key
minp = value
print((minf, minp))
results.append(minf)
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]}
return results

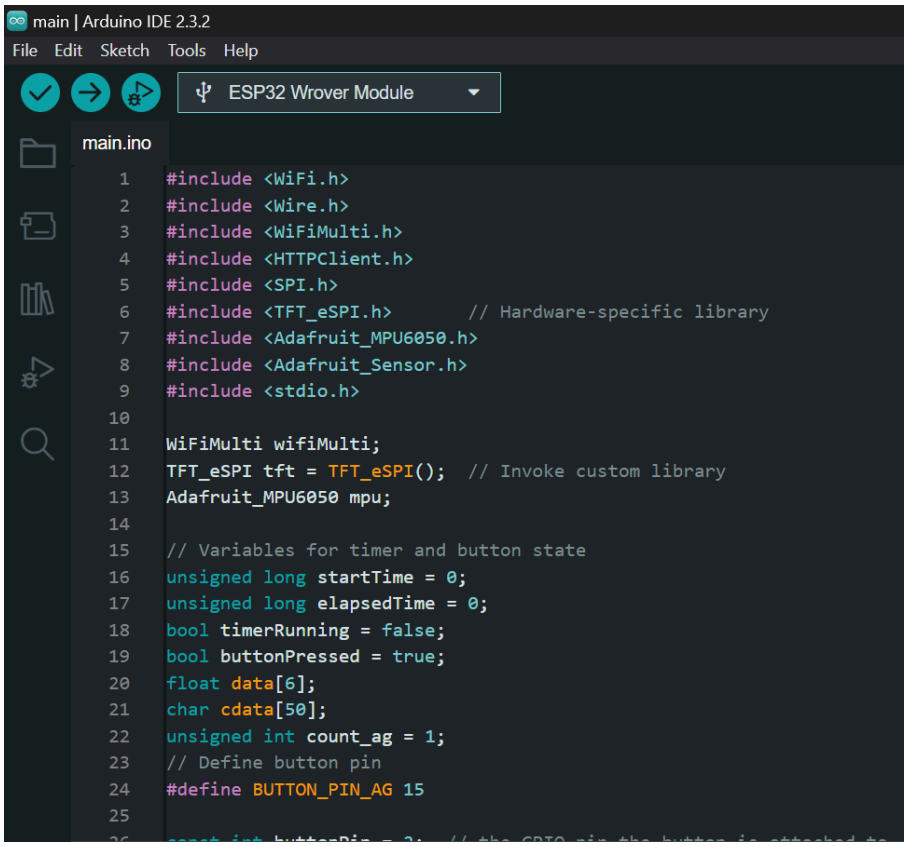

File Edit Selection View Go Run ···
Search
firebase_app.py 2
freqs_app.py
weights_app.py
C: > Users > anush > Projects > iFit_Explore_Studio > weights_app.py > ...
def findWeights(df, freqs, goal=10):
# freq = freqs.get(col)
# for key in freqs.keys():
# print(f'comparing {key} and {col}')
# if key[0] == col[0]:
# freq = df[key]
# print('checking for freq ', freq)
filtered = lp.low_pass_filter(df, col, 2, freqs[col], 10)
op = argrelextrema(filtered[col + '_lowpass'].values, np.greater)
peaks = filtered.iloc[op]
preds.append(len(peaks))
print(preds)
weights = []
for i in preds:
if i == goal:
weights.append(1)
else:
weights.append(round(1/(abs(i - goal)+1), 2))
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]}
return weights


File Edit Selection View Go Run
Search
firebase_app.py 2
freqs_app.py
weights_app.py
infer.py
plan_func.py
C: > Users > anush > Projects > iFit_Explore_Studio > infer.py > ...
def findReps(df, freqs, weights):
if weights['x'] == weights['y'] == 1:
if reps[0] == reps[1]:
result = reps[0]
else:
if abs(reps[2] - reps[0]) < abs(reps[2] - reps[1]):
result = reps[0]
else:
result = reps[1]
elif weights['x'] == weights['z'] == 1:
if reps[0] == reps[2]:
result = reps[0]
else:
if abs(reps[1] - reps[0]) < abs(reps[1] - reps[2]):
result = reps[0]
else:
result = reps[2]
elif weights['y'] == weights['z'] == 1:
if reps[1] == reps[2]:
result = reps[1]
else:
if abs(reps[0] - reps[1]) < abs(reps[0] - reps[2]):
result = reps[1]
else:
result = reps[2]
elif weights['x'] == 1:
if abs(reps[0]-reps[1]) > 1 and abs(reps[0]-reps[2]) > 1:
result = reps[0] + 1
elif abs(reps[0]-reps[1]) < 1 and abs(reps[0]-reps[2]) < 1:
0 2
0
Live Share
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.10.9 64-bit
Go Live

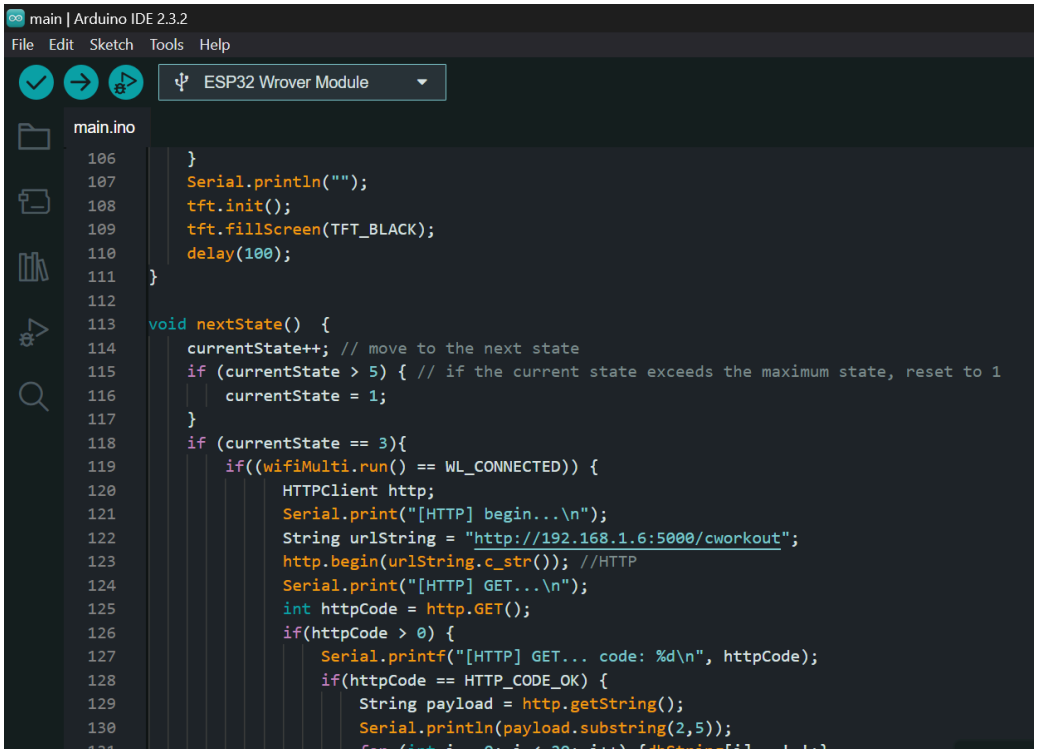

File Edit Selection View Go Run
Search
firebase_app.py 2
freqs_app.py
weights_app.py
infer.py
plan_func.py
C: > Users > anush > Projects > iFit_Explore_Studio > plan_func.py > ...
def makePlan(formResponse, exerciseOptions):
deleteHard = False
if formResponse['difficulty'] == 'easy':
deleteHard = True
new_options = []
# Filter options
for index in range(len(exerciseOptions)):
ex = exerciseOptions[index]
exerciseOptions[index]['time'] = int(exerciseOptions[index]['time'])
if deleteHard:
if ex['difficulty'] == 'hard':
continue
for target in formResponse['target_areas']:
if target in ex['target_areas']:
new_options.append(exerciseOptions[index])
break
final_options = new_options.copy()
# To handle tiny workout times with large dataset
rep_bool = 0
while totalTime(final_options) > formResponse['time']:
index = randint(0, len(final_options)-1)
if type(final_options[index]['reps']) == bool:
if rep_bool:
final_options.pop(index)
rep_bool = not rep_bool
Live Share
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.10.9 64-bit
Go Live


File Edit Selection View Go Run
Search
firebase_app.py 2
freqs_app.py
weights_app.py
infer.py
plan_func.py
C: > Users > anush > Projects > iFit_Explore_Studio > plan_func.py > ...
def makePlan(formResponse, exerciseOptions):
else:
final_options[index]['time'] = ex['time']/2
# To increase reps/time until target time
while totalTime(final_options) < formResponse['time']:
for index in range(len(final_options)):
ex = final_options[index]
if ex['reps']:
final_options[index]['time'] += ex['time']*5
if type(final_options[index]['reps']) == bool:
final_options[index]['reps'] = 5
else:
final_options[index]['reps'] += 5
else:
final_options[index]['time'] += ex['time']*1.5
return final_options

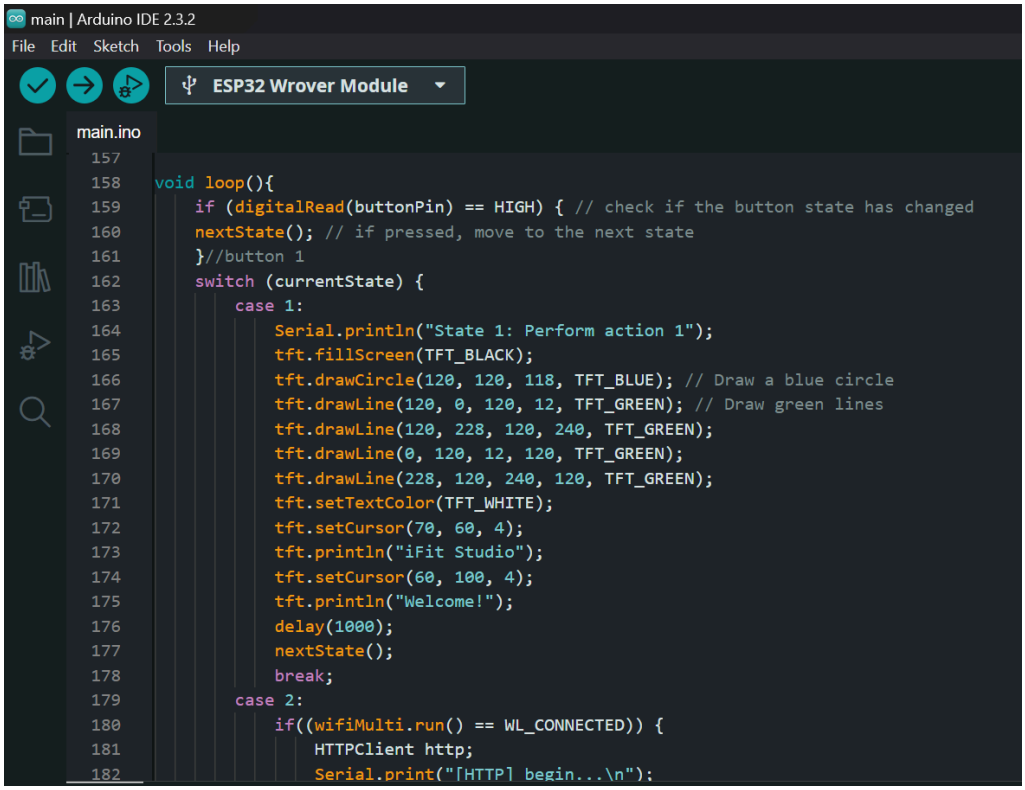
4.3.5 SMARTWATCH

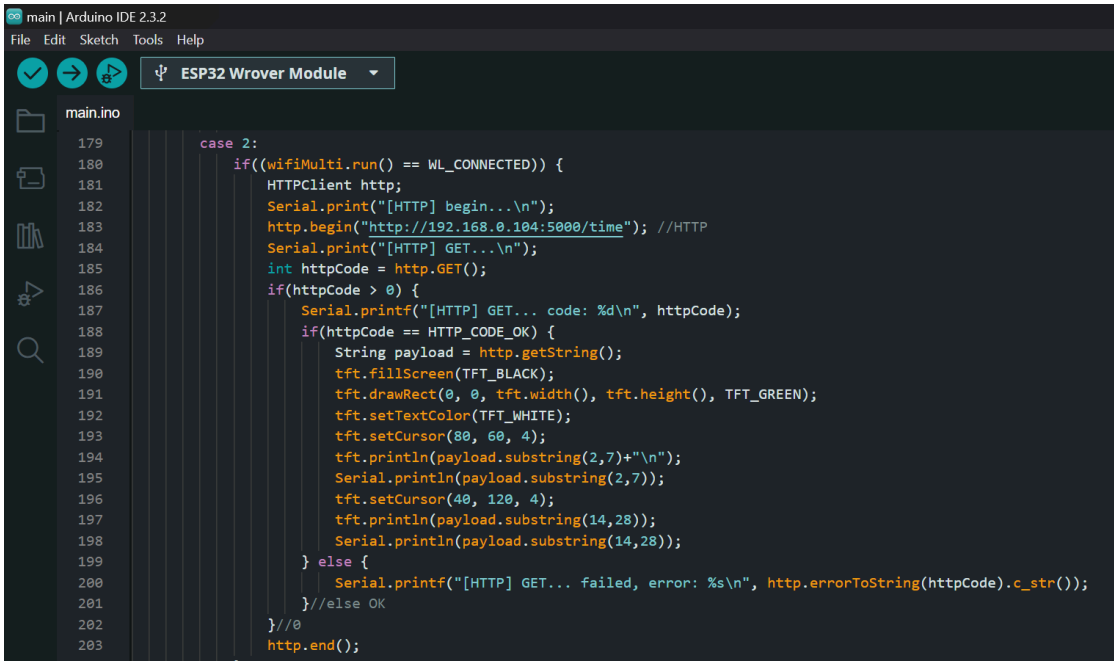


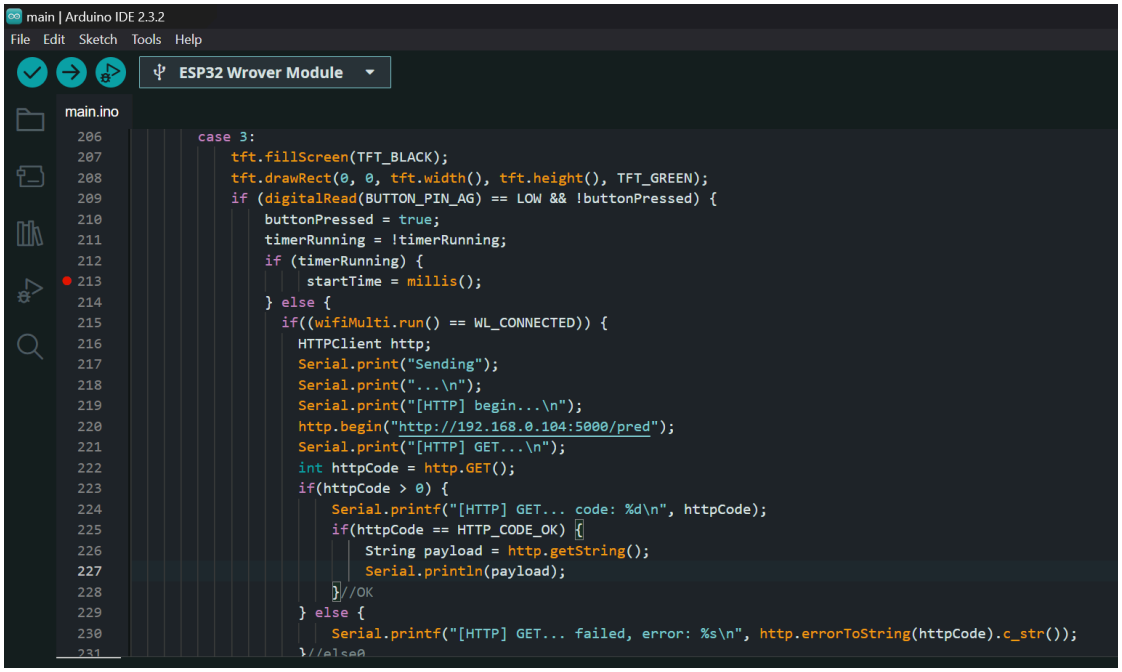

```
#include <WiFi.h>
#include <Wire.h>
#include <WiFiMulti.h>
#include <HTTPClient.h>
#include <SPI.h>
#include <TFT_eSPI.h>       // Hardware-specific library
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <stdio.h>

WiFiMulti wifiMulti;
TFT_eSPI tft = TFT_eSPI();  // Invoke custom library
Adafruit_MPU6050 mpu;

// Variables for timer and button state
unsigned long startTime = 0;
unsigned long elapsedTime = 0;
bool timerRunning = false;
bool buttonPressed = true;
float data[6];
char cdata[50];
unsigned int count_ag = 1;
// Define button pin
#define BUTTON_PIN_AG 15
```

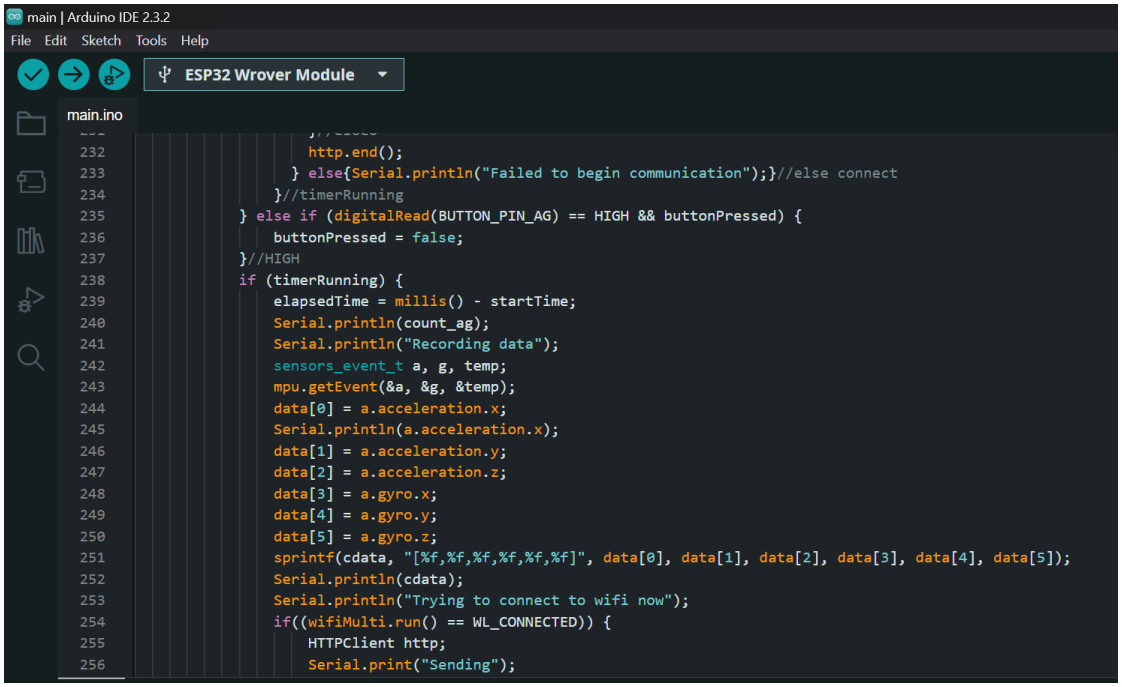


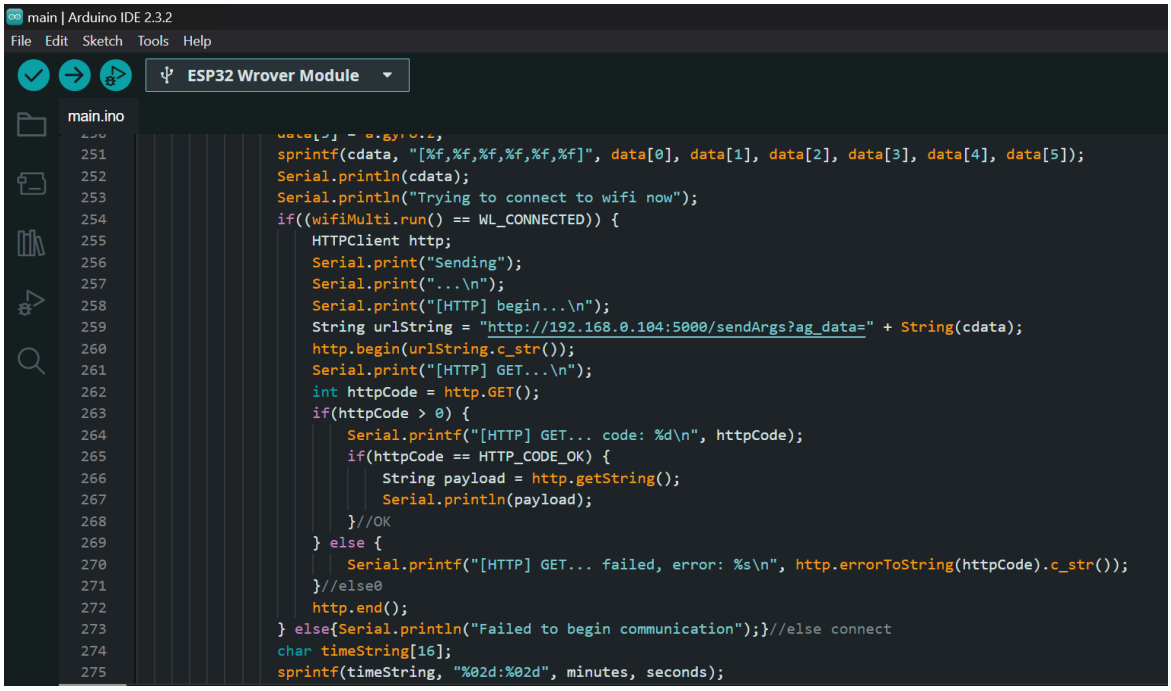

```
    }
    Serial.println("");
    tft.init();
    tft.fillScreen(TFT_BLACK);
    delay(100);
}

void nextState()  {
    currentState++; // move to the next state
    if (currentState > 5) { // if the current state exceeds the maximum state, reset to 1
        currentState = 1;
    }
    if (currentState == 3){
        if((wifiMulti.run() == WL_CONNECTED)) {
            HTTPClient http;
            Serial.print("[HTTP] begin...\n");
            String urlString = "http://192.168.1.6:5000/cworkout";
            http.begin(urlString.c_str()); //HTTP
            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.substring(2,5));
```

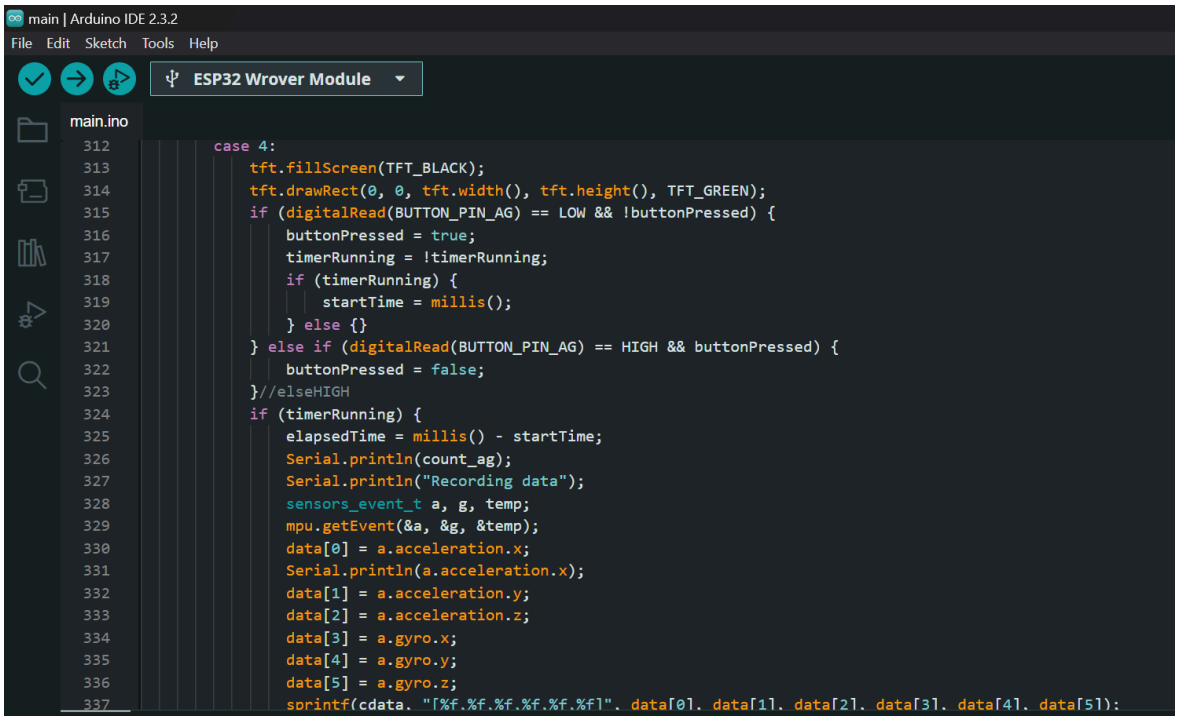

main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
main.ino
157
158 void loop(){
159 if (digitalRead(buttonPin) == HIGH) { // check if the button state has changed
160 nextState(); // if pressed, move to the next state
161 }//button 1
162 switch (currentState) {
163 case 1:
164 Serial.println("State 1: Perform action 1");
165 tft.fillScreen(TFT_BLACK);
166 tft.drawCircle(120, 120, 118, TFT_BLUE); // Draw a blue circle
167 tft.drawLine(120, 0, 120, 12, TFT_GREEN); // Draw green lines
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");

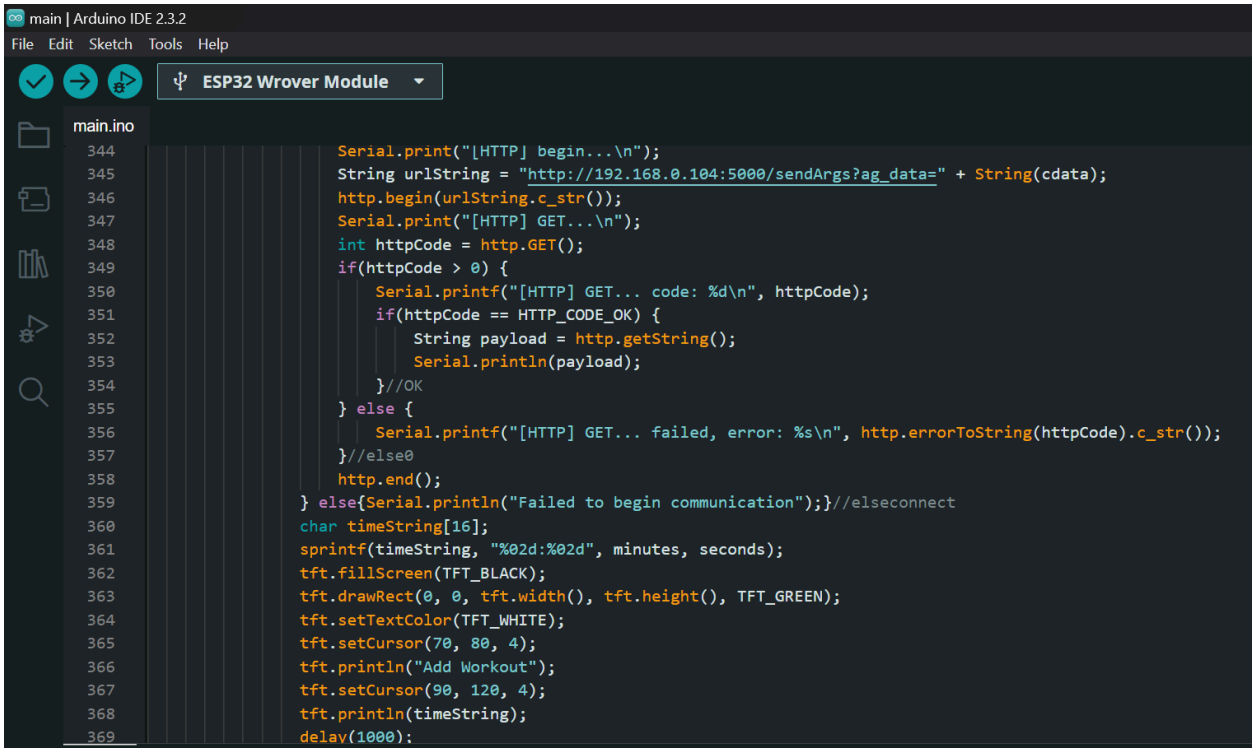

main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
main.ino
179 case 2:
180 if((wifiMulti.run() == WL_CONNECTED)) {
181 HTTPClient http;
182 Serial.print("[HTTP] begin...\n");
183 http.begin("http://192.168.0.104:5000/time"); //HTTP
184 Serial.print("[HTTP] GET...\n");
185 int httpCode = 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.fillScreen(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();


main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
main.ino
206 case 3:
207 tft.fillScreen(TFT_BLACK);
208 tft.drawRect(0, 0, tft.width(), tft.height(), TFT_GREEN);
209 if (digitalRead(BUTTON_PIN_AG) == LOW && !buttonPressed) {
210 buttonPressed = true;
211 timerRunning = !timerRunning;
212 if (timerRunning) {
213 startTime = millis();
214 } else {
215 if((wifiMulti.run() == WL_CONNECTED)) {
216 HTTPClient http;
217 Serial.print("Sending");
218 Serial.print("...\n");
219 Serial.print("[HTTP] begin...\n");
220 http.begin("http://192.168.0.104:5000/pred");
221 Serial.print("[HTTP] GET...\n");
222 int httpCode = http.GET();
223 if(httpCode > 0) {
224 Serial.printf("[HTTP] GET... code: %d\n", httpCode);
225 if(httpCode == HTTP_CODE_OK) {
226 String payload = http.getString();
227 Serial.println(payload);
228 }//OK
229 } else {
230 Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());


main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
main.ino
232 http.end();
233 } else{Serial.println("Failed to begin communication");}//else connect
234 }//timerRunning
235 } else if (digitalRead(BUTTON_PIN_AG) == HIGH && buttonPressed) {
236 buttonPressed = false;
237 }//HIGH
238 if (timerRunning) {
239 elapsedTime = millis() - startTime;
240 Serial.println(count_ag);
241 Serial.println("Recording data");
242 sensors_event_t a, g, temp;
243 mpu.getEvent(&a, &g, &temp);
244 data[0] = a.acceleration.x;
245 Serial.println(a.acceleration.x);
246 data[1] = a.acceleration.y;
247 data[2] = a.acceleration.z;
248 data[3] = a.gyro.x;
249 data[4] = a.gyro.y;
250 data[5] = a.gyro.z;
251 sprintf(cdata, "[%f,%f,%f,%f,%f,%f]", data[0], data[1], data[2], data[3], data[4], data[5]);
252 Serial.println(cdata);
253 Serial.println("Trying to connect to wifi now");
254 if((wifiMulti.run() == WL_CONNECTED)) {
255 HTTPClient http;
256 Serial.print("Sending");


main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
main.ino
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");
Serial.print("...\n");
Serial.print("[HTTP] begin...\n");
String urlString = "http://192.168.0.104:5000/sendArgs?ag_data=" + String(cdata);
http.begin(urlString.c_str());
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());
}//else0
http.end();
} else{Serial.println("Failed to begin communication");}//else connect
char timeString[16];
sprintf(timeString, "%02d:%02d", minutes, seconds);


main | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
main.ino
case 4:
tft.fillScreen(TFT_BLACK);
tft.drawRect(0, 0, tft.width(), tft.height(), TFT_GREEN);
if (digitalRead(BUTTON_PIN_AG) == LOW && !buttonPressed) {
buttonPressed = true;
timerRunning = !timerRunning;
if (timerRunning) {
startTime = millis();
} else {}
} else if (digitalRead(BUTTON_PIN_AG) == 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;



Code Summary:

The Arduino code for the prototype of the iFit Studio project establishes a connection with a server to fetch data and display it on an LCD screen. The code uses a state machine approach, where different states correspond to display pages and functionalities.

The variable declarations section declares various variables for timer management, button state, data storage, connection status, etc. These variables are used throughout the code for various purposes.

The code handles HTTP requests to a specified URL to fetch data from the server. Depending on the response status (HTTP_CODE_OK), the data is extracted and parsed to update variables.

The display pages are managed using a case statement. Each case corresponds to a specific page on the LCD screen. The welcome page 1 automatically changes to the

next state, and page 2, the time and date page, requests time from the server and displays it. However, for the remaining pages, the user can use the button to switch the display page to the next.

When the button is pressed, the workout page displays workout information fetched from the server. The user can then begin the workout by pressing the timer button to start the timer. The accelerometer and gyroscope sensor then begin to collect the data. This data is sent to the server in real-time while the user performs exercises. Once the workout is completed and the button is pressed, the data is processed and updated on the progress summary page in the mobile application.

The "Add workout" page allows users to add custom workout sessions. Similar to the workout page, accelerometer data is collected during the workout and processed for future predictions.

In summary, the code connects with the server to fetch and display data, manages display pages using a state machine approach, and collects accelerometer data from the MPU6050 sensor during workouts. This data is sent to the server over Wi-Fi for processing and updating the progress summary page in the mobile application.

The server manages the tracking, plan generation, and other tasks. It recognises patterns by applying a low-pass filter at various frequencies. At different frequencies between 0 and 1, different levels of variations in data are passed or removed, smoothening the pattern. After that, the ‘peaks’ or high points in the data are found, the number of which are the repetitions we need. The accelerometer, as does the gyroscope, gives three values for the three axes. So, each of these six values has to be evaluated, and six frequencies are the results. But in every exercise, specific axes capture this pattern better than others. Taking the concept of changing weights & parameters & observing changes in output from AI, we assign weights to these columns depending on how close they are to the actual count. So, these frequencies and weights

are the basis of finding repetition. Every time we receive data for that exercise from then on, it is passed with these values to predict the count.

For the plan generation, three questions are asked - target areas, time, and difficulty level. The first step is to filter challenging exercises for those who choose easy. Then, we iterate through all target areas of every exercise and add it to the options if anything matches. Finally, repetitions are incremented by five each round to match the time until the time is close enough.

5. TESTING & EVALUATION

5.1 TESTING APPROACHES

1. Unit testing - The hardware was first tested with example codes from their libraries to ensure functionality. Each subsequent step was added and uploaded to the device individually to ensure proper working. Otherwise, alternative approaches were used before proceeding to the next step. The application was first developed with non-functional buttons to finalise the design. Each button was made functional one by one. The server first printed the received data before editing the Firebase data.

2. Integration, System and Performance Testing - The server is dedicated to connecting the other two components together. So certain functions are triggered from the app, reach the server, reach Firestore, access data from the watch, and go back and display them. So, these functions were thoroughly tested with different parameters to confirm proper communication & functioning.

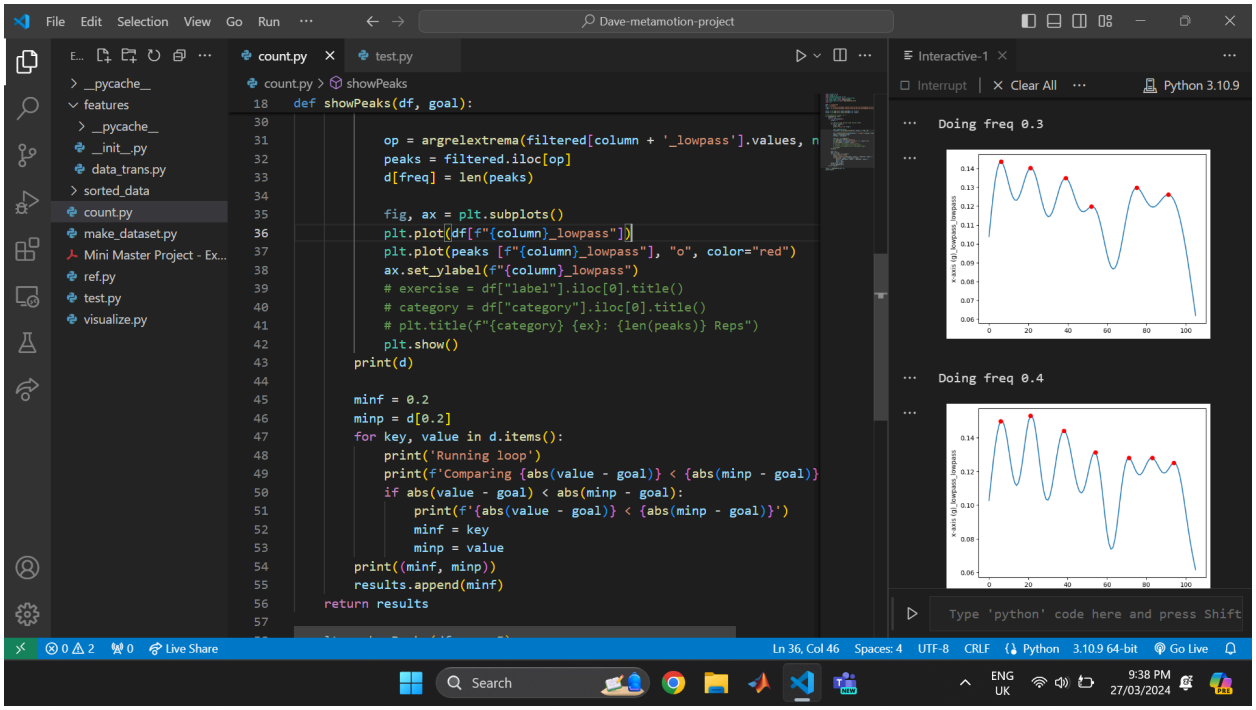
3. Acceptance Testing - Users wore the hardware, used the application, and gave feedback on the positive & negative aspects. While the provided features were

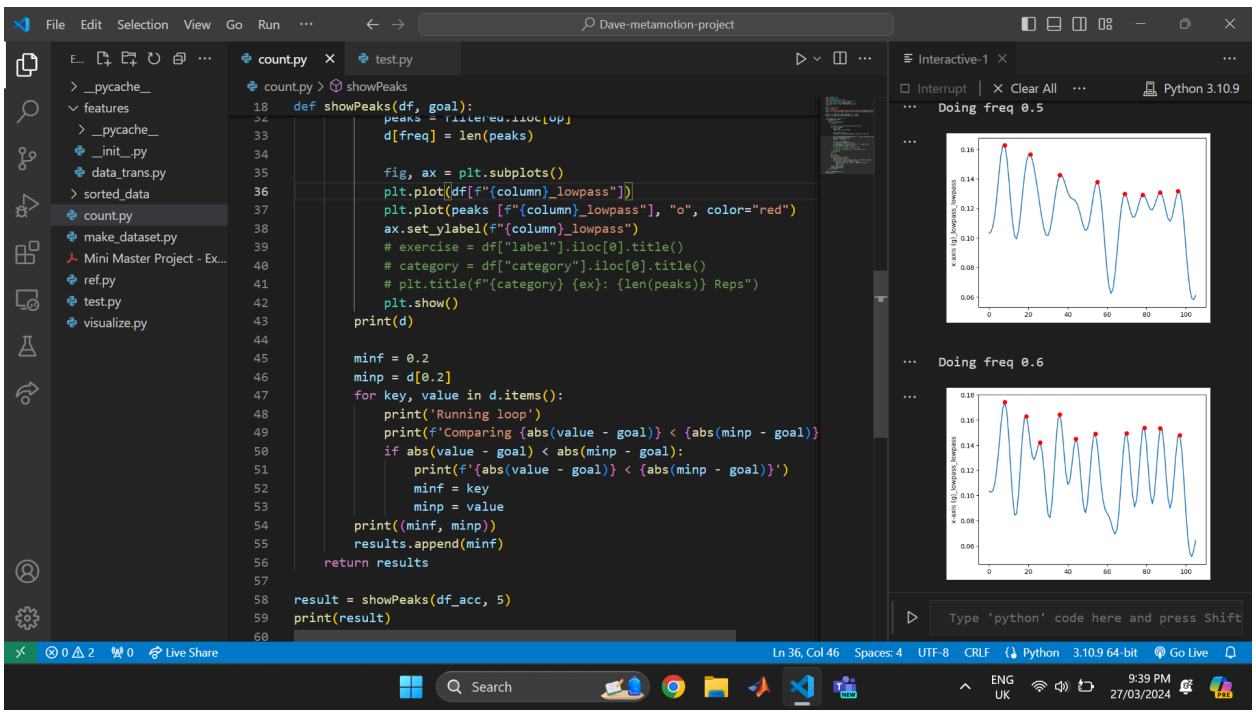
liked, improvements in the Design were suggested, which have been added to the future prospects.

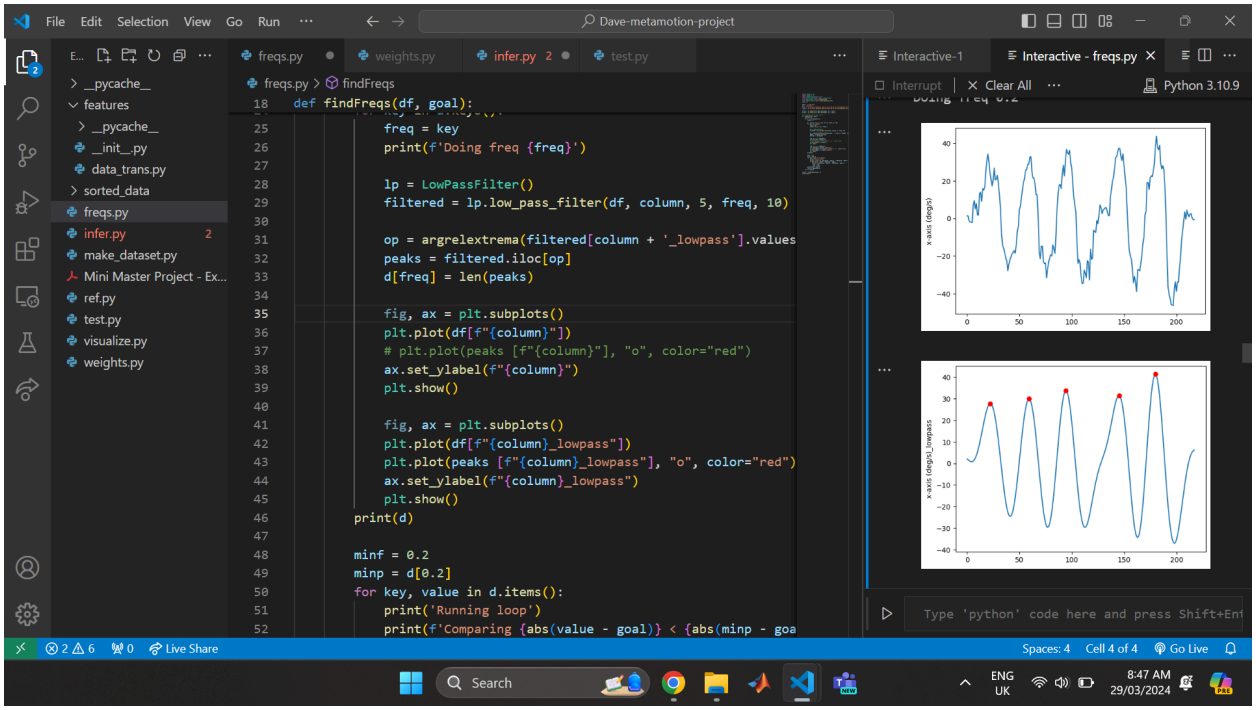
4. Edge cases - Values not expected as inputs were sent, to test whether the system will crash with errors, respond with appropriate messages or something else. Errors were handled when noticed.

5.2 ANALYSIS OF RESULTS

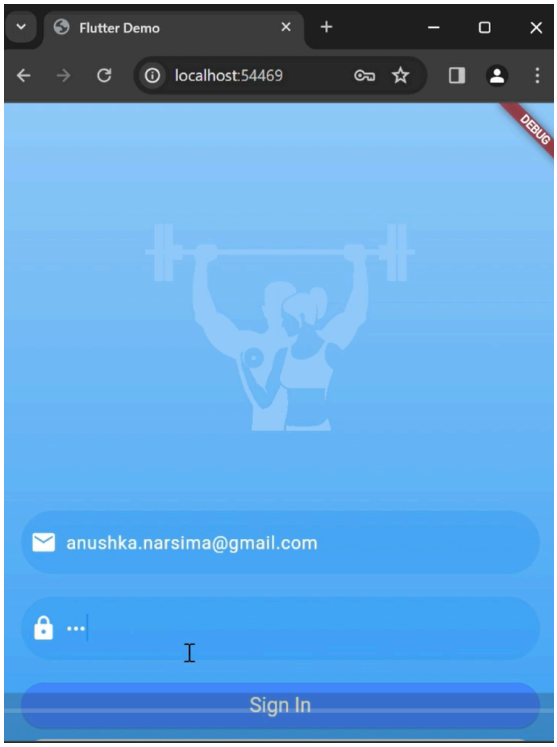
The graphs shown in the images below all represent the same data set but at different frequencies. The red dots at the top mark the peaks, and the number of them are the repetitions. The frequency is printed above the graph. It is clearly visible that a specific frequency can be decided to be the best for getting a certain number as a result.

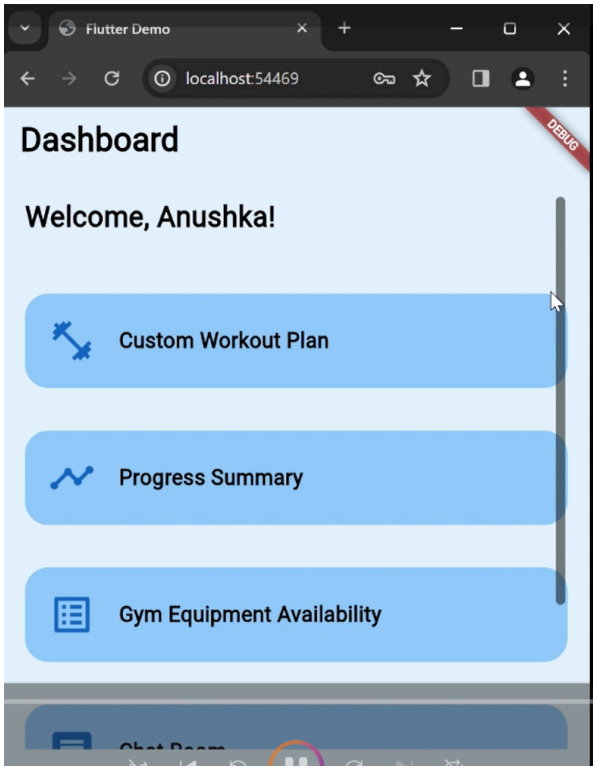

File Edit Selection View Go Run
Dave-metamotion-project
count.py
test.py
count.py > showPeaks
def showPeaks(df, goal):
op = argrelextrema(filtered[column + '_lowpass'].values, n
peaks = filtered.iloc[op]
d[freq] = len(peaks)
fig, ax = plt.subplots()
plt.plot(df[f"{column}_lowpass"])
plt.plot(peaks [f"{column}_lowpass"], "o", color="red")
ax.set_ylabel(f"{column}_lowpass")
# exercise = df["label"].iloc[0].title()
# category = df["category"].iloc[0].title()
# plt.title(f"{category} {ex}: {len(peaks)} Reps")
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)}
if abs(value - goal) < abs(minp - goal):
print(f'{abs(value - goal)} < {abs(minp - goal)}')
minf = key
minp = value
print((minf, minp))
results.append(minf)
return results
Interactive-1
Interrupt
Clear All
Python 3.10.9
Doing freq 0.3
Doing freq 0.4
Type 'python' code here and press Shift
Ln 36, Col 46 Spaces: 4 UTF-8 CRLF Python 3.10.9 64-bit Go Live
Search
ENG UK
9:38 PM 27/03/2024

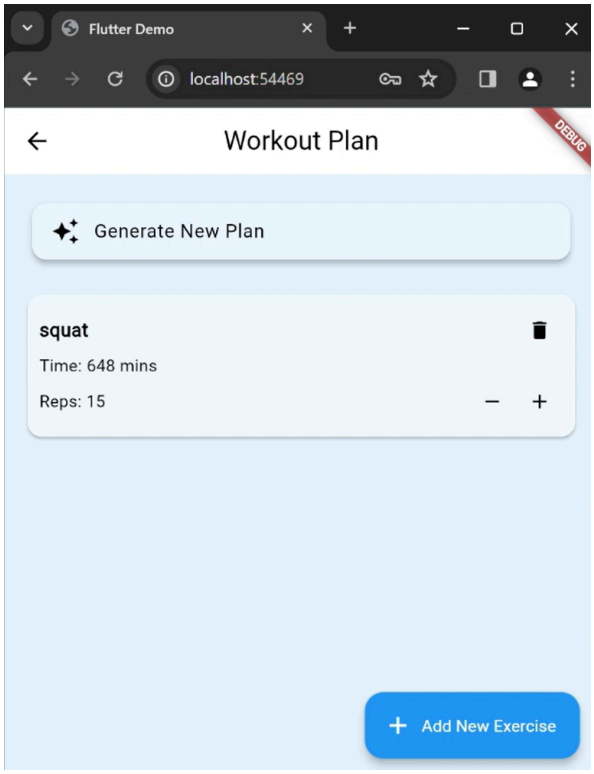

File Edit Selection View Go Run
Dave-metamotion-project
count.py
test.py
count.py > showPeaks
def showPeaks(df, goal):
d[freq] = len(peaks)
fig, ax = plt.subplots()
plt.plot(df[f"{column}_lowpass"])
plt.plot(peaks [f"{column}_lowpass"], "o", color="red")
ax.set_ylabel(f"{column}_lowpass")
# exercise = df["label"].iloc[0].title()
# category = df["category"].iloc[0].title()
# plt.title(f"{category} {ex}: {len(peaks)} Reps")
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)}
if abs(value - goal) < abs(minp - goal):
print(f'{abs(value - goal)} < {abs(minp - goal)}')
minf = key
minp = value
print((minf, minp))
results.append(minf)
return results
result = showPeaks(df_acc, 5)
print(result)
Interactive-1
Interrupt
Clear All
Python 3.10.9
Doing freq 0.5
Doing freq 0.6
Type 'python' code here and press Shift
Ln 36, Col 46 Spaces: 4 UTF-8 CRLF Python 3.10.9 64-bit Go Live
Search
ENG UK
9:39 PM 27/03/2024

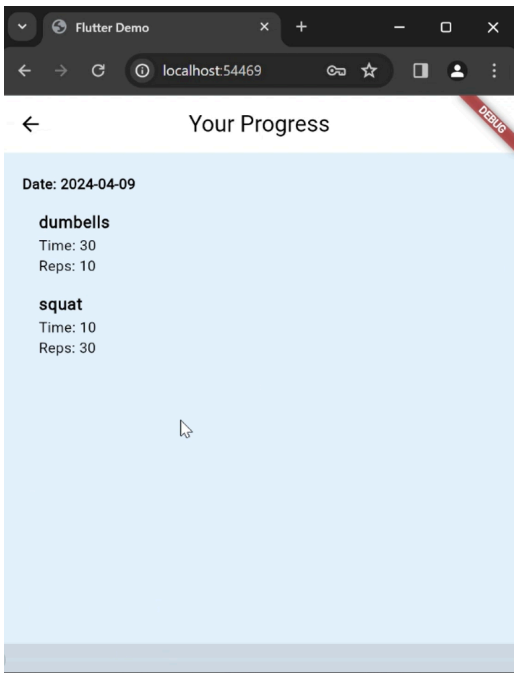


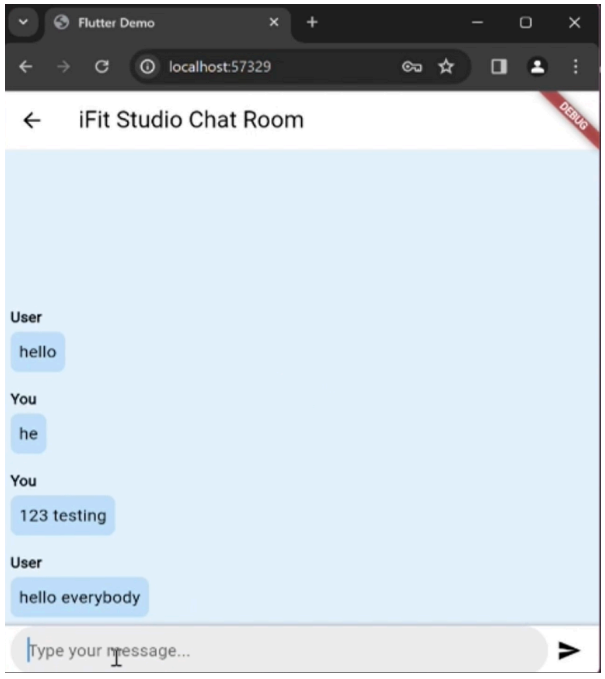
Snapshots of the applicaion are shown below. The full demo video can be found at: https://drive.google.com/file/d/12qZoK2mq690vt8tCjXizODcbNkykJB8g/view?usp=drive_link




Flutter Demo
localhost:54469
Dashboard
DEBUG
Welcome, Anushka!
Custom Workout Plan
Progress Summary
Gym Equipment Availability


Flutter Demo
localhost:54469
Workout Plan
DEBUG
Generate New Plan
squat
Time: 648 mins
Reps: 15
Add New Exercise


Flutter Demo
localhost:54469
Your Progress
DEBUG
Date: 2024-04-09
dumbells
Time: 30
Reps: 10
squat
Time: 10
Reps: 30


Flutter Demo
localhost:57329
iFit Studio Chat Room
DEBUG
User
hello
You
he
You
123 testing
User
hello everybody
Type your message...

6. CONCLUSION

6.1 SUMMARY OF WORK DONE

The iFit project has successfully developed and implemented a comprehensive IoT-based fitness management system. This system integrates a smartwatch and a companion mobile application, providing users with real-time exercise tracking, personalized workout guidance, and community engagement features. Through the collaboration of hardware and software components, the iFit project aims to enhance the overall fitness experience for users, promoting motivation and progress towards fitness goals.

6.2 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, it will contribute to its long-term success by exploring partnerships and expanding the system's capabilities to support diverse workout routines and fitness goals.

Five main prospects we would like to highlight are:

1. Advancements in Hardware:

The field of fitness technology is constantly evolving, and one area with significant potential for growth is the improvement of hardware components. As technology advances, we expect to see the development of more sophisticated sensors, smaller and more powerful microcontrollers, and longer-lasting batteries. These advancements will create even more accurate and versatile fitness devices, providing users with better insights into their health and performance.

2. Influencer Workout Led Content:

There is a growing popularity of fitness influencers and online fitness communities. There is a significant opportunity to integrate influencer workout plans and ideas into fitness management systems. By partnering with renowned fitness experts and influencers, the system could offer users access to curated workout routines, training tips, and motivational content tailored to their goals and interests.

This integration could take various forms, such as providing direct access to influencer-led workouts within the mobile app, allowing users to follow along with their favourite trainers in real time. Additionally, the system could incorporate social features that enable users to share their progress, participate in challenges, and engage with their peers and influencers for support and motivation.

Overall, if we embrace these advancements in hardware technology and integrate influencer-led content, the future of fitness management systems holds immense potential to enhance the user experience further, drive motivation, and promote overall well-being.

3. UI/UX

While the design was visualised at the beginning of the project, the functional back-end tasks were prioritised and achieved first. Thus, a few minor points in the design should have been included to enhance the overall experience. For example, the sign-in page has constraints to accept only properly formatted emails and passwords but doesn't state the problem on the page for the user to be aware of and fix. The progress page lists progress by date & exercise, which will become hard to read when used for long. Visual representations like pie charts on daily, weekly & monthly durations would help summarise the list.

4. MedPALM Integration

Exercises are done for health improvement, but the same can sometimes be dangerous depending on age and health conditions. Google's MedPALM is a large language model

that is specifically trained to answer medical questions accurately, so integrating it into the app would help protect users from engaging in activities that aren't suitable for their conditions.

5. Compact PCB

PCBs in our iFit watch can be used to create compact and efficient circuits that integrate various electronic components essential for its functionality. The PCB design in our iFit watch will accommodate the watch's compact form factor while maintaining high reliability and durability, as these devices are worn on the wrist and subjected to daily wear and tear. Advanced PCB technologies, such as high-density interconnects and flexible circuits, are commonly used to meet such demands. This allows our iFit watch to offer features like touchscreens, GPS and heart rate monitoring in a sleek and portable package.

7. 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/wpcontent/uploads/2020/05/IJSRET_V6_issue3_493.pdf

4. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013). "Internet of things (IoT): A Aision, 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: Mhealth 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/acsmhealthfitness/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 - "IoTbased 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/acsmhealthfitness/fulltext/2017/09000/motivational_dynamics_of_wearable_activity.8.aspx+ more
9. Johnson, T. (2021). "User Consent in IoT Systems." Journal of Data Protection, 15(2), 98-110.
10. Smith, J. (2020). "Data Security in Smart Devices." IEEE Transactions on Information Security, 25(3), 123-130.
11. Williams, R. (2019). "Privacy Regulations and IoT Applications." International Journal of Technology Law, 12(4), 201-212.
12. Brown, A. (2021). "Implementing Secure Authentication with Firebase in Flutter." Mobile Development Review, 18(2), 45-56.
13. Jones, L. (2020). "Building Dynamic Workout Plans with Firestore." Journal of App Development, 22(3), 78-89.

14. Smith, K. (2019). "Creating Real-Time Chat Applications with Flutter and Firestore." Journal of Mobile Computing, 29(1), 102-115.
15. Paulo Alexandre Azevedo Ferreira (2020). "*A context-aware system architecture to assist workout plans*". Available at: https://repositorium.sdum.uminho.pt/bitstream/1822/79703/1/Paulo%20Alexandre%20Azevedo%20Ferreira.pdf

https://www.puregym.com/blog/uk-fitness-report-gym-statistics/#gym-usage (Add to doc)

16. Fernández-Rodríguez, E.F., Merino-Marban, R., Romero-Ramos, O., & López-Fernández, I. (2020). "*A systematic review about the characteristics and patterns of use of outdoor gyms*". Available at: https://www.researchgate.net/publication/344379572_A_systematic_review_about_the_characteristics_and_patterns_of_use_of_outdoor_gyms
17. Coates, W.; Wahlström, J. (2023). "*LEAN: Real-Time Analysis of Resistance Training Using Wearable Computing*". Available at: https://doi.org/10.3390/s23104602
18. Mark Hoogendoorn, Burkhardt Funk(2018). "*Machine Learning for the Quantified Self - On the Art of Learning from Sensory Data*". Available at: https://link.springer.com/book/10.1007/978-3-319-66308-1
19. Morris, D., Saponas, T. S., Guillory, A., & Kelner, I. (2014). "*RecoFit: using a wearable sensor to find, recognize, and count repetitive exercises*". Available at: https://github.com/microsoft/Exercise-Recognition-from-Wearable-Sensors/tree/main
20. Dave Ebbelar (2019). "*Exploring the Possibilities of Context-Aware Applications for Strength Training".* Available at: https://docs.datalumina.io/s1eFpcdtbvRTMb

a.