Personalized workout planner

By

Cornelius Wong Qin Jun

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

June 2025

Personalized workout planner

By

Cornelius Wong Qin Jun

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

June 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ms Chai Meei Tyng who has given me this bright opportunity to engage in a machine learning and mobile application design project. It is my first step to establish a career machine learning and mobile application design field. A million thanks to you.

To a very special person in my life, Mr Chong Chee Wai, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

In modern times, people often neglect exercise due to busy schedules and reliance on conveniences like advanced transportation, leading to decreased physical activity. Additionally, the effectiveness of workouts is a major concern, as not all routines suit everyone. A personalized workout planner is essential to recommend effective exercise plans based on factors like height, weight, gender, level of activity, and specific goals. However, traditional workout plans may not adapt to injury or difficulties encountered by the user, which can lead to further complications or reduced progress. To address these issues, this project proposes the development of a Personalized Workout Planner mobile application that incorporates injury and failure adjustment features. This app aims to provide users with adaptable workout routines, ensuring safety and efficiency even in cases of injury or when the user struggles with certain exercises. The proposal outlines the project's background, objectives, existing systems, and the proposed method and system to offer an effective, customized fitness solution. The final product of the project is the combination of FastAPI backend, Google Cloud Platform, Supabase and an React native mobile application with the ability to recommend workouts, fetch workouts, update profile for recommendation needs, feedback for 4 type of uncertainty cases and link with Google Calendar for free slot schedules.

Area of Study: Machine learning, Mobile app development

Keywords: Recommendation system, cosine similarity, Feedback System, Mobile Application, Google Calendar integration

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

HIIT	High-Intensity Interval Training
BPM rate	Beats per minute rate
BMI	Body mass index
KNN	k-nearest neighbors
HRpeak	Maximum Heart Rate
VO2peak	Volume of oxygen uptake during peak exercise
MICT	Moderate-Intensity Continuous Training
RLS	Row-level security
JWT	JSON Web Token
BaaS	Backend as a Service

CHAPTER 1
Project Background

1.1.1 Introduction

In today's fast-paced world, physical inactivity has become a widespread issue. Modern conveniences, such as advanced transportation, have led to a decline in daily physical activity, as people often rely on vehicles instead of walking. Additionally, the demands of a busy lifestyle make it challenging for individuals to find time for regular exercise.

Effective workouts require careful consideration of various factors, including weight, height, activity level, age, and gender [1]. According to [2], research has shown that most beginners struggle to maintain their workout routines due to interdependent psychological processes, such as the human tendency to follow the law of least effort. The additional effort required to manually plan and adhere to a workout routine can further discourage individuals from pursuing a healthier lifestyle.

To address these challenges, personalized diet and fitness planners have emerged as convenient and accessible tools that help individuals create and follow workout plans suited to their unique circumstances and even motivate them. Recent research has explored the use of machine learning techniques, such as Random Forest and Support Vector Machine to develop personalized workout recommendation systems [1].

In this proposal, I propose the development of a personalized workout planner mobile application that leverages these advanced technologies to provide tailored exercise recommendations.

1.1.2 Information Background

Initially, wearable fitness trackers presented a universal metric-focused goal, like burning 500 calories and walking 10000 steps, which did not yield effective long-term results for users. These metrics may not align with individual user goals, causing motivation to diminish after a few months, user just acts it as numbers. To enhance this experience, Human-Computer Interaction (HCI) is introduced to have better to support workout experience that suit to user. Jasmine Niess suggests that hedonic and eudaimonic well-being serve as fundamental concepts for developing adaptive fitness goals for users.   Hedonic wellbeing emphasizes positive emotions, often associated with pleasure and enjoyment. In contrast, eudaimonic wellbeing centers on self-fulfillment and pursuing a meaningful life. These can ensure the user stays motivated and improves adherence to the fitness app to stay healthy and achieve their goals with HCI based personalized workout planner.[3]

High-Intensity Interval Training

High-Intensity Interval Training (HIIT) consists of cardiovascular exercises characterized by short, intense aerobic intervals followed by rest or low-intensity periods to optimize the body's oxygen consumption. High-intensity exercises should reach 85-95% of the peak heart rate (HRpeak), without exceeding the body's peak oxygen uptake (VO2peak), which is approximately 80-90% for short durations like 4 minutes. The recovery or low-intensity phase allows users to recover, enabling them to repeat the high-intensity intervals later. This recovery phase should maintain a heart rate of 60-70% HRpeak, such as through walking, for a duration of 2-3 minutes. HIIT depends on individual HRpeak, so it can vary among people and done in several interval sets. HIIT can improve cardiovascular disease risk factors such as blood pressure, weight and fat regulation. Since it short duration properties, it is more time-efficient way to get health benefits. [4]

1.2 Problem Statement

According to Patrick G. Kidman, 1 in 3 adults suffers from chronic diseases, leading to a rise in health-related app usage for lifestyle and physical activity management. However, the abandonment rate of these apps is notably high, with 66% of health apps and 69% of fitness apps being discarded within 90 days. [5] There are two main stages of abandonment: the early stage and the later stage. During the early stage, users leave apps due to generic workout plans that fail to meet their goals, overwhelming notifications, and a lack of personalization. In the later stage, users stop using the app because they feel unmotivated and don't see any results. [6] Additionally, changes such as a job switch, a new living environment, or injury can disrupt routines, making app usage impractical. Shifting goals can also lead users to quit if the app doesn't provide solutions for their new objectives.

Today, many workout planner apps center around daily routines aimed at specific fitness goals, which is quite standard. However, life can be unpredictable, and users might face injuries, like those to the arm or leg. In such circumstances, a personalized planner that ignores these uncertainties and persists with pre-injury guidelines can exacerbate the injury and lead to adverse effects. Research indicates that factors such as excessive loading, insufficient recovery, and lack of preparedness can increase the risk of injury by subjecting individuals to unsustainable levels of strain [7]. The musculoskeletal system, when exposed to excessive strain, can experience different types of overuse injuries that may impact bones, muscles, tendons, and ligaments. This highlights the importance of adapting workout plans to avoid exacerbating injuries and to ensure proper recovery.

1.3 Motivation

This project aim to propose a mobile application for a personalized workout planner that addresses the uncertainties of life, such as the challenge of dealing with injuries. The system will provide workout recommendations tailored to the user's injury condition, ensuring that exercises do not worsen the injury. This feature will not suggest a workout that worsens the user’s injury and will replan user's workout planner.

Additionally, project includes the failure adjustment feature improves flexibility, allowing users to adjust their workout plans if the exercises are too difficult or cause discomfort. This adaptability ensures that the workout plan remains beneficial while aligned with the user's current physical condition. Besides, adjusting workout times using a calendar API to sync with the user's schedule enhances the user experience by ensuring workouts fit seamlessly into their routine to prevent disruptions due to a busy lifestyle.

1.4 Project Scope

The scope of this project included the development of a personalized workout planner mobile application designed to accommodate user-specific needs, particularly in managing injuries and adapting workout plans based on user feedback. The application uses AI-driven recommendations to generate a personalized workout plan tailored to individual data and user input, including weight, height, activity level, and fitness goals. Besides standard fitness tracking, the app will offer unique functionalities, such as injury-aware fitness planning, where users can report their injuries and receive modified exercise recommendations that avoid exacerbating their condition. Another key feature is the failure adjustment feature, which allows the workout plan to be adjusted if users find the exercises too difficult, ensuring that the plan remains effective and let users still able to continue following. The Calendar feature will sync users' calender to plan the users' routine to schedule the workout that suits the user.

The key features include smart workout recommendations, personalized feedback-based adjustments, alternative workout suggestions, streak tracking for motivation, and an AI-powered assistant for guiding users. The apps also track user progress with streak recorder, encourage notifications, and provide workout performance insight to motivate users and prevent abandonment. The project results a fully functional mobile application that engages user's workout experience by providing adaptive personalization and motivation tracking and injury awareness fitness planning.

1.5 Project Objectives

The objective of the project is to develop an AI-driven personalized workout planner mobile app based on user data to tackle the key causes of user abandonment problem.

Objective 1: Reduce early-stage and later-stage abandonment

To address the high abandonment rate of fitness app in the early and later stages, personalized workout plan recommendations will be developed to achieve different user goals, such as fat loss, muscle gain, flexibility, and endurance. This can help reduce user dissatisfaction caused by the generic plan that fell short of their expectations.

Objective 2: Handle user's life uncertainly

To address the issue of sudden changes of the user's routine, the project includes 2 major categories.

First is the project aim to handle the injury uncertainty in user life by implementing injury awareness workout recommendations. The injury-aware workout recommendation adjusts the plan to prevent worsening injuries based on user feedback to the system. This helps the user work out safely without risking injury and stay consistent when the injury isn't too serious.

Second is the project's aim to handle time uncertainty, like busy weeks or timetable reschedules due to emergencies or switching jobs. The app will implements Calender sync workout adjustment to sync their workout routines with their daily schedule. This provides flexibility by rescheduling the workout plan or shifting to high-intensity interval training(HIIT) plan recommendation for those who are busy instead of skipping.

Objective 3: Maintain motivation and promote consistency

To maintain the long-term user motivation and prevent later-stage abandonment, the app will implement streak tracking, encourage notifications, and interactive feedback options that allow users to adjust their workout plans based on difficulty level that causes failure to be consistence. This is capable for workout plan that is too challenging or painful, ensuring continued adherence and effectiveness. These features can solve the common issues like annoying notifications, lack of motivation, and workout difficulty mismatches that lead them to suffer and quit. To tackle the problem that user lose motivation when not visually see the result, track progress and useful insight metrics are implemented to enable the user to view their progress through metrics like weight and performance status on reports generated and referring suggestions for improvement. This visual insight serves to provide tangible help and help users stay motivated to ensure they are on the right track.

While this project focuses on personalized workout planning, it does not provide medical diagnoses, wearable device, dietary planning, or real-time exercise correction. The scope is narrowed to encounter routine uncertainties and user feedback adjustment workout recommender. The project aims to fill a gap in the current market of workout planners, offering a more personalized to suit the user's routine, a safer fitness experience for injury cases, and feedback-adjusted workout planning.

1.6 Impact, significance and contribution

The impact of the project is to contribute a healthier, safer, and more consistent fitness culture. Many user abandoned fitness apps within 90 days due to lack of personalization and routine change. [5] This project directly addresses these issues by user feedback workout recommendations for injury, busy, and failure cases. By keeping users engaged and providing solutions for uncertain cases, this app ensures more people maintain an active lifestyle without getting frustrated on impractical workout plan.

This project is significant as it bridges the gap between static fitness plans and real-world user challenges. Unlike existing apps, it provides personalization and feedback adjustment on uncertainty, preventing users from quitting due to workout difficulty. pain and schedule conflict. Calendar sync, rescheduling options, and alternative workout recommendations, which is HIIT for busy people to support their fitness experience. In the long run, the app has the potential to reduce fitness dropouts, reduce low adherence exercise problem, thus improving public health. By implementing these features, the app ensures flexibility, motivation, and consistency in workout habits to safeguard a sustainable and enjoyable workout experience.

CHAPTER 2

Literature Reviews

2.1 Reviewed Application and Previous Works on Workout Planner

Four Diet and Workout Planner mobile applications reviewed are HealthifyMe, Fitbod, MyFitnessPal, and Lose Weight in 30 Days and Injurymap [8] [9] [10] [11] [12]. The strength/sale points and limitations, application features, and machine learning on personalized diet and workout recommender will be discussed.

2.1.1 Features in four reviewed Workout Planner mobile applications

Table 2.1.1.1 presents a comparison of five diet and workout planner applications: HealthyFity, Lose Weight in 30 Days, Fitbod, MyFitnessPal and Injurymap. Each application is evaluated based on features that are critical to users aiming to manage their diet and exercise routines effectively. The common features that can be categorized into 10 which are Prompts for Essential Details, Target Settings, Nutrition Tracking, Custom Diet Planning, Custom Workout Planning, Integration with Other Health Applications, Social Element, Progress Tracking, Report and Insight, and Coach Appointment.

1) Prompts for Essential Details:

All applications prompt users to enter essential details such as gender, weight, height, active level, and age. This is crucial for personalized recommendations so that can help the user according to their situation. In Healthifyme, myfitnesspal and Lose weight in 30-day, users are also asked about their dietary preferences (e.g., vegetarian or non-vegetarian) and any food allergies, which allows the app to recommend a diet that suits their dietary restrictions. The proposed app consists this feature.

2) Target Settings:

Setting goals is a feature present in all applications. In Healthifyme and Lose weight in 30-day, there is target weight validation feature for the validate the entered target weight fit the BMI normal and healthy range by calculate using weight and height that entered. For the Fitbod , and Lose weight in 30-day and myfitnesspal, more specific goal for specific muscle training can be setted. Then the system will recommend the suitable workout task based on the active level (e.g. not active such as working as deskjob, normal such as standing like waiter and active such as construction job). There are also prompt user to enter their workout level either beginner, moderate or expert. The proposed app consists this feature. Figure 2.1.1 shows the target settings in Lose weight in 30-day.

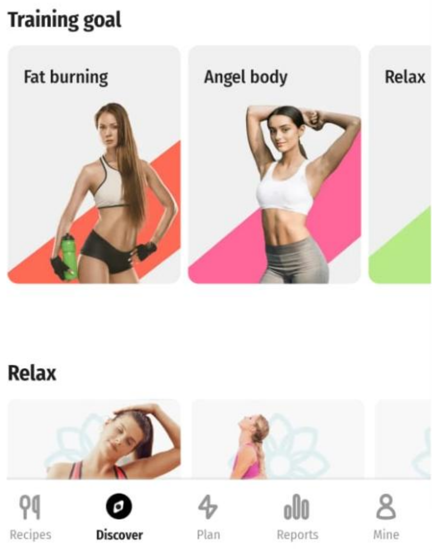


Figure 2.1.1 Target settings in Lose weight in 30-day.

3) Nutrition Tracking:

Healthifyme and MyFitnessPal offer robust nutrition tracking features, which are vital for users who want to monitor their diet. These two applications can show and record the recommended diet nutrients and micronutrients such as protein, carbohydrate, vitamin A, calcium, and more. The user can view and manage to take more food with nutrients that the user's body most needed. Lose Weight in 30 Days , Injurymap and Fitbod do not provide this feature, focusing more on exercise rather than dietary management. The proposed app do not consists this feature.

4) Custom Diet Planning:

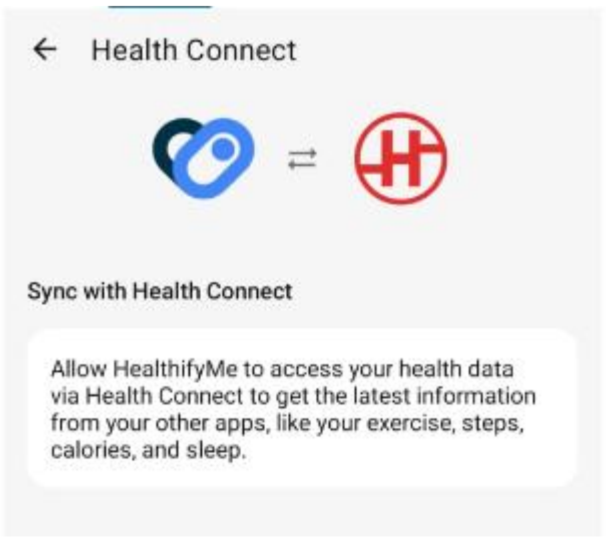
Healthifyme and myfitnesspal offers a custom diet planning feature, making it stand out for users who need specific dietary guidance. In HealthyFity, user able to choose their diet preferences to let the system to recommend such as according to the type of cuisine like Malay, Chinese, Indian, western. This allows the user to eat healthy without pain and be forced to eat the food that they do not like. The other applications do not provide this level of customization, which could be a drawback for those with particular dietary needs. The proposed app do not consists this feature.

5) Custom Workout Planning:

All application except Healthifyme excel in custom workout planning. They offer detailed workout customization options by the level for expertise of user in workout and the target of muscle selected to train. In Lose Weight in 30 Days, the user able to feedback to the system is that too difficult to complete the work plan and do modifications. while HealthyFity do not emphasize too much workout planning but with simple walking tracker only. Injurymap offers treatment-based exercise to reduce body pain. The proposed app consists this feature.

6) Integration with Other Health Applications:

All applications except injurymap offer integration with other health apps, which is beneficial for users who want to have a holistic view of their health metrics. The system will integrate with the health data of the user so that can do a more accurate recommendation and record. The health application that able to connect such as Health Connect, Samsung Health, Google Fit and more. In Healthifyme, Bluetooth connection for the automated recording using smart scale is also included. The proposed app do not consists this feature. Figure 2.1.2 shows integration with other health applications in HealthifyMe app.



Sync with Health Connect

Allow HealthifyMe to access your health data via Health Connect to get the latest information from your other apps, like your exercise, steps, calories, and sleep.

Figure 2.1.2 Integration with other health applications in HealthifyMe app.

7) Social Element:

MyFitnessPal and HealthifyMe include social networking features that allow users to share their diet or exercise progress on social media or within the app. There is a leaderboard feature that can compare points between users which is significant as the completed workout plan. According to H. E. Lee, this fosters encouragement, competition, and a sense of community among users, which can significantly boost user engagement and enhance adherence to health activities [13]. This networkability will improve the willingness of users to continue to use the application. The proposed app do not consists this feature.

8) Progress Tracking:

Progress tracking is available in all the applications. The tracker allow user to monitor their change throughout the whole diet and workout activities by time such as their weight. The system also allows user to take photos to log their body shape changes. Besides, other trackers such as water tracker, sleep tracker, weight tracker and more are included to track and monitor user overall health better. The integrated with other health app like Health Connect ease the progress record. The proposed app consists this feature. Injurymap consists phases progress for rehabilitation exercise in reduce pain. Figure 2.1.3 shows the Progress tracker in Fitbod app.

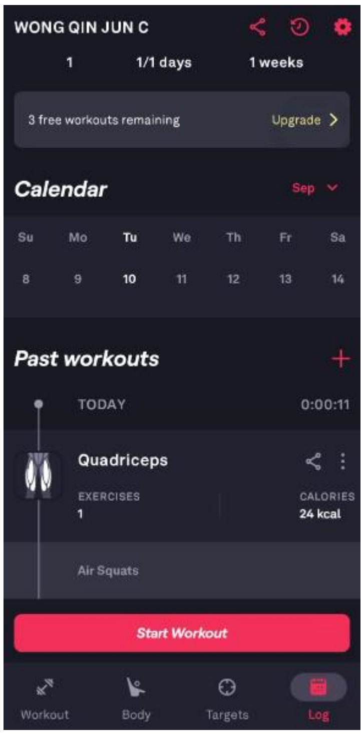


Figure 2.1.3 Progress tracker in Fitbod app.

9) Report and Insight:

Each application includes features for generating reports and providing insights that help users understand their progress and make informed decisions. The system generates reports on aspects such as calorie intake and weight changes over time, highlighting areas where users may need to improve, such as reducing calorie consumption. The graphical representation of weight change against target weight helps users visualize their progress. Injurymap include pain rating in graph for weeks to visualize the progress in reduce pain. The proposed app consists this feature. Figure 2.1.4 shows report for workout in MyFitnessPal app.

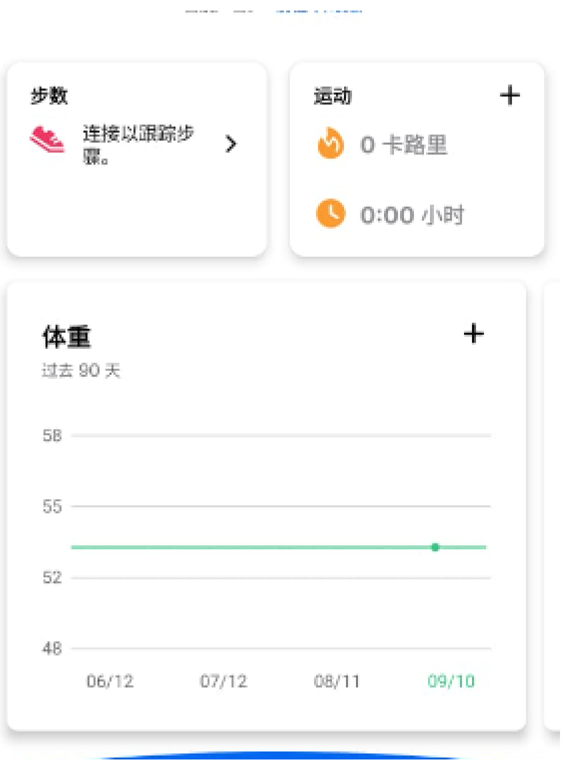


Figure 2.1.4 Report for workout in MyFitnessPal app.

10) Coach Appointment:

Healthifyme and Lose Weight in 30 Days stands out by offering the option to appoint a real person coach, providing users with personalized guidance. Coaches recommend specific actions or dietary choices based on the user's needs, and users can contact their coach anytime and anywhere for support. The proposed app do not consists this feature.

Feature	Healthifyme	Lose Weight in 30 Days	Fitbod	MyFitnessPal	Injurymap	Proposed App
Prompts for Essential Details (e.g., gender, weight, height, age)	✓	✓	✓	✓	✓	✓
Target settings	✓	✓	✓	✓	✓	✓
Nutrition Tracking	✓	✗	✗	✓	x	x
Custom Diet Planning	✓	✗	✗	✓	x	x
Custom Workout Planning	✗	✓	✓	✓	✓	✓
Integration with other health applications	✓	✓	✓	✓	x	x
Social element	✓	✗	✗	✓	x	x
Progress Tracking	✓	✓/	✓	✓	✓	✓
Report and insight	✓	✓	✓	✓	✓	✓
Coach appointment	✓	✓	x	x	x	x

Table 2.1.5 Common feature comparison between 5 applications

2.1.2 Strengths and Limitations in four reviewed Workout Planner mobile applications

The strengths and limitations are compared in the table 2.1.2.1.

1) HealthifyMe

The first sale point in HealthifyMe is embedded with AI called RIA as a coach/assistant. The AI will provide insight into the diet and workout plan about how should be modified to yield better results with clear explanations to the user. Besides, RIA also will motivate the user by giving encouraging words to let the user continue to stay with the plan instead of giving up. Ai can improve the information quality in the application which will motivate the user to continue to use the application. Information is only valuable if users can understand and apply it, and RIA ensures this through its clear and actionable insights [13]. Another key strength of HealthifyMe is its extensive food database, which includes a wide variety of foods, including localized options. This allows users to easily search for foods with detailed information on calories and nutrients. The comprehensive database also enables more personalized recommendations based on user's preferences and the foods available to them. Users can even eliminate foods that they are allergic to from recommendations. Additionally, HealthifyMe offers the option to consult with a real coach or expert at any time, providing users with convenience and access to professional advice.

However, HealthifyMe has limited multilingual support. The food name that is searched must be in English and this makes users inconvenient when do not know the food name in English. The users need to google the food name and translate it by themselves to search and still, it can be inaccurate.

2) Lose weight in 30 days

Lose Weight in 30 Days app appeals strongly to female users by offering targeted body-shaping goals. The app includes various beauty-focused objectives, such as slimming the face, legs, or eliminating a double chin. It also features programs for overall body shaping and getting "bikini ready," which means achieving a body shape that looks good in a bikini. Each target comes with clear, step-by-step workout instructions, complete with videos, to help users understand and follow the routines. The app's female-focused workout plans offer specific options tailored to meet the unique needs of women. In addition to its workout routines, the app provides a well-structured 30-day diet and exercise plan designed to help users achieve their goals within a month. These plans are customized based on the user's activity level—beginner, intermediate, or advanced—and even consider rest times for optimal recovery. Furthermore, the app includes a feedback and adjustment feature that allows users to modify their plans. If a user finds a workout too difficult, they can provide feedback, and the system will suggest a more manageable yet still effective alternative.

However, a limitation of "Lose Weight in 30 Days" is its fixed food recommendations. The app does not customize meal suggestions based on individual preferences while still considering the necessary nutrients or calorie intake. This lack of flexibility may make it harder for users to stick to the diet plan, potentially reducing the program's effectiveness.

3) Fitbod

Fitbod's standout feature is its focus on training specific muscle groups. Users can select which muscles they want to target and specify whether they have access to workout equipment, are training at the gym, or have weightlifting experience. This makes Fitbod particularly valuable for advanced users. Additionally, Fitbod offers AI-powered workouts that adapt over time. The AI learns from users' previous workouts and adjusts future plans accordingly. For example, advanced exercises are only recommended later in the program if the user starts as a beginner.Another key feature is the app's customizable workout plans. Users can tailor their routines based on their preferences, with options to adjust workout equipment weight or increase rest times according to their needs.

However, Fitbod has a limitation which it is less beginner-friendly for those without prior workout knowledge. Users need a basic understanding of exercises and their purposes, as well as clear fitness goals. Without this knowledge, users may struggle with the terminology and end up following routines without fully understanding their benefits. Information loses its value if users do not comprehend it, especially for those who lack clear targets and cannot connect specific exercises to their goals.

4) MyFitnessPal

MyFitnessPal's primary selling point is its barcode scanner feature. This feature allows users to scan the barcode on food packages to instantly check nutritional information, such as calories, protein, and macronutrients. Users can then directly record this information in their MyFitnessPal diary, enabling the app to generate a detailed diet analysis. This makes it convenient for users to track their food intake when a barcode is available. Another strength of MyFitnessPal is its extensive food database, which includes accurate calorie and nutrient data for a wide range of foods. The app also provides macronutrient recommendations, which is particularly helpful for users looking to improve specific aspects of their nutrition. Additionally, the app incorporates social features, like recipe sharing, allowing users to share their diet recipes with friends or copy them for their own use. This fosters a sense of community and provides users with more references and ideas for maintaining a healthy diet. The app also allows users to export reports or data into a CSV file for more advanced analysis.

However, MyFitnessPal has some limitations. Manually entering food data can be time-consuming, especially if the food item doesn't have a barcode. Users must search for and manually log these foods into their diary, which can be burdensome for those who are not in the habit of tracking their meals regularly. Additionally, the app does not offer localized food recommendations, which can be a drawback for users in regions where certain foods are not readily available.

5) Injurymap

Injurymap's main sale points is pain recovery goal-based workout recommendations. The app prompts user to select the area of the body where they feel pain, then suggests a corresponding workout plan in 3 phases with professional doctor advice and validation. It also provides personalized focus areas, activities to avoid, and useful tips to help users get effective results. Injurymap includes a wide range of guided exercises, expert advice, and tutorial videos.

However, Injurymap has some limitations. The advice of professional doctor and expert is written in fully text-based paragraph. This will overwhelm the user to read and obviously know what should take care. The long paragraph can make users lose what they should actually pay attention to. The user can lose patience and quit due to long reading.

6) Proposed app

The proposed app's strength is it tackles and solves the uncertain in real life by feedback back to the system to adjust the workout plan. The user allows to feedback on when they are busy or get injured to adjust the workout plan to align with their current situation. Unlike other applications that suggest workouts for the general case without these uncertainties, the proposed more flexible and personalized to the user's real life. The workout that worsens the injury will be avoided so ensure users have a safe workout experience. The time feedback allows user to adjust their calendar event plan seamlessly for the workout plan allocation and provide HIIT exercise for time-efficient workouts with effective results for busy people.

However, this proposed app is limited to integrate with the wearable device to track the user's body data. These will reduce the users efficiency when the user is required to manually enter the value. Besides, the proposed app do not include a diet recommendation system which is useful in managing user health to support and improve workout plan effectiveness. An incorrect dietary plan will reduce or worsen the workout plan results. While integration of wearable device and diet recommender can be good to improve user workout journey, this are not considered in the current scope due to the project time and resource constraints.

Application reviewed	Strength/sale points	limitation
HealthifyMe	• AI or RIA as a coach/assistant (insights, motivation) • Real coach or human expert available for chat • Strong food database and localised • Customized diet plan with preference and allergy options	• limited multilingual support
Lose weight in 30 days	• Different types of beauty targets (slim legs, angel body) • Organized 30-day plan with rest time • Feedback and replan if workouts are too hard	• Do not have a personalized diet according to preferences
Fitbod	• AI-powered workouts that adapt over time • Customizable workout plans (e.g., add weight, rest timer， gym or home) • Target specific muscle groups • Advanced report metric	• Less friendly to beginners basic knowledge about the workout • A lot of terminology burden user understanding
MyFitnessPal	• Barcode scanner for quick nutrient/calorie detail entry • Food recommendations according to target nutrients • Social feature: Share own recipes of food with others • Option to export reports in CSV format • Strong food database	• Manually record daily food taken requires some time • Do not have localized food recommendation.
Injurymap	• Pain recovery on different body parts	• paragraph-like advice in text which

	• Professional doctor's advice • Multiple workout type with video tutorials	overwhelmed the user to understand
Proposed app	• Uncertainty feedback system (injury, time) • Calendar sync workout scheduler	• Do not integrate with a wearable tracker device to record user data. • Do not have diet recommendation plan for supporting user's workout journey.

Table 2.1.2.1 Strengths and weaknesses of the reviewed application

2.2 Machine learning for the diet and workout recommendations

Previous studies did cover on types of machine learning to build the diet and workout recommender system which are four different algorithms (k-nearest neighbors, Support vector machine, Random forest, AdaBoost) [1]. The researcher had found out that the random forest get the highest F1 score among others which mean high precision and high recall. The C4.5 classifier is used as it consists of additional features such as pre-pruning, handling continuous attributes and missing values, and rule induction to improve accuracy. The step can be concluded into few steps. First, the global entropy and entropy of each attribute will be calculated. Next, calculate the information gain and compute splinting information. The highest information gain ratio attribute will be chosen become the root of the decision tree. Then the dataset is splited based on the root to create subtrees by recursion. The multiple-decision tree will be created to become random forest. The database chosen is using USDA Food Composition Database by the researcher. The algorithm will will receive user input to do recommendation on diet and workouts.

2.3 Rehabilitation application

The rehabilitation app that was reviewed is Sports Injury Rehabilitation [14]. The app consists of two types of users: athletes and coaches. The system will consist of an injury information page for the user to choose the injury type and provide details and causes of the injury. The user can choose the rehabilitation exercise plan after choosing the injury type. The system will provide a plan in multiple phases for the user to follow for effective recovery. Besides the system has a tracker for the progress in days and the phases. With these features, the user is able to know more details about the user’s injury and monitor it to prevent worsening it or to have effective recovery. The Figure 2.3.1 and 2.3.2 show the information page for the injury type and the plan for rehabilitation exercise.

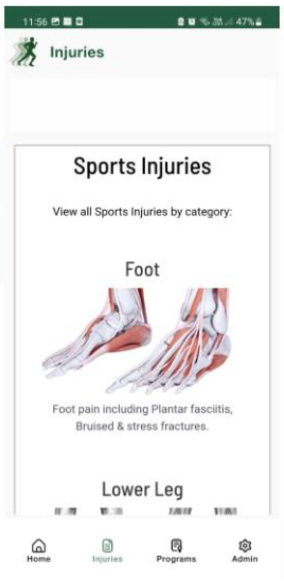

Injuries
Sports Injuries
View all Sports Injuries by category:
Foot
Foot pain including Plantar fasciitis, Bruised & stress fractures.
Lower Leg
Home
Injuries
Programs
Admin

Figure 2.3.1 Information page for the injury type in Sports Injury Rehabilitation app.

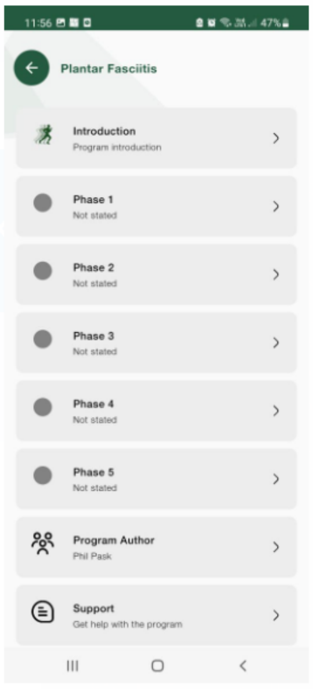


Figure 2.3.2 Plan for rehabilitation exercise in Sports Injury Rehabilitation app.

2.4 High Intensity interval Training (HIIT) Benefits

As introduced in the background information, HIIT is a time-efficient workout that involves switching between short, high-intensity exercise and recovery phase. HIIT have double better performance in improving VO2peak compared to Moderate-Intensity Continuous Training (MICT). Short time and better result can be obtained with HIIT in cardiorespiratory fitness and VO2peak with is suitable for busy people to stay healthy. The HIIT is good in improving cardiovascular risk factor such as blood pressure, cholesterol, body weight and fat regulation [4]. The Merling Phaswana team had proved that HIIT gain better result which the HIIT group trained for about 60% less time than the MICT group, but still ended up burning more energy weekly and recommend that HIIT had potential to be included in vocational working hours [15]. However, MICT is still better in weight loss goal. HIIT is safe to perform basically but it increases risk in cardiometabolic disease such as heart attack. Despite that, its benefit is still outweigh its risk under proper monitoring. The key to performing HIIT are warming up, adjusting the workload, and measuring HRpeak properly. Warming up can help in reaching targeted heart rate safely. Besides, workouts should depend on individual HRpeak and should gradually increase over time, such as after a few weeks if capable to improving the outcome. Last, the HRpeak should be measured properly to prevent underestimating or over. By offering HIIT training in a personalized workout planner app, busy people can gain better results with less time. [4]

2.5 Proposed App

This project aims to propose a personalized workout planner with adaptive, goal-based workout recommendations based on user details. The project includes common features like prompting essential user data, target settings, a custom workout planner with recommendations, a progress tracker, and a report generator. Prompting essential user data is for further analysis and recommendation purposes. AI-powered customizable workout plans will be based on user details such as equipment, fitness level, and fitness goals. The progress tracker allows users to track their progress through various metrics like weight and performance stats. The report and insight generator will create reports based on user activities and workouts over time. Automatic difficulty scaling based on progress completion percentage to ensure user can keep consistency after adjust more suitable difficulty.

A unique aspect of this project is its feedback adjustment feature. The user able to feedback user's injury to the system. The system will offer injury-prevention workout advice. Based on this information, it will avoid recommending exercises that could worsen the injury. This injury-focused approach addresses a significant gap that is not commonly found in existing workout applications. Furthermore, the failure/difficulty adjustment feedback system is designed to enhance the user experience by offering flexibility. If a user finds it challenging to complete their workout plan, the system will suggest adjustments to make the plan more manageable while still effective. Users can manually indicate if a plan is too difficult, and the system will recommend easier alternatives to ensure that the exercise routine remains beneficial without being overly painful.
If the user feedback busy to the system, HIIT exercise will be suggested on a busy day instead of allowing the user to skip or reduce workload to get convenience. The Calendar API is integrated to get the free slot of user to plan the workout. The app can smartly check the user calendar event to suggest a workout schedule time for user. If user feedback is boring to the system, system will suggest alternative workouts that have the same goal to the user. A streak recorder is also included to keep the user motivated and consistent on user workout journey. Notifications such as workout reminders and streak lost reminders are included to notice the user to keep consistent

2.6 Review of Technologies

2.6.1 React Native

React Native, an open-source framework developed by Meta Platforms, Inc. which its significant advantages in cross-platform development stand up in the market. The framework's core principle is "Learn once, write anywhere". React native allows the creation of applications for both Android and iOS with a single JavaScript codebase by bridging the JavaScript with the native platform. Developer can use React and the app platform's native capabilities to build the app. This can reduces development time and operational costs because it eliminates the need for separate development teams and codebases for each platform and speed up the development while having uniform experience for user[16].

Apart from that, Expo has further enhanced the React Native development experience. It is an open-source platform and toolset built around the framework with file-based routing, and standard library of native modules. Expo simplifies development a lot through its "managed workflow," which conceals much of native configuration overhead such that developers only write their React/JavaScript code. A key strength is the Expo Go client, an application that enables developers to instantly run and test their projects on physical iOS and Android devices by using Expo go app. Additionally, Expo provides a robust SDK for seamless access to device capabilities like notifications, which is useful in proposed app for notice user on workouts[17].

However, React native will have slightly less performance than the native platform. React native has a "bridge" to enable JavaScript code to communicate with native rendering APIs . Native Android applications generally maintain a performance advantage since they access a platform's optimized runtime, Android Run-time directly alongside native APIs with absolutely no overhead present in the JavaScript bridge or reliance upon third-party components.[18] Despite this weakness, react native is suitable for developing the proposed app as its shared codebase for different mobile platforms and easy in setup and test with expo.

2.6.2 Supabase

Supabase is a Postgres development platform that serve as the backend infrastructure, which is an alternative of Firebase. It's an open-source platform that provides a suite of backend tools built directly on a dedicated PostgreSQL database [19]. Supabase offers a portable, realtime and extendable easily PostgreSQL database with security enhancement by using powerful Row Level Security (RLS). It enables fine-grained control over data access and ensures that each user's data remains private and secure. Furthermore, Supabase Auth delivers a comprehensive user management system that supports various methods, including email/password and other providers such as Google. It can also be used to store large files such as images and video, which are the needs for the proposed app for workouts images and video. Additionally, Postgres Vector database in Supabase supports using pgvector to store vector embedding of machine learning and large language model with AI Toolkit. These strengths of Supabase meet the requirements of the proposed app for storing a CSV data format dataset and user authentication [19]. Although Supabase has a feature called edge function that can run code directly, it is not compatible with the Python and sklearn library as it is in node.js.

2.6.3 Fastapi

FastAPI, a modern, high-performance web framework for building APIs with Python for custom server-side logic [20]. One of its greatest strengths is its high performance, which is not as heavy as NodeJS and Go, making it one of the fastest Python frameworks available. FastAPI uses standard Python type hints for data validation by Pydantic, a python data validation library and editor support in Visual Studio Code‘s auto-completion to speed up the development process. Another is its interactive automated API documentation. FastAPI generates a valid OpenAPI schema with absolutely no extra work and serves it through two distinct web interfaces: a Swagger UI (at /docs) and ReDoc (at /redoc). It allows one to instantly see, test, and interact with each individual one of the API endpoints directly within a browser, making development easier and future integrations [20]. This FastAPI is suitable for handling the custom server-side workout recommendation logic in the proposed app with quick testing, deployment, and the same Python environment as the machine learning Python environment.

2.6.4 Google Cloud Platform

To facilitate integration with users' personal schedules, the project utilizes Google Cloud Platform (GCP). Google Cloud Platform is behind managing and securely accessing Google's extensive collection of APIs, such as the Google Calendar API[21]. The Google Cloud Platform enables the implementation of the OAuth 2.0 standard, which is a standard for delegated access with user approval to get the access token to ensure a robust and secure authentication and authorization[22]. This protocol allows the proposed app to securely request specific, user-consented permissions (scopes) to access their calendar data with the least privilege, but never having to handle or store a user's Google password. By managing the API credentials and user consent flow through the Google Cloud Console, the application ensures that the calendar integration is secure, private, and reliable.

Chapter 3

System Methodology

3.1 System Design Diagram

3.1.1 System Architecture Diagram

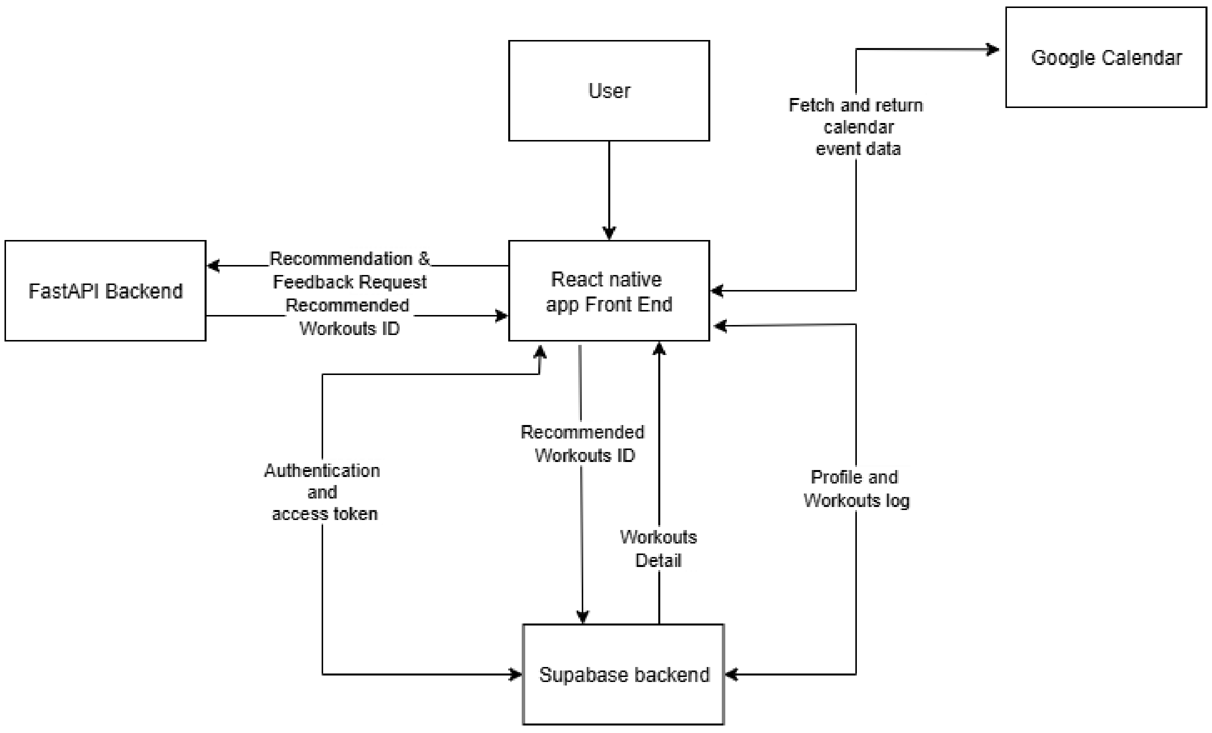


Figure 3.1.1.1: System Architecture Diagram

This diagram illustrates a hybrid system architecture for a mobile fitness application. The design is centered around a React native App Front-End. It acts as the user interface that interacts directly with the user and links to other services. Instead of depending on a single monolithic backend, the architecture divides tasks 3 parts of services: a custom FastAPI Backend for complex recommendation logic, a Supabase Backend for core functionalities like data storage and management, authentication, and user management and Google Calendar from Google Cloud Console to authorize and fetch calendar events. Unlike the traditional architiuecture (Front-End -> API -> Database), this method allows the front-end mobile app to directly communicate with the supabase database (Front-End -> Supabase) with low latency and securely. This is capable due to the supabase acting as a Backend as a Service and its feature, Row Level security. All requests include a new JSON Web Token (JWT) that authenticates the user’s identity as long as they remain signed in with Supabase Auth [23]. JWT provides the foundation of RLS and RLS at PostgreSQL checks the token against policies that are pre-specified, so that users can only view or modify data that they’re authorized to[24]. JWT authentication and database-level policies together make it secure for to leave out a custom backend. This architecture pattern can ease and speed up development, reduce latency, and remain secure.

Component Roles and Rationale

React native App Frontend: This is the user's entry point to the system. It is responsible for rendering the user interface, capturing user input, and communicating directly with the others services. By acting as the center of the system, it controls and calls service to ensure the workflow success and displays to the user to meet user needs. It will requires suggested workouts from FastAPI, update user profile and retrieve workouts detail from supabase and connect with google calendar to get permission and fetch the user calendar event.

FastAPI Backend: This is a specialized microservice whose primary role is to handle computationally intensive business logic, which includes recommending and feedback on workouts. It runs cosine similarity on the precomputed encoding vector of workouts and returns the top 5 similar workouts back to the front end based on user inputs. For the feedback case, it is similar as normal recommendation but with feedback required user input obtained from the frontend. It returns the workouts ID to the front for frontend to fetch the workouts detail directly from Supabase by using the workouts ID. FastAPI with python backend can run complex algorithm for cosine similarity easily with heavy reliance on the frontend and isoslates the complex logic. FastAPI is selected instead of Supabase Edge Function because edge functions do not support the sklearn library for computing cosine similarity.

Supabase Backend: Supabase serves as the foundational backend for the application, handling all standard data operations and storage. It is responsible for user authentication (providing access tokens), storing workout details, and logging user profiles and workout history. Using a BaaS like Supabase can accelerates development as it provides ready-to-use solutions for common needs like database management, user sign-ups, and secure file storage, thus allowing the development to focus more on application-specific aspects. The mobile app communicates directly with Supabase to fetch and store data with the presence of RLS..

Google Calendar API: This is an external, third-party service integrated into the system to provide scheduling functionality. The mobile app makes direct calls to this API to fetch calendar events data and suggest the workouts schedule based on user free time and preference, workout time, and frequency per week. This can enhance the user experience by connecting their fitness plan to their daily schedule.

3.1.2 Use Case Diagram and Description

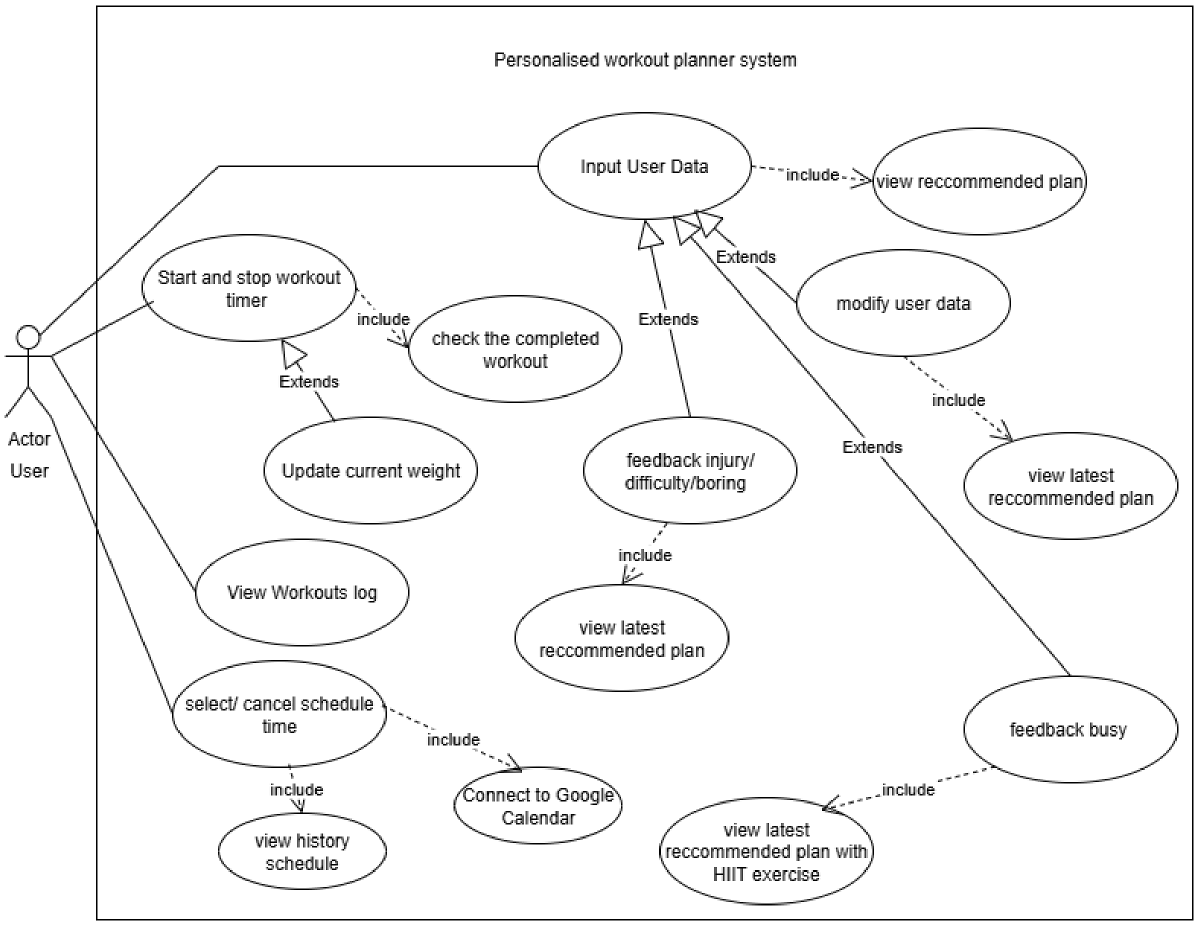

Personalised workout planner system
Input User Data
include
view reccommended plan
Start and stop workout timer
include
check the completed workout
Extends
Extends
Extends
modify user data
include
view latest reccommended plan
Actor
User
Update current weight
feedback injury/ difficulty/boring
Extends
include
View Workouts log
view latest reccommended plan
select/ cancel schedule time
include
include
Connect to Google Calendar
view history schedule
feedback busy
include
view latest reccommended plan with HIIT exercise

Figure 3.1.2.1 Use case diagram

Use case name: Get recommended workout plan	ID : 1	Important level: High
Primary actor: User	Use case type: Detail, Essential	

Brief description: This use case describes how user can obtain the recommended workout plan according to user' situation.
Trigger: User inputs user data such as height, weight, goal, fitness level, injury notes, equipment, workouts per week, preferred workout time, and favorite workout to the personalized workout planner system. Type: external
Relationship: Association: User Extend: feedback injury/difficulty/boring/busy, modify user data Include: view recommended workout plan Generalization: -
Normal Flow of Events: 1) The user inputs user data such as height, weight, goal, fitness level, injury notes, equipment, workouts per week, and preferred workout time to the personalized workout planner system during the initial setup. User can connect to Google Calendar. 2) The personalized workout planner system recommends a suitable workout plan to the user to achieve the user's goal. 3) The user can view the recommended workout plan. 4) User can select or cancel the schedule time after the user connect with Google Calendar. 5) The user starts following the workout plan using the timer feature. 6) The system marks the workout plan task as progress based on the completation percentage after done or stop. 7) The personalized workout planner system will show the workouts logs such as average percentage of completion and its trend, BMI, weight changes, number of workouts completed this month, current weekly workout streak in weeks, and Muscle Group Focus bar. 8) Repeat 4-7 until reaching the goal.
SubFlows: -
Alternate/Exceptional Flows: 5a) If the user gets injured,

1. The user reports the injury to the system.
2. The system will identify the injury and recommend the latest workout plan that not worsen the injury.

5b) If the user felt the workout is too challenging,

1. The user reports the workout is too challenging to the system.
2. The system will recommend the latest workout plan that is easier and painless but still effective.

5c) If the user felt the workout is boring,

1. The user reports the workout is boring.
2. The system will recommend the latest workout plan that alternative workout with the same goal.

5d) If the user is busy and no time for exercise,

1. The user reports is busy.
2. The system will recommend the latest workout plan that alternative workout which is HIIT exercise, and adjust the workout per week to 1.

6a) If the user has a workout complete rate <50% for 3 in a row,

1. The system will recommend the latest workout plan that is easier and painless but still effective.

6b) if the user stop and mentions failure,

a) The system will calculate the current workout completion rate and record into the database.

6c) if the user clicks discard,

1. The system will not save the current workout data.

.

3.1.3 Activity Diagram

3.1.3.1 Flow 1: The Core Loop - Generating and Starting a Workout

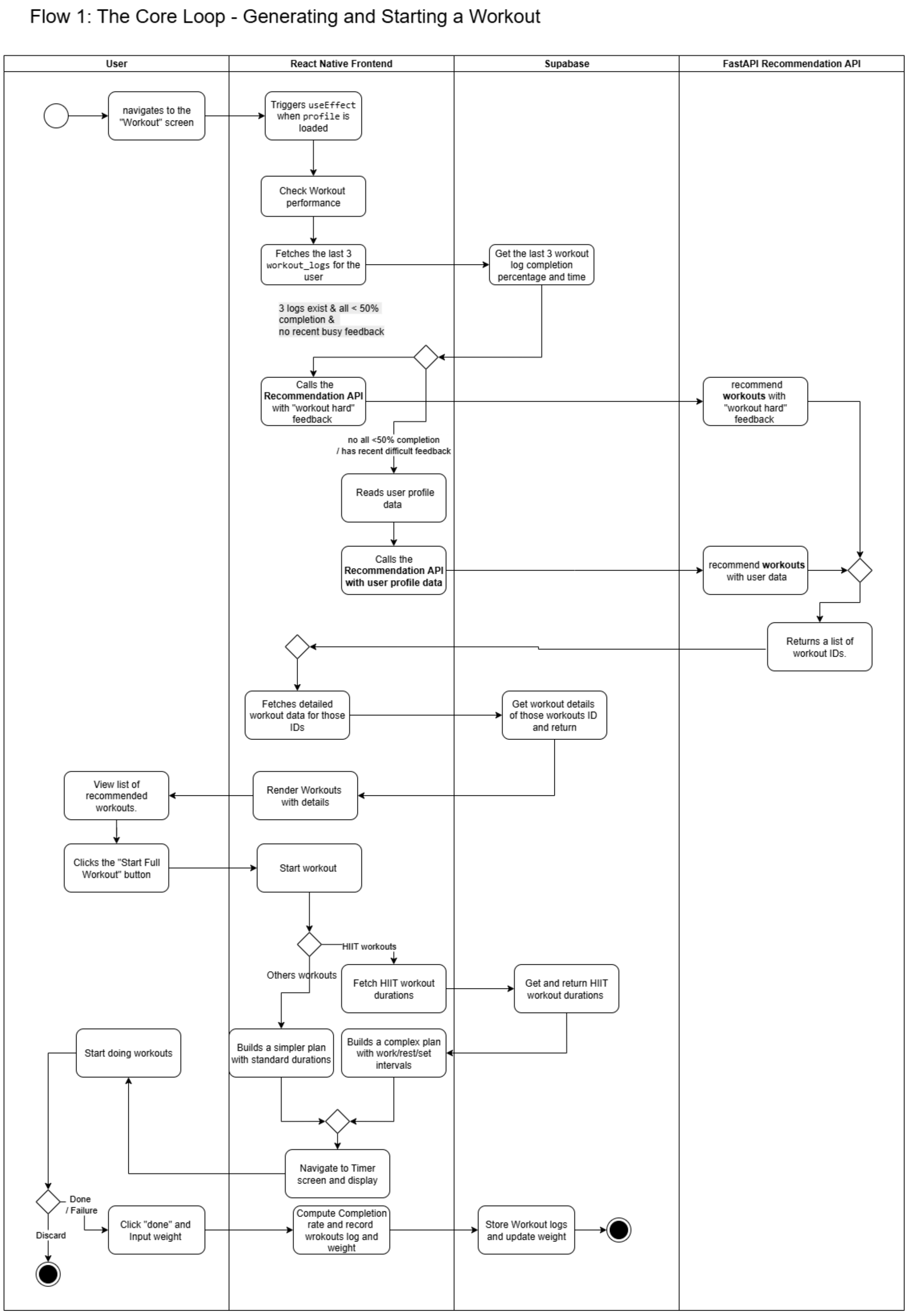

Flow 1: The Core Loop - Generating and Starting a Workout
User
React Native Frontend
Supabase
FastAPI Recommendation API
navigates to the "Workout" screen
Triggers useEffect when profile is loaded
Check Workout performance
Fetches the last 3 workout_logs for the user
Get the last 3 workout log completion percentage and time
3 logs exist & all < 50% completion & no recent busy feedback
Calls the Recommendation API with "workout hard" feedback
recommend workouts with "workout hard" feedback
no all <50% completion / has recent difficult feedback
Reads user profile data
Calls the Recommendation API with user profile data
recommend workouts with user data
Returns a list of workout IDs.
Fetches detailed workout data for those IDs
Get workout details of those workouts ID and return
View list of recommended workouts.
Render Workouts with details
Clicks the "Start Full Workout" button
Start workout
HIIT workouts
Others workouts
Fetch HIIT workout durations
Get and return HIIT workout durations
Start doing workouts
Builds a simpler plan with standard durations
Builds a complex plan with work/rest/set intervals
Navigate to Timer screen and display
Done / Failure
Discard
Click "done" and Input weight
Compute Completion rate and record wrokouts log and weight
Store Workout logs and update weight

Figure 3.1.3.1: Flow 1, The Core Loop - Generating and Starting a Workout

This activity flow outlines the application's core functionality, dynamic generation and initiation of a personalized workout recommendation. This process is core to deliver a tailored fitness experience to the user based on user profile data.

The flow starts when the authenticated user navigates to the main "Workout" screen. Upon loading, the frontend system initiates a checking process. Firstly, it performs an automated check on the user's recent performance by querying the `workout_logs` table in the Supabase database. If it detects a three consecutive workouts completed at less than 50% of low completion rates, the system automatically triggers a difficult feedback request to the Recommendation API, indicating that the workouts have been too difficult to lower one difficulty level. Then, it will proceeds with workout recommendation.

If no performance-based adjustment is needed, the system proceeds with the standard workout recommendation. It retrieves the user's profile data, including their fitness level, goals, available equipment, and any noted injuries, from the `user_profile_new` table in Supabase. This data is then packed into a request to the Recommendation API. The API processes these parameters and returns a list of recommended workout IDs.

With these IDs, the frontend system connects to Supabase's `Workouts` database table to fetch respective details for each exercise, such as instructions and muscle groups targeted. The data from the recommendation API and the Supabase database are then merged and presented to the user as a list of recommended workouts in a visual display.

After that, the user is ready to start a workout. The user clicks the "Start Full Workout" button, and then the system constructs a detailed, step-by-step workout timer plan. It dynamically adjusts the structure of this plan based on the workout category. For instance, a HIIT workout session is built with specific work_duration, rest, and set intervals fetched from the `HIIT_duration` table. The workout duration, rest and number of set based on fitness level is referred to the DAREBEE website to create[25]. For others workout will use a standard workout timer plan. Finally, the system navigates the user to the timer interface, passing the structured plan to begin the guided workout. If the user completes the workout session or presses stop to declare failure, the system will calculate the current percentage of completion, ask the user to input their latest weight, and then record it to the Supabase workout_logs table. The user can also choose to discard the timer, and this workout log will not be recorded for misclick purpose.

3.1.3.2 Flow 2: Providing Manual Feedback

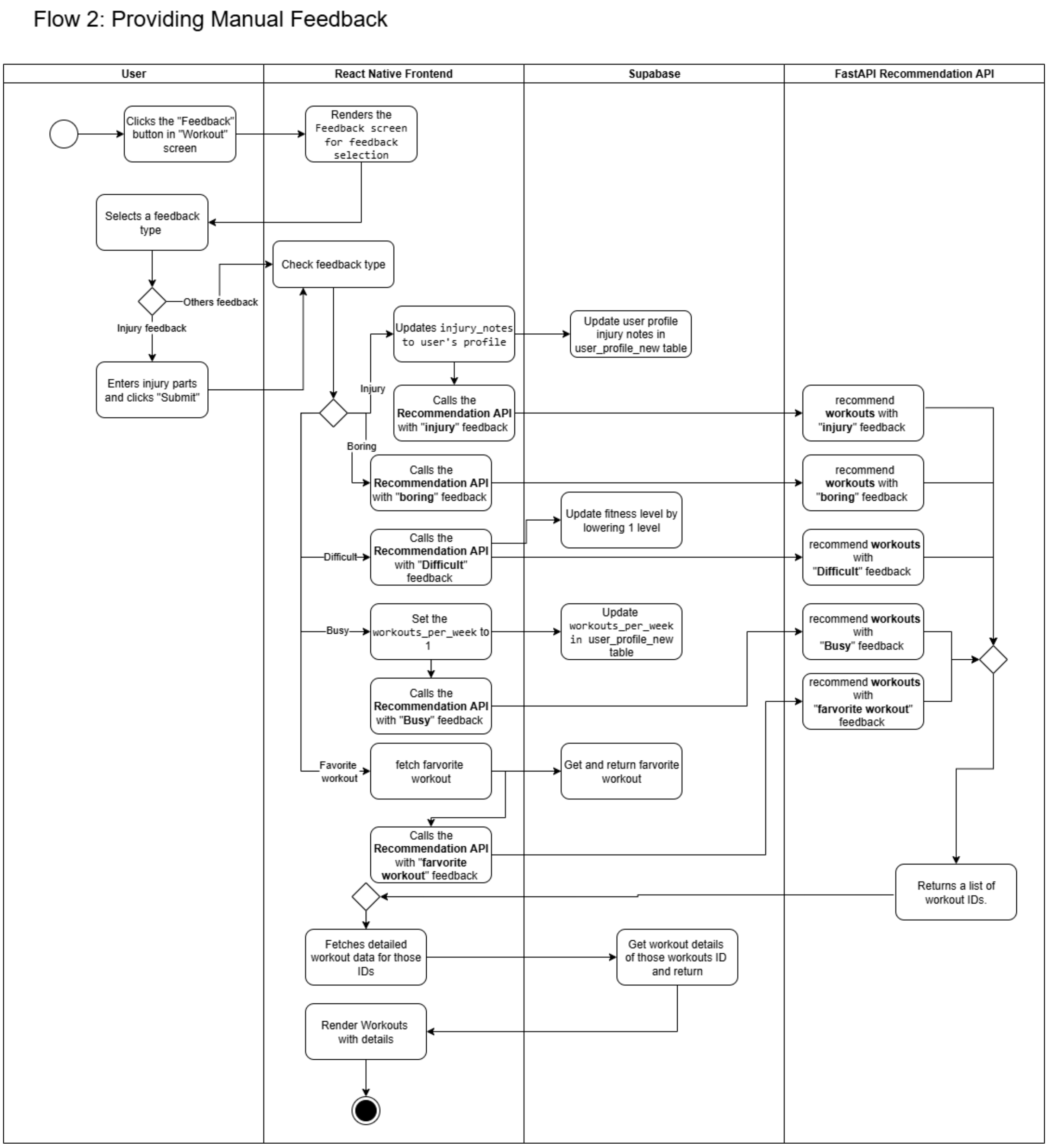

Flow 2: Providing Manual Feedback
User
React Native Frontend
Supabase
FastAPI Recommendation API
Clicks the "Feedback" button in "Workout" screen
Renders the Feedback screen for feedback selection
Selects a feedback type
Check feedback type
Others feedback
Injury feedback
Enters injury parts and clicks "Submit"
Updates injury_notes to user's profile
Update user profile injury notes in user_profile_new table
Injury
Calls the Recommendation API with "injury" feedback
recommend workouts with "injury" feedback
Boring
Calls the Recommendation API with "boring" feedback
recommend workouts with "boring" feedback
Update fitness level by lowering 1 level
Difficult
Calls the Recommendation API with "Difficult" feedback
recommend workouts with "Difficult" feedback
Busy
Set the workouts_per_week to 1
Update workouts_per_week in user_profile_new table
recommend workouts with "Busy" feedback
Calls the Recommendation API with "Busy" feedback
recommend workouts with "farvorite workout" feedback
Favorite workout
fetch farvorite workout
Get and return farvorite workout
Calls the Recommendation API with "farvorite workout" feedback
Returns a list of workout IDs.
Fetches detailed workout data for those IDs
Get workout details of those workouts ID and return
Render Workouts with details

Figure 3.1.3.2: Flow 2, Providing Manual Feedback

This flow details how the system adjusts workout recommendations based on direct user feedback, ensuring the fitness plan remains aligned with the user's changing needs and preferences to counter real-life uncertainty.

The interaction starts when the user clicks the "Feedback" button on the main workout screen, which it will renders a selection of feedback options. The system's response is tailored to the specific type of feedback submitted in the `handleFeedbackSubmit` function.

- **Injury Feedback**: If the user reports an injury, the system directly updates the `injury_notes` field in the `user_profile_new` table in Supabase. This ensures that all future workout generations will automatically consider this new physical limitation without call this injury feedback again. After the user recover from injury, the user can manually remove the injury notes field in the account profile.
- **Boring Feedback:** The system makes a direct call to the Recommendation API to recommend similar workouts from the previous randomly but this meet the user current needed by their user profile parameters.
- **Difficult Feedback**: the system reduces 1 level of fitness level and call the Recommendation API with the rest of the user profile parameters to request a new set of workouts. The lower fitness level will be updated in supabase.
- **Availability Feedback (Busy)**: If the user indicates they are "busy," the system performs two actions. It updates the `workouts_per_week` in their Supabase profile to a lower frequency (e.g., 1) and simultaneously calls the Recommendation API with the "busy" feedback to generate a shorter, more time-efficient HIIT workout plan.
- **Preference Feedback (Favorite)**: When the user provides feedback on a favorite workout by marking a workout as favorite, the system calls the Recommendation API with the `favorite_workout_id` to recommend similar workouts that within the user profile parameters. This ensures the favorite workout can still achieve user needs

In all cases where the API is called, it returns a new list of workout IDs. The frontend then fetches the corresponding workout details from Supabase and refreshes the UI, presenting the user with the latest workout recommendation list.

3.1.3.3 Flow 3: Scheduling a Workout

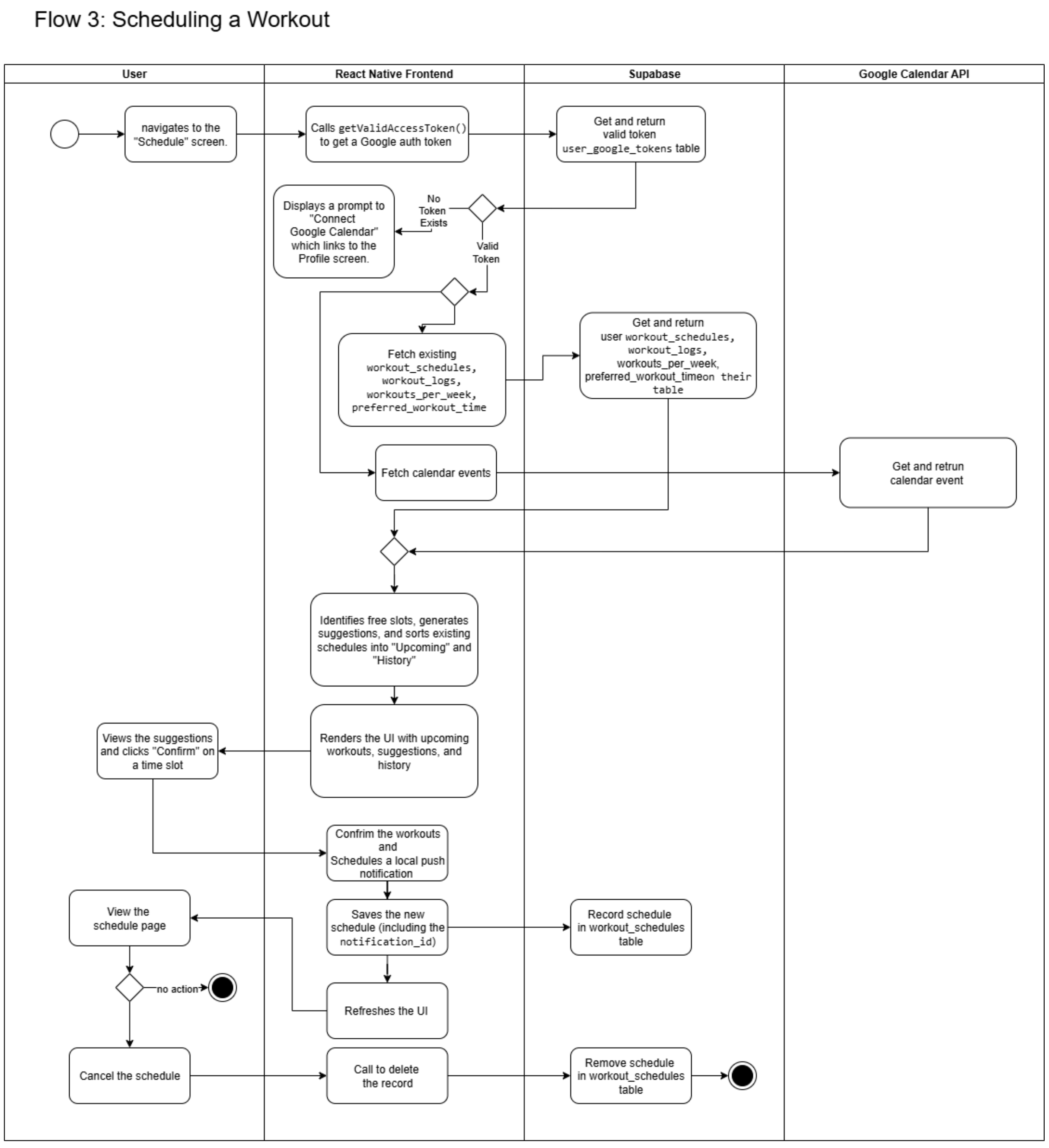

Flow 3: Scheduling a Workout
User
React Native Frontend
Supabase
Google Calendar API
navigates to the "Schedule" screen.
Calls getValidAccessToken() to get a Google auth token
Get and return valid token user_google_tokens table
No Token Exists
Valid Token
Displays a prompt to "Connect Google Calendar" which links to the Profile screen.
Fetch existing workout_schedules, workout_logs, workouts_per_week, preferred_workout_time
Get and return user workout_schedules, workout_logs, workouts_per_week, preferred_workout_timeon their table
Fetch calendar events
Get and retrun calendar event
Identifies free slots, generates suggestions, and sorts existing schedules into "Upcoming" and "History"
Renders the UI with upcoming workouts, suggestions, and history
Views the suggestions and clicks "Confirm" on a time slot
Confrim the workouts and Schedules a local push notification
Saves the new schedule (including the notification_id)
Record schedule in workout_schedules table
View the schedule page
no action
Refreshes the UI
Cancel the schedule
Call to delete the record
Remove schedule in workout_schedules table

Figure 3.1.3.3: Flow 3, Scheduling a Workout

This flow describes the adaptive scheduling feature, which integrates with the user's Google Calendar to suggest free slots times and manage their fitness schedule.

The process is start when the user navigates to the "Schedule" screen. First, the system will calls `getValidAccessToken()` to get a Google auth token from the `user_google_tokens` table in Supabase. If valid token is not found, the user is prompted to connect their Google account via the profile screen. The user will be navigated to the profile page after clicking the "Go to Profile".

With a valid token, the frontend fetches existing `workout_schedules`, `workout_logs`, `workout_per_week, and preferred_workoit_time` from Supabase. It also making an API call to the Google Calendar API to retrieve the user's events for the upcoming week simultaneously . After that, the system processes these data, identifying free time slots that align with the user's `preferred_workout_time` and generates a list of suggestions.

The system will displays the user's upcoming scheduled workouts, their recent history, and the new suggestions. The user can select and "Confirm" a suggested time slot same as the number of `workout_per_week` set by the user in their profile. Upon confirmation, the `handleConfirmWorkout` function performs two key actions. It schedules a local push notification on the device to remind the user of their session and saves the new schedule, including the `notification_id`, to the `workout_schedules` table in Supabase. Next, UI is refreshed to reflect the newly added workout. The user also able to cancel a scheduled workout, which removes the associated notification and deletes the record from the database.

3.1.3.4 Flow 4: User Onboarding, Authentication, and Profile Management

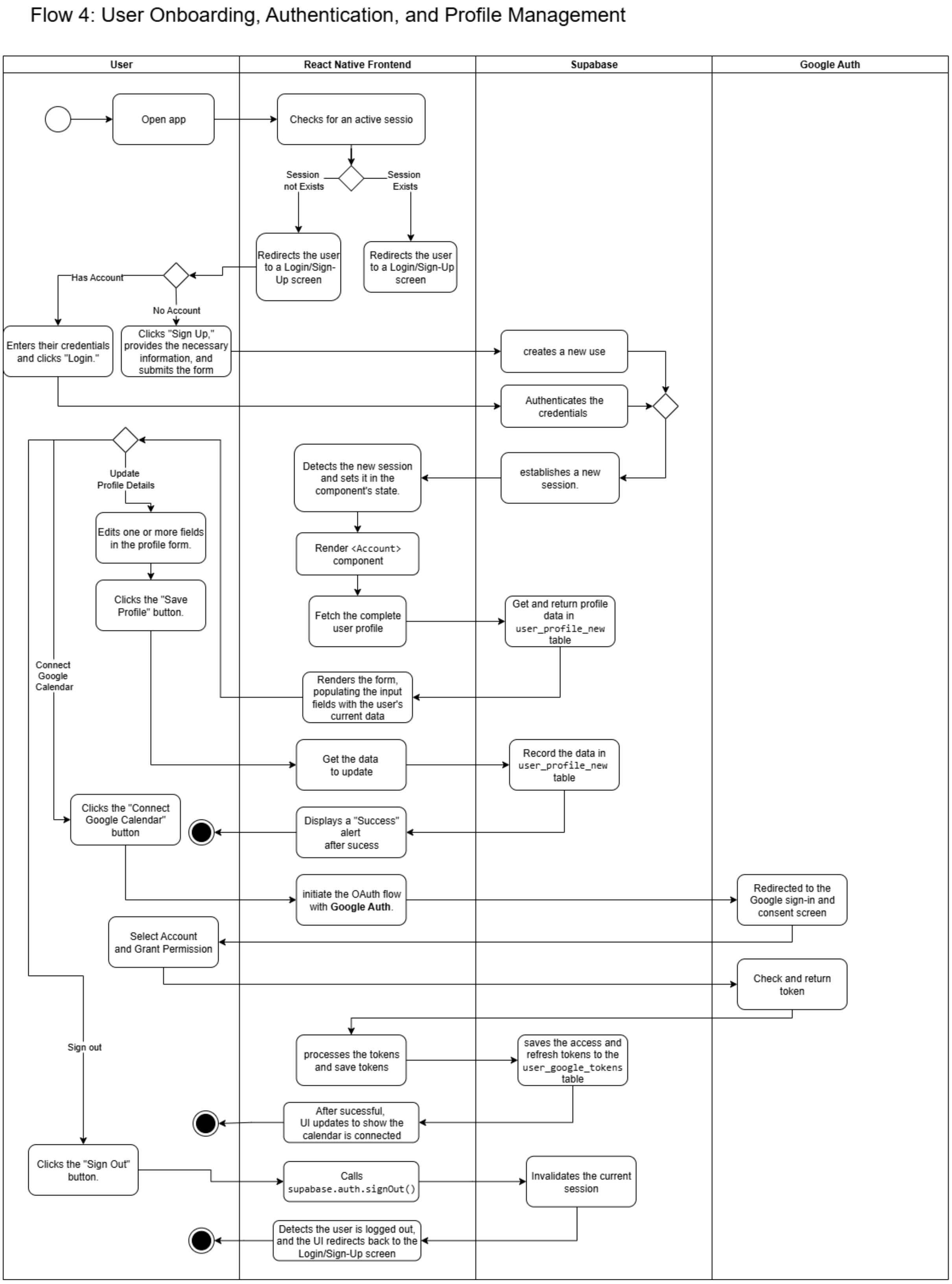

Flow 4: User Onboarding, Authentication, and Profile Management
User
React Native Frontend
Supabase
Google Auth
Open app
Checks for an active sessio
Session not Exists
Session Exists
Redirects the user to a Login/Sign-Up screen
Redirects the user to a Login/Sign-Up screen
Has Account
No Account
Enters their credentials and clicks "Login."
Clicks "Sign Up," provides the necessary information, and submits the form
creates a new use
Authenticates the credentials
Detects the new session and sets it in the component's state.
establishes a new session.
Update Profile Details
Edits one or more fields in the profile form.
Render <Account> component
Clicks the "Save Profile" button.
Fetch the complete user profile
Get and return profile data in user_profile_new table
Connect Google Calendar
Renders the form, populating the input fields with the user's current data
Get the data to update
Record the data in user_profile_new table
Clicks the "Connect Google Calendar" button
Displays a "Success" alert after sucess
initiate the OAuth flow with Google Auth.
Redirected to the Google sign-in and consent screen
Select Account and Grant Permission
Check and return token
Sign out
processes the tokens and save tokens
saves the access and refresh tokens to the user_google_tokens table
After sucessful, UI updates to show the calendar is connected
Clicks the "Sign Out" button.
Calls supabase.auth.signOut()
Invalidates the current session
Detects the user is logged out, and the UI redirects back to the Login/Sign-Up screen

Figure 3.1.3.4: Flow 4, User Onboarding, Authentication, and Profile Management

This flow covers the complete user lifecycle, from initial sign-up and authentication to profile management and Google Calendar service integration.

When a user opens the app, the frontend checks for an active session with Supabase immediately. If no session exists, the user is redirected to a login/sign-up screen. The user can either sign up to create an account or log in to authenticate their credentials using email and password. After successful authentication, Supabase establishes a new session, and the frontend detects this change, allowing user to navigate to the main application content.

Once logged in, the user can manage their profile. The system fetches their complete profile data from the `user_profile_new` table in Supabase and display the form in the `<Account>` component in the profile page. The user has several options:

- **Update Profile Details**: The user can edit their information and click "Save Profile." The changes will be recorded to update in the `user_profile_new` table in Supabase.
- **Connect Google Calendar**: By clicking "Connect Google Calendar," the system will initiate an OAuth flow with Google. User will be redirected to the Google sign-in and consent page to select their Google account and authenticate as an authorized user. After granting permission, Google returns authentication tokens. The frontend system will process and save securely to the `user_google_tokens` table in Supabase. The UI re-renders to reflect the successful connection with the connected green button.
- **Sign Out**: The user can click "Sign Out" to calls the `supabase.auth.signOut()` method. Supabase will invalidates the current session, and the frontend redirects the user back to the login screen, completing the session lifecycle.

Chapter 4: System Design

4.1 System Block Diagram

4.1.1 React native app frontend Block Diagram

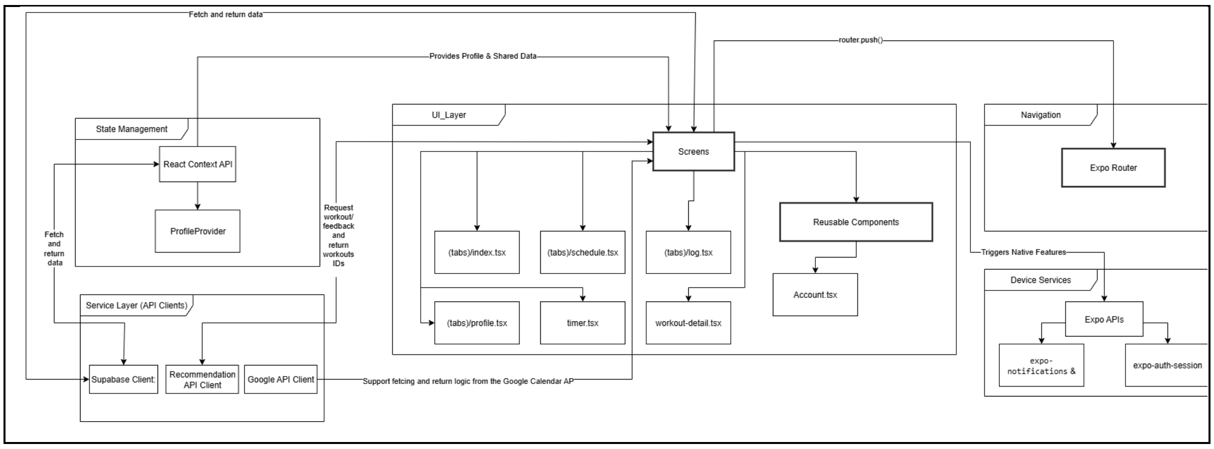


Figure 4.1.1: React native app frontend Block Diagram

a) Screens & Components

The block is responsible for rendering all pages, visual elements of the application, and capturing user input for the user control. This is the main interface that users interact with the system. The UI is built using react native components and a pre-built kit from react native paper, a UI component library. This allows speeding up development with a standard outlook instead of building from scratch. The application is divided into several key screens, including the main workout recommendation screen (`index.tsx`), the scheduling screen (`schedule.tsx`), the user's progress log screen (`log.tsx`), the workout timer (`timer.tsx`), and the profile management screen (`profile.tsx` and `Account.tsx`).

b) Navigation

This block is responsible to manage the navigation and flow among different screen with Expo Router. By putting the file in the "(tabs)" folder in the project and defining navigation tabs in "`(tabs)/_layout.tsx`", the Expo router can easily route users using the navigation tab bar that is located at the bottom of the screen. Screens such as `timer.tsx` which is not included in the tabs navigation bar, can use useRouter hook from Expo router to moving to the timer screen after a user starts a workout.

c) State Management

This block is responsible for a centralized way to manage and share application-wide data, such as user profile information, without having to pass props through many layers of components. ProfileProvider from `ProfileContext.tsx` include all the necessary data that is needed across all pages and is used to wrap the screen parent layout, which is "`_layout.tsx`" in the app folder. This ensures the screens can access data. It holds and provides user profile data for screens.

d) Service Layer (API Clients)

This block is responsible to all external communication, acting as a single gateway for fetching or sending data to the external service such as Supabase, FastAPI recommendation backend, and Google Calendar. This separates the UI logic from the data-fetching logic.

The Supabase Client is an interface used for all interactions with the Supabase, including user account authentication and database operations like fetching workout logs in `log.tsx` or saving scheduled workouts in `schedule.tsx`. This supabase client is created in "`lib/supabase.ts`".

The Recommendation API Client part is responsible for handling all HTTP fetch requests to the FastAPI backend. This is used in `index.tsx` to get workout recommendations and send feedback request. This can reduce duplication of code and centralize the HTTP request in "`lib/api.ts` ".

The Google API Client part is responsible for logic for authenticating and fetching events from the Google Calendar API, which is included in the schedule.tsx and Account.tsx components..

e) Device Services (Expo APIs)

This block is responsible for handling the mobile device's native hardware and software capabilities for a better user experience. The Expo library is used to simplify the usage and integration. The `expo-notifications` is used to to schedule local push notifications to remind the user of an upcoming workout session in the schedule page. For the `expo-auth-session`, it is used to handle the OAuth 2.0 flow required for secure user authentication with Google Calendar.

4.1.2 FastAPI Backend Block Diagram

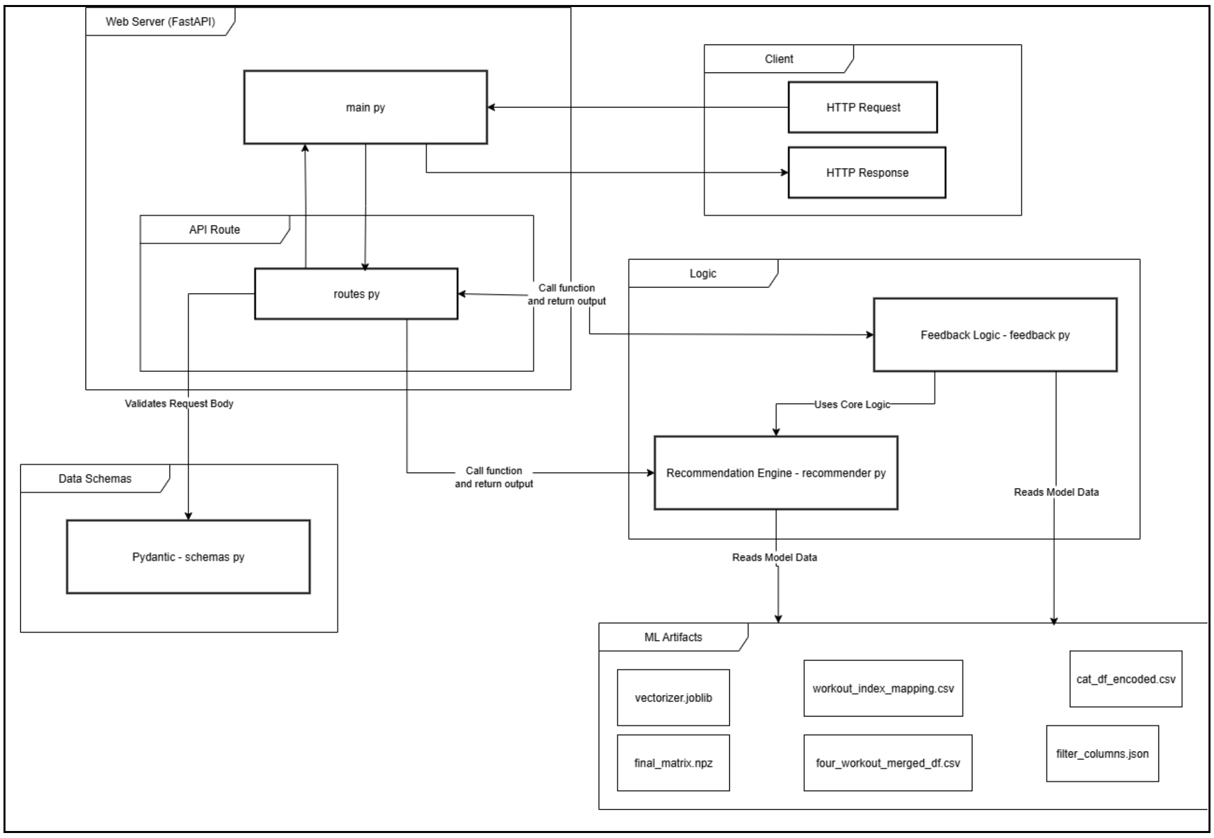


Figure 4.1.2: FastAPI Backend Block Diagram

1. **Web Server (FastAPI)**

This block is responsible for acting as the main entry point for the backend service. It will listen to the HTTP requests and the application lifecycle. The FastAPI framework is used and created in `main.py`. It also configures Cross-Origin Resource Sharing (CORS) middleware to securely accept requests from the React Native frontend. The CORS middleware is a special handler for the web application that has a different origin, including different port number, so that they can communicate[26]. While mobile app requests from Android and IOS are not subject to browser CORS restrictions. This is still included for future web application possibilities and ease for testing. At the beginning, it is defined with the @app.on_event("startup") decorator for loading machine learning artifacts into memory. This allows those artifacts to be ready before any requests are served.

2. **API Router**

This block is inside the web server block and is responsible for mapping incoming request URLs to the specific business logic functions that can handle them within the backend application. It is created in `routes.py` and used to manage all API endpoints such as POST request to the /recommend path is routed to the recommendation function for workout recommendation, and then returns the output for the HTTP response.

.

3. Data Schemas (Pydantic)

This block is responsible for data validation so that the data in JSON format will not causes error and fit to the python data format. This ensures data integrity by validating the structure and types that are expected for data processing. It is created in `schemas.py` using Pydantic BaseModel. Classes is created, such as RecommendRequest, InjuryFeedback and more. They are used as type hints in the router functions for automatically parsing and validating API requests.

4. Logic - Recommendation Engine

This block is the core logic of the recommendation system by running the algorithms for user input filter, which is based on user profile data, running cosine similarity, and generating a ranked list of personalized workouts with ID to return. It is created in `recommender.py` and relies on ML Artifacts for running the operation. The operation includes filters workouts, performs the core similarity calculation against the pre-loaded matrix, `final_matrix.npz`, and gets the top 5 high score workouts as recommendations.

5. Logic - Feedback Logic

This block is the special handler for modifying the recommendation process based on specific types of user feedback for solving their real-life uncertain needs. It is created in feedback.py and has 5 types which are busy, injury, boring, difficulty, and favorite. Special functions for handling data are also created, such as filter_workouts_by_injury for preprocess the user's request before it is passed to the main recommendation engine.

6. ML Artifacts

This block is the pre-computed of data or models that created from the Jupiter notebook for the recommendation engine and feedback logic to use. This allows the system can perform efficiently by recomputing the heavy and repetitive logic, such as matrix encoding for the workout data and for mapping the matrix with its workout ID. The artifacts includes 6 types.

- a Scikit-learn TF-IDF vectorizer (vectorizer.joblib)
- SciPy sparse matrix of workout features (final_matrix.npz
- CSV files for mapping workout IDs to matrix indices (workout_index_mapping.csv)
- Workout dataset for referring if needed (four_workout_merged_df.csv)
- List of filter columns needed for recommendations for documentation purposes (filter_columns.json)
- Reference of efficient pre-filtering for the user input filter before cosine similarity step (cat_df_encoded)

4.2 System component top-down flow diagram

4.2.1 Workout page, index.tsx

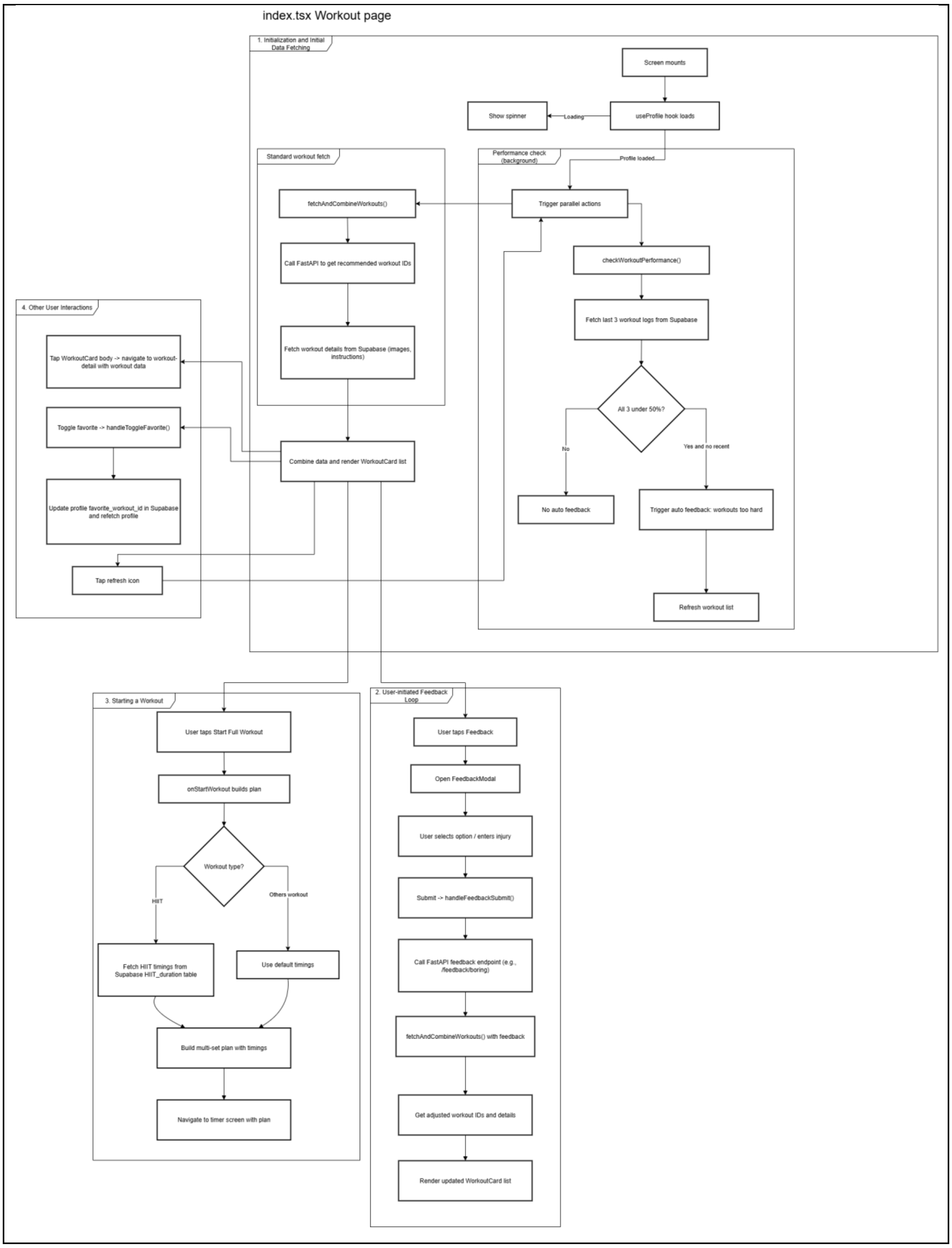

index.tsx Workout page
1. Initialization and Initial Data Fetching
Screen mounts
useProfile hook loads
Loading
Show spinner
Profile loaded
Standard workout fetch
fetchAndCombineWorkouts()
Call FastAPI to get recommended workout IDs
Fetch workout details from Supabase (images, instructions)
Combine data and render WorkoutCard list
Performance check (background)
Trigger parallel actions
checkWorkoutPerformance()
Fetch last 3 workout logs from Supabase
All 3 under 50%?
No
Yes and no recent
No auto feedback
Trigger auto feedback: workouts too hard
Refresh workout list
4. Other User Interactions
Tap WorkoutCard body -> navigate to workout-detail with workout data
Toggle favorite -> handleToggleFavorite()
Update profile favorite_workout_id in Supabase and refetch profile
Tap refresh icon
3. Starting a Workout
User taps Start Full Workout
onStartWorkout builds plan
Workout type?
HIIT
Others workout
Fetch HIIT timings from Supabase HIIT_duration table
Use default timings
Build multi-set plan with timings
Navigate to timer screen with plan
2. User-initiated Feedback Loop
User taps Feedback
Open FeedbackModal
User selects option / enters injury
Submit -> handleFeedbackSubmit()
Call FastAPI feedback endpoint (e.g., /feedback/boring)
fetchAndCombineWorkouts() with feedback
Get adjusted workout IDs and details
Render updated WorkoutCard list

Figure 4.2.1: Workout page, index.tsx

1. **Initialization and Initial Data Fetching**

The process start when the user navigate to the workout page(index.tsx). The system will directly calls the useProfile hook. While the profile is loading, a spinner which is "<ActivityIndicator />" in react native paper library, is shown. After the user's user's profile is successfully loaded, it runs two independent processes in parallel.

- **Performance Check:**
  The `checkWorkoutPerformance` function runs in the background and fetchesthe user's last three workout logs from the Supabase database. If all 3 logs are < 50% of completion rate and no recent feedback difficult record in Supabase (lastFeedbackTimestamp < lastWorkoutTimestamp), it automatically triggers the feedback difficult flow and refreshes the workout list. Otherwise, it will do a normal workout recommendation.
- **Standard Workout Fetch:**
  The fetchAndCombineWorkouts function will run and start normal recommendation flow. If the injury notes field is not null for the user profile, it will call the feedback injury instead. It calls FastAPI backend to get a list of recommended workout IDs.

After getting the list of workout IDs, it is used to fetch full workout details such as instructions, image, primary, and secondary muscles from the Supabase Workouts table. The data of from the backend and Supabase are then combined and displayed to the user in a list of WorkoutCard components.

2. **User-Initiated Feedback Loop**

When user clicks the "Feedback" button, which opens the FeedbackModal pop up screen. The user can choose 5 options which are boring, busy, difficult, injury and favorite workouts. For the injury option, the user needs to type the injury parts with a comma as delimiter and submit. The modal's onSubmit function calls handleFeedbackSubmit in the main screen then calls fetchAndCombineWorkouts again with a specific feedback function to reach specific FastAPI endpoint. The list of recommended workout IDs will return and undergo the normal flow to display.

3. **Starting a Workout**

When the user clicks the "Start Full Workout" button, `onStartWorkout` function analyzes the workout type. If it is a HIIT workout, an extra call to Supabase to get specific timings (work, rest, sets) from the HIIT_duration table to build a detailed, multi-set plan. Otherwise, it will use a standard default plan which are 3 minutes for doing workout and 2 minutes of rest for 1 workout. The constructed plan will be pass to `timer` screen for display timer.

Other actions, such as Mark Favorite by tapping the heart icon on a WorkoutCard calls handleToggleFavorite. This will update the favorite_workout_id in the user's profile on Supabase and then refetch the profile to update the UI. To get more information, the user can tap on the view instruction button of a WorkoutCard to navigate to the workout-detail screen, passing all the workout data for that card. The user can also tap the the refresh button to refresh to do a normal recommendation flow.

4.2.2 Profile page, profile.tsx

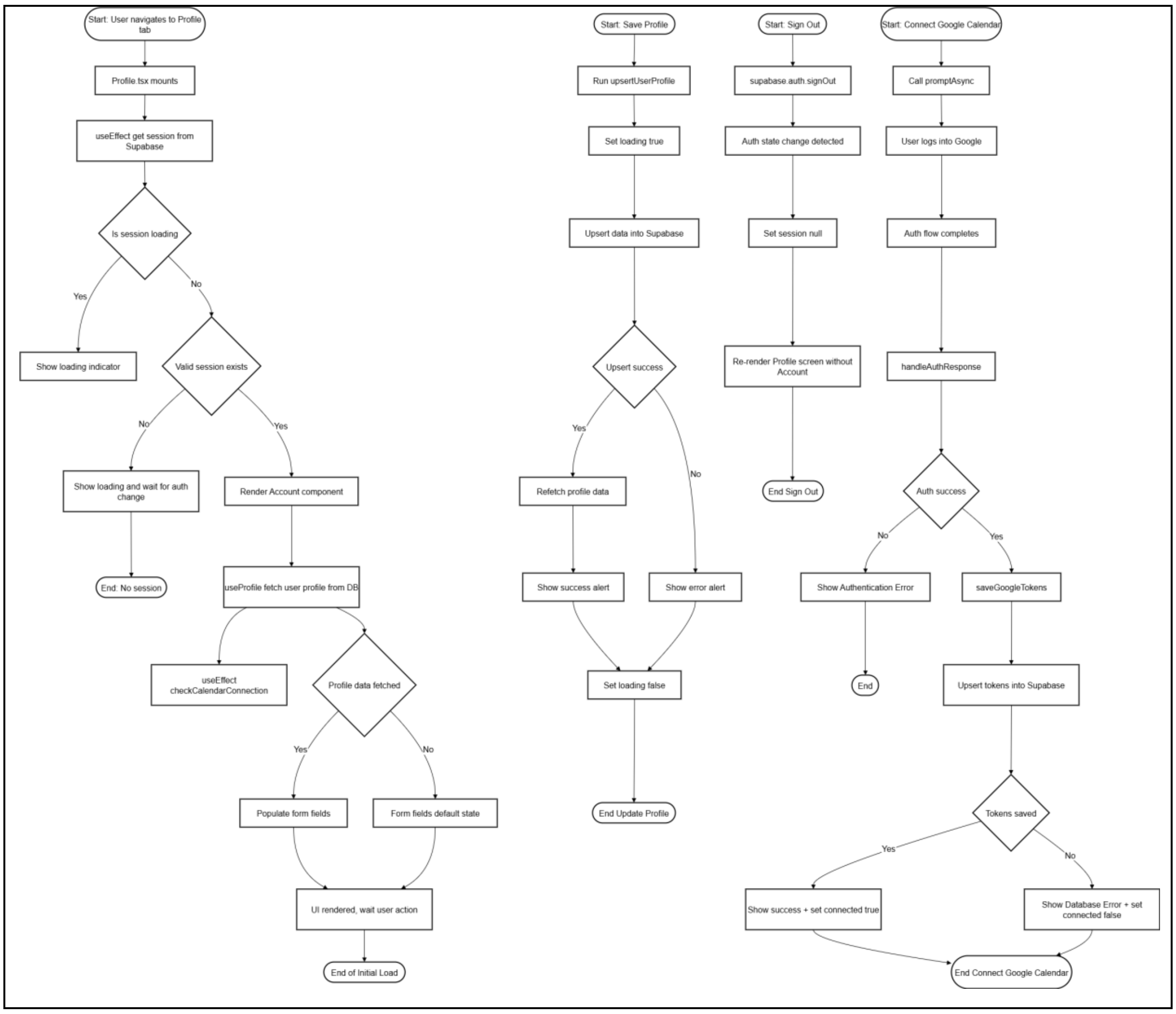


Figure 4.2.2: Profile page, profile.tsx

a) Initial Screen and Data Loading

When a user navigates to the profile tab, the application first checks for an active user session with Supabase. If a valid session exists, the Account component is rendered. Otherwise, the screen waits for a change in authentication state. This is for secure purpose since if not authenticated, the user will navigate to sign up or sign in. After successful, system will uses the useProfile hook to fetch the user's detailed informationsuch as their name, goal, and fitness level from the user_profile_new table in the database The screen will display the field with user information.

b) Update profile, Sign out and Connect to Google Calendar

There are three main actions the user can perform on this screen. First, in order to update their profile, the user is able to modify the fields of the form and press "Save Profile." This executes the upsertUserProfile function, which puts the new data into the user_profile_new table. On a successful update, the profile details are refreshed so that the UI reflects current data, and a success alert is given to the user. Second, the user can connect their Google Calendar by pressing the appropriate button. This executes Google authentication flow. In the case that the user grants the permissions, authentication tokens are acquired and are added to the user_google_tokens table via an upsert command. An alert is given to the user stating that the integration was successful and the button is updated in appearance. Third, the user is able to logout via the "Sign Out" button, which executes the supabase.auth.signOut() command. This is caught by an authentication state listener, who clears out the session and removes the Account component from the screen. This will leads to navigate to the sign in page as no active user session

4.2.3 schedule page, schedule.tsx

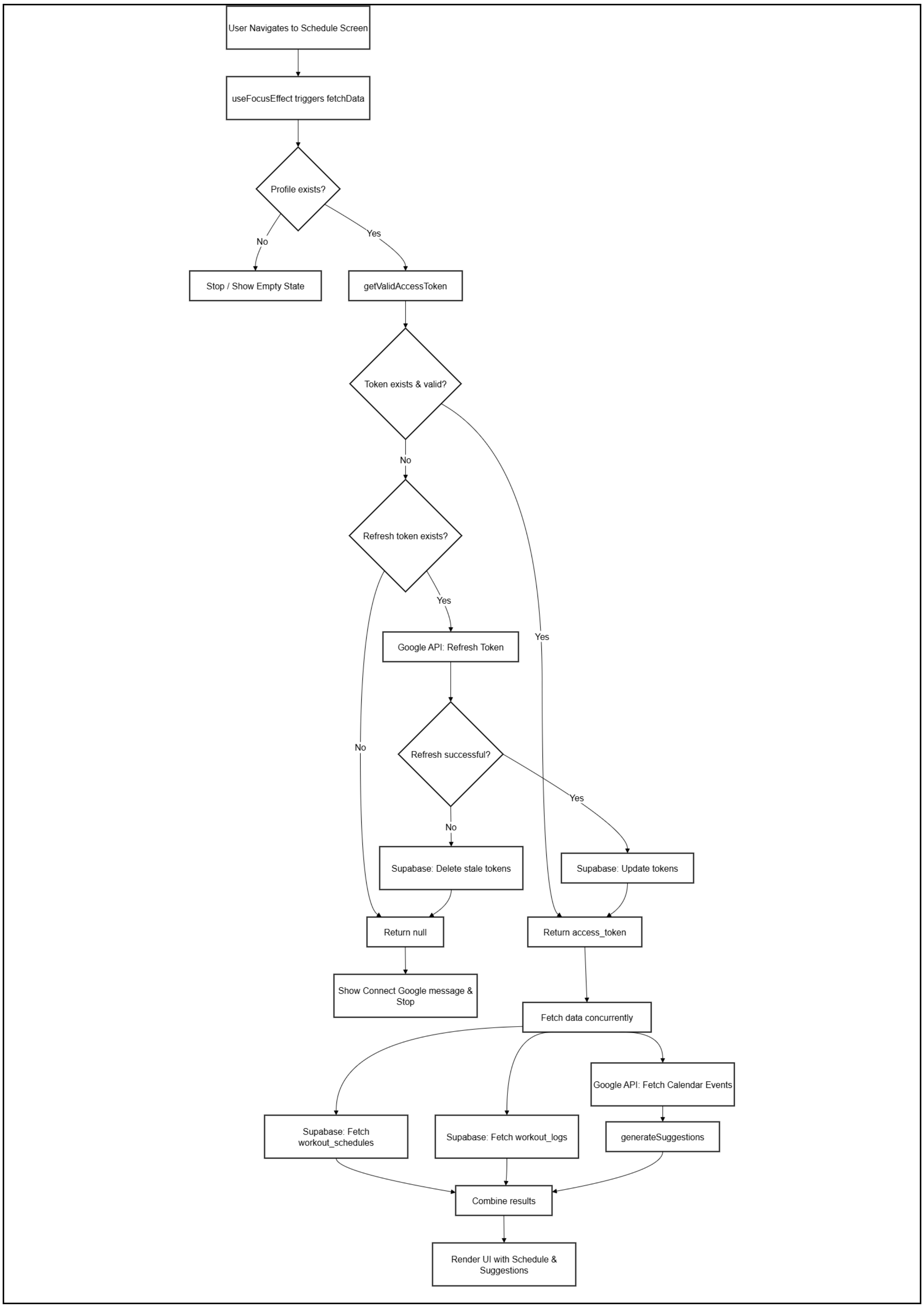

User Navigates to Schedule Screen
useFocusEffect triggers fetchData
Profile exists?
No
Yes
Stop / Show Empty State
getValidAccessToken
Token exists & valid?
No
Yes
Refresh token exists?
No
Yes
Google API: Refresh Token
Refresh successful?
No
Yes
Supabase: Delete stale tokens
Supabase: Update tokens
Return null
Return access_token
Show Connect Google message & Stop
Fetch data concurrently
Google API: Fetch Calendar Events
Supabase: Fetch workout_schedules
Supabase: Fetch workout_logs
generateSuggestions
Combine results
Render UI with Schedule & Suggestions

Figure 4.2.3.1: schedule page, schedule.tsx, Data Loading and Suggestion Generation

Data Loading and Suggestion Generation

The schedule screen is used for managing a user's weekly workout plan, suggest a scheduling system integrated with their Google Calendar. When the user navigates to the schedule page, the useFocusEffect hook to trigger the main fetchData function. It validates the user's profile and then checks for a valid Google Calendar access token with the getValidAccessToken helper. If the token is expired but a refresh token exists, a new access token is requested from Google's OAuth service and the user's record in the Supabase database is updated. If authorization fails, the token will be removed and the user is prompted to connect their Google account. After validation of the token, it can proceed with fetching all existing workout_schedules from Supabase, retrieves the user's workout_logs to cross-check completed sessions, and calls the Google Calendar API to obtain events scheduled for the upcoming week concurrently. This can optimize loading time.

Then data preprocessing is undergone by analyzing calendar events that mark "busy" slots, and adding buffer time which is 30 minutes before and after the event, and the remaining availability is used to generate potential workout times based on the user's saved preferences workout time (e.g., "morning" or "evening"). These suggestions help to filter out any already confirmed slots in the calendar event. At the same time, past schedules are compared against workout logs to classify them as "Completed" or "Missed." Last, upcoming workouts, new suggestions, and recent history are rendered on the screen.

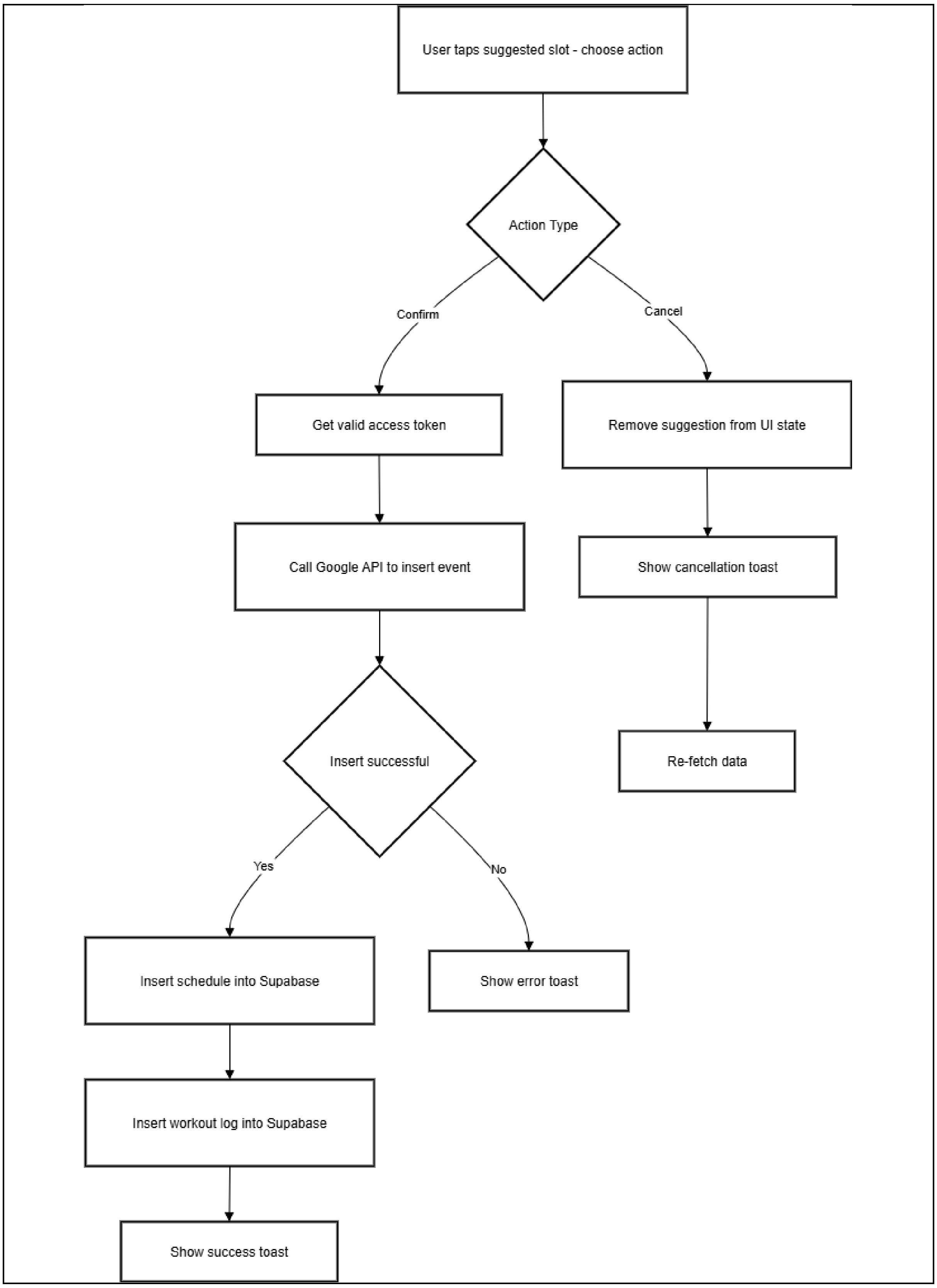

User taps suggested slot - choose action
Action Type
Confirm
Cancel
Get valid access token
Remove suggestion from UI state
Call Google API to insert event
Show cancellation toast
Insert successful
Re-fetch data
Yes
No
Insert schedule into Supabase
Show error toast
Insert workout log into Supabase
Show success toast

Figure 4.2.3.2: schedule page, schedule.tsx, Confirm and Cancel schedule

Confirm and Cancel schedule

For Confirming a Workout, user selects “Confirm” on a suggestion, the handleConfirmWorkout function runs. It first checks whether the user has reached their configured workouts_per_week limit, which is set at the profile page. If not, it performs two actions. First, it calls Expo Notifications to schedule a local push notification reminding the user at the workout’s start time, storing its unique ID. Next, it inserts a new record into the workout_schedules table in Supabase with the user ID, start/end times, and notification_id. The data is then re-fetched so the new workout appears in the UI.

For Canceling a Workout, user taps “Cancel” on a previously confirmed workout. The handleCancelWorkout function cancels the scheduled push notification using the stored notification_id via Expo Notifications, and it deletes the corresponding record from the workout_schedules table in Supabase. After success, the screen’s data is re-fetched to update the UI.

4.2.4 log page, log.tsx

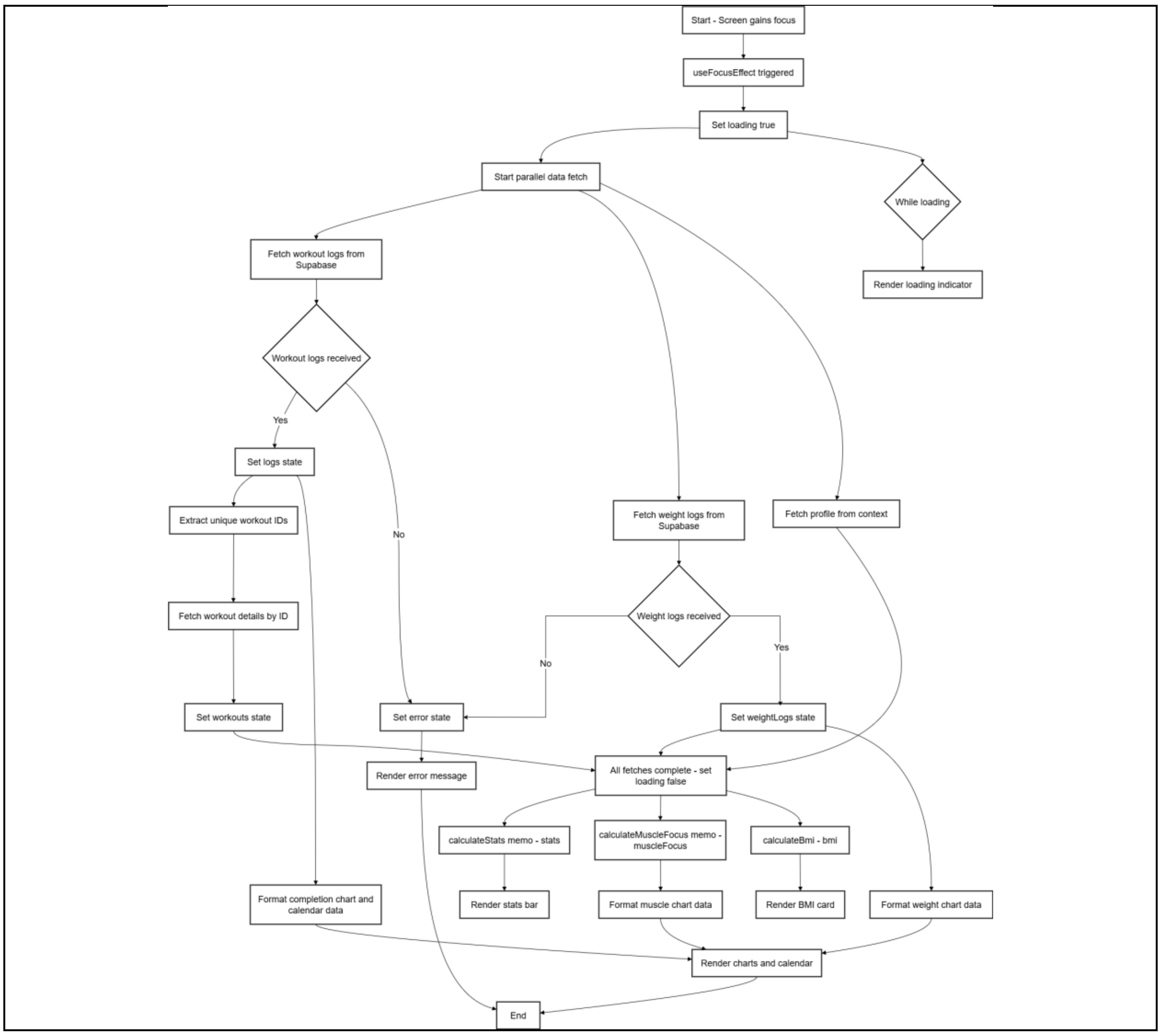

Start - Screen gains focus
useFocusEffect triggered
Set loading true
Start parallel data fetch
While loading
Render loading indicator
Fetch workout logs from Supabase
Workout logs received
Yes
No
Set logs state
Extract unique workout IDs
Fetch workout details by ID
Set workouts state
Set error state
Render error message
Fetch weight logs from Supabase
Weight logs received
No
Yes
Set weightLogs state
Fetch profile from context
All fetches complete - set loading false
calculateStats memo - stats
calculateMuscleFocus memo - muscleFocus
calculateBmi - bmi
Render stats bar
Format muscle chart data
Render BMI card
Format weight chart data
Format completion chart and calendar data
Render charts and calendar
End

Figure 4.2.4: log page, log.tsx

The Log Screen is the user's exercise dashboard. It is used workout logs, weight history, and user profile data, sorting it all out into meaningful stats and trends, and displaying it back with a set of summary cards and graphs.

When user navigate to log page, it triggers a data-fetching to a Supabase backend and simultaneously loads all the weight logs and workout logs of the user. Then, it loads the actual details for each exercise, such as main muscles worked, using the IDs of the workout logs. Meanwhile, weight and height is loaded from the ProfileContext which is the provider that store user profile data for entire system.

After all the needed data is pulled, the component will process the data by doing some calculations for the dashboard. The calculateStats method from *log-helpers.ts* makes an assessment of the logs depending on the number of workouts achieved in the current month, the number of weeks the customer worked out in the past, and their completion percentage on average. The calculateMuscleFocus method counts the distribution of muscle groups the customer worked out the most. The user's BMI is calculated based on the latest weight and height from their profile. The data is then formatted to display in the charts, and then the UI is rendered, presenting the customer with a detailed and easy-to-understand analysis of their progress.

4.2.5 Recommendation and Feedback in FastAPI

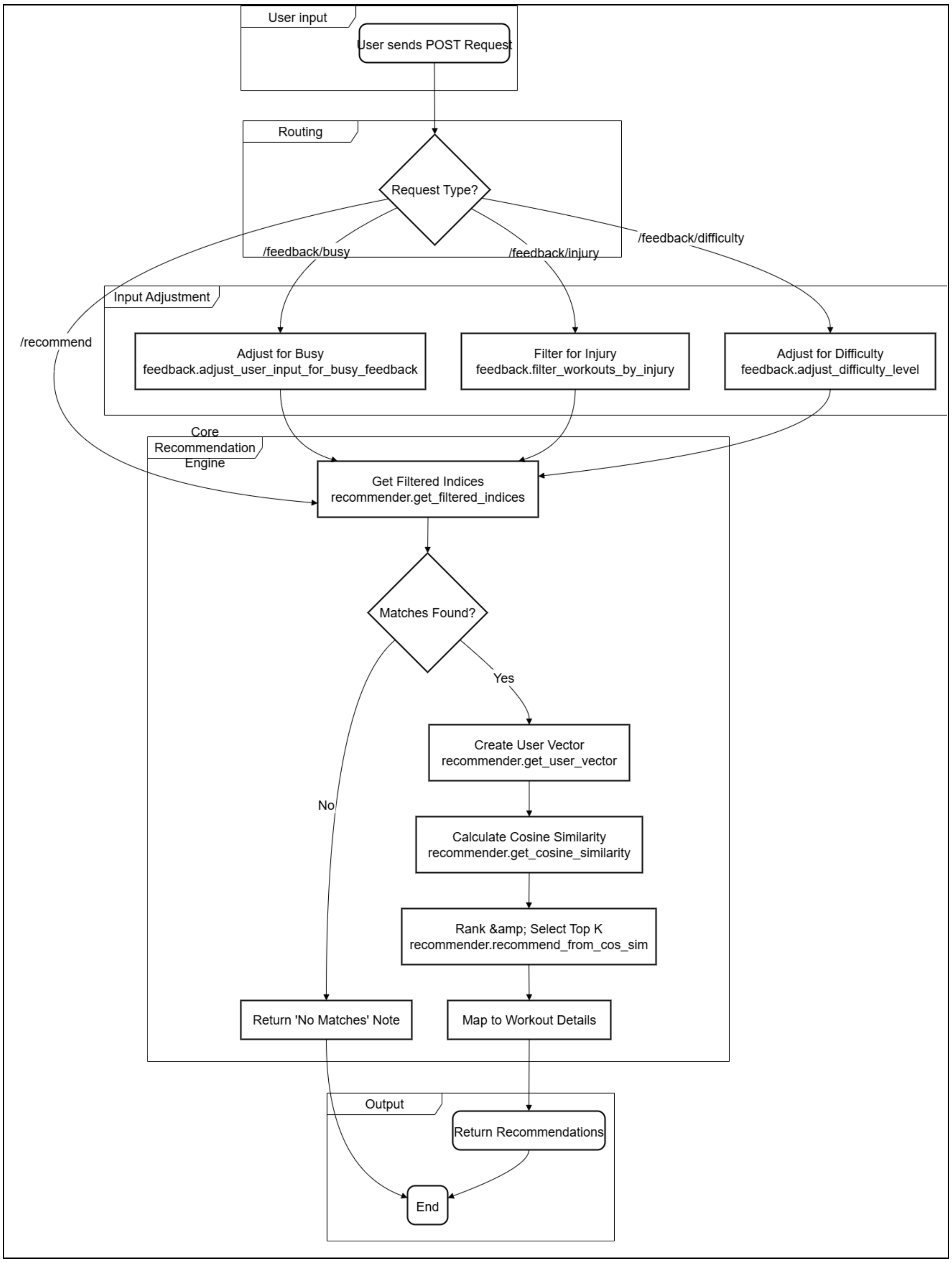


Figure 4.2.5.1: Recommendation and feedback injury, difficult, busy

This flowchart shows how most of the feedback routes act as an up-front "input adjustment" step before feeding into a common, core recommendation engine. The "Favorite" and "Bored" feedback routes each have special logic and are represented separately.

The process starts when a request is received from the user's HTTP request. For standard recommendations or feedback related to being busy, injured, and difficult, the system first adjusts the input. For a busy user, the request is modified so that the 'hiit' category is prioritized. For an injury, a pre-filtered list of safe workouts is created first by mapping the injury part with the primary and secondary muscles of the workouts, and then exclude those affected workouts, and then other preferences are applied. If a workout is too difficult, it lowers the user's workout level (e.g., intermediate → beginner) if the workout is marked too hard or after three consecutive fails. If it is already the lowest level, then remain that level (beginner). Once the input is prepared, it is passed into the Core Recommendation Engine.

This engine start with the get_filtered_indices function in recommender.py processes the user's filters. For each filter, such as level='beginner', it creates the matching column name (e.g., level_beginner), which exists in the cat_df_encoded.csv file. It then builds a boolean mask to select only the rows where all specified filter columns equal 1 and returns the integer indices of the corresponding workouts.

The next step is to create user vector. The get_user_vector function takes these indices to extract the relevant workout vectors (rows) from the full final_matrix. It calculates the column-wise mean of these vectors, producing a single averaged vector that numerically represents the user's taste based on the filtered workouts.

Next, it will be chosen as the seed base for computing cosine similarity. The get_cosine_similarity function performs the core mathematical operation. Using *sklearn.metrics.pairwise.cosine_similarity*, it efficiently computes the similarity between the single seed user vector and every workout vector in the final_matrix. The result is an array of scores (ranging from 0 to 1), where each score indicates how closely a workout aligns with the user's needs.

Finally, the recommend_from_cos_sim takes this array of scores and applies argsort() to identify the indices of the highest scores, ranking all workouts from most to least similar. It selects the top *5* indices, retrieves their corresponding workout_id and name from the mapping DataFrame, and formats the data into a JSON list to send back to the client.

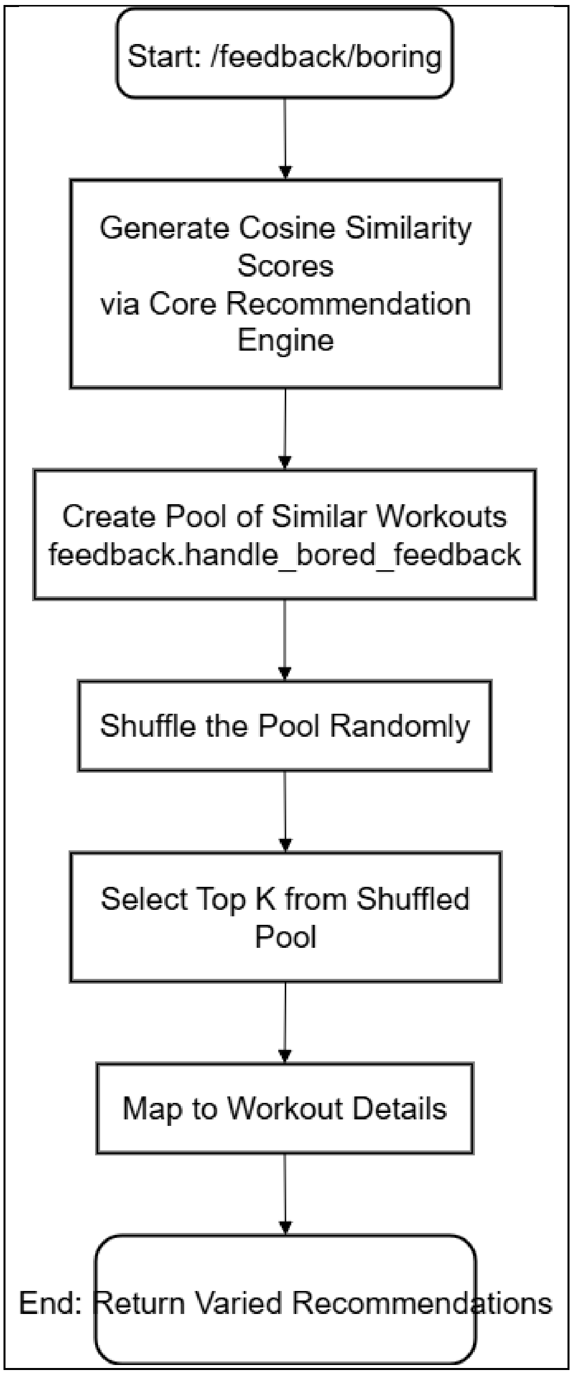

Start: /feedback/boring
Generate Cosine Similarity Scores via Core Recommendation Engine
Create Pool of Similar Workouts feedback.handle_bored_feedback
Shuffle the Pool Randomly
Select Top K from Shuffled Pool
Map to Workout Details
End: Return Varied Recommendations

Figure 4.2.5.2: feedback boring

For feedback boring, to introduce variety, handle_bored_feedback in feedback.py first retrieves the similarity scores from the core engine. It then loops through these scores and creates a new list containing only workouts that score above the defined similarity_threshold, 0.6. Next, it uses Python's random.shuffle() function to randomize the order of this pool of sufficiently similar workouts before selecting the top k, which is 5 as default. This ensures the user still gets relevant recommendations but not always the same top results. The rest will be similar to the normal flow.

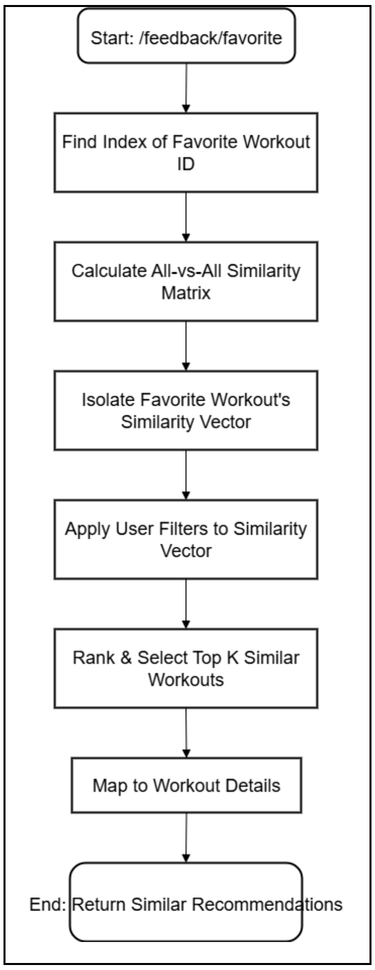

Start: /feedback/favorite
Find Index of Favorite Workout ID
Calculate All-vs-All Similarity Matrix
Isolate Favorite Workout's Similarity Vector
Apply User Filters to Similarity Vector
Rank & Select Top K Similar Workouts
Map to Workout Details
End: Return Similar Recommendations

Figure 4.2.5.3: feedback favorite

For feedback favorite, this finds workouts similar to one specific item within their user input needs. It begins by searching the mapping DataFrame for the numerical row index that matches the user's favorite_workout_id and its corresponding encoded matrix. Instead of building a user preference vector as a seed, it computes a full similarity matrix of every workout compared to every other workout by "cosine_similarity(recommender.final_matrix)". From this, it extracts the similarity scores of the favorite workout against all others. This list is then filtered to include only workouts that also satisfy the user's additional criteria (such as level and goal) before being sorted to return the top *k* matches.

4.3 Core Component diagram and relative function

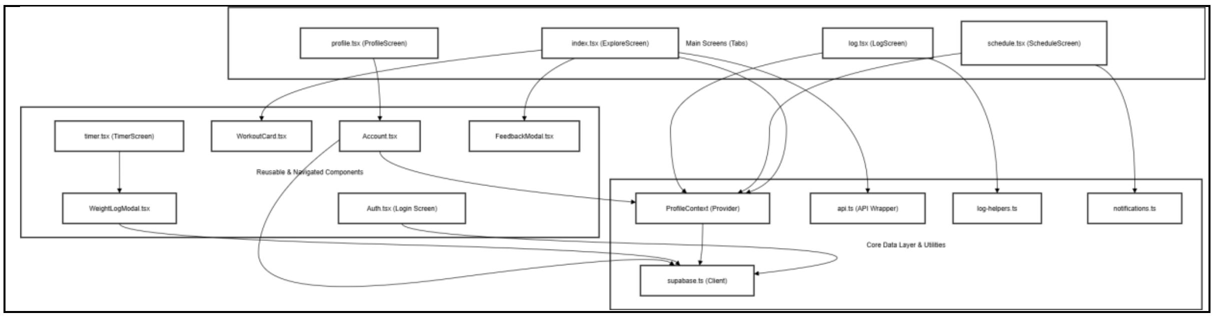


Figure 4.3.1: Core Component diagram

The figure is the component diagram of frontend. The below are the functions inside them and descriptions of each component

ProfileContext.tsx

Component for central manage the user profile date for global usage easily.

- **fetchProfile():** Retrieves the logged-in user's profile by getting the current session from Supabase Auth, querying the **user_profile_new** table using the user ID, and storing the details in state.
- **refetchProfile():** Refreshes the profile data by calling **fetchProfile()** again, ensuring changes like profile updates or favorite workouts are reflected immediately.

index.tsx (Workout Screen)

Component for workout screen display for display workout recommendation and viewing detail.

- **fetchAndCombineWorkouts():** Generates a personalized list of workouts by reading user details from **ProfileContext**, calling the external recommendation API, fetching full details from the Supabase **Workouts** table, and combining everything into a **DisplayWorkout[]** array.
- **checkWorkoutPerformance():** Detects when a user is struggling by checking if the last three workouts are below 50% completion and validating the **difficulty_feedback_triggered_at** timestamp before suggesting easier workouts.
- **handleToggleFavorite():** Updates the user's **favorite_workout_id** in the **user_profile_new** table and calls **refetchProfile()** to refresh the UI after setting or removing a favorite workout.
- **onStartWorkout():** Prepares the workout plan by converting recommendations into a **WorkoutStep[]** array, retrieving timing from the **HIIT_duration** table for HIIT workouts, applying default durations for others, and passing the plan as JSON to the Timer screen.

schedule.tsx (Schedule Screen)

Component for schedule page display to show and suggest workout schedule.

- **handleConfirmWorkout():** Schedules a workout by calling **schedulePushNotification()** to create a reminder and inserting the workout details with the notification ID into the **workout_schedules** table.
- **handleCancelWorkout():** Cancels a workout by removing its notification using the notification ID and deleting the corresponding entry from the **workout_schedules** table.

timer.tsx (Timer Screen)

Component for the timer page display for the user to use after starting a workout.

- **logWorkout():** Records the results of a workout session by calculating completion percentage, inserting a log into the **workout_logs** table, checking the latest weight from the profile, and prompting the user with the **WeightLogModal** to update weight.

log.tsx (Log Screen)

Component for the log page display for the user to access the dashboard for performance and useful metrics.

- **fetchLogData():** Retrieves workout logs and weight logs in parallel, fetches workout details like muscle groups, and provides the data for the "Muscle Focus" chart.
- **calculateBmi():** Computes BMI using the formula **weight_kg ÷ (height_cm/100)^2** and returns a text result such as "Normal" or "Overweight."

Account.tsx and Profile.tsx (Profile Screen)

Component for the show and update user profile data and connect to google calendar which call by the profile.tsx.

- **upsertUserProfile():** Saves user profile information by collecting form data (e.g., name, height, weight, goal) into an update object and calling Supabase's upsert to create or update the record.
- **saveGoogleTokens():** Stores Google Calendar authentication tokens by receiving access and refresh tokens after login and saving them into the **user_google_tokens** table linked to the user's ID.

Auth.tsx
This component manages user sign-in and sign-up, serving as the app's entry point for unauthenticated users.

- **signInWithEmail():** Collects the user's email and password from state and sends them to `supabase.auth.signInWithPassword` to authenticate and establish a session.
- **signUpWithEmail():** Registers a new user via `supabase.auth.signUp`. If successful but no session is created, it alerts the user to check their email for verification.

WeightLogModal.tsx
This modal appears after a workout, prompting the user to log their weight.

- **handleSaveWeight():** The core function that performs a dual-write operation:
    1. Inserts a time-stamped entry into the `weight_logs` table for history tracking.
    2. Updates the `weight_kg` field in the `user_profile_new` table, ensuring the app always has the most recent weight

FeedbackModal.tsx
This modal centralizes user feedback on workout plans.

- **handleFavoriteSubmit():** Instead of being disabled, the button now checks if a `favorite_workout_id` exists in the user profile. If not, it shows an alert guiding the user to set a favorite.
- **handleInjurySubmit():** Cleans up the comma-separated input and passes it as an array of strings to the `onSubmit` handler

Utility & Helper Files

1. **supabase.ts**: Initializes and configures the Supabase client using project URL and anon key from environment variables. It sets up `AsyncStorage` for mobile session persistence and exports a ready-to-use `supabase` object for database and auth interactions.
2. **api.ts**: A centralized wrapper for the FastAPI backend. Its `post` function standardizes POST requests—handling headers, body serialization, and error checking—keeping UI code clean.
3. **log-helpers.ts**: Extracts business logic from UI. Includes pure functions like `calculateStats` (monthly workouts, weekly streak) and `calculateMuscleFocus`, which transform raw logs into structured data for charts and stats.
4. **notifications.ts**: Manages push notification setup. `registerForPushNotificationsAsync` abstracts platform-specific details, including permissions and Android channel configuration, to ensure consistent behavior.

Components and Dependencies

Component	**Depends On**
Account.tsx	ProfileContext.tsx, supabase.ts
api.ts	None (Independent)
Auth.tsx	supabase.ts
ExploreScreen (`index.tsx`)	FeedbackModal.tsx, WorkoutCard.tsx, api.ts, ProfileContext.tsx, supabase.ts
FeedbackModal.tsx	ProfileContext.tsx
log-helpers.ts	None (Independent)
LogScreen (`log.tsx`)	log-helpers.ts, ProfileContext.tsx, supabase.ts
notifications.ts	None (Independent)
Profile.tsx (the screen)	Account.tsx, supabase.ts
ProfileContext.tsx	supabase.ts
ScheduleScreen (`schedule.tsx`)	ProfileContext.tsx, supabase.ts, notifications.ts
supabase.ts	None (Independent)
TimerScreen (`timer.tsx`)	WeightLogModal.tsx, supabase.ts
WeightLogModal.tsx	supabase.ts
WorkoutCard.tsx	None (Independent)
WorkoutDetailScreen.tsx	None (Independent)

Table 4.3.2: Table of Components and Dependencies

This table shows the dependency among the components in the react native frontend app.

4.4 Supabase table design and Data schema

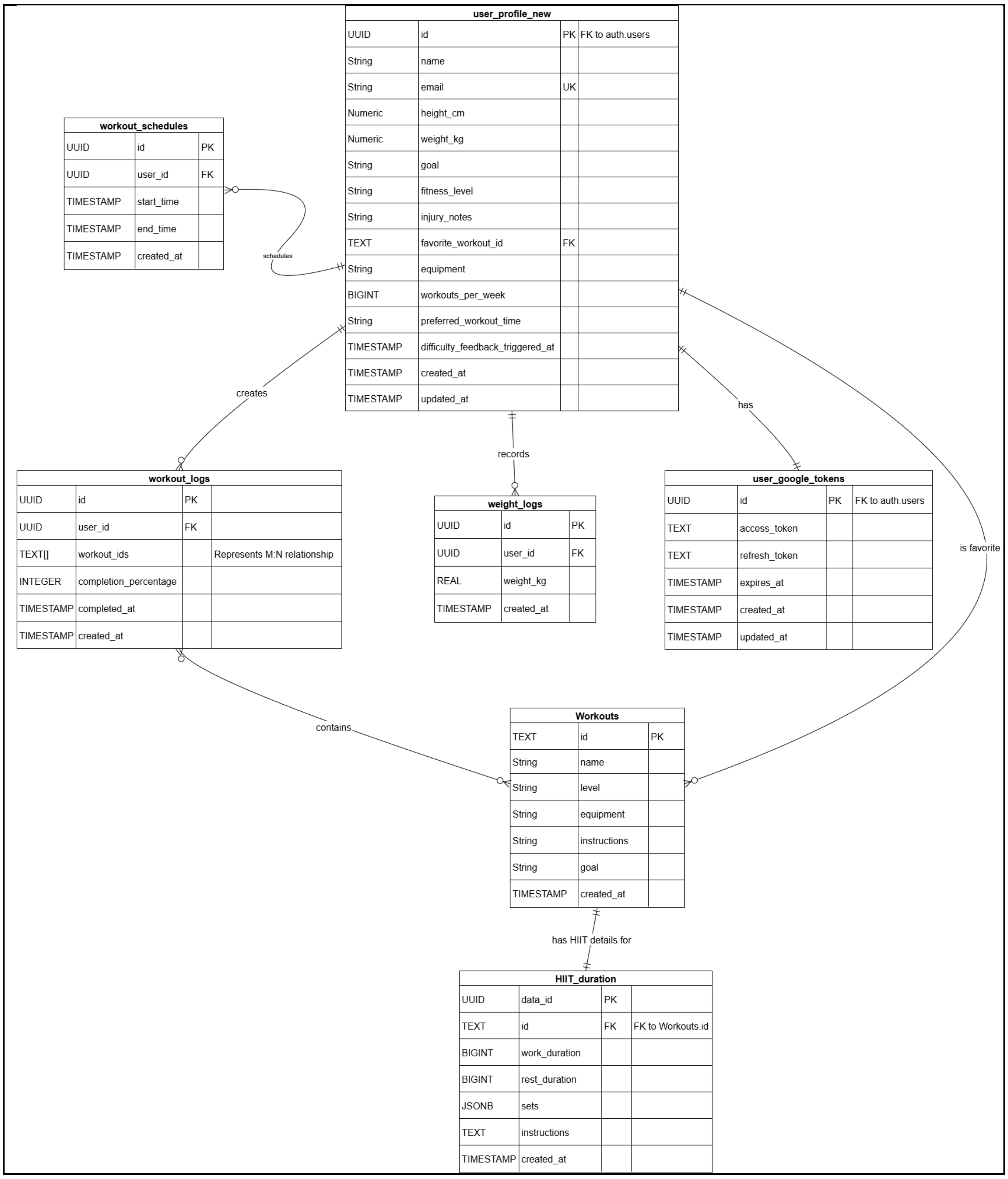


Figure 4.4: ERD diagram for Supabase tables

This Entity-Relationship Diagram (ERD) illustrates the database table needed for storing user data to provide full services. The **user_profile_new** table, which acts as the main of the system. It stores basic user identification details like name and email and key fitness-related attributes such as `height_cm`, `weight_kg`, personal goals (e.g., “fat loss”), and current `fitness_level`. It directly integrates with the authentication system using supabase Auth, with its primary key `id` also serving as a foreign key to `auth.users`, ensuring both data integrity and security.

From this central profile table, several one-to-many relationships extend to track different aspects of user activity. The workout_schedules table manages planned fitness sessions ,store notification_id, linking each schedule to the user through `user_id`. The workout_logs table records completed workouts, capturing metrics like `completion_percentage` and the `completed_at` timestamp. Weight over time is tracked via the weight_logs table, which stores historical weight data. Together, these relationships allow the application to build a start-to-end workout service for each user's engagement and progress.

The diagram also includes one-to-one relationships for specialized data. Each user profile is associated with a single record in the **user_google_tokens** table, which securely manages API tokens from Google Calendar integration. Apart from that, the **Workouts** table, serving as the catalog of all available exercises, is linked one-to-one with the **HIIT_duration** table which store the HIIT workouts plan time data such as rest duration, workout duration and number of sets of workout to perform for different fitness level. This structure enhances workout data by attaching detailed timing information only to HIIT workouts to handle HIIT workout due to their high intensity and rest interval flow.

To solve the many-to-many relationship between workouts and logs , the workout_logs table use a workout_ids array field for later retrieval from the workout table. This allows a single workout log to point to multiple exercises from the Workouts table, making it easier to record sessions that include many workouts. This is also useful for creating the muscle focus chart for the log page in the frontend. This method keeps the database structure simpler while still capturing the variety of real workout routines.

Below is the SQL code for creating table and an explanation of the column data purpose.

user_profile_new

```
create table public.user_profile_new (
  id uuid not null,
  created_at timestamp with time zone not null default now(),
  updated_at timestamp with time zone not null default now(),
  name text not null,
  email text not null,
  height_cm numeric not null,
  weight_kg numeric not null,
  goal text not null default 'fat loss'::text,
  fitness_level text not null default 'beginner'::text,
  injury_notes text null,
  favorite_workout_id text null,
  equipment text null,
  workouts_per_week bigint null default '3'::bigint,
  preferred_workout_time text null default 'any'::text,
  difficulty_feedback_triggered_at timestamp with time zone null,
  constraint user_profile_new_pkey primary key (id),
  constraint user_profile_new_email_key unique (email),
  constraint user_profile_new_id_fkey foreign KEY (id) references auth.users (id) on delete
CASCADE,
  constraint fitness_level_check check (
    (
      fitness_level = any (
        array[
          'beginner'::text,
          'intermediate'::text,
          'expert'::text
        ]
      )
    )
  ),
  constraint user_profile_new_workouts_per_week_check check ((workouts_per_week >=
0))
) TABLESPACE pg_default;
```

For the user_profile_new table, the id (UUID): The Primary Key that uniquely identifies each user profile. At the same time, it's also a Foreign Key referencing the id in auth.users, establishing a direct one-to-one connection between the profile data and the corresponding authentication record. The name, email, goal, fitness_level, injury_notes, equipment, favorite_workout_id are attributes that are used as user profile data for workout recommendation. The goal will be mapped to the workouts category in the frontend. The injury_notes and equipment are optional for physical limitations or available workout gear. The height_cm and weight_kg is for weight tracking and BMI calculation. The workouts_per_week store the user's target number of weekly workouts, with a default of 3 and preferred_workout_time for the time zone for schedule.

The difficulty_feedback_triggered_at is an attribute that store the latest feedback difficult timestamp to prevent duplication, lowering fitness level before the user starts a new workout. For the RLS, the authenticated users can view and write their record.

workout_schedules

```
create table public.workout_schedules (
  id uuid not null default gen_random_uuid (),
  user_id uuid not null,
  start_time timestamp with time zone not null,
  end_time timestamp with time zone not null,
  created_at timestamp with time zone not null default now(),
  notification_id text null,
  constraint workout_schedules_pkey primary key (id),
  constraint workout_schedules_user_id_fkey foreign KEY (user_id) references auth.users
(id) on delete CASCADE
) TABLESPACE pg_default;
```

For the workout_schedules, user_id (UUID) column, which is a Foreign Key that links the record back to the user who created it. The start_time and end_time are the schedule time zone and notification_id is to store the schedule local notification in the frontend device. For the RLS, the authenticated users can view, delete, and write their record.

workout_logs

```
create table public.workout_logs (
  id uuid not null default gen_random_uuid (),
  user_id uuid not null,
  completion_percentage real not null,
  completed_at timestamp with time zone not null default now(),
  workout_ids text[] not null,
  constraint workout_logs_pkey primary key (id),
  constraint workout_logs_user_id_fkey foreign KEY (user_id) references auth.users (id)
on delete CASCADE,
  constraint workout_logs_completion_percentage_check check (
    (
      (completion_percentage >= (0)::double precision)
      and (completion_percentage <= (100)::double precision)
    )
  )
) TABLESPACE pg_default;
```

For the workout_logs, similarly user_id (UUID) column is for retrieve how created and authentication. The completion_percentage is record the completion percentage of the workout progress from 0-100 and a completed_at is very important as checking and proof for the is that user has completed the workout during the scheduled time in the schedule page in the app. The workout_ids (an array of TEXT) is to store workout ids that the user takes during the workout session.

For the RLS, the authenticated users can view and write their record.

Workouts

```
create table public."Workouts" (
  name text not null,
  force text null,
  level text null,
  mechanic text null,
  equipment text null,
  primarymuscles jsonb null,
  secondarymuscles jsonb null,
  instructions text null,
  category text null,
  images text null,
  id text not null,
  musclesaffected text null,
  goal text not null,
  constraint Workouts_pkey primary key (id),
  constraint Workouts_id_key unique (id)
) TABLESPACE pg_default;
```

For Workouts table, it store name, level, equipment, instructions, goal, image, category, primary and secondary muscles to display to the user for more detail information.

For the RLS, the authenticated users can view all the record.

HIIT_duration

```
create table public."HIIT_duration" (
  data_id uuid not null default gen_random_uuid (),
  created_at timestamp with time zone not null default now(),
  id text not null,
  work_duration bigint null,
  rest_duration bigint null default '120'::bigint,
  sets jsonb null default '{"expert": 7, "beginner": 3, "intermediate": 5}'::jsonb,
  instructions text null,
  constraint HIIT_duration_pkey primary key (data_id),
  constraint HIIT_duration_id_key unique (id),
  constraint HIIT_duration_id_fkey foreign KEY (id) references "Workouts" (id) on delete
CASCADE
) TABLESPACE pg_default;
```

For HIIT_duration table, it is to store specific timing details (work_duration , rest_duration , sets) that are only relevant for HIIT workouts, avoiding having many null for others workout category in Workouts table. This is used for the create a workout plan for timer components in app. Both work_duration and rest_duration are stored as unit of second. A FOREIGN KEY constraint points to Workouts(id), enforcing a strict one-to-one relationship. This ensures that each HIIT entry corresponds to exactly one workout, and each workout can have at most one associated set of HIIT details that match with Wrokouts table.

For the RLS, the authenticated users can view all record.

user_google_tokens

```
create table public.user_google_tokens (
  id uuid not null,
  created_at timestamp with time zone not null default now(),
  updated_at timestamp with time zone not null default now(),
  access_token text not null,
  refresh_token text null,
  expires_at timestamp with time zone not null,
  constraint user_google_tokens_pkey primary key (id),
  constraint user_google_tokens_id_fkey foreign KEY (id) references auth.users (id) on
delete CASCADE
) TABLESPACE pg_default;
```

For user_google_tokens table, id field (UUID) serves as the primary key while also acting as a foreign key to auth.users(id), creating a one-to-one connection between each user and their stored tokens. The table holds two key credentials, access_token and refresh_token, which are used to interact with the Google API, along with an expires_at timestamp that specifies when the access_token will expire and require renewal. To protect sensitive data, the foreign key on id is defined with ON DELETE CASCADE, ensuring that a user's tokens are automatically removed if their account is deleted.

For the RLS, the authenticated users can view, delete and write their record.

weight_logs

```
create table public.weight_logs (
  id uuid not null default gen_random_uuid (),
  user_id uuid not null,
  weight_kg real not null,
  created_at timestamp with time zone not null default now(),
  constraint weight_logs_pkey primary key (id),
  constraint weight_logs_user_id_fkey foreign KEY (user_id) references auth.users (id) on
delete CASCADE,
  constraint weight_logs_weight_kg_check check ((weight_kg >= (0)::double precision))
) TABLESPACE pg_default;
```

For the weight_logs, it store the weight record in kg of the user with the created_at timestamp for tracking the weight changes using the app. The constraint that the weight should be positive is checked to prevent invalid data.

For the RLS, the authenticated users can view and write their record

4.5 Data Flow diagram

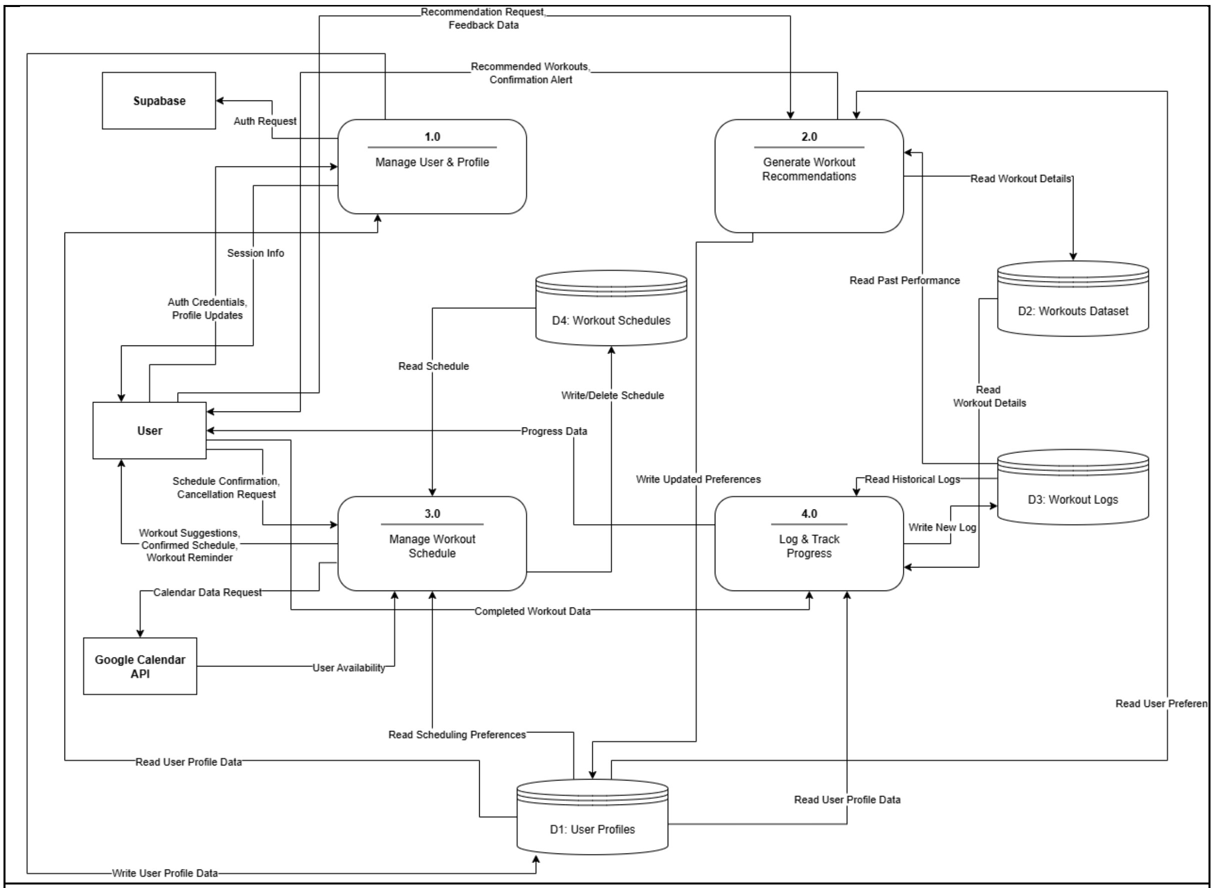


Figure 4.5 data flow diagram

Data Flow for User Management (Process 1.0)

The user management process starts when the user provides authentication credentials and profile updates, which is to 1.0 Manage User & Profile process. In this process, to verify the credentials, the process sends an Auth Request to the external Supabase Auth entity. After verification, the system produces a Session Info output, such as an authentication token, which is returned to the user to grant access to the application. After that System interacts with the D1 User Profiles data store, which is a Supabase database, involving both read and write operations to ensure accurate profile management for user update and profile management.

Data Flow for Recommendations (Process 2.0)

The recommendation engine is initiated when the user submits either a Recommendation Request or Feedback Data to the 2.0 Generate Workout Recommendations process. This process packs information from multiple sources, User Preferences are retrieved from D1 User Profiles as recommendation input, Workout Details are obtained from the D2 Workouts Dataset, and Past Workout Performance is read from the D3 Workout Logs for checking failure for difficulty adjustment. In cases where recommendations lead to changes in the user's profile, an Updated Preferences data flow is written back into D1 User Profiles. The final output is a structured list of Recommended Workouts, which flows back to the user for display.

Data Flow for Scheduling (Process 3.0)

The scheduling process is triggered when the user submits a Schedule Confirmation or Cancellation Request to the 3.0 Manage Workout Schedule process. This interaction prompts a Calendar Data Request to the Google Calendar API, which responds with the user's Availability data. At the same time, the process references Scheduling Preferences from D1 User Profiles and manages Workout Schedule Entries through read, write, and delete operations in the D4 Workout Schedules data store. As outputs, the user receives Workout Suggestions, a Confirmed Schedule, and Workout Reminders in the form of push notifications.

Data Flow for Progress Logging (Process 4.0)

The progress logging process begins when Completed Workout Data from the user's session flows into the 4.0 Log & Track Progress process. This information is transformed into a structured log entry and written into the D3 Workout Logs data store. To generate meaningful insights, the process performs additional read operations which is retrieving Historical Logs from D3, Workout Details from D2, and User Profile Data from D1. These datasets are combined and analyzed to create a comprehensive Progress Data output, which is delivered back to the user in the form of charts and statistics.

5.1 Software Setup

1. Development Environment

This section covers the foundational tools required to run and develop your project.

- Runtime Environment: Node.js
- Programming language frontend: JavaScript and Typescript
- Package Manager: npm for managing project dependencies.
- Mobile Development Tool: Expo, and using the Expo Go app for testing.
- Python Version: Python 3.12.10 for the backend API.
- Code Editor: Visual Studio Code (or a similar modern editor)

2. Frontend (Client-Side) Setup

This details the technologies and libraries used to build your mobile application.

- Core Framework: React Native with Expo was used to build the cross-platform mobile application.
- Key Libraries & Packages:
  - Navigation: Expo Router for file-based routing and navigation between screens.
  - UI Components: React Native Paper.
  - Backend Communication: The Supabase Client Library (@supabase/supabase-js) is used for all interactions with the Supabase database and authentication services.
  - Notifications: Expo Notifications for local push notifications for workout reminders.
  - Data Visualization: react-native-chart-kit and react-native-calendars for displaying user progress charts and the workout calendar on the log screen.
  - External Authentication: expo-auth-session is used to handle the OAuth 2.0 flow for connecting with Google Calendar.

3. Backend (Server-Side) Setup

This section lists the components that make up your recommendation API.

- Core Framework: Python with the FastAPI framework for quick and debug-friendly.
- Key Libraries & Packages:
  - Data Validation: Pydantic for data validation for HTTP requests.
  - Recommendation Engine: scikit-learn, pandas, and NumPy for computing cosine similarity for returning top 5 workouts similar to user needs.
  - Model Persistence: Joblib and for vectorizer for encoder to create model artifacts into the application.
  - Sparse Matrix Handling: SciPy for efficiently load and handle the sparse matrix (final_matrix.npz).

4. Database & External Services

This covers the third-party platforms your application relies on to function.

- Database & Authentication: Supabase (using a PostgreSQL database) serves as the primary backend-as-a-service platform.
- Calendar Integration: The Google Calendar API is used to fetch a user's calendar events, allowing the app to suggest free time slots for workouts.

5.2 Setting and Configuration

5.2.1 Machine learning model implementation

Before starting development the App, machine learning model part is built in a Jupyter notebook.

5.2.1.1 Dataset Collection

Before starting to develop the workout recommendation system, 4 datasets were collected. Below shows the datasets and their column:

There are 4 datasets is used:

1) User Exercise Dataset (age, gender, weight, height, BPM stats, session duration, calories burned, workout type, fat percentage, water intake, workout frequency, experience level, BMI). [27]
2) Workout Dataset (name, force, level, mechanic, equipment, primary/secondary muscles, instructions, category, images, ID). [28]
3) Cardio Dataset (name, force, level, mechanic, equipment, primary/secondary muscles, instructions, category, images, ID).
4) HIIT Dataset (name, force, level, mechanic, equipment, primary/secondary muscles, instructions, category, images, ID).

Additionally, two datasets generated by ChatGPT, namely Cardio and HIIT workouts, are combined with the Workout dataset to address the skewness in the workout category column. In future, more detailed workout information will be manually gathered from a fitness website to enhance the ChatGPT-generated dataset for improved robustness and accuracy.

5.2.1.2 Pre-processing and Data Cleaning

In this process, the datasets are cleaned for better performance in the later machine learning process. The null value should be filled or removed.

The following are the steps of pre-process and data cleaning process:

i. User exercise and workout dataset CSV files are imported into Jupyter notebook.

ii. Null values in both datasets are checked and the column with missing value are identified using the function `isnull().sum()`, which are force, mechanic, secondaryMuscles, and instructions columns in the workout dataset.

iii. The workout tuples with null are removed in the instructions column. The workouts without instructions are meaningless for the user.

iv. The missing values in force and mehanic columns are handled with their corresponding mode value.

v. Since a workout may not have a secondary muscle and only involve primary muscle, a muscleAffected column is created with following steps:

   a. The missing values in the secondaryMuscles columns are filled with “Primary Muscle Only”.

   b. Since secondaryMuscle are stored as multiple of muscles using delimiter comma, the column data are converted into list and intra-row duplicates (e.g., "shoulder, shoulder, lower back") are removed to prevent distraction in the later cosine similarity process. If “Primary Muscle Only” is found, an empty list is returned.

   c. The primaryMuscles is duplicated to increase weight and are combined with secondaryMuscles column to create a new column called muscleAffected.

   d. The muscleAffected column is converted from list to string with a delimiter space for later cosine similarity process. For example, duplication of primary muscle, hamstrings combined with secondary muscle, calves to get "hamstrings hamstrings calves" for muscleAffected column

5.2.1.3 Specific workout recommendation with user’s input

The planning to predict the user's suitable workout type in FYP1 is removed due to the low quality of data to predict the correct workout type. The reason will be discussed in the project challenges section. It switched to a direct Workout goal and workout type direct mapping.

In this process, the user input, such as goal, level of fitness, and equipment will be obtained. For the initial setup of the app, the system will use the rule-based workout type recommendation to specific fitness goals. Workout goal and workout type will be direct mapping. The workout type will also be included as user input for the specific workout recommendation.

The following are the steps of the specific workout recommendation with user's input process:

i. In Figure 5.2.1.3.1, the workout category or type is highly skewed as strength workout is the majority, followed by stretching. Only 14 cardio workout type and HIIT workout is not available. Figure 5.2.1.3.2 and 5.2.1.3.3 shows bar chart for the dataset category distribution. A balanced workout dataset is created.

ii. The 40 strength and 40 stretching workout tuples are randomly sampled and combined with the 40 cardio and 40 HIIT workouts generated by ChatGPT to create a new dataframe called "four_workout_merged_df".

iii. The category names are refined to lowercase, and the "stretching" category is renamed to "yoga" to keep naming consistent with the workout type in the user exercise dataset.

iv. The workout categories are mapped to specific fitness goals. HIIT maps to fat loss, isolation strength to muscle gain, cardio to endurance, yoga to flexibility, and compound strength to general fitness. These mappings are based on the typical benefits of each workout category to create goal column.

v. The rare equipments, which is below 5 in the dataset are grouped as others to reduce overfitting and better performance.

vi. Meaningless column such as id, image, primaryMuscles and secondaryMuscles are removed.

vii. The one hot encoding of pandas is applied to categorical column such as force, level, mechanic, equipment, category, goal to converting each unique category into a separate binary column.

viii. The preprocessing is applied on the instructions and muscleAffected columns with following steps and refer Table 5.2.1.3.4 for text preprocessing sample:

a. All characters are converted to lowercase to prevent duplication of semantically identical tokens with different casing.
b. The punctuation are removed to reduce noise.
c. Texts are tokenized using NLTK's word tokenizer to enable granular processing
d. Stopwords are removed to retain informative words
e. Lemmatization is applied to reduce words into their base form to prevent them from being identified as different features.
f. The tokens of words are joined with spacing to create a string.

ix. The instructions and muscleAffected columns are combined to create a new column called ‘combined_text’

x. The name of workouts with each index are recorded into a workout_names variable for reference.

xi. Term Frequency–Inverse Document Frequency (TF-IDF) vectorization was applied to the ‘combined_text’ column for converting textual data to numerical data which suitable for machine learning model. Each combined text is convert into vector that reflects the importance of each term relative to the entire corpus. This help to capture meaningful information for model.

xii. The encoded categorical binary columns and the vectorized text column are combined to create a SciPy sparse matrix called “final matrix”.

xiii. User input of goal, equipment, fitness level and workout type are collected.

xiv. The workouts that met the requirement are filtered to create a dataframe with suitable workouts called “user_input_filtered_df”

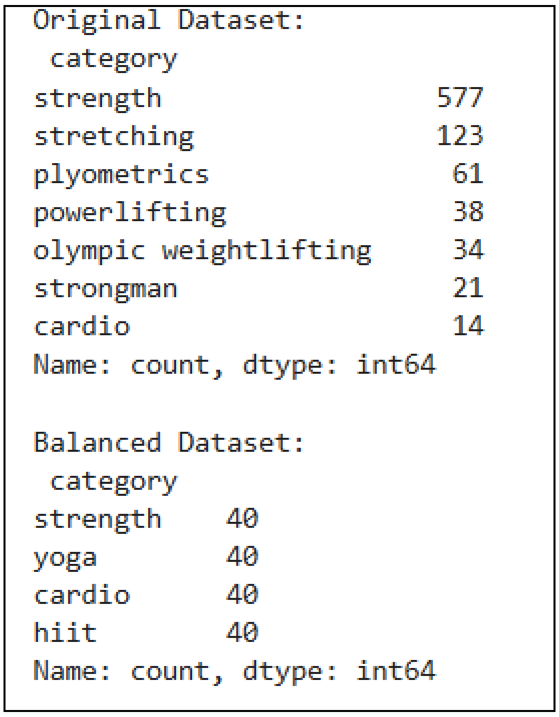
xv. The indices of the filtered workouts is mapped with the “final matrix” to get filtered workouts with the encoded and vectorized data.

xvi. If the workouts that meet the requirements are available, the first workout is selected to be suggested for initial stage.

xvii. Cosine similarity is applied to the first selected workout to compare with the rest of the filtered workouts to find similar workout.

xviii. The indices of the top 5 workouts that are similar to the first selected workout are recorded and mapped with the name. The first selected workout is included in the top 5.

xix. The top 5 workout names are printed which are shown in Figure 5.2.1.3.5



```
Original Dataset:
 category
strength                577
stretching              123
plyometrics              61
powerlifting             38
olympic weightlifting    34
strongman                21
cardio                   14
Name: count, dtype: int64

Balanced Dataset:
 category
strength    40
yoga        40
cardio      40
hiit        40
Name: count, dtype: int64
```

Figure 5.2.1.3.1 Value counts of the category in the original and balanced datasets.

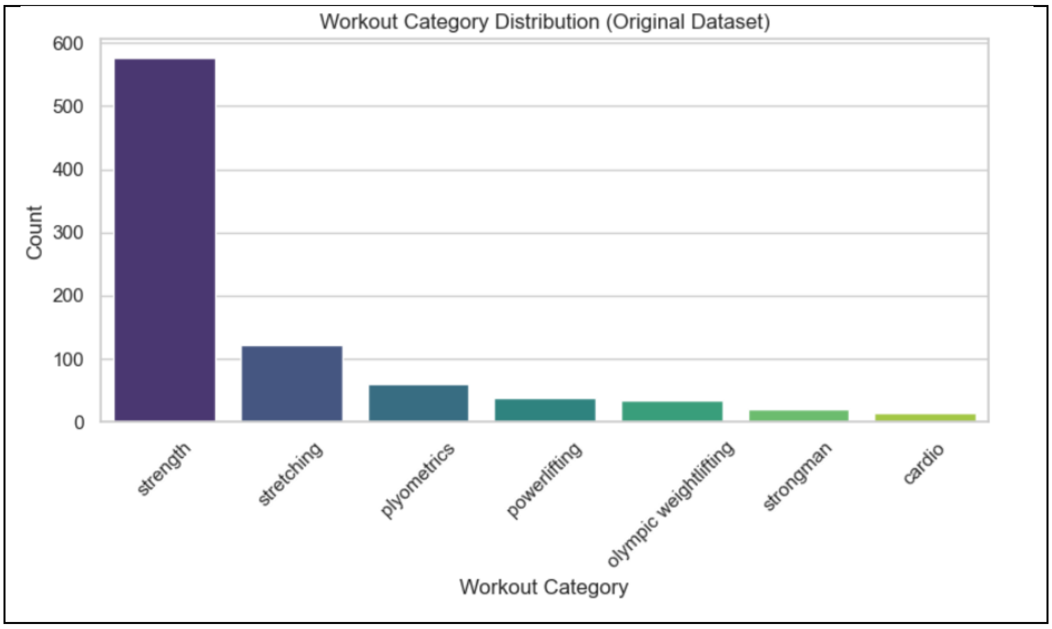


Figure 5.2.1.3.2Bar chart for original workout category distribution.

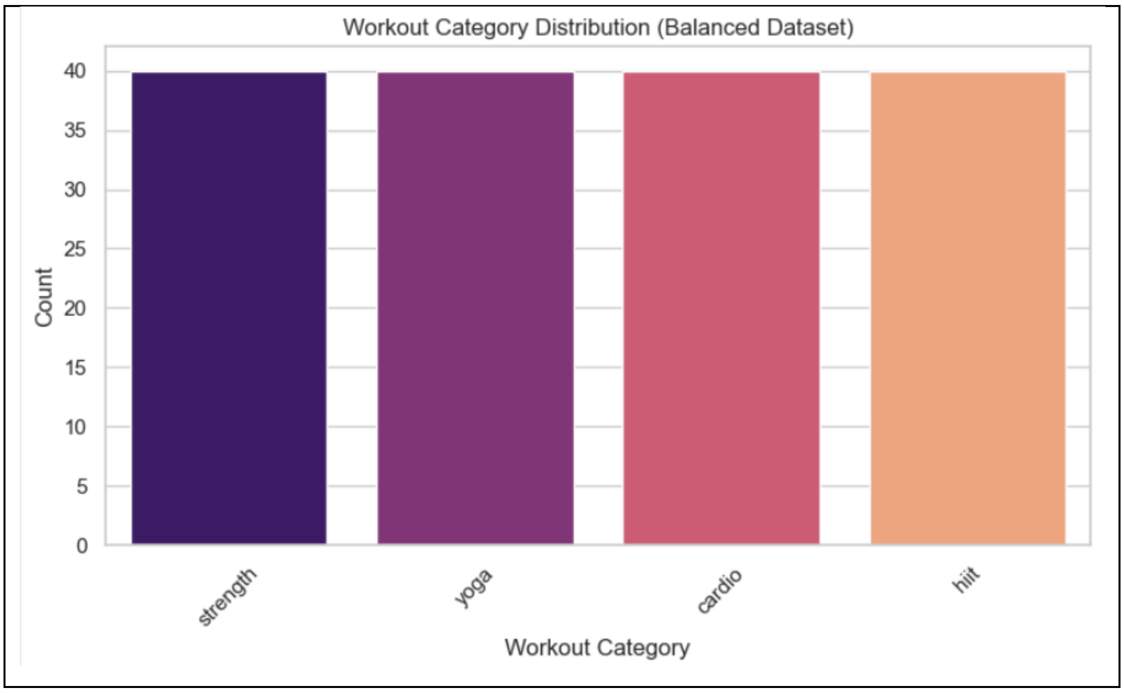


Figure 5.2.1.3.3 Bar chart for balanced workout category distribution.

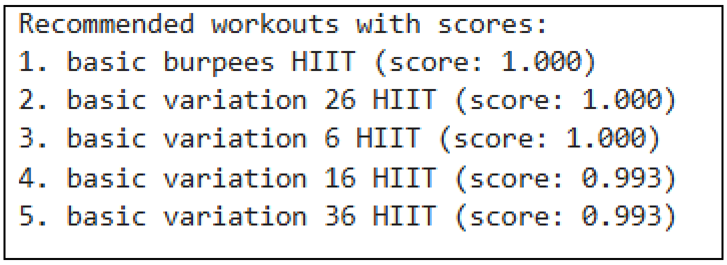
In Figure 5.2.1.3.1, the value count of the category column for the original workout dataset is shown. The strength workout consists of 577 and stretching with 123. There is only 14 cardio and no HIIT workouts. In Figure 5.2.1.3.2, the original data were skewed, with strength and stretching workouts dominating, while cardio and HIIT were underrepresented. ChatGPT generated dataset is included to balance the distribution to get a fair distribution for the later process. In Figure 5.2.1.3.3, a balanced dataset is formed in which each category consists of 40 workouts.

Field	Before	After
Instructions	"Start by having a dumbbell in each hand with y..."	"start dumbbell hand arm fully extended side us..."
muscleAffected	"Shoulders Shoulders Triceps"	"shoulder shoulder triceps"
Combined Text	-	"start dumbbell hand arm fully extended side us... shoulder shoulder triceps"

Table 5.2.1.3.4 Example of Text Preprocessing Example

Table 5.2.1.3.4 shows the before and after of textual pre-processing. In Both instruction and muscleAffected, the text is lowercased and stopwords such as “by” and “a” are removed. The punctuations are removed to reduce noise, and texts are undergo lemmatization. They are both combined to form a combined text column.

Result



```
Recommended workouts with scores:
1. basic burpees HIIT (score: 1.000)
2. basic variation 26 HIIT (score: 1.000)
3. basic variation 6 HIIT (score: 1.000)
4. basic variation 16 HIIT (score: 0.993)
5. basic variation 36 HIIT (score: 0.993)
```

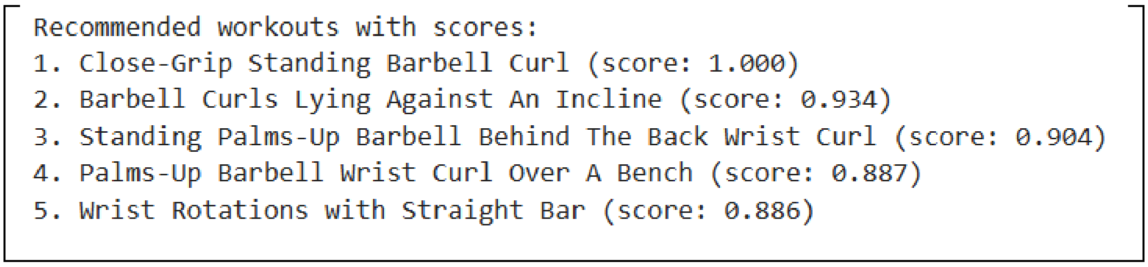
Figure 5.2.1.3.5 Top 5 workout recommendations with user input for fat loss

In Figure 5.2.1.3.5, top 5 workouts are recommended to user. The user input example is as below:

User input:

- level: intermediate
- goal: fat loss
- equipment: body only
- workout type: HIIT

For level, goal, and equipment are obtained from the user input and the workout type is from the initial rule-based logic. The displayed score serves as an indicator of how similar the workout to the chosen recommended workout. This score is only show for checking and not to the user. The closer the score to 1, the more similar to the chosen recommended workout and vice versa. A score of 1 indicates an exact match in feature vectors between the chosen recommended workout (represented as user_vector = filtered_matrix[0]) and the rest of filtered workouts. In Figure 5.2.1.3.5, the "basic burpees HIIT," "basic variation 26 HIIT," and "basic variation 6 HIIT" workouts all receive a score of 1 because "basic burpees HIIT" is the chosen recommended workout (filtered_matrix[0]). The other two workouts share identical features due to issues with the ChatGPT-generated dataset, which includes different names for the same features, resulting in the same score. This can be addressed by manually creating a more reliable dataset in the future by sourcing workout details from trusted websites. The system will recommend the top five workouts with the highest scores to help the user achieve their goal.



```
Recommended workouts with scores:
1. Close-Grip Standing Barbell Curl (score: 1.000)
2. Barbell Curls Lying Against An Incline (score: 0.934)
3. Standing Palms-Up Barbell Behind The Back Wrist Curl (score: 0.904)
4. Palms-Up Barbell Wrist Curl Over A Bench (score: 0.887)
5. Wrist Rotations with Straight Bar (score: 0.886)
```

Figure 5.2.1.3.6Top 5 workout recommendations with user input for muscle gain

In Figure 5.2.1.3.6, top 5 workouts are recommended to user for muscle gain. The user input example is as below:

User input:

- level: beginner
- goal: muscle gain
- workout type: strength

The user can optionally input equipment if the user does not know the specific equipment. All type of equipment will be suggested to the user. This provides dynamic to the user to not be forced to enter all types of input. Workout of Close-Grip Standing Barbell Curl is the chosen workout recommended since it has a score of 1. The similar workouts will be suggested for users with high similarity score.

5.2.1.4 Feedback workout recommendation system

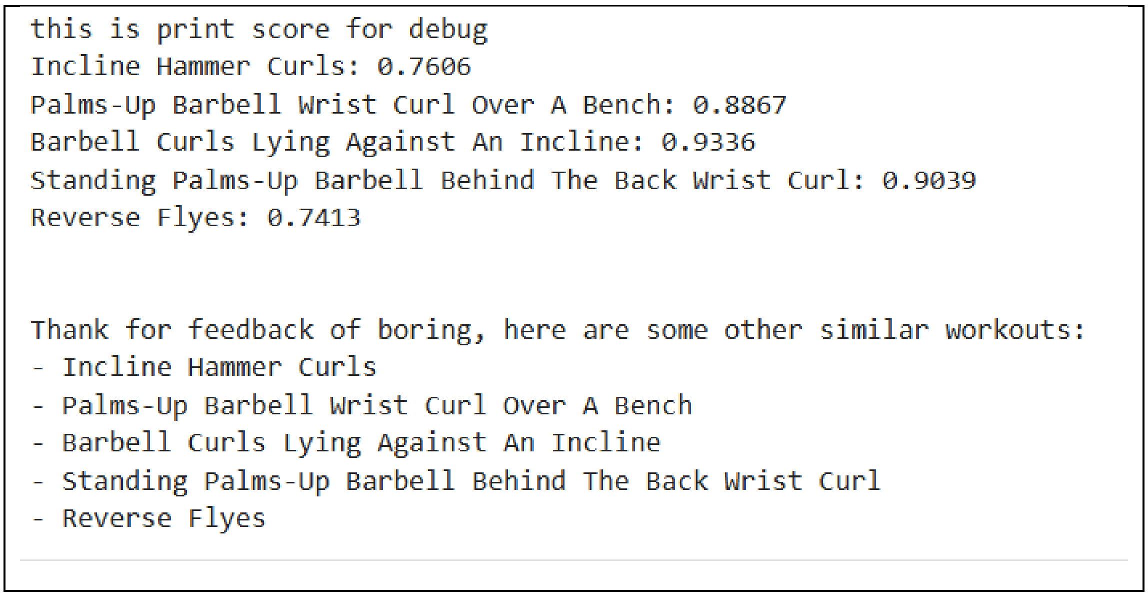
In this process, feedback from the user is handled to adjust workout plan recommendations to align with the user's condition. The feedback system is divided into four categories: injury, difficulty, boring, and time. User workout preferences are also considered to suggest similar alternatives.

5.2.1.5 Boring Feedback

When users give feedback on being bored with their current workout, the system use uses a content-based filtering approach powered by cosine similarity with score threshold to recommend workouts.

The following are the steps of handling the boring feedback process:

i. Boring variable is recorded from user feedback and passed to the handle_bored_feedback() function.
ii. When boring variable is true, the function is triggered.
iii. The cosine similarity scores of the current workout and filtered workouts are evaluated.
iv. Workouts with a similarity score above the 0.6 threshold are added to the pool. The 0.6 threshold is chosen to ensure the workouts are reasonably similar, avoiding both unrelated and overly strict matches.
v. When no workouts match the threshold, a fallback message is returned to inform the user.
vi. The pool is randomly shuffled and 5 workouts are selected to recommend the user to avoid repetitiveness.



```
this is print score for debug
Incline Hammer Curls: 0.7606
Palms-Up Barbell Wrist Curl Over A Bench: 0.8867
Barbell Curls Lying Against An Incline: 0.9336
Standing Palms-Up Barbell Behind The Back Wrist Curl: 0.9039
Reverse Flyes: 0.7413


Thank for feedback of boring, here are some other similar workouts:
- Incline Hammer Curls
- Palms-Up Barbell Wrist Curl Over A Bench
- Barbell Curls Lying Against An Incline
- Standing Palms-Up Barbell Behind The Back Wrist Curl
- Reverse Flyes
```

Figure 5.2.1.5.1, Boring feedback results for 5 workouts are recommended to user for muscle gain.

In Figure 5.2.1.5.1 , top 5 workouts are recommended to user for muscle gain. The user input example is as below:

User input:

- level: beginner
- goal: muscle gain
- workout type: strength

The 5 workouts that are more than 0.6 similarity threshold are suggested. The randomization allows users to have more diversity in workout plan with similar benefits to user to achieve goal.

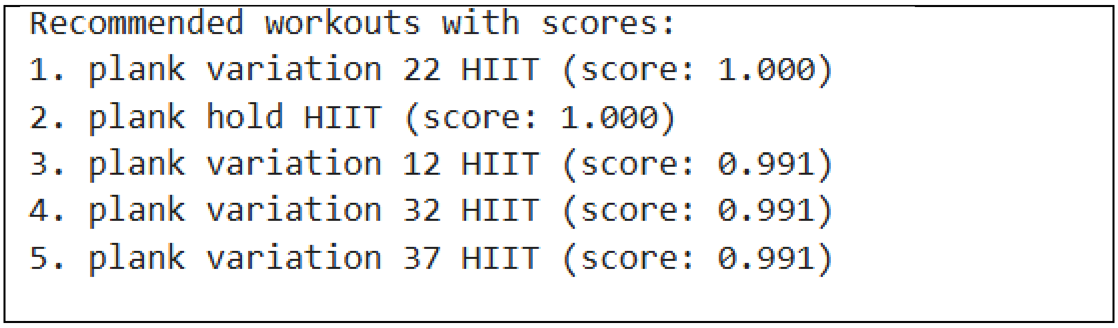
5.2.1.6 Time Feedback

When users give feedback on they are busy and no time for workouts, the system will set the workout category to HIIT which is time-efficient workouts.

The following are the steps of handling the busy feedback process:

i. User input will be saved as backup user input to store the original input.
ii. Busy variable is recorded from user feedback and pass to adjust_user_input_for_busy_feedback() function.
iii. The category entered by user for user input will be modified into hiit.
iv. The modified user input will be sent to filter the workout for the busy case.
v. The filtered workout will go through the normal process to get the recommended workouts.

This feature will be combined with the calendar API to give better output. The system will fetch calendar events to check the user's schedule to identify how busy the day. When the day is packed, the system will ask for user to switch to HIIT workout and reallocate the workout time.



```
Recommended workouts with scores:
1. plank variation 22 HIIT (score: 1.000)
2. plank hold HIIT (score: 1.000)
3. plank variation 12 HIIT (score: 0.991)
4. plank variation 32 HIIT (score: 0.991)
5. plank variation 37 HIIT (score: 0.991)
```

5.2.1.6.1, Time feedback results for 5 workouts are recommended

In Figure 5.2.1.6.1, top 5 workouts are recommended to user for muscle gain. The user input example is as below:

User input:

- level: beginner
- goal: muscle gain
- workout type: strength

Adjusted User input:

- level: beginner
- workout type: HIIT

The user input is altered by changing the workout type into HIIT. The original workout input will be store as backup user input and will be reused when during not busy. The adjusted user input will be go through normal workflow to get recommended workout plan.

5.2.1.7 Injury Feedback

In this process, system will handle the user injury body parts feedback to recommend safe workouts that do not include those corresponding muscle parts. The system will map the common body part term with the corresponding muscle parts and exclude those workouts that include those muscles.

The following are the steps of handling the injury feedback process:

i. The muscles are collected from the cleaned muscleAffected column that was previously done and recorded by using `set()` to get unique muscles.
ii. A dictionary of common body parts terms to muscles is defined and it is shown in Table 5.2.1.7.1 for reference.

iii. Serious injury case is handled by responding an advice message to rest instead of workouts.

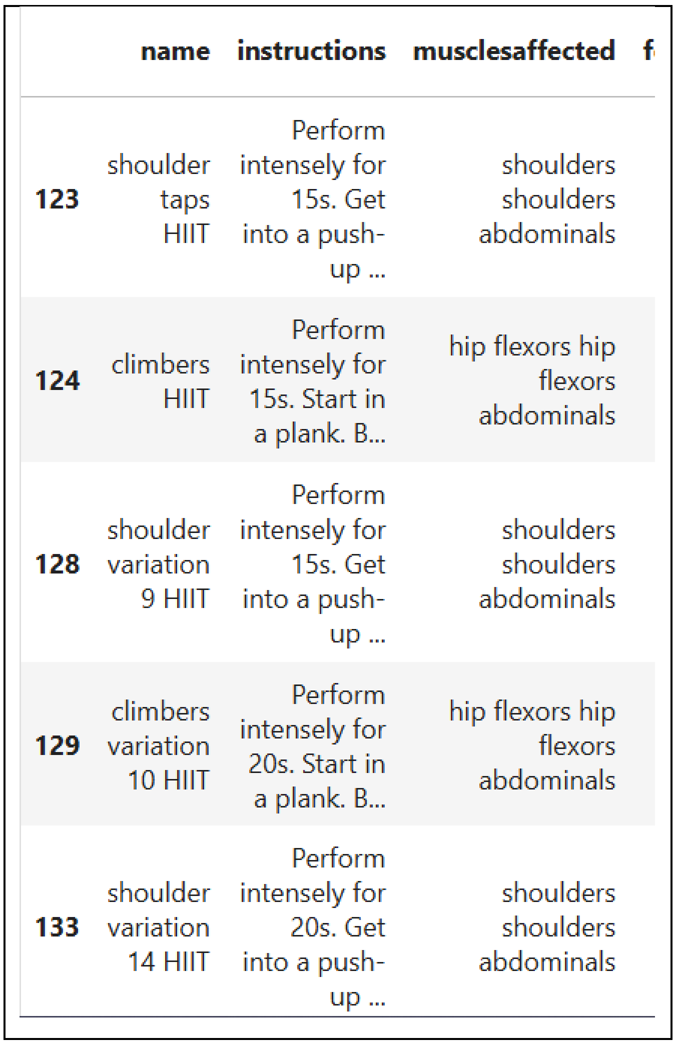
iv. The common body parts entered by users will be mapping to their corresponding muscles.

v. The workouts that target those muscles will be filtered out by using disjoint().

vi. The safe workouts will remain and stored in a dataframe, then pass to undergo user input filter and cosine similarity as stated in Section 4.4.

Injury Common Body Parts	**Muscles**
shoulder	shoulder, trap, deltoid
neck	neck, trap
arm	biceps, triceps, forearm
chest	chest, pectorals
back	back, lat, trap, middle
lower back	lower, back
core	abdominal
hip	hip, glute, flexor
leg	leg, quadriceps, hamstring, calf
knee	quadriceps, hamstring, calf
foot	foot
inner thigh	adductor
outer thigh	abductor
full body	*(empty – all muscles to be avoided)*

Table 5.2.1.7.1 Injury common body parts term and muscles mapping



	name	instructions	musclesaffected	f
123	shoulder taps HIIT	Perform intensely for 15s. Get into a push-up ...	shoulders shoulders abdominals	
124	climbers HIIT	Perform intensely for 15s. Start in a plank. B...	hip flexors hip flexors abdominals	
128	shoulder variation 9 HIIT	Perform intensely for 15s. Get into a push-up ...	shoulders shoulders abdominals	
129	climbers variation 10 HIIT	Perform intensely for 20s. Start in a plank. B...	hip flexors hip flexors abdominals	
133	shoulder variation 14 HIIT	Perform intensely for 20s. Get into a push-up ...	shoulders shoulders abdominals	

Figure 5.2.1.7.2 The filtered dataframe without injury muscle part on knee

The user feedback injury part as knee. Knee is mapping to muscle of hamstring, quadriceps, and calf. In Figure 5.2.1.7.2, a snapshot of 5 data tuple from the injury filtered dataframe is shown, which excludes hamstring, quadriceps, and calf. The filtered dataframe will be used to The adjusted user input will be go through the section normal workflow to get a recommended workout plan.

5.2.1.8 Difficulty Feedback

In this process, user feedback on difficult workouts will be handled. The difficulty level will be downgraded to provide easier workouts. When the user consistently fails in completing the workout, the difficulty will also decrease.

The following are the steps of handling the Difficulty feedback process:

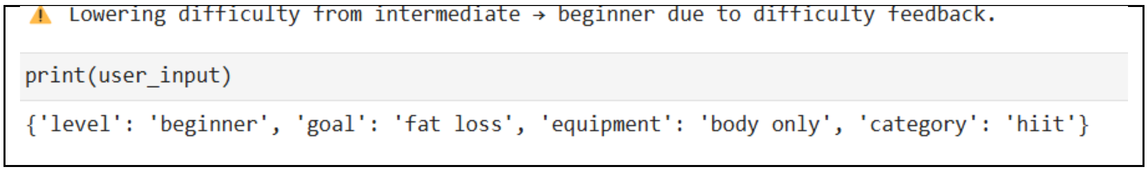
i. The "hard" difficult feedback is recorded and passed to the `adjust_difficulty_level()` function.

ii. If the function triggered when user feedback as hard and 3 times in a row (`fail_streak >= 3`).

iii. The current level of the user input is checked.

iv. If the user is not already at the lowest level ("beginner"), the level will be shifted one level down.

v. Fallback message of to inform already in the lowest message will be informed if the current level is “beginner”.

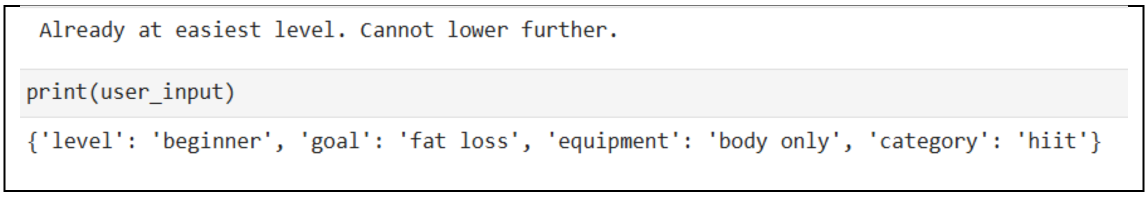


```
⚠️ Lowering difficulty from intermediate → beginner due to difficulty feedback.

print(user_input)

{'level': 'beginner', 'goal': 'fat loss', 'equipment': 'body only', 'category': 'hiit'}
```

Figure 5.2.1.8.1 Success message for lowering difficulty



```
Already at easiest level. Cannot lower further.

print(user_input)

{'level': 'beginner', 'goal': 'fat loss', 'equipment': 'body only', 'category': 'hiit'}
```

Figure 5.2.1.8.2 Fallback message for already easiest level

In Figure 5.2.1.8.1, the initial user input is shown below:

User input:

- level: intermediate
- goal: fat loss
- equipment: body only
- workout type: HIIT

Then the difficulty is successfully lowered from intermediate to beginner.

In Figure 5.2.1.8.2, the initial user input is shown below:

User input:

- level: beginner
- goal: fat loss
- equipment: body only
- workout type: HIIT

Since it is already in the easiest level, it remains and fallback message to notice user is displayed. Both user input will be go through the normal workflow to get a recommended workout plan.

5.2.1.9 User preference workout recommendation

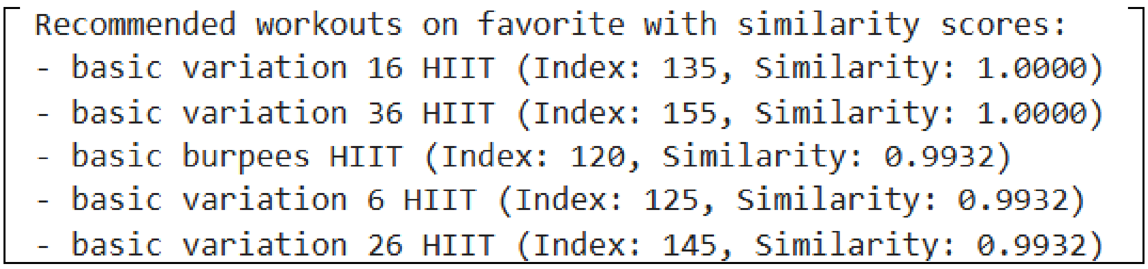
In this process, the index of the favorite workout marked by the user will be used to process to get similar workouts within the user input requirement context.

i. The user's marked favorite workout index is recorded.
ii. The cosine similarity for all workouts to all workouts is calculated and saved as `all_vs_all_cos_similarity_matrix`.
iii. The favorite workout index related scores are selected out from `all_vs_all_cos_similarity_matrix` and stored as `fav_workout_similarity`.
iv. The filtered workouts from the previous normal workflow is used and check the favorite workout exists in the filtered workouts to ensure the favorite workouts can help the user to achieve goal. If not, fallback message that the favorite workout is not suitable for the user to achieve the current goal.
v. The filtered workouts score will be checked and arranged top 5 similar workouts, including the favorite workout itself.

In the `all_vs_all_cos_similarity_matrix`, all cosine similarity scores between workouts are stored. Since there are 160 workouts (40 for each workout type), the matrix is sized 160×160, where each workout is compared against every other workout, including itself.

The `fav_workout_similarity` contains the similarity scores for a selected workout (favorite workout) against all 160 workouts. This includes the score of the workout compared with itself.

Next, `filtered_similarities = [(i, fav_workout_similarity[i]) for i in filtered_indices]` is used to get the similarity scores between the favorite workout and only the workouts that match the user's current input filters. It checks if those filtered workouts exist in the 160 similarity scores linked to the favorite workout.Finally, the filtered similarities are sorted, and the top 5 workouts with the highest similarity scores are selected. The favorite workout itself is included in the top 5 since its self-similarity score is 1.



```
Recommended workouts on favorite with similarity scores:
- basic variation 16 HIIT (Index: 135, Similarity: 1.0000)
- basic variation 36 HIIT (Index: 155, Similarity: 1.0000)
- basic burpees HIIT (Index: 120, Similarity: 0.9932)
- basic variation 6 HIIT (Index: 125, Similarity: 0.9932)
- basic variation 26 HIIT (Index: 145, Similarity: 0.9932)
```

Figure 5.2.1.9.1 top 5 recommended workouts with favorite workout

Figure 5.2.1.9.1 shows the top 5 recommended workout by using the favorite workout. The favorite workout is basic variation 36 HIIT (index 155). The system gets two scores of 1 due to the ChatGPT-generated datasets problem which is the same reason in normal workflow, Figure 5.2.1.9.1 discussion.

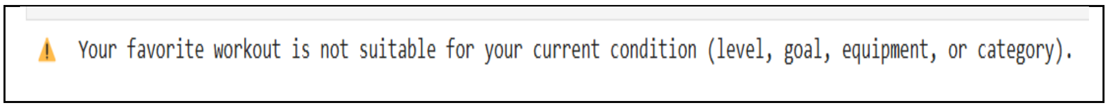

⚠ Your favorite workout is not suitable for your current condition (level, goal, equipment, or category).

Figure 5.2.1.9.2 Favorite workout not suitable message

Figure 5.2.1.9.2 shows the fallback message when the Favorite workout not suitable to the current user input requirement.

5.2.2 Service configuration

Backend configuration

Once the machine learning model part is done, the model artifact is created to build for API usage instead of re-computing the heavy operation.

The below artifact is created:

- a Scikit-learn TF-IDF vectorizer (vectorizer.joblib)
- SciPy sparse matrix of workout features (final_matrix.npz
- CSV files for mapping workout IDs to matrix indices (workout_index_mapping.csv)
- Workout dataset for referring if needed (four_workout_merged_df.csv)
- List of filter columns needed for recommendations for documentation purposes (filter_columns.json)
- Reference of efficient pre-filtering for the user input filter before cosine similarity step (cat_df_encoded)

The backend API in FastAPI will load the model artifact for use. FastAPI use uvicorn to host the API app using running “uvicorn app.main:app --host 0.0.0.0 --port 8000 –reload”. This command allows the FastAPI backend to host the app in port 8000 and automatic update if changes in the code, ease for development. In the future, the FastAPI Backend can be hosted in the Replit platform for a quick, easy, and lightweight backend service for production. The current setup is enough for the demo.

Supabase configuration

To use the Supabase services, a project needs to be created in the Supabase account. In this project, FYP_Personalized_Workout_Planner project is created. The Supabase anon key and the Supabase project URL are saved for the frontend app env file for safety access. Create the tables and RLS based on the previous chapter.

Google cloud console configuration

To integrate Google Calendar, a project needs to be created in Google cloud console, called MyWorkoutPlannerApp. Next, Google Calendar API is searched and enabled. Enabling this API authorizes the project to access the Calendar endpoints and accept requests from the application. Before credentials can be generated, the OAuth 2.0 consent screen must be configured. This interface is displayed to users when the application requests access to their Google data.

The setting is shown below:

1. The User Type is set to External to allow authorization from any Google account.
2. The application name, user support email, and developer contact information is filled to show the frontend app name to mention this app want to have access and support email for users to contact to ask questions during production.
3. Scope of access id defined. The Access privileges should be least privileges. The two scope are:
    - https://www.googleapis.com/auth/calendar.events.readonly – allows the application to read events from the user’s primary calendar. The app is planned not to write the schedule to the calendar to block the user's calendar event to provide flexibility for their job event. The workout is expected to be lesser priority and flexible to fit their life
    - https://www.googleapis.com/auth/userinfo.profile – grants access to basic user profile information.

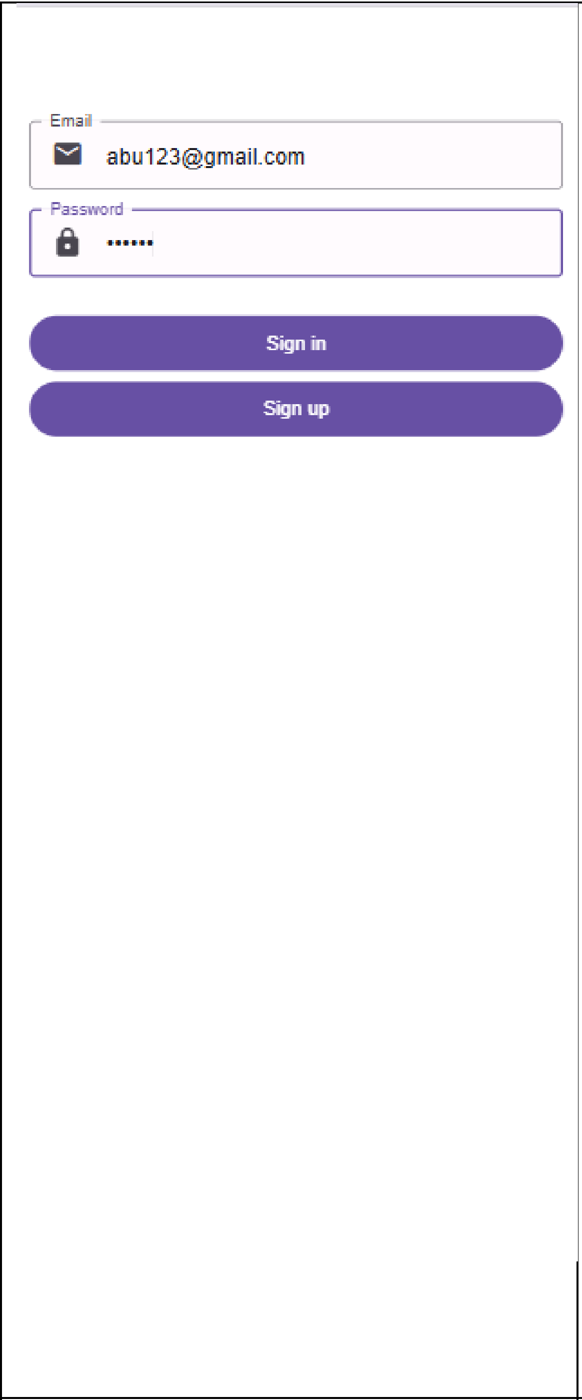
4. Test Users of specific Google accounts were added to allow secure testing during development and to bypass the unverified app screen

Different OAuth 2.0 Client IDs is needed for Web, Android, and iOS platforms to ensure secure, platform-specific authentication. Web client ID can be very useful for testing. For Web Client ID, “Web Application” type is selected, with an Authorized Redirect URI pointing to the localhost:8081 of the expo react native app. This URI handles redirect flows during development and in web builds. It a new web client ID can be create when the app is ready for production. For the mobile app Android and IOS, “Android” and “IOS” types are selected and enter a package name and SHA-1 certificate fingerprint for Android and Bundle ID for IOS when ready for production. These OAuth 2.0 Client IDs are saved and put in the env file of the frontend app to use them for better security instead of hardcoding.

5.3 System Operation

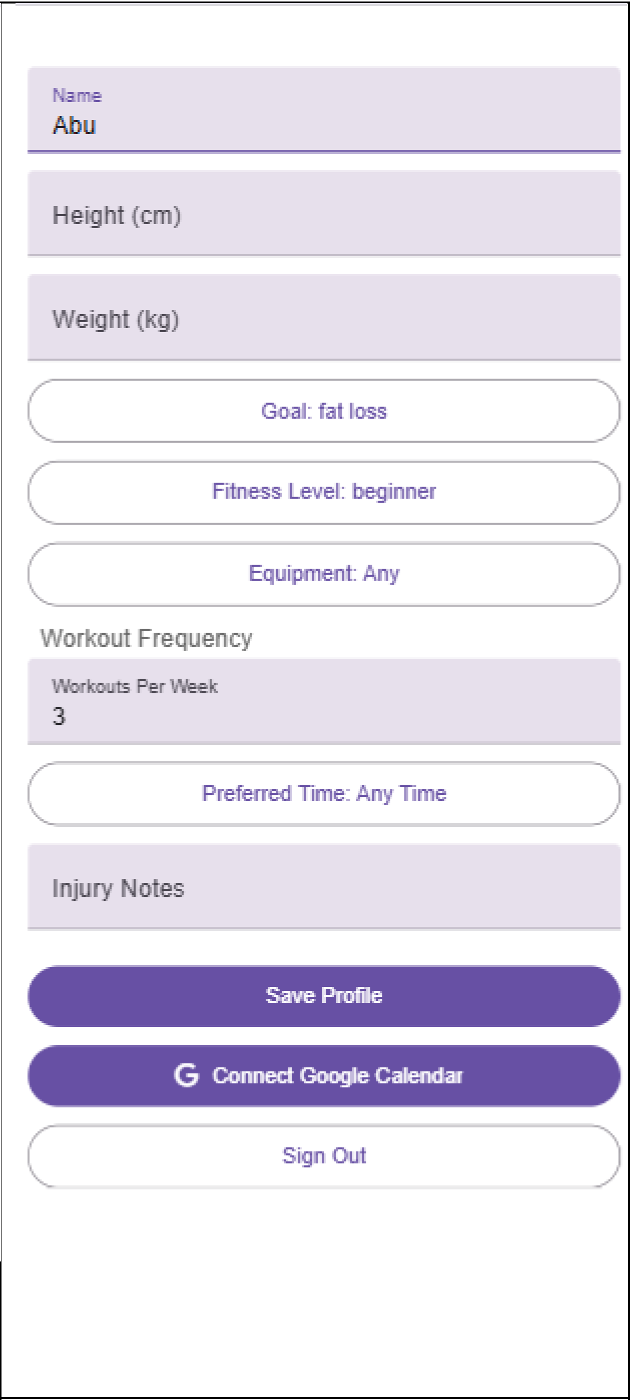
5.3.1 User Authentication and profile setup .

If the user does not have an account, new users are able to register by creating an account, while existing users can securely sign in using their email and password. Authentication is managed through Supabase Auth, which is responsible for verifying credentials, generating session tokens, and maintaining secure access control throughout the application.



Email
abu123@gmail.com
Password
••••••

Figure 5.3.1.1 Create an account and sign-in page



Name
Abu
Height (cm)
Weight (kg)
Goal: fat loss
Fitness Level: beginner
Equipment: Any
Workout Frequency
Workouts Per Week
3
Preferred Time: Any Time
Injury Notes

Figure 5.3.1.2 Pre-user profile interface before enter main screen

In Figure 5.3.1.1, a sign-up and sign-in page is shown. User can enter email and password to sign for an existing user and sign up for new user. The password will be hide for security purpose. In this case, a new user is created with email, abu123@gmail.com and password, abu123. This is a testing email for demo. It is not a real google email so it will not able to use the Google Calendar feature. In figure 5.3.1.2, a pre user profile interface is shown for update necessary user information for workout recommendations before entering the main app screen. This can prevent no information for recommended. User can fill their name, height, weight, workout frequency per week and injury notes. User can also select options for goal, preferred workout time, and equipment. Once done, click save profile to sign in. In figure 5.3.1.3, Abu is successful create an account and enters the main screen with evidence having a navigation bar. The user can do the same way to update the profile to have latest information for workout recommendation.

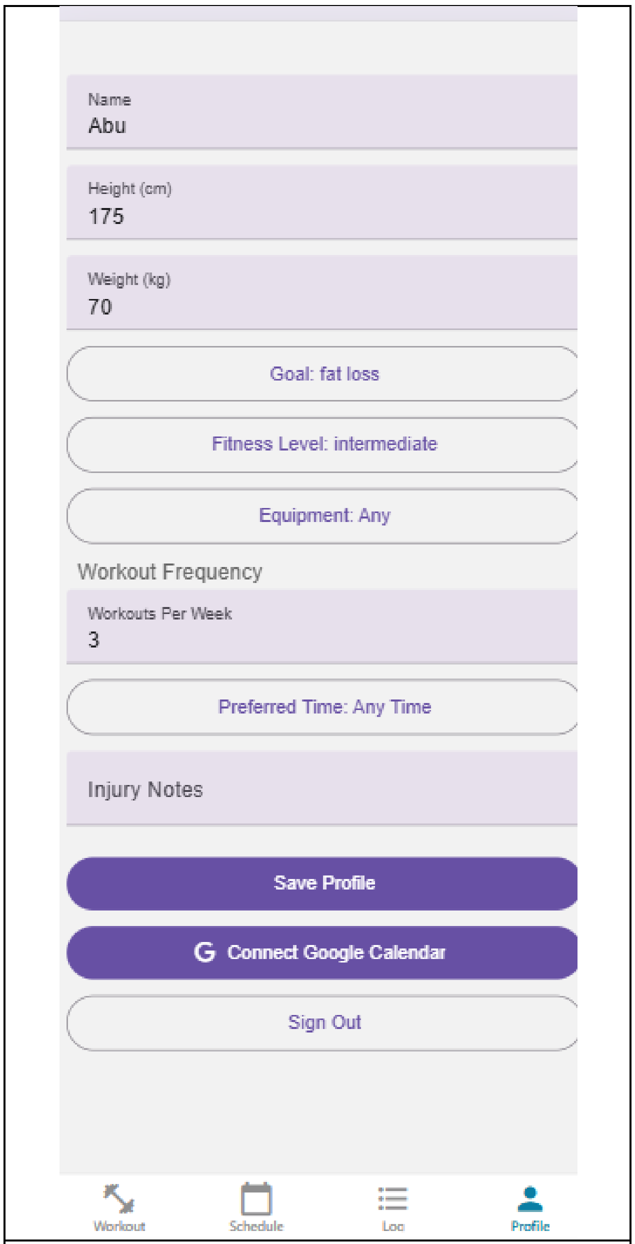
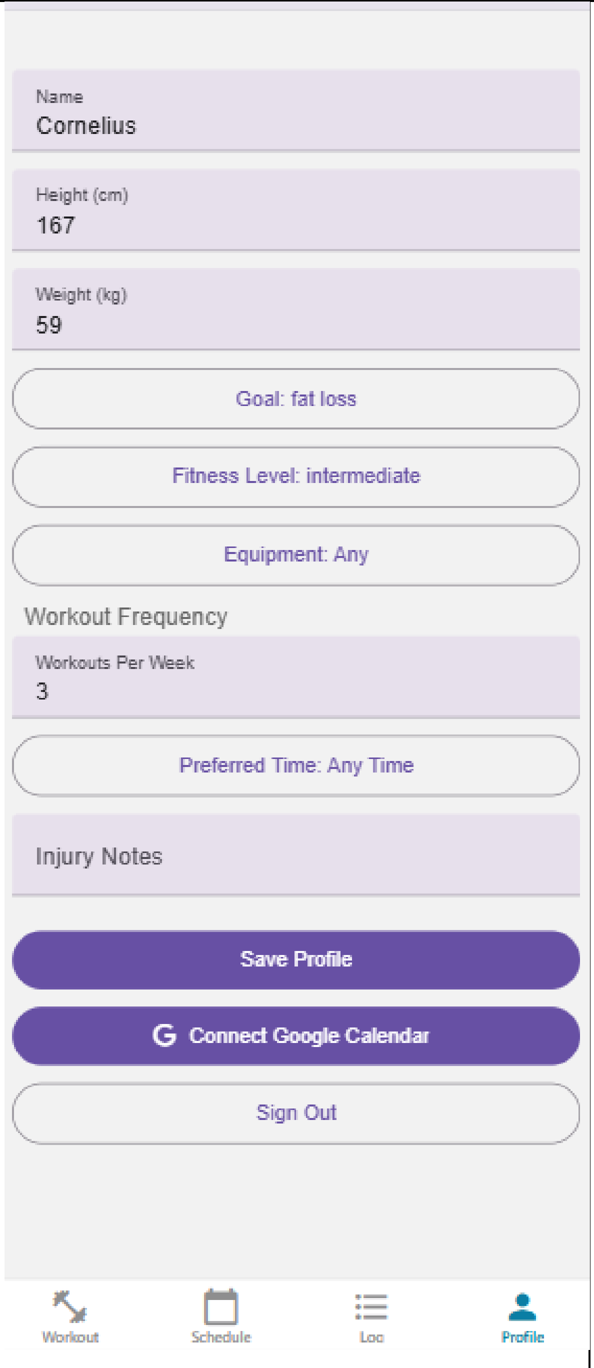
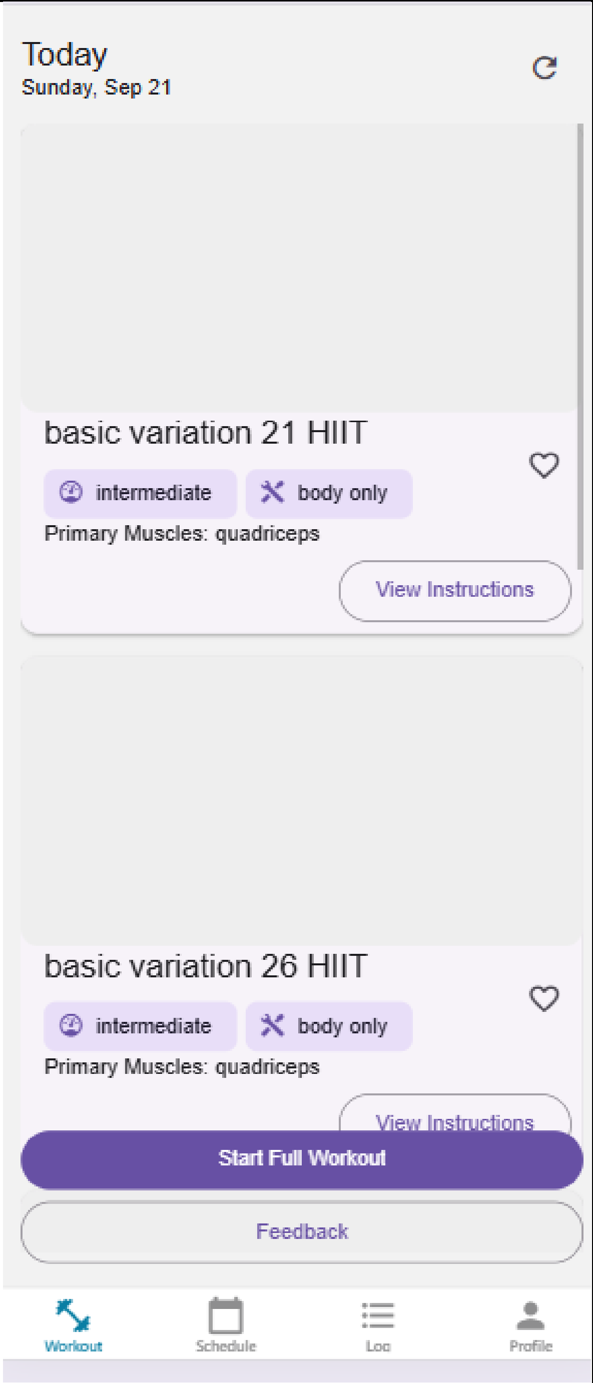

Name
Abu
Height (cm)
175
Weight (kg)
70
Goal: fat loss
Fitness Level: intermediate
Equipment: Any
Workout Frequency
Workouts Per Week
3
Preferred Time: Any Time
Injury Notes

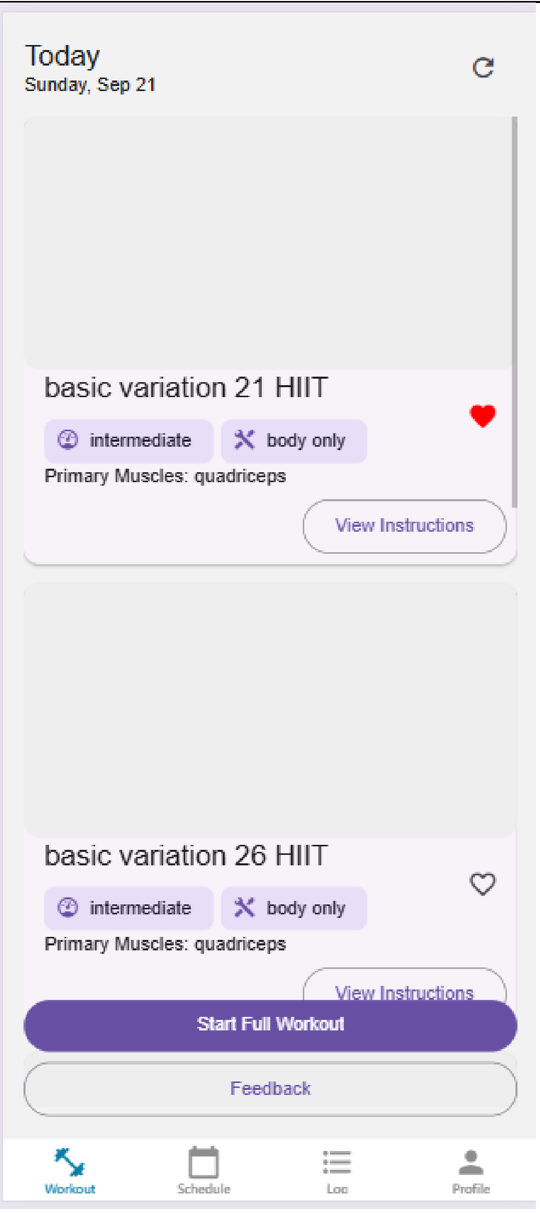
Figure 5.3.1.3 Successful create account for Abu

5.3.2 Generating and Viewing Recommendations

For workout recommendation, it will use the data stored in the profile, the system fetches and displays a tailored list of workouts on the workout screen. In this case, the existing account "Cornelius" is used. In Figure 5.3.2.1, the profile is store user data as fat loss for goal, intermediate for fitness level, any for equipment and null for injury notes. This data will be used for the workout recommendation.

 Name: Cornelius; Height (cm): 167; Weight (kg): 59; Goal: fat loss; Fitness Level: intermediate; Equipment: Any; Workout Frequency — Workouts Per Week: 3; Preferred Time: Any Time; Injury Notes Save Profile Connect Google Calendar Sign Out Workout Schedule Log Profile	 Today, Sunday, Sep 21; basic variation 21 HIIT — intermediate, body only, Primary Muscles: quadriceps; basic variation 26 HIIT — intermediate, body only, Primary Muscles: quadriceps View Instructions View Instructions Start Full Workout Feedback Workout Schedule Log Profile
Figure 5.3.2.1 Profile data of Cornelius for recommendation	Figure 5.3.2.2 Returned recommendation workouts list.

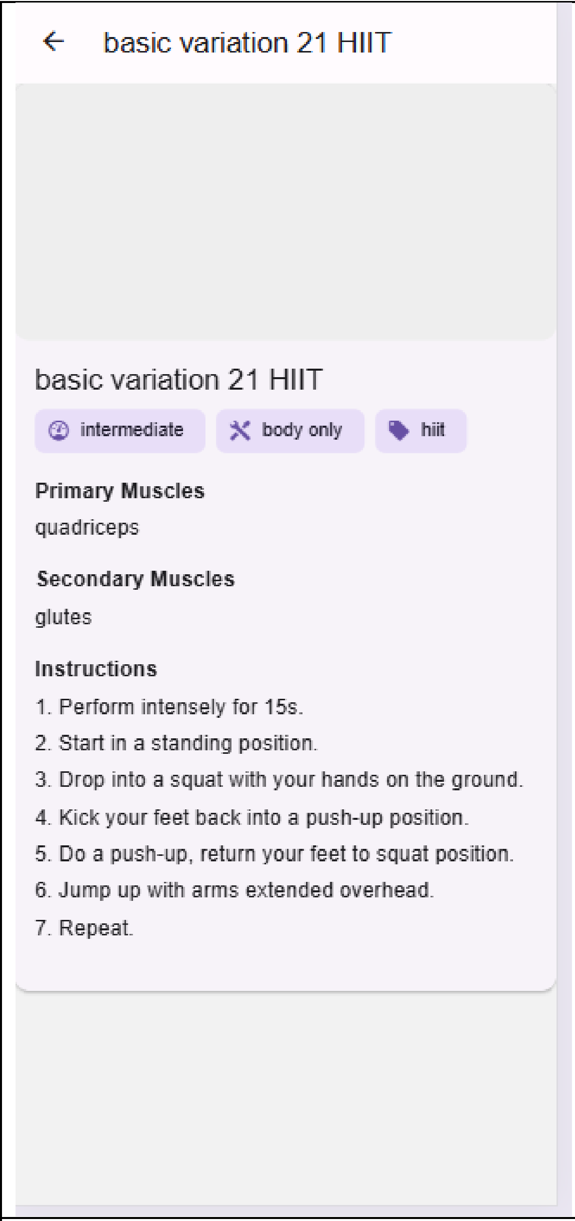
In figure 5.3.2.2, a list of 5 workouts is returned when user navigate to the workout page. The workouts is shown in a workout card box with image, name, level, equipment, primary muscle. The image section is empty and shows a rectangle placeholder due to the ChatGPT-generated dataset without images. In the future, the image can be added. The user can click the heart icon to mark the workouts as favorites for later feedback favorite. The heart icon will turn into red after update shown in figure 5.3.2.3. When the user clicks the "view instruction" button, the user will be navigated to the workout detail page, which shows additional workout categories, secondary muscles, and instructions. The image problem is similar to the previous situation.


Today
Sunday, Sep 21

basic variation 21 HIIT
intermediate | body only
Primary Muscles: quadriceps
View Instructions

basic variation 26 HIIT
intermediate | body only
Primary Muscles: quadriceps
View Instructions

Figure 5.3.2.3 Mark as favorite


basic variation 21 HIIT

basic variation 21 HIIT
intermediate | body only | hiit

Primary Muscles
quadriceps

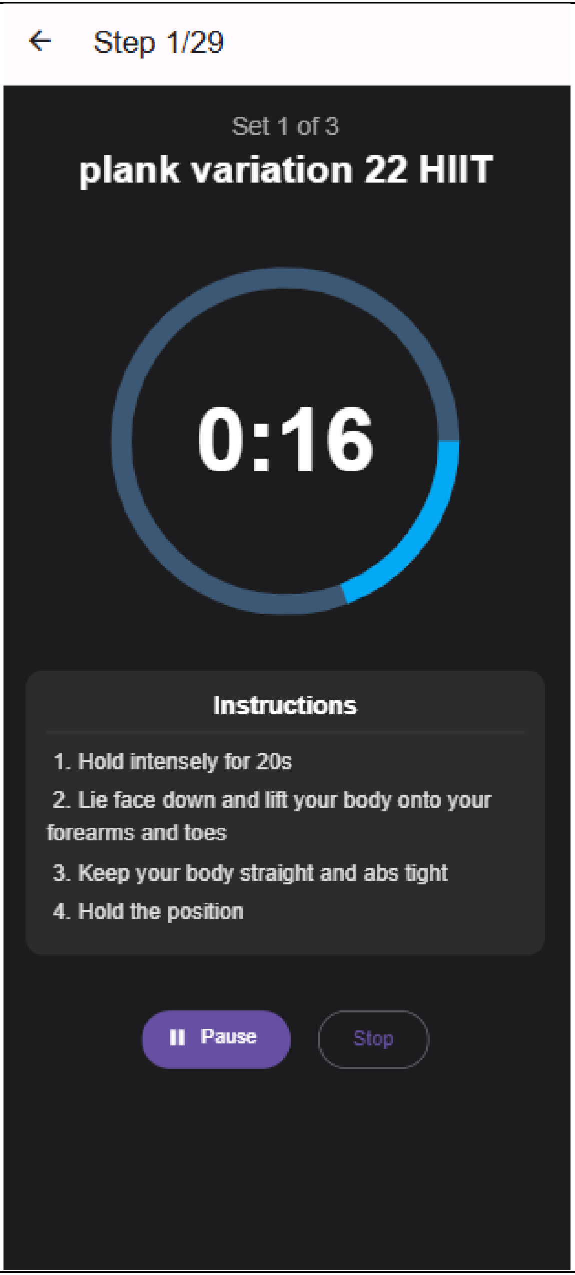
Secondary Muscles
glutes

Instructions
1. Perform intensely for 15s.
2. Start in a standing position.
3. Drop into a squat with your hands on the ground.
4. Kick your feet back into a push-up position.
5. Do a push-up, return your feet to squat position.
6. Jump up with arms extended overhead.
7. Repeat.

Figure 5.3.2.4 Workout detail

5.3.3 Executing a Workout

To start a workout session, the user can click the "Start Full Workout" button. User will be navigated to the timer screen. In figure 5.3.3.1, the a timer screen is display with a countdown for the workout duration, workout name, number of sets (1 set is equal to the whole recommendation list), current progress of steps, and sets and instructions. The sets are considered repeating the recommended workouts list. In the future, an image or a video can be shown instead of only instructions for a clearer guide. The user can click the pause button for pause and stop button to declare failure or cancel the workout session. If the user chooses to declare failure, it will ask the user to input the current weight and record the workout log. If the user chooses discard, no workout log will be recorded for a mis-click to start a workout case. These are shown in Figure 5.3.3.2.


Step 1/29

Set 1 of 3

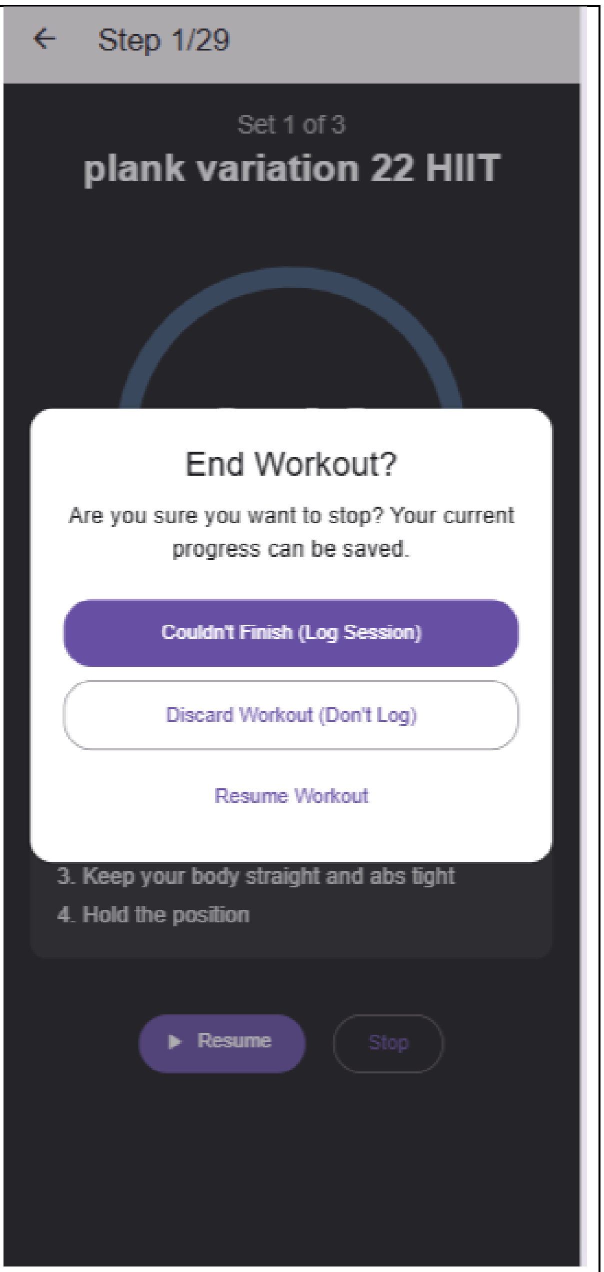
plank variation 22 HIIT

0:16

Instructions

1. Hold intensely for 20s
2. Lie face down and lift your body onto your forearms and toes
3. Keep your body straight and abs tight
4. Hold the position

Figure 5.3.3.1 Timer screen


Step 1/29

Set 1 of 3

plank variation 22 HIIT

End Workout?

Are you sure you want to stop? Your current progress can be saved.

3. Keep your body straight and abs tight
4. Hold the position

Figure 5.3.3.2 Stop timer

When the timer reaches the end, the user can click the "finish and save" button and it will be show a pop-up menu for the user to input their current weight. The user can choose to record or not to record. This is shown in Figure 5.3.3.3 and Figure 5.3.3.4.

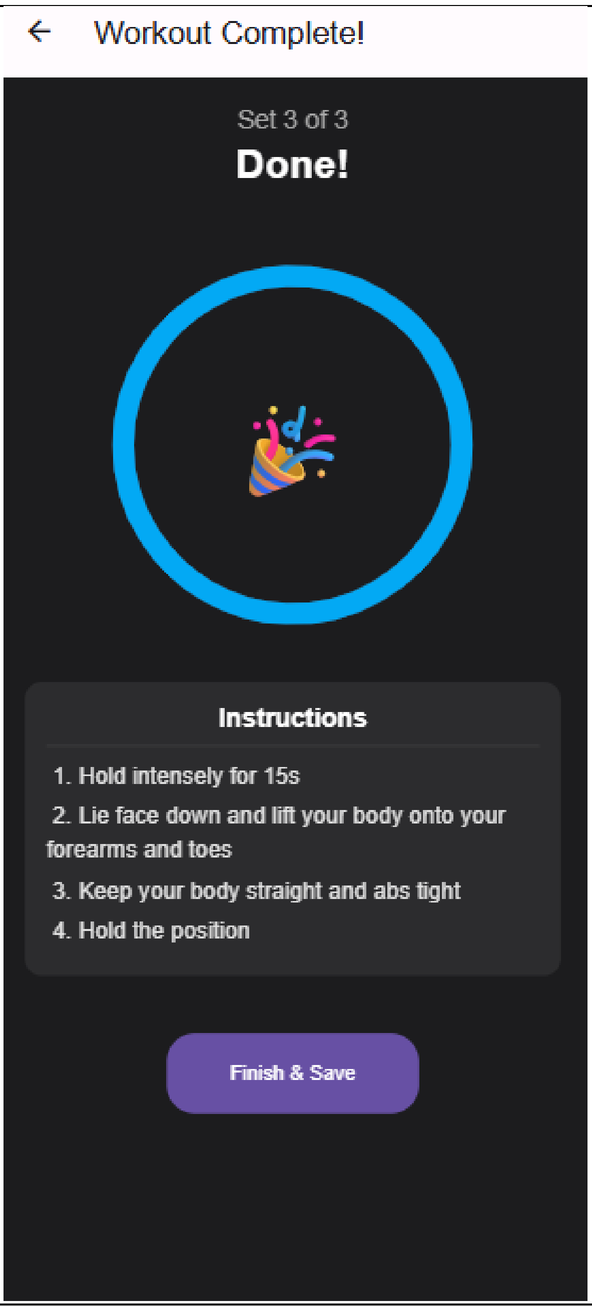
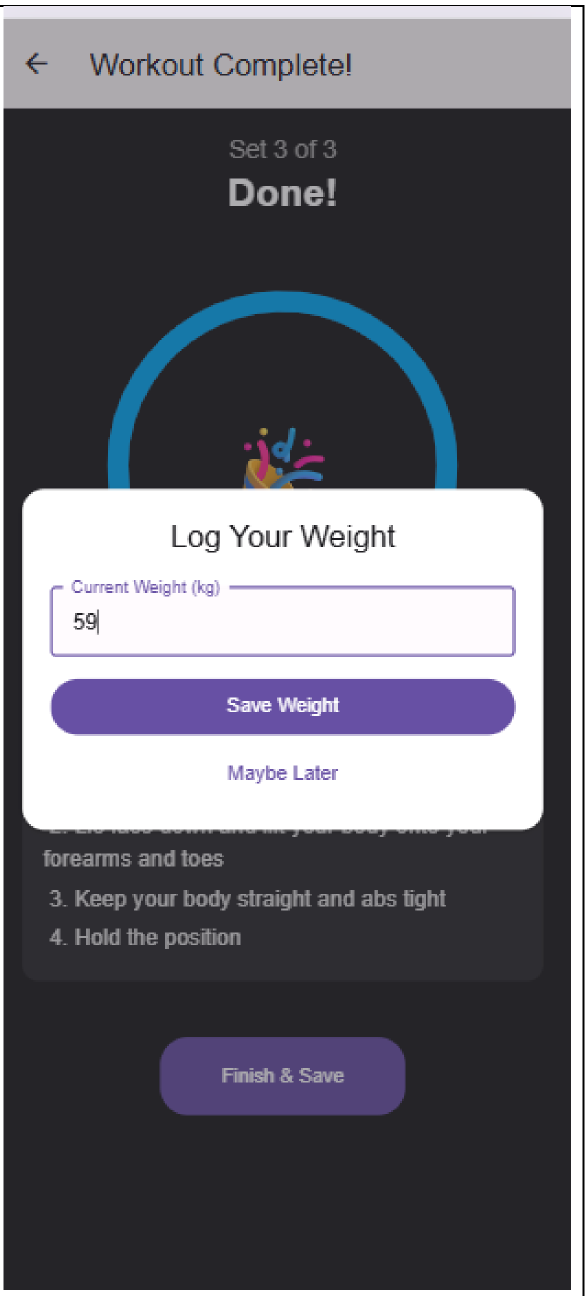
	

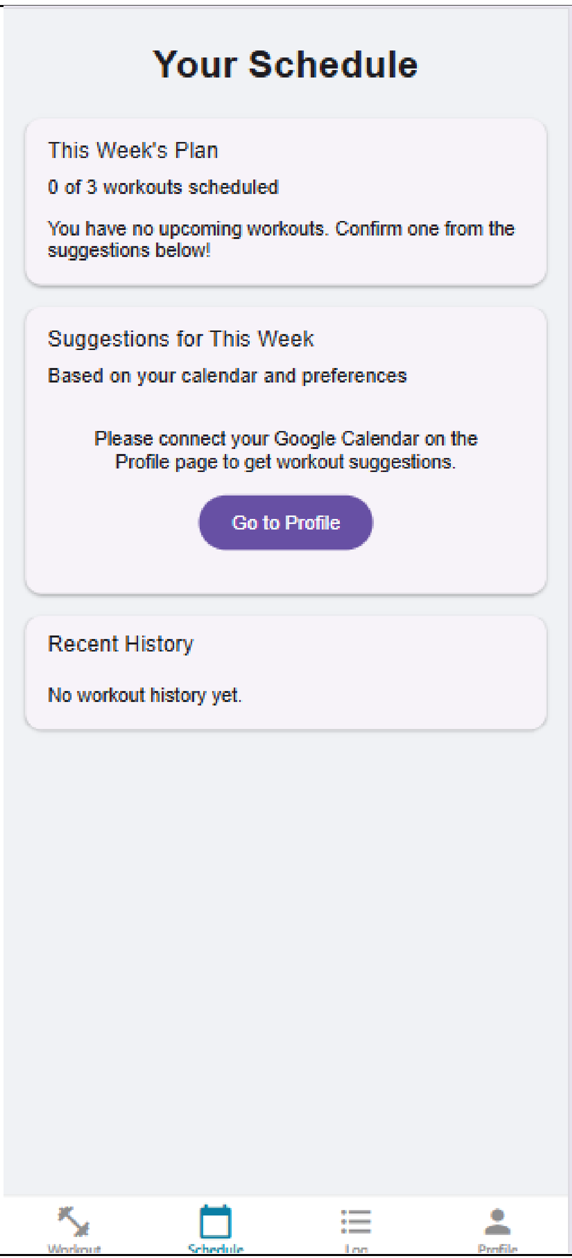
Figure 5.3.3.3 Success shown after complete	Figure 5.3.3.4 Record the weight

Workout Complete!
Set 3 of 3
Done!
Instructions
1. Hold intensely for 15s
2. Lie face down and lift your body onto your forearms and toes
3. Keep your body straight and abs tight
4. Hold the position
Finish & Save

5.3.4 Schedule workout session and log dashboard.

When the user navigates to the schedule page, the system will check for google calendar connection. If not connected, the user can click the "Go to profile" button to navigate to the profile page to connect the Google Calendar shown In figure 5.3.4.1. It will nagivate to the

Google consent page for choosing Google account and granting permission shown In figure 5.3.4.2.



Your Schedule

This Week's Plan

0 of 3 workouts scheduled

You have no upcoming workouts. Confirm one from the suggestions below!

Suggestions for This Week

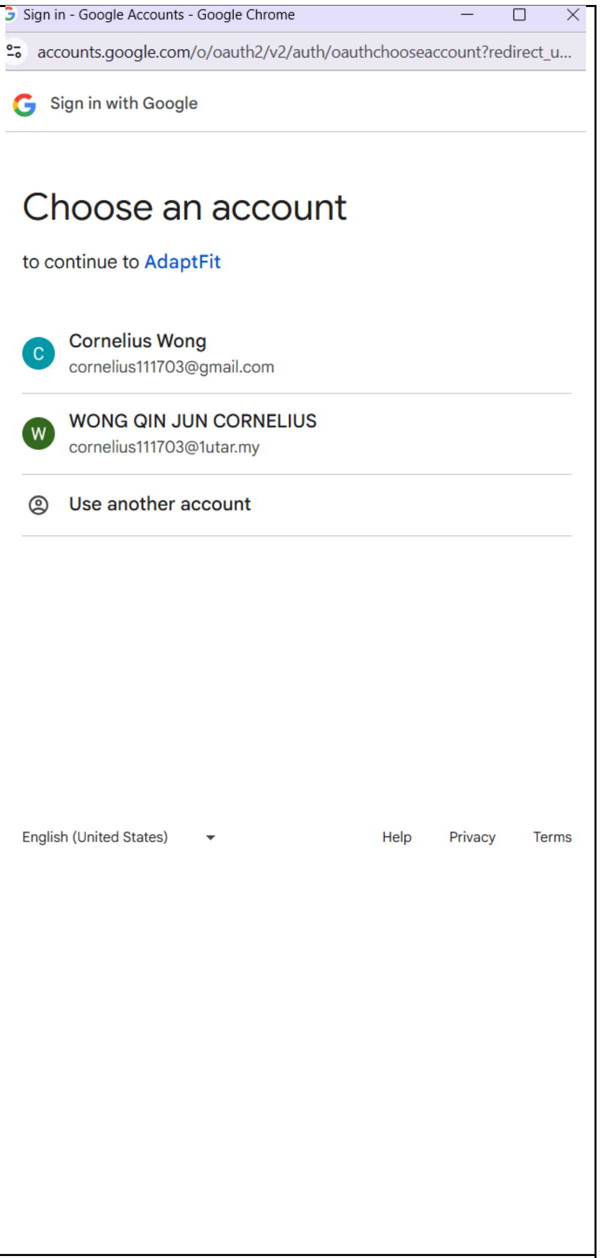
Based on your calendar and preferences

Please connect your Google Calendar on the Profile page to get workout suggestions.

Recent History

No workout history yet.

Figure 5.3.4.1 schedule page without connection



accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?redirect_u...

Sign in with Google

Choose an account

to continue to AdaptFit

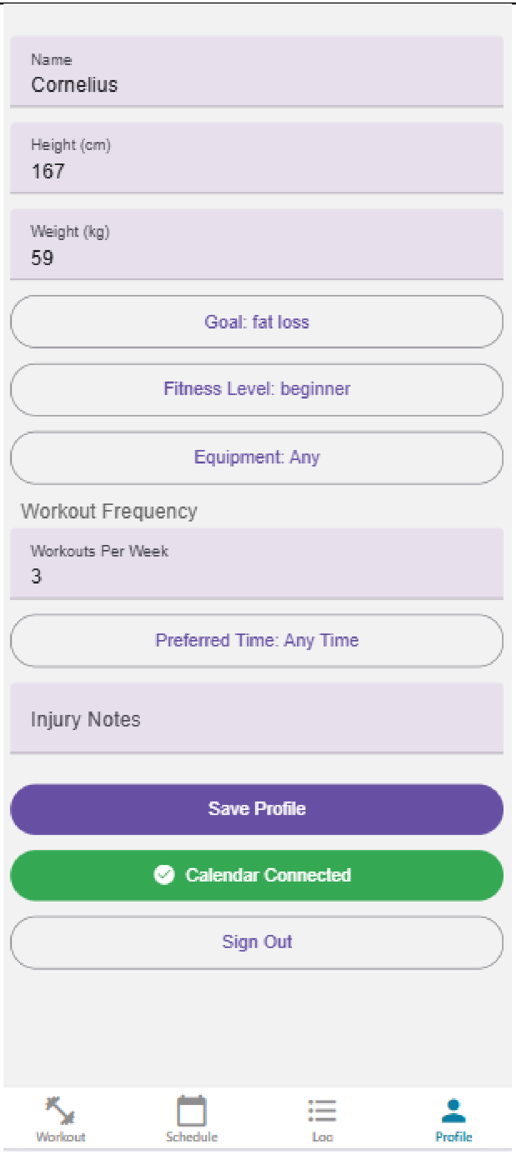
Cornelius Wong
cornelius111703@gmail.com

WONG QIN JUN CORNELIUS
cornelius111703@1utar.my

Use another account

Figure 5.3.4.2 Authentication and grant permission for Google account for the Google Calendar connection

After the connection is successful, the button will turn green and display as "Calendar Connected" shown in figure 5.3.4.3. Now, when navigate to the schedule page, the schedule content is updated shown in figure 5.3.4.4. It can be categorized into 3 sections: Weekly plan, schedule suggestion, and history schedule. For the Weekly plan, the number of schedule limits is based on the workout per week data set in the profile page. Currently, it has no schedule yet. For the schedule suggestion part, the suggestion is given based on the user's free slot in the Google Calendar. It will return the nearest time zone of free slot based on the user's preferred workout time set in the profile page. The user can click on the "confirm" button to select the schedule to put into the weekly plan. For the history schedule part, it shows the user's history schedule and completion status. If not workout logs is created in within the timestamp of schedule, it will be marked as missed, otherwise completed.



Name
Cornelius

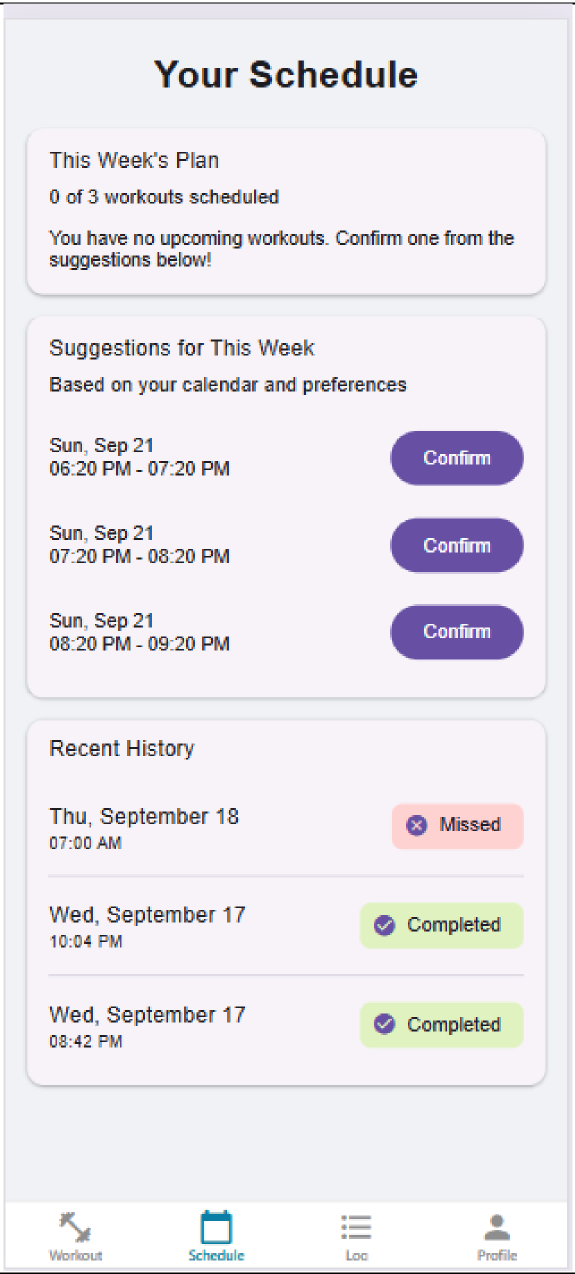
Height (cm)
167

Weight (kg)
59

Workout Frequency

Workouts Per Week
3

Injury Notes



Your Schedule

This Week's Plan

0 of 3 workouts scheduled

You have no upcoming workouts. Confirm one from the suggestions below!

Suggestions for This Week

Based on your calendar and preferences

Sun, Sep 21
06:20 PM - 07:20 PM

Sun, Sep 21
07:20 PM - 08:20 PM

Sun, Sep 21
08:20 PM - 09:20 PM

Recent History

Thu, September 18
07:00 AM — Missed

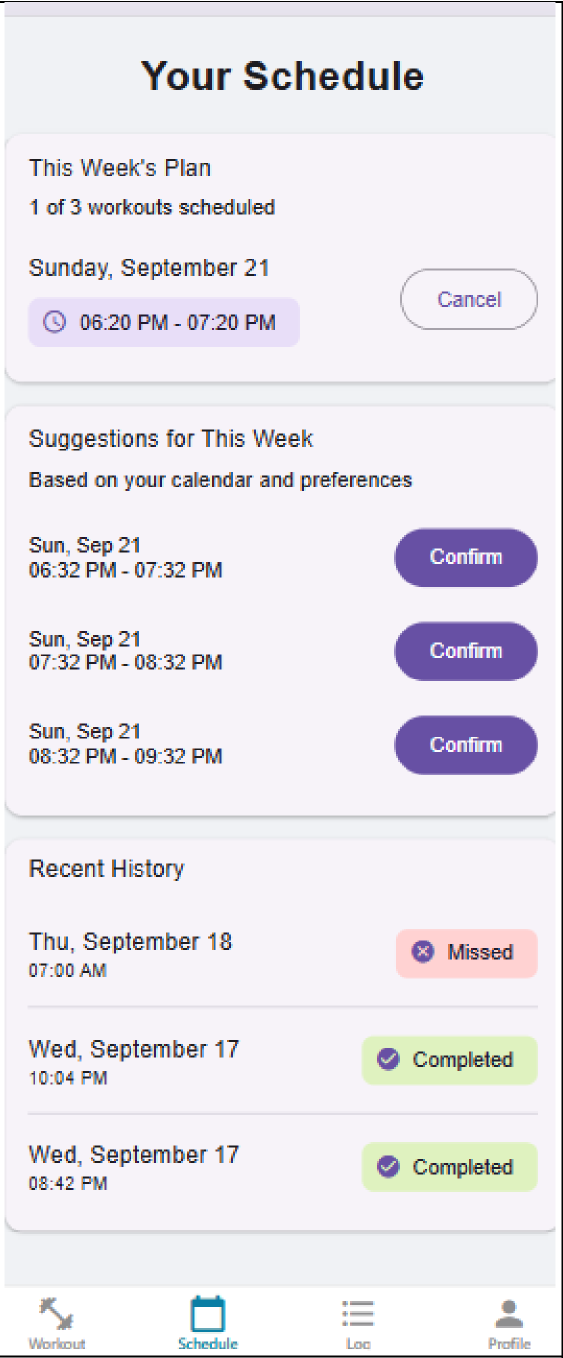
Wed, September 17
10:04 PM — Completed

Wed, September 17
08:42 PM — Completed

Figure 5.3.4.3 Success connection to Google Calendar

Figure 5.3.4.4 schedule page without connection

When the user confirms the schedule, it will be moved into the weekly plan section and the suggestion section will fill the gap to show the next suggestion shown in figure 5.3.4.5. Meanwhile, the notification will be scheduled to notice user when the schedule time zone reached. If the schedule reach the workout per week limit, the schedule suggestion section will be close shown in figure 5.3.4.6.



Your Schedule

This Week's Plan
1 of 3 workouts scheduled

Sunday, September 21
06:20 PM - 07:20 PM

Suggestions for This Week
Based on your calendar and preferences

Sun, Sep 21
06:32 PM - 07:32 PM

Sun, Sep 21
07:32 PM - 08:32 PM

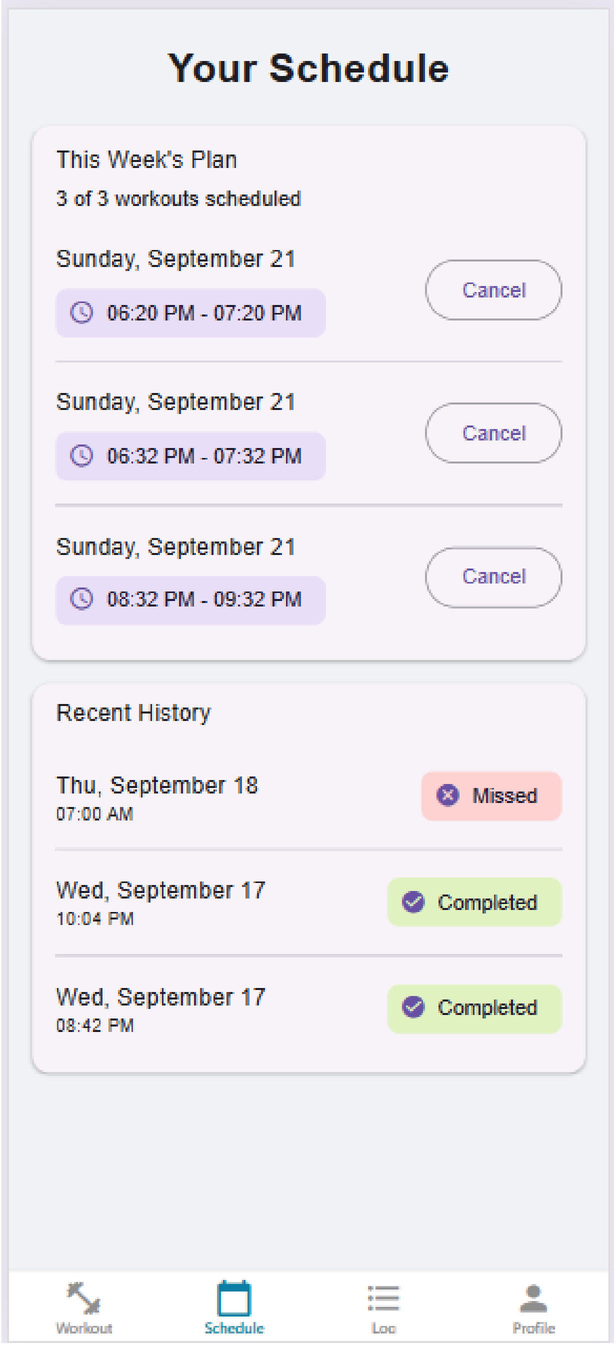
Sun, Sep 21
08:32 PM - 09:32 PM

Recent History

Thu, September 18
07:00 AM
Missed

Wed, September 17
10:04 PM
Completed

Wed, September 17
08:42 PM
Completed



Your Schedule

This Week's Plan
3 of 3 workouts scheduled

Sunday, September 21
06:20 PM - 07:20 PM

Sunday, September 21
06:32 PM - 07:32 PM

Sunday, September 21
08:32 PM - 09:32 PM

Recent History

Thu, September 18
07:00 AM
Missed

Wed, September 17
10:04 PM
Completed

Wed, September 17
08:42 PM
Completed

Figure 5.3.4.5 Confirm schedule from suggestion	Figure 5.3.4.6 Schedule reach limit set by the user

When the user navigate to the log page, the system will fetch the needed data to construct the dashboard. In Figure 5.3.4.7, the statistical data is shown such as number of workouts in this month, weekly streak, and average completion percentage of workouts and BMI. The weight changes are shown in a chart for the user to view the trend easily. In Figure 5.3.4.8, the primary muscle group focus distribution is shown in the bar chart to show which muscles users train the most. The completion rate of the workout chart against date is shown to view the trend of completion for consistency evaluation. In Figure 5.3.4.9, a workout calendar is shown with the dot indicator at the date that the user has a workout log to have overview of workout days.

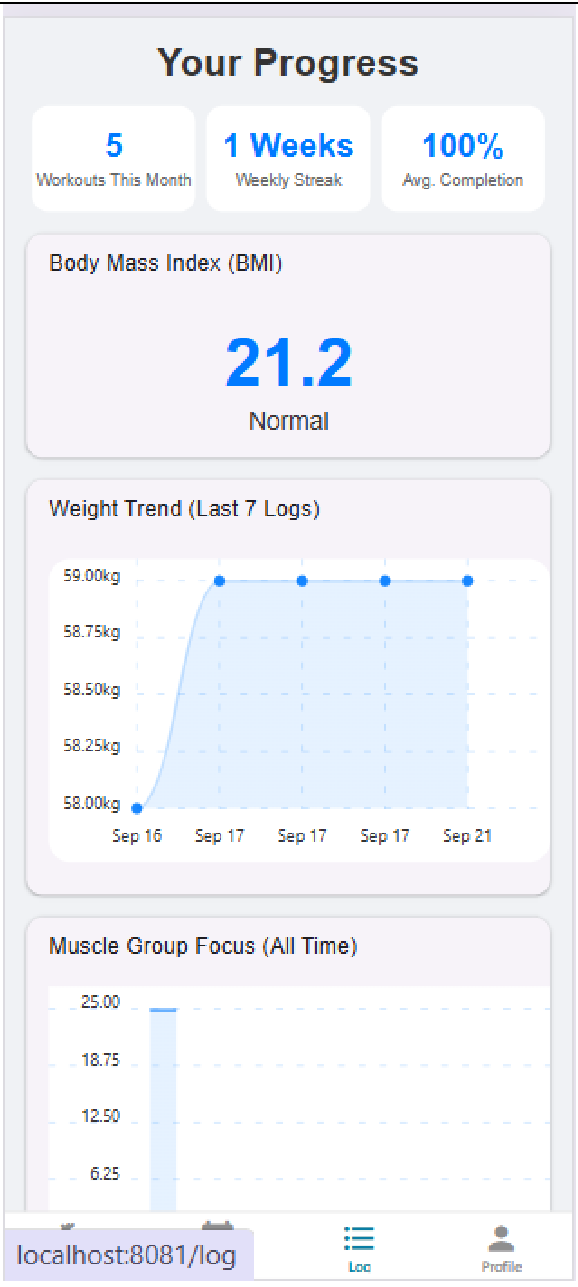


Figure 5.3.4.7 log screen part 1

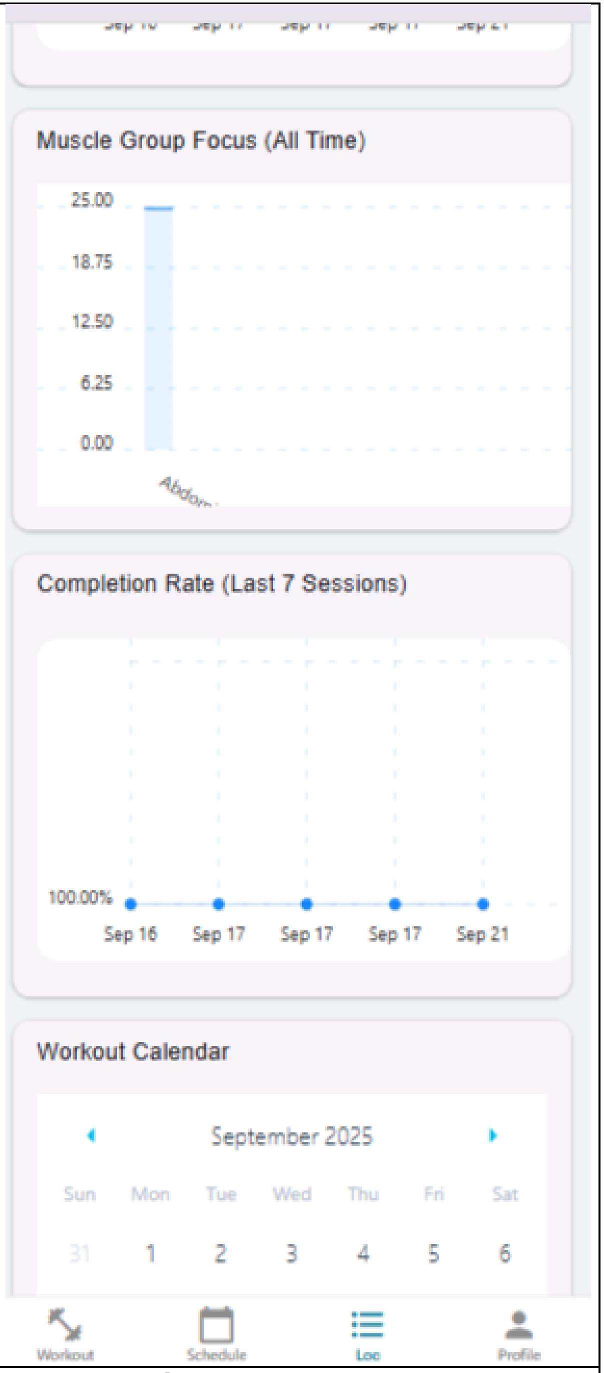


Figure 5.3.4.8 log screen part 2

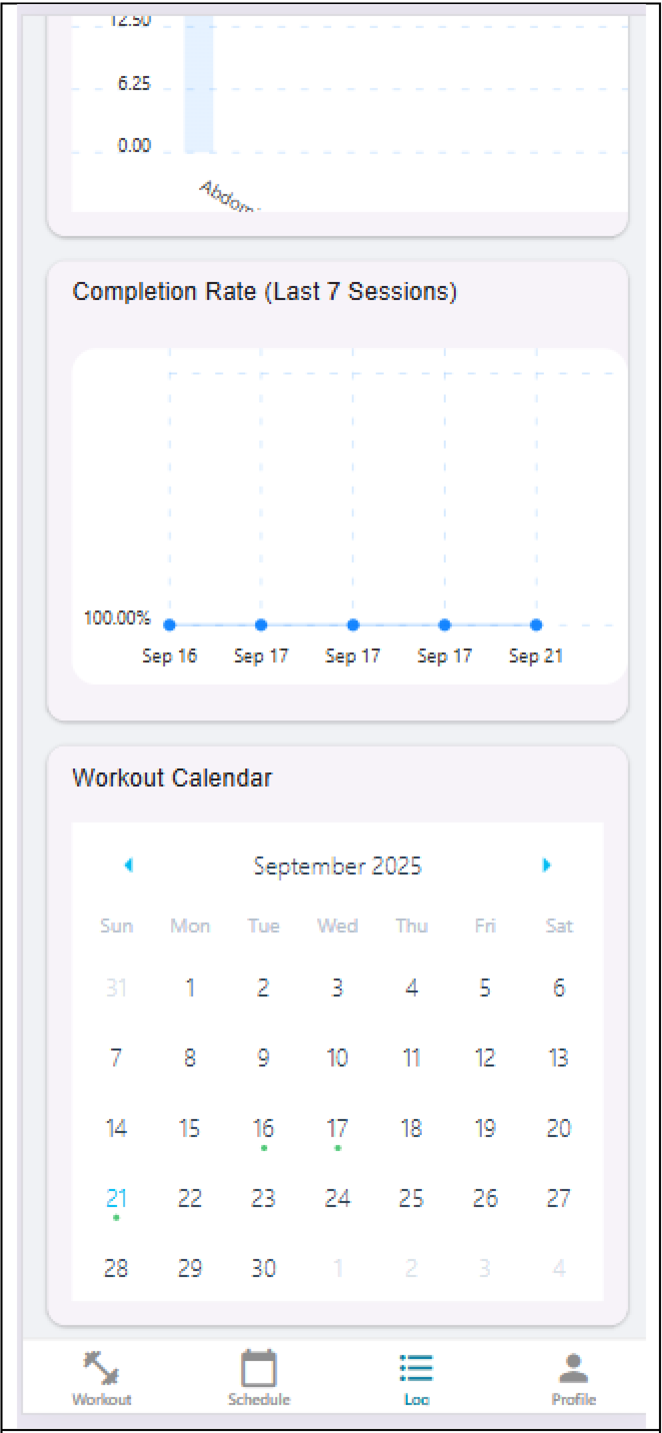


Figure 5.3.4.9 log screen part 3

5.4 Implementation Issues and Challenges

For the challenges, the first is the incompleteness of the dataset of workout details. The dataset of workouts is highly skewed to strength and yoga. The cardio and HIIT workout dataset is generated by ChatGPT which causes low quality of the dataset and there are no images and videos. To overcome this in the future, the workout dataset can be obtained from the authorized fitness website manually, and get the image and video. If there is no images and videos, Image generation model can be used to generate an image or video of a workout with the instructions and a few sample images and videos. The generated image and video are then evaluated by an authorized expert.

Apart from that, the decision of not to include the integration of health apps such as the Samsung app, causes the app to lack rich, contextual health metrics such as heart rate. The proposed app does not plan to ask the user to manually count the heart rate and input the system due to the concern of time-consuming manual input, causing user fatigue, and inaccuracy for the data. This will cause the loss of user, which breaks the main goal of the proposed app. The HIIT exercise is highly dependent on the heart rate as only reaching 85-95% of the peak heart rate (HRpeak) can be considered as High intensity, and it varies among different user body situations. In this case, the user might under intensity when doing a HIIT workout and reduce the effectiveness of the workouts. To solve this, the application should support the health app integration and smart device integration to record the user's heart rate while they are doing a HIIT workout.

5.5 Concluding Remark

In conclusion, the system implementation starts with creating a machine learning model for workout recommendation. After that, services needed, such as Supabase, FastAPI backend, and Google Cloud Console, are configured before proceeding to frontend development. The last one is the frontend react native development. The needed libraries are imported and the components. The components are connected and interact with each other to have a complete app.

Chapter 6 :System Evaluation And Discussion

6.1 System Testing and Performance Metrics

In this section, a structured testing strategy was developed and executed. This ensure the reliability, functionality, and effectiveness of the personalized fitness application to achive the objective**.** Testing can be categorized into 3 types, which are response time, recommendation relevancy, and functional correctness.

1. Response Time (Quantitative)

These metric measures backend and frontend latency, measuring the time (in seconds) from when a request start to when a response is returned. It is a critical indicator of system efficiency and has a direct impact on the responsiveness of the app so that the user will not to keep waiting until fatigue. The user will quit the app if the waiting time is too long

Target: For all update profile, schedule confirmation, recommendation, and feedback endpoints such as /recommend, /feedback/injury, the target response time was set at under 1s, and under 2s for the log dashboard loaded, ensuring smooth and responsive interactions

2. Recommendation Relevance (Qualitative)

This metric assesses the appropriateness and logical quality of workout recommendations produced by the system. It evaluates whether the recommendation engine provides results that are contextually relevant and meet the user's personalized needs.

Test scenarios were created to simulate specific contexts, such as submitting feedback for an injury, and the resulting recommendations were manually inspected. For example, recommendations for a user who reported a “knee” injury were checked to ensure that exercises that could hurt the knee would not be recommended.

3. Functional Correctness (Binary)

This metric of a binary Pass/Fail metric determines whether a feature or function operates exactly as expected, without producing errors or unintended behavior. It spans the full application workflow, from user interface interactions to database transactions.

Functional correctness was tested through predefined test cases covering core user journeys, including registration, profile updates and feedback submission with the edge cases. Each test outcome was classified as “Pass” if the feature executed correctly, or “Fail” if deviations or errors were observed.

6.2 Testing Setup and Result

6.2.1 Response Time test case.

For the recommendation and all feedback flow, the latency test is < 1s. For the schedule suggestion and workout log dashboard load, it is also < 1s and <2s. This shown that the latency is very low and it will not causes user to keep waiting. However, to confirm the schedule time zone, the latency is > 1s. To solve it in the future, the process can be done in parallel to reduce the latency. Another thing is the current test is a test on a localhost and it do not represent the actual networking flow latency. When the app in moves to production, the tools like Postman can be used to check the latency when passing through the network.

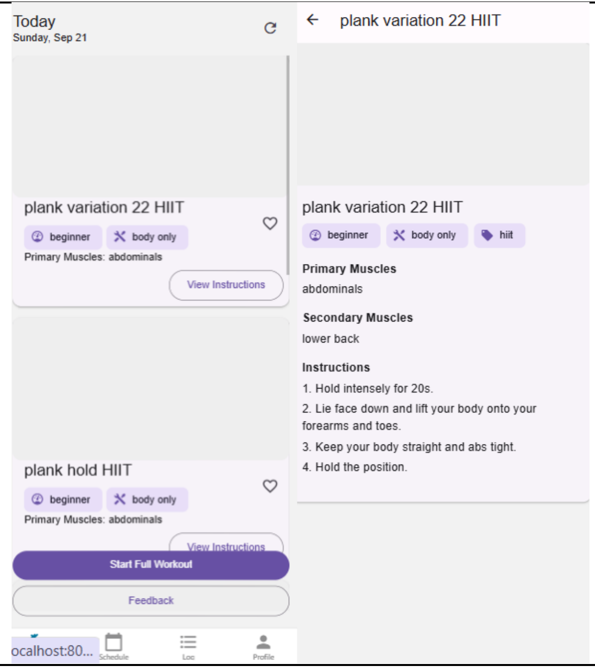
6.2.2 Recommendation Relevance Test Case

The test case will go through each goal test case and each feedback test case. First, the profile data is set to a different goal with the same fitness level, which is beginner, and the equipment is any. The injury notes remain null. Fat loss should get HIIT, muscle gain get strength, endurance get cardio and flexibility get yoga as expected.

Profile data

- Goal: fat loss/ muscle gain /endurance/flexibility
- fitness level: beginner
- equipment: any
- injury not: null

For the fat loss goal case, the HIIT workout is returned as expected shown in the side-by-side figure 6.2.2.1 of recommendation workouts list and details. For the muscle gain case goal case, the strength workout is returned as expected, shown in the side-by-side figure 6.2.2.2 of recommendation workouts list and details. For the endurance goal case, the cardio workout is returned as expected shown in the side-by-side figure 6.2.2.3 of recommendation workouts list and details. For the flexibility goal case, the yoga workout is returned as expected, shown in the side-by-side figure 6.2.2.4 of the recommendation workouts list and details.


Today
Sunday, Sep 21

plank variation 22 HIIT
beginner | body only
Primary Muscles: abdominals
View Instructions

plank hold HIIT
beginner | body only
Primary Muscles: abdominals
View Instructions
Start Full Workout
Feedback

plank variation 22 HIIT

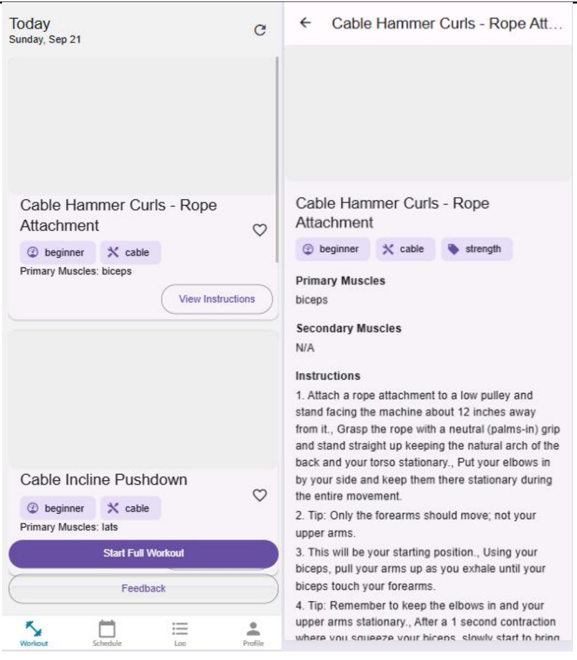
plank variation 22 HIIT
beginner | body only | hiit

Primary Muscles
abdominals

Secondary Muscles
lower back

Instructions
1. Hold intensely for 20s.
2. Lie face down and lift your body onto your forearms and toes.
3. Keep your body straight and abs tight.
4. Hold the position.

Figure 6.2.2.1 Test fat loss return HIIT


Today
Sunday, Sep 21

Cable Hammer Curls - Rope Attachment
beginner | cable
Primary Muscles: biceps
View Instructions

Cable Incline Pushdown
beginner | cable
Primary Muscles: lats
Start Full Workout
Feedback

Cable Hammer Curls - Rope Att...

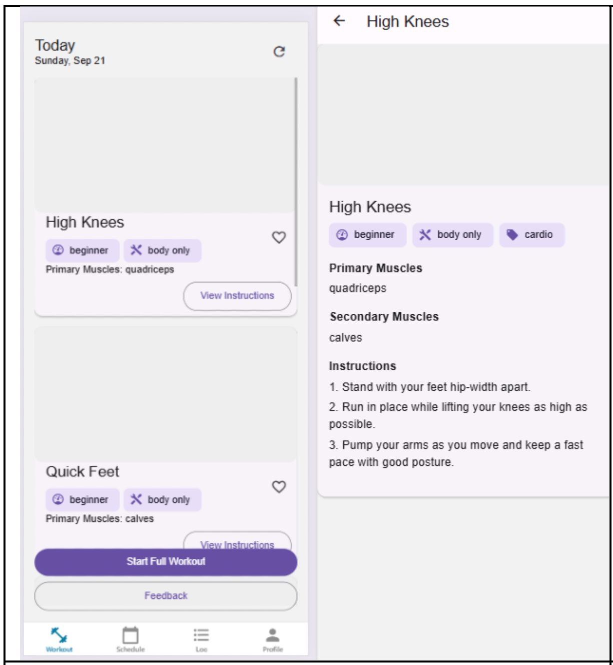
Cable Hammer Curls - Rope Attachment
beginner | cable | strength

Primary Muscles
biceps

Secondary Muscles
N/A

Instructions
1. Attach a rope attachment to a low pulley and stand facing the machine about 12 inches away from it., Grasp the rope with a neutral (palms-in) grip and stand straight up keeping the natural arch of the back and your torso stationary., Put your elbows in by your side and keep them there stationary during the entire movement.
2. Tip: Only the forearms should move; not your upper arms.
3. This will be your starting position., Using your biceps, pull your arms up as you exhale until your biceps touch your forearms.
4. Tip: Remember to keep the elbows in and your upper arms stationary., After a 1 second contraction where you squeeze your biceps, slowly start to bring

Figure 6.2.2.2 Test muscle gain return strength



High Knees
beginner · body only
Primary Muscles: quadriceps

Quick Feet
beginner · body only
Primary Muscles: calves

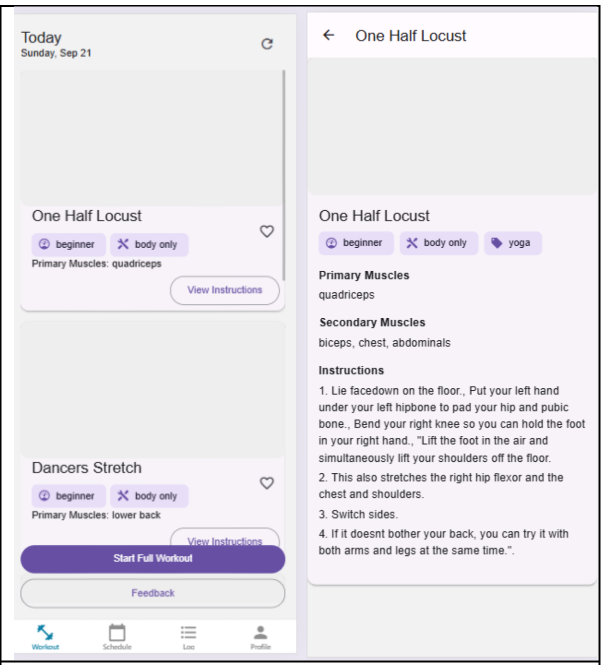
High Knees
beginner · body only · cardio

Primary Muscles
quadriceps

Secondary Muscles
calves

Instructions
1. Stand with your feet hip-width apart.
2. Run in place while lifting your knees as high as possible.
3. Pump your arms as you move and keep a fast pace with good posture.

Figure 6.2.2.3 Test endurance return cardio



One Half Locust
beginner · body only
Primary Muscles: quadriceps

Dancers Stretch
beginner · body only
Primary Muscles: lower back

One Half Locust
beginner · body only · yoga

Primary Muscles
quadriceps

Secondary Muscles
biceps, chest, abdominals

Instructions
1. Lie facedown on the floor., Put your left hand under your left hipbone to pad your hip and pubic bone., Bend your right knee so you can hold the foot in your right hand., "Lift the foot in the air and simultaneously lift your shoulders off the floor.
2. This also stretches the right hip flexor and the chest and shoulders.
3. Switch sides.
4. If it doesnt bother your back, you can try it with both arms and legs at the same time.".

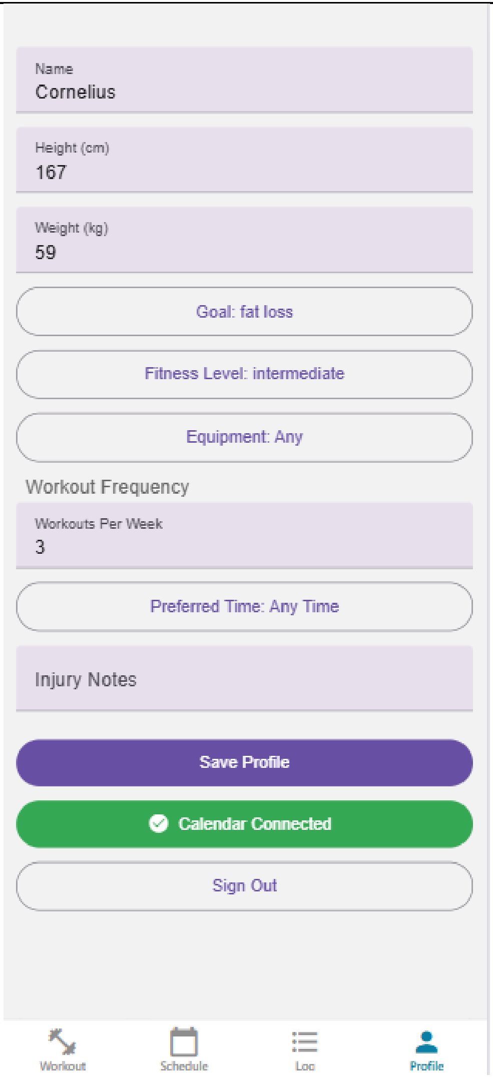
Figure 6.2.2.4 Test flexibility return yoga

Next is for testing the feedback system, the profile data will be set up for goal as fat loss, fitness level as intermediate, the equipment as any and the injury notes as null. For testing busy feedback, the goal is set to muscle gain. The expectation for boring feedback is returned different workouts, but still remains the profile data input request. For feedback busy, it expects to return a HIIT workout and set the workout per week to 1 in the profile. For feedback difficult, it is expected to lower one level of fitness level and return those workouts. For feedback injury, it expected to exclude workout that worsen the injured parts for example set the injury notes to knee and shoulder. For favorite feedback, it is expected to return workouts similar to the favorite workout while retaining the user profile data request needs.

Profile data

- Goal: fat loss/ muscle gain (for testing busy)
- fitness level: intermediate
- equipment: any
- injury not: null / knee, shoulder (for testing injury)

In Figure 6.2.2.5 and Figure 6.2.2.6, the profile and the workout recommendation before the feedback logic is triggered. In Figure 6.2.2.7, different workouts are recommended, but retain user needs, which are HIIT, intermediate, and any equipment, as expected. In Figure 6.2.2.8, the workout recommendation is lowering 1 level from intermediate to beginner as expected. But the update is not reflected in the profile data to lower 1 level for this manual case which is not expected. This issue can be solved in the future by rechecking the update flow. In Figure 6.2.2.9, the goal is set to muscle gain, and the workout per week is 3 before testing the feedback busy. In Figure 6.2.2.10, the workout recommendation turns into HIIT workouts and the workout per week turns into 1 as expected. In Figure 6.2.2.11, the knee and shoulder is submit for a feedback injury. In Figure 6.2.2.12, the adjusted workouts that were excluded the injury parts are returned. The primary and second muscle that affected by those workouts will not causes further injury to the mentioned injury based on the table of body part and muscle mapping in table 5.2.1.7.1. However, the climber HIIT exercise do will further worsen the knee as movement puts repetitive stress and pressure on the knee joint. The current logic is not robust as it only considers the primary and secondary muscles that will be trained by the workout without the joint affection. The climber exercise do not directly train the muscles of affected by the knee, such as the quadriceps, hamstrings, and calves, but it will affect as the joint. For Figure 6.2.2.13, the workout that similar to the favorite workout is return as expected which is basic variation of HIIT series workouts, including the favorite workout itself since user love it. These still meet the user profile data needs. However, if the user switches to another goal, such as muscle gain, it will prioritize to meet the user's needs instead of similarity to the favorite workouts, but still consider similarity with the favorite one shown in Figure 6.2.2.14.



Name
Cornelius

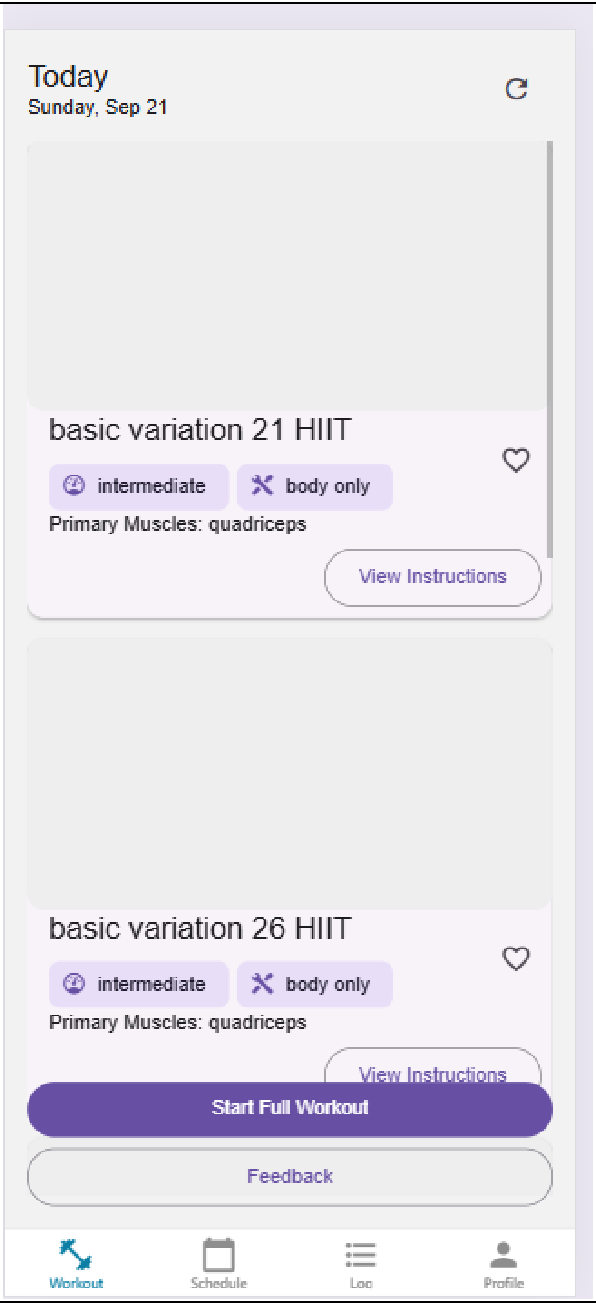
Height (cm)
167

Weight (kg)
59

Workout Frequency

Workouts Per Week
3

Figure 6.2.2.5 Profile before feedback test



Today
Sunday, Sep 21

basic variation 21 HIIT
intermediate | body only
Primary Muscles: quadriceps

basic variation 26 HIIT
intermediate | body only
Primary Muscles: quadriceps

Figure 6.2.2.6 Workout before feedback test

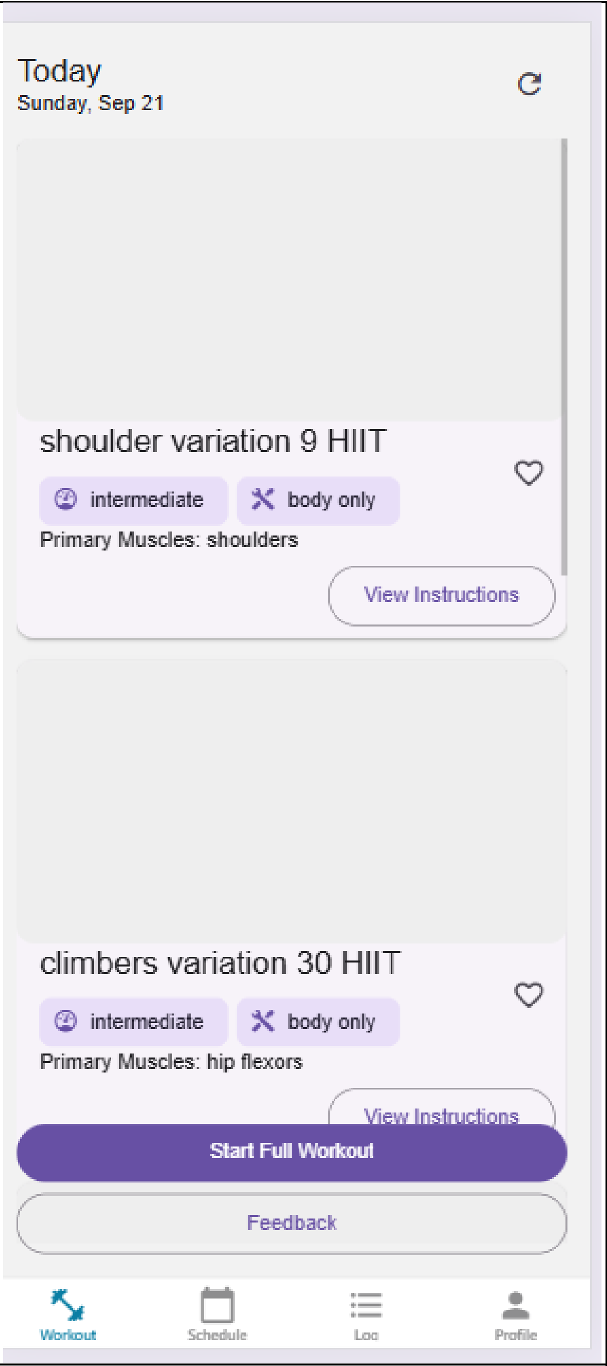
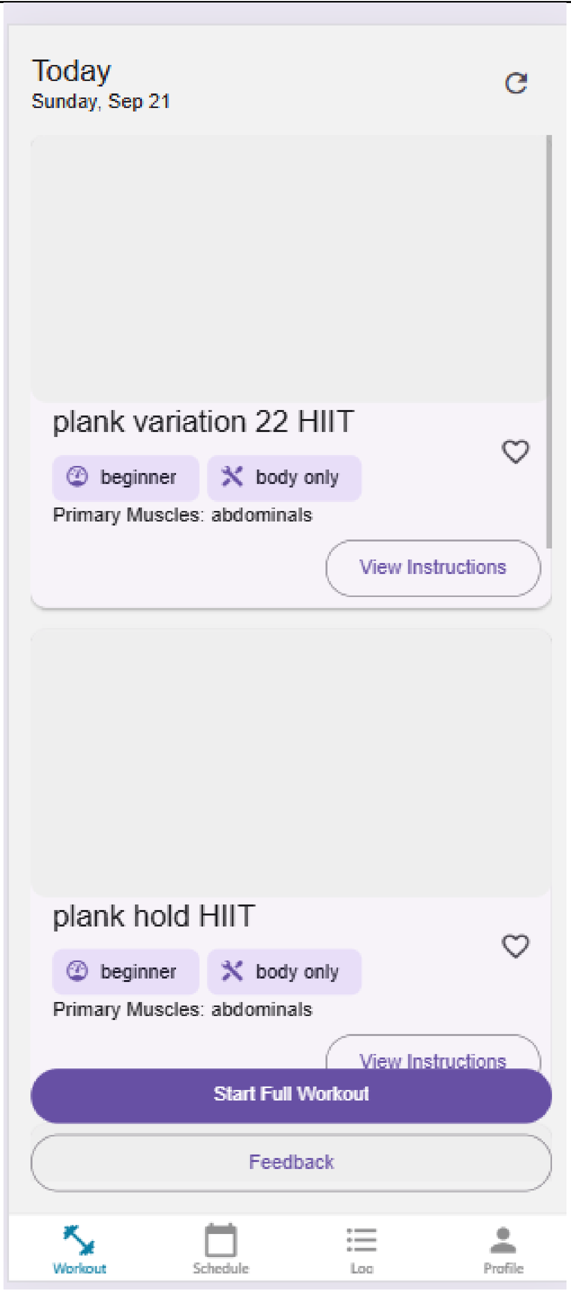
	

Figure 6.2.2.7 feedback boring with different workouts but retain user needs	Figure 6.2.2.8 feedback difficult lowering 1 level from intermediate to beginner.

Today
Sunday, Sep 21

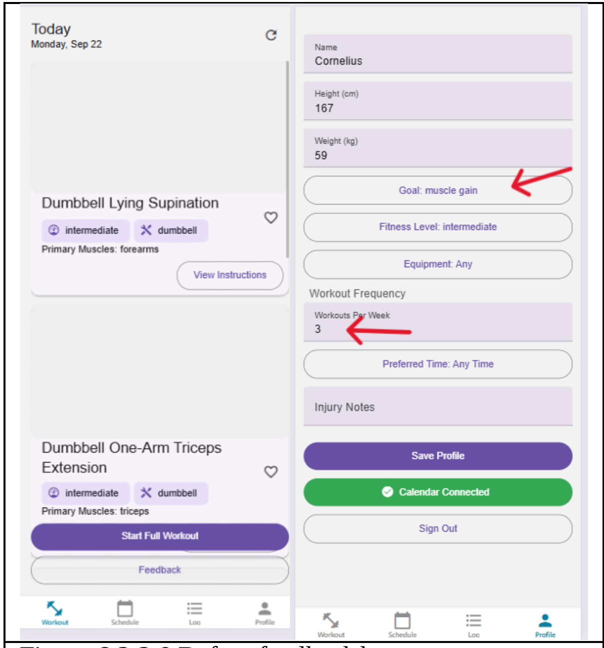
shoulder variation 9 HIIT
intermediate | body only
Primary Muscles: shoulders

climbers variation 30 HIIT
intermediate | body only
Primary Muscles: hip flexors

Today
Sunday, Sep 21

plank variation 22 HIIT
beginner | body only
Primary Muscles: abdominals

plank hold HIIT
beginner | body only
Primary Muscles: abdominals



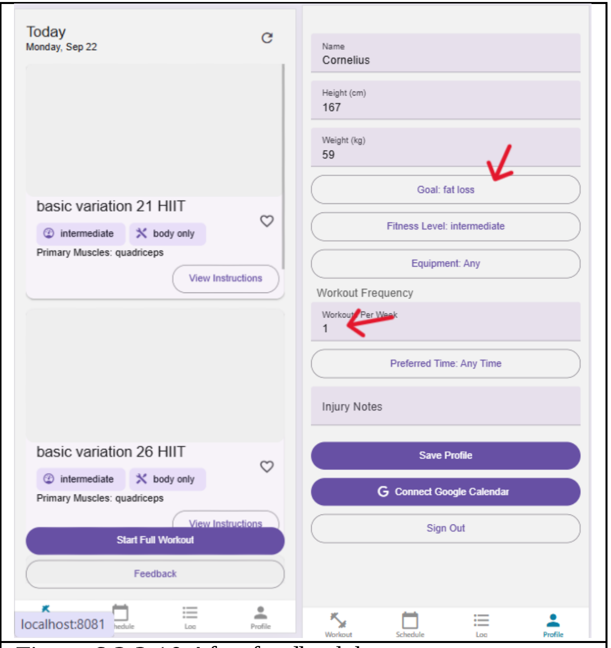
Today
Monday, Sep 22

Dumbbell Lying Supination
intermediate · dumbbell
Primary Muscles: forearms
View Instructions

Dumbbell One-Arm Triceps Extension
intermediate · dumbbell
Primary Muscles: triceps
Start Full Workout
Feedback

Name: Cornelius
Height (cm): 167
Weight (kg): 59
Goal: muscle gain
Fitness Level: intermediate
Equipment: Any
Workout Frequency
Workouts Per Week: 3
Preferred Time: Any Time
Injury Notes
Save Profile
Calendar Connected
Sign Out

Figure 6.2.2.9 Before feedback busy



Today
Monday, Sep 22

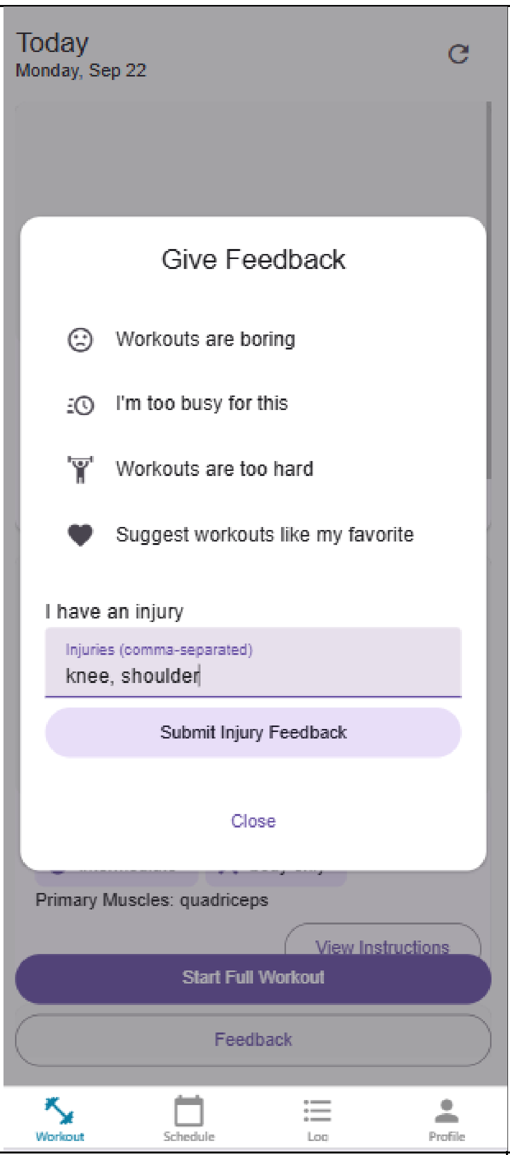
basic variation 21 HIIT
intermediate · body only
Primary Muscles: quadriceps
View Instructions

basic variation 26 HIIT
intermediate · body only
Primary Muscles: quadriceps
View Instructions
Start Full Workout
Feedback

localhost:8081

Name: Cornelius
Height (cm): 167
Weight (kg): 59
Goal: fat loss
Fitness Level: intermediate
Equipment: Any
Workout Frequency
Workouts Per Week: 1
Preferred Time: Any Time
Injury Notes
Save Profile
Connect Google Calendar
Sign Out

Figure 6.2.2.10 After feedback busy



Today
Monday, Sep 22

Give Feedback

Workouts are boring

I'm too busy for this

Workouts are too hard

Suggest workouts like my favorite

I have an injury

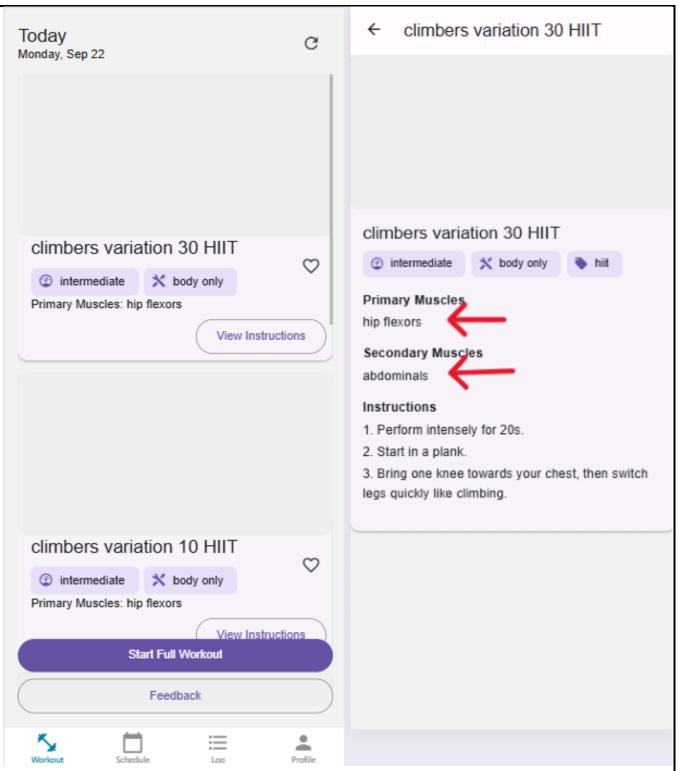
Injuries (comma-separated)
knee, shoulder

Submit Injury Feedback

Close

Primary Muscles: quadriceps

Figure 6.2.2.11 Filling injury with knee and shoulder



Today
Monday, Sep 22

climbers variation 30 HIIT
intermediate body only
Primary Muscles: hip flexors

climbers variation 10 HIIT
intermediate body only
Primary Muscles: hip flexors

climbers variation 30 HIIT

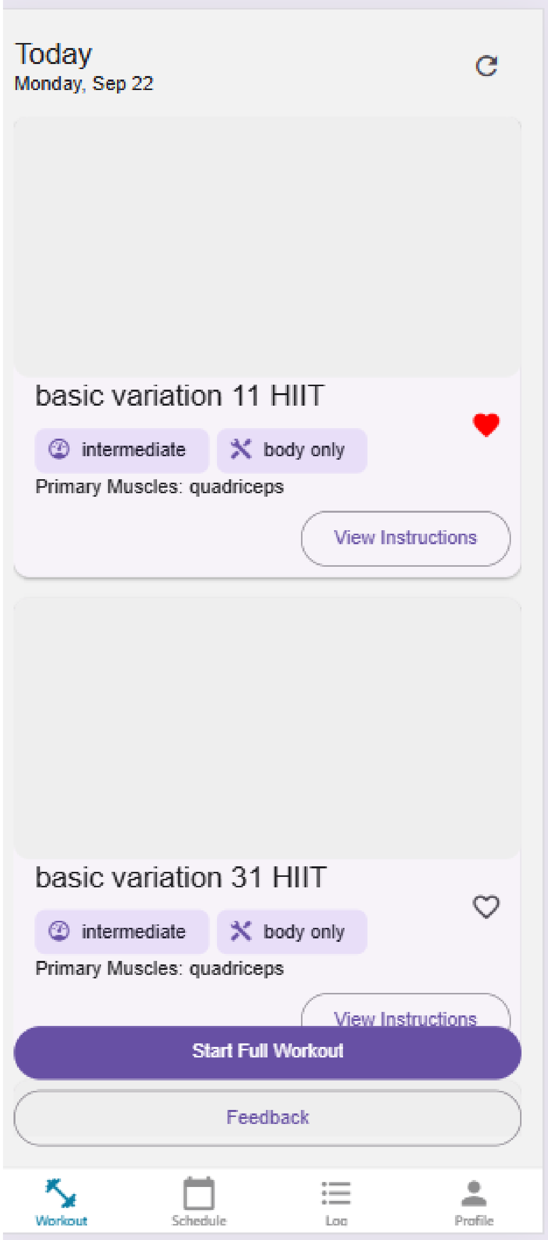
climbers variation 30 HIIT
intermediate body only hiit

Primary Muscles
hip flexors

Secondary Muscles
abdominals

Instructions
1. Perform intensely for 20s.
2. Start in a plank.
3. Bring one knee towards your chest, then switch legs quickly like climbing.

Figure 6.2.2.12 After feedback injury

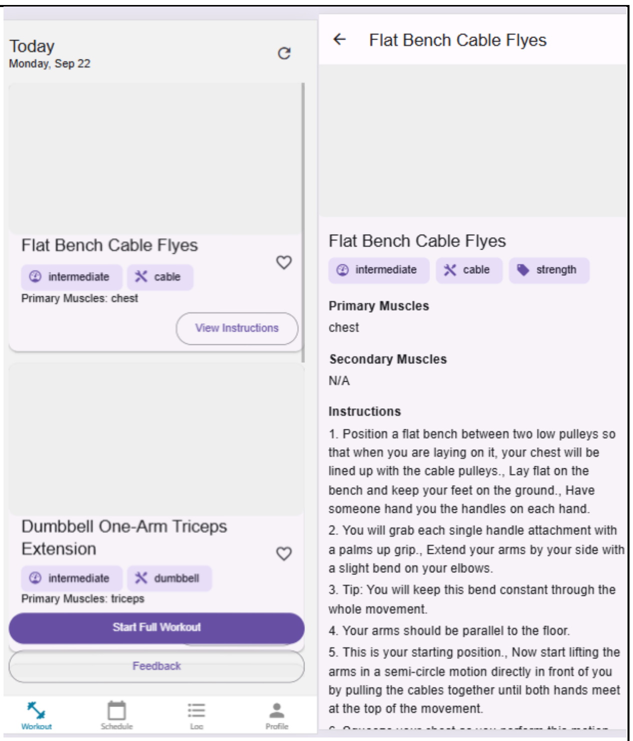


Today
Monday, Sep 22

basic variation 11 HIIT
intermediate body only
Primary Muscles: quadriceps
View Instructions

basic variation 31 HIIT
intermediate body only
Primary Muscles: quadriceps
View Instructions

Figure 6.2.2.13 favorite feedback output



Today
Monday, Sep 22

Flat Bench Cable Flyes
intermediate cable
Primary Muscles: chest
View Instructions

Dumbbell One-Arm Triceps Extension
intermediate dumbbell
Primary Muscles: triceps

Flat Bench Cable Flyes

Flat Bench Cable Flyes
intermediate cable strength

Primary Muscles
chest

Secondary Muscles
N/A

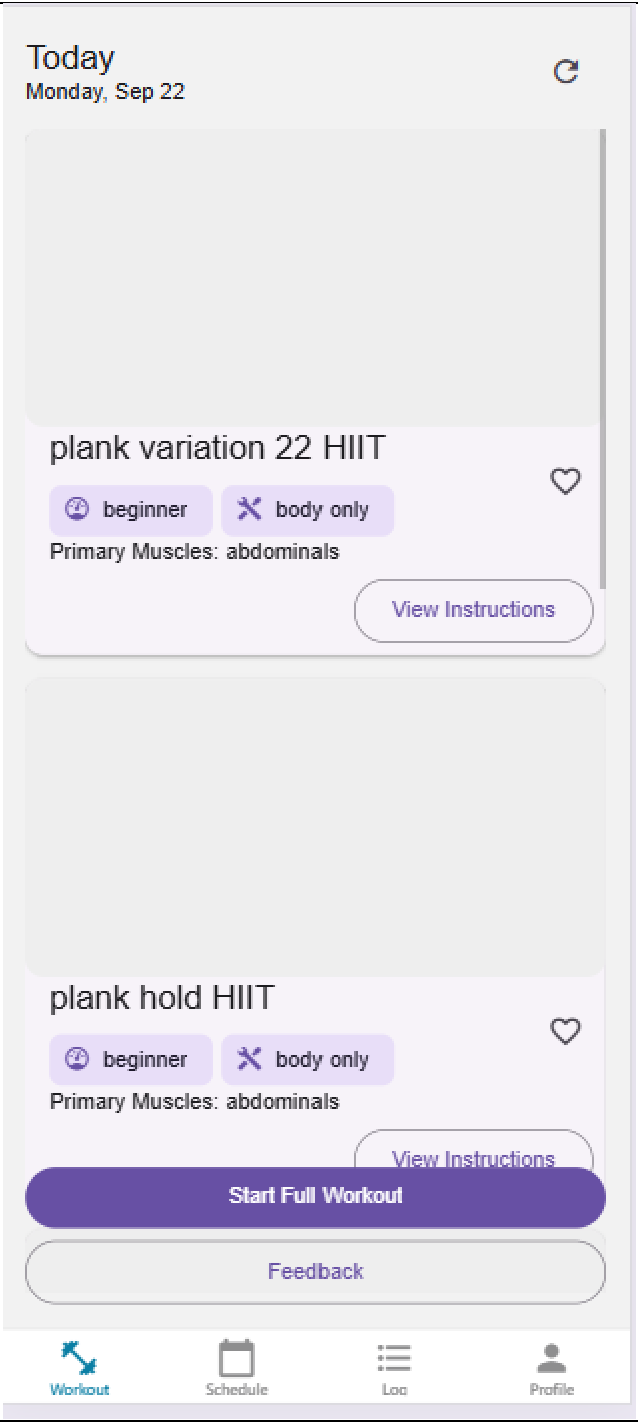
Instructions
1. Position a flat bench between two low pulleys so that when you are laying on it, your chest will be lined up with the cable pulleys., Lay flat on the bench and keep your feet on the ground., Have someone hand you the handles on each hand.
2. You will grab each single handle attachment with a palms up grip., Extend your arms by your side with a slight bend on your elbows.
3. Tip: You will keep this bend constant through the whole movement.
4. Your arms should be parallel to the floor.
5. This is your starting position., Now start lifting the arms in a semi-circle motion directly in front of you by pulling the cables together until both hands meet at the top of the movement.

Figure 6.2.2.14 favorite feedback output when goal is not the same as the favorite

6.2.3 Functional Correctness

For this section, edge cases are tested. The first test is input the weight with a negative value. The expected output is the action fails. In this case, the weight is still set to -1 despite setting constraints in the table. To solve this, the additional logic checking need to be handle in the frontend app. The test failed.

The second is to test the 3 consecutive workouts < 50 % completion rate should trigger the difficult feedback. The expected output is similar to the feedback difficult. By using the same condition in the previous feedback difficult, the 3 consecutive workouts < 50 % completion rate is created. The fitness level is lower and then recommend the corresponding workouts as expected when reload shown in figure 6.2.3.1. It does not refresh immediately due to the useffect checking. This can be solve by switching to usefocuseffect.,



Today
Monday, Sep 22

plank variation 22 HIIT
beginner
body only
Primary Muscles: abdominals

plank hold HIIT
beginner
body only
Primary Muscles: abdominals

Figure 6.2.2.15 lowering fitness level after 3 consecutive <50 % completion rate of workout logs

6.3 Project Challenges

The current challenges is the implementation relies on a direct mapping between injured body parts and their associated primary muscles. For example, a reported "knee" injury excludes workouts that target the quadriceps, hamstrings, and calves. This approach is effective for filtering out exercises that explicitly train the muscles surrounding an injured joint. However, it fails to consider exercises that, while not directly targeting those muscles, still place biomechanical stress on the joint itself. For instance, the climber HIIT exercise under the current logic would remain available to a user with a knee injury. But the movement inherently involves repetitive knee flexion and extension under load, which could worsen the injury.

This exposes a fundamental gap in the muscle-centric filtering strategywhich does not account for the kinematic and biomechanical demands of an exercise. To address this limitation, the filtering model would need to be expanded to include data on joint movement, range of motion, and impact load for each exercise, thereby providing a more robust and injury-aware recommendation system.

The second challenge is the abandoned workout type prediction model that build in FYP1. In the planning, personal and physiological data of the user is used to predict the user's suitable workout type (Cardio, HIIT, Strength, or Yoga). The methodology involved data preprocessing, training a baseline Random Forest model, analyzing feature importance, and retraining the model with selected features. To ensure robustness, three additional models—Logistic Regression, K-Nearest Neighbors (KNN), and Gradient Boosting were also tested for comparison. The results show that all models performed poorly. The best-performing model, Logistic Regression, achieved an accuracy of only 0.2872, which is significantly below a practical threshold for reliable classification. The root cause is the lack of separability between the workout categories within the dataset. Input features such as average and Calories Burned, considerable overlap across users labeled for different workout types is shown in Figure 6.2.3. This causes the model to hard to differentiate the workout type.

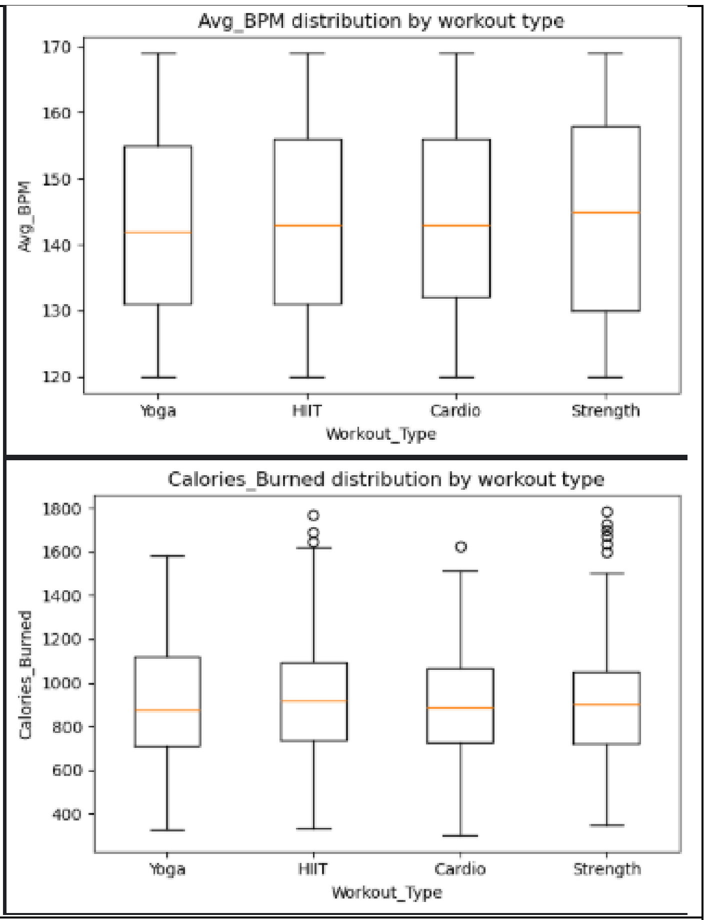


Figure 6.2.3 Overlapping label for average heart rate beat per Minutes and calories burned for 4 workout types

6.4 Objectives Evaluation

This section evaluates the success of the project by comparing the final implemented system against the three primary objectives

Objective 1: Reduce Early-Stage and Later-Stage Abandonment

Evaluation: achieve

The first objective was achieved by making a personalization application. Rather than relying on a generic workout plan, the system generates tailored workouts list from the very first interaction using user profile data such as their primary fitness goal, fitness level, and available equipment. These steps are minimal but effective in reducing the massive of data input needs before start-up for user friendly. This approach directly addresses early abandonment driven by dissatisfaction with generic plans and fatigue, while ongoing personalization mitigates later-stage abondon.

Objective 2: Handle User's Life Uncertainty

This objective was divided into two sub-categories: injury-related uncertainty and time-related uncertainty.

Sub-objective 2a: Injury Uncertainty
Evaluation: partially achieved

The system includes an injury-aware recommendation feature, enabling users to report injuries via the feedback modal. The backend then filters workouts, excluding exercises that directly target the injured muscles. This prevents worsening injuries and can remains consistency. However, testing revealed a limitation, which is that the filtering is muscle-centric and does not account for exercises that stress joints without directly targeting the surrounding muscles. This may worsen the injury, and the user blind trust the app will think that there is not problem and make it worse.

Sub-objective 2b: Time Uncertainty
Evaluation: achieved

Time flexibility was fully addressed through two mechanisms. First, the application integrates with Google Calendar, allowing the system to fetch user availability from calendar event and suggest optimal workout slots. Second, the feedback busy option, which generates an alternative plan centered on short, high-intensity interval training (HIIT) sessions for a time-efficient workout. These features provide a robust solution for managing time uncertainty, ensuring that users can adapt their routines without losing momentum.

Objective 3: Maintain Motivation and Promote Consistency

Evaluation: Achieved

Several features were implemented to sustain user engagement and motivation. A "Weekly Streak" is calculated and displayed on the progress screen to encourage consistency. Push notifications act as motivational reminders for scheduled workouts. For stay motivation, the instant manual difficult adjustment feedback system allows users to report when workouts feel too hard and prompts the system to immediately adjust workout plans. This can prevent user from losing motivation due to the difficulty of the workouts and failure. In addition, an automated mechanism detects repeated failures to complete workouts and proactively suggests easier alternatives. These allow addressing motivational challenges, reducing the risk of abandonment caused by difficulty mismatch.

6.5 Concluding Remark

In conclusion, the app is considered to achieve the expected output in the recommendation and feedback logic, except the injury feedback. It also performs well in scheduling the workout and log presentation. This can be helpful to keep the user consistent by reducing the friction of doing a workout by automatically scheduling and adapting to real-life different cases.

Chapter 7: Conclusion and Recommendation

7.1 Conclusion

In conclusion, this project is aimed at solving the abandonment issue in the early and later stages, which commonly results from a lack of personalization, uncertainty in daily routines, and loss of motivation. The Personalized Workout Planner was proposed as a comprehensive solution to tackle the issue by fulfilling the need for customized fitness experiences that adapt to individual goals and conditions. A novel aspect of this project is the development of a dynamic, user-centric workout planner that provides a seamless experience in handling real-life uncertainty. Personalized workouts are recommended with a feedback system based on boredom, injury, time constraints, and difficulty level. Safe workouts are recommended by excluding exercises that could worsen injuries. Due to the muscle centrix only problem, the feedback injury system needs to refine to consider joints for more robust and safe recommendation. Next, HIIT exercise alternatives for a busy schedule is provided to ensure time-efficient workouts. Automated scheduling is implemented by accessing with users' Google Calendars to plan the workout plan and improve the workout experience. Personalized, seamless and dynamic workouts that are provided can improve the user workout experience and reduce the abandonment rate.

7.2 Recommendation

The first recommendation is to integrate the system with health app platforms and smart wearable devices such as Samsung health. The current design lacks rich metrics such as heart rate. This allows access to real-time biometric data, enabling the ability to accurately check heart rate for workout intensity checking for meet HIIT standard. Real-time feedback can be included during HIIT sessions, ensuring users remain within their optimal training zone or need to have rest. This enhancement would effectively transform the system from a static workout planner into an interactive coach. Furthermore, integration would enable more precise tracking such as calories and fat percentage.

The second is that exercises should be tagged with metadata for joint stress and biomechanical impact. For instance, exercises involving “high knee impact” could be automatically excluded for users reporting a knee injury. This refinement would allow the system to make more nuanced and safer recommendations, significantly reducing the risk of injury and strengthening user trust in the platform. The generative AI can be integrated to have a double evaluation for workout safety and provide chat-based responses for things to take care of or rehabilitation.

The third is to improve user-friendliness and experience by providing short, looping instructional videos that demonstrate proper form. This is particularly important for beginners, who often require visual guidance to correctly perform complex movements. This make application would make learning safer, easier, and more engaging. Besides, the option to choose a free slot by user direct input instead of the nearest free slot in the schedule feature can let the user have more control.

REFERENCES

[1] D. Mogaveera, V. Mathur, and S. Waghela, “e-Health Monitoring System with Diet and Fitness Recommendation using Machine Learning,” in Proc. 6th Int. Conf. Inventive Comput. Technol. (ICICT), Jan. 2021, doi: 10.1109/ICICT50816.2021.9358605.

[2] S. Iso-Ahola, “Conscious-nonconscious processing explains why some people exercise but most don’t,” J. Nat. Sci., vol. 3, no. 6, e384. [Online]. Available: https://www.researchgate.net/publication/317411823_Conscious-Nonconscious_Processing_Explains_Why_Some_People_Exercise_but_Most_Don't

[3] J. Niess and P. W. Woźniak, “Supporting meaningful personal fitness: the Tracker Goal Evolution Model,” in Proc. 2018 CHI Conf. Human Factors Comput. Syst. (CHI '18), New York, NY, USA: ACM, 2018, pp. 1–12, doi: 10.1145/3173574.3173745.

[4] T. Karlsen, I.-L. Aamot, M. Haykowsky, and Ø. Rognmo, “High Intensity Interval Training for Maximizing Health Outcomes,” Prog. Cardiovasc. Dis., vol. 60, no. 1, pp. 67–77, 2017, doi: 10.1016/j.pcad.2017.03.006.

[5] P. Kidman, R. Curtis, A. Watson, and C. Maher, “When and why adults abandon lifestyle behavior and mental health mobile apps: scoping review,” J. Med. Internet Res., vol. 26, 2024, Art. no. e56897, doi: 10.2196/56897.

[6] C. Attig and T. Franke, “Abandonment of personal quantification: A review and empirical study investigating reasons for wearable activity tracking attrition,” Comput. Human Behav., vol. 102, pp. 223–237, 2020, doi: 10.1016/j.chb.2019.08.025.

[7] R. Aicale, D. Tarantino, and N. Maffulli, “Overuse injuries in sport: a comprehensive overview,” J. Orthop. Surg. Res., vol. 13, no. 1, Dec. 2018, doi: 10.1186/s13018-018-1017-5.

[8] HealthifyMe, 16 Apr. 2025, “HealthifyMe – Calorie Counter,” distributed by Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.healthifyme.basic

[9] Fitbod, 18 Apr. 2025, “Fitbod: Workout & Gym Planner,” distributed by Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.fitbod.fitbod

[10] MyFitnessPal, Inc., 22 Apr. 2025, “MyFitnessPal: Calorie Counter,” distributed by Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.myfitnesspal.android

[11] Simple Design Ltd., 21 Apr. 2025, “Lose Weight in 30 Days,” distributed by Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=loseweight.weightloss.workout.fitness

[12] Injurymap, 27 Aug. 2024, “Injurymap,” distributed by Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.injurymap.injurymap

[13] H. E. Lee and J. Cho, “What motivates users to continue using diet and fitness apps? Application of the uses and gratifications approach,” Health Commun., vol. 32, no. 12, pp. 1445–1453, Jun. 2016, doi: 10.1080/10410236.2016.1167998.

[14] www.sportsinjuryclinic.net, 27 Jan. 2025, “Sports Injury Rehabilitation,” distributed by Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.sportsinjuryclinic.rehab.program

[15] M. Phaswana, D. V. Khumalo, D. Constantinou, et al., “Effectiveness of high-intensity interval training and moderate-intensity continuous training on cardiometabolic health in university labourers,” World J. Clin. Med., vol. 6, no. 1, pp. 25–32, 2024, doi: 10.18772/26180197.2024.v6n1a4.

[16] Meta Platforms, Inc., "React Native Documentation," 2025. [Online]. Available: https://reactnative.dev/

[17] Expo, Inc., "Expo Documentation," 2025. [Online]. Available: https://docs.expo.dev/

[18] W. Danielsson, ‘React Native application development : A comparison between native Android and React Native’, Dissertation, 2016. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A998793&dswid=1904

[19] Supabase, Inc., "Supabase Documentation," 2025. [Online]. Available: https://supabase.com/docs

[20] S. Tiangolo, "FastAPI," 2025. [Online]. Available: https://fastapi.tiangolo.com/

[21] Google, "Google Calendar API Overview," 2025. [Online]. Available: https://developers.google.com/calendar/api/guides/overview

[22] Google, "OAuth 2.0 for Mobile & Desktop Apps," 2025. [Online]. Available: https://developers.google.com/identity/protocols/oauth2

[23]Supabase, “JWTs,” *Supabase Documentation*. [Online]. Available: https://supabase.com/docs/guides/auth/jwts. [Accessed: Sept. 18, 2025].

[24] Supabase, “Row Level Security,” *Supabase Documentation*. [Online]. Available: https://supabase.com/docs/guides/database/postgres/row-level-security. [Accessed: Sept. 18, 2025].

[25] DAREBEE, “100 HIIT Workouts by DAREBEE,” Darebee.com, 2025. https://darebee.com/100-hiit-workouts.html (accessed Sep. 21, 2025).

[26]FastAPI, “CORS (Cross-Origin Resource Sharing),” *FastAPI Tutorial*, FastAPI. [Online]. Available: https://fastapi.tiangolo.com/tutorial/cors/ (accessed **Sep. 20, 2025**).

[27] V. Khorasani, Sep. 2024, “Gym Members Exercise Dataset,” distributed by Kaggle. [Online]. Available: https://www.kaggle.com/datasets/valakhorasani/gym-members-exercise-dataset

[28] R. Gradien, Apr. 2023, “exercises_cleaned,” distributed by Github. [Online]. Available: https://github.com/RalphGradien/HomeWorkoutRecommendations/blob/main/data/exercises_cleaned.csv

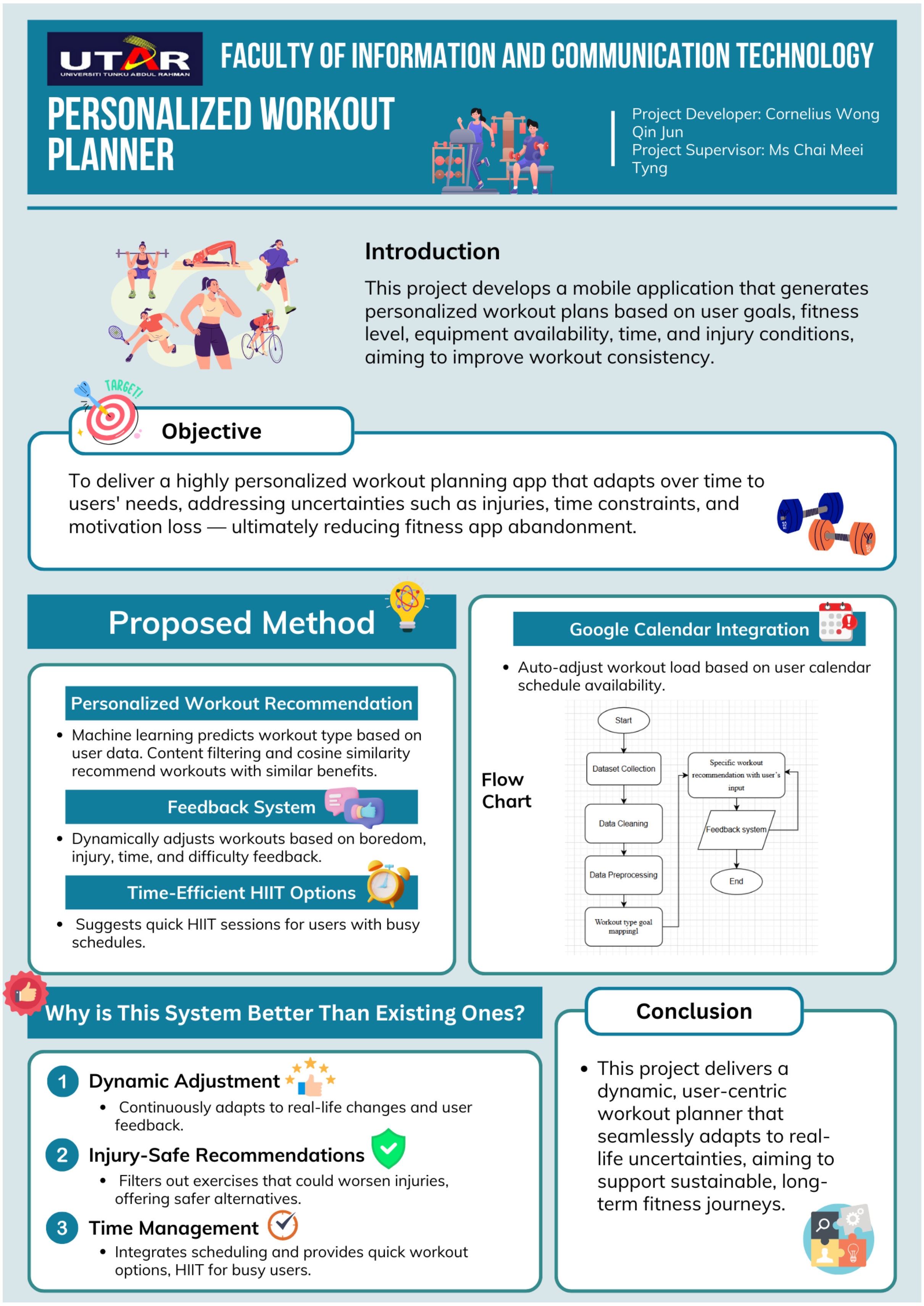
APPENDIX

GitHub link:

https://github.com/CWUtar/Personalized_Workout_Planner.git

https://github.com/CWUtar/backend_api.git

POSTER



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

PERSONALIZED WORKOUT PLANNER

Project Developer: Cornelius Wong Qin Jun
Project Supervisor: Ms Chai Meei Tyng

Introduction

This project develops a mobile application that generates personalized workout plans based on user goals, fitness level, equipment availability, time, and injury conditions, aiming to improve workout consistency.

Objective

To deliver a highly personalized workout planning app that adapts over time to users' needs, addressing uncertainties such as injuries, time constraints, and motivation loss — ultimately reducing fitness app abandonment.

Proposed Method

Personalized Workout Recommendation

- Machine learning predicts workout type based on user data. Content filtering and cosine similarity recommend workouts with similar benefits.

Feedback System

- Dynamically adjusts workouts based on boredom, injury, time, and difficulty feedback.

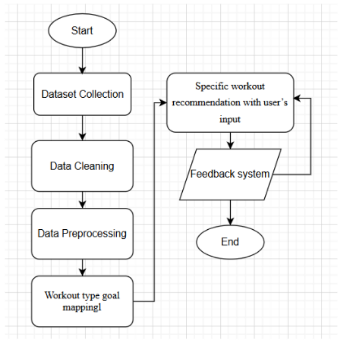
Time-Efficient HIIT Options

- Suggests quick HIIT sessions for users with busy schedules.

Google Calendar Integration

- Auto-adjust workout load based on user calendar schedule availability.

Flow Chart


Start
Dataset Collection
Data Cleaning
Data Preprocessing
Workout type goal mappingl
Specific workout recommendation with user's input
Feedback system
End

Why is This System Better Than Existing Ones?

1. **Dynamic Adjustment**
   - Continuously adapts to real-life changes and user feedback.
2. **Injury-Safe Recommendations**
   - Filters out exercises that could worsen injuries, offering safer alternatives.
3. **Time Management**
   - Integrates scheduling and provides quick workout options, HIIT for busy users.

Conclusion

- This project delivers a dynamic, user-centric workout planner that seamlessly adapts to real-life uncertainties, aiming to support sustainable, long-term fitness journeys.