National University of Science and Technology (NUST)
School of Mechanical & Manufacturing Engineering(SMME)
ME-16: Section – A: Batch 2024

4/25/2025

Programming Lab
Project Lab Proposal

National University of Science and Technology (NUST)
School of Mechanical & Manufacturing Engineering(SMME)
ME-16: Section – A: Batch 2024

Lab Project Proposal

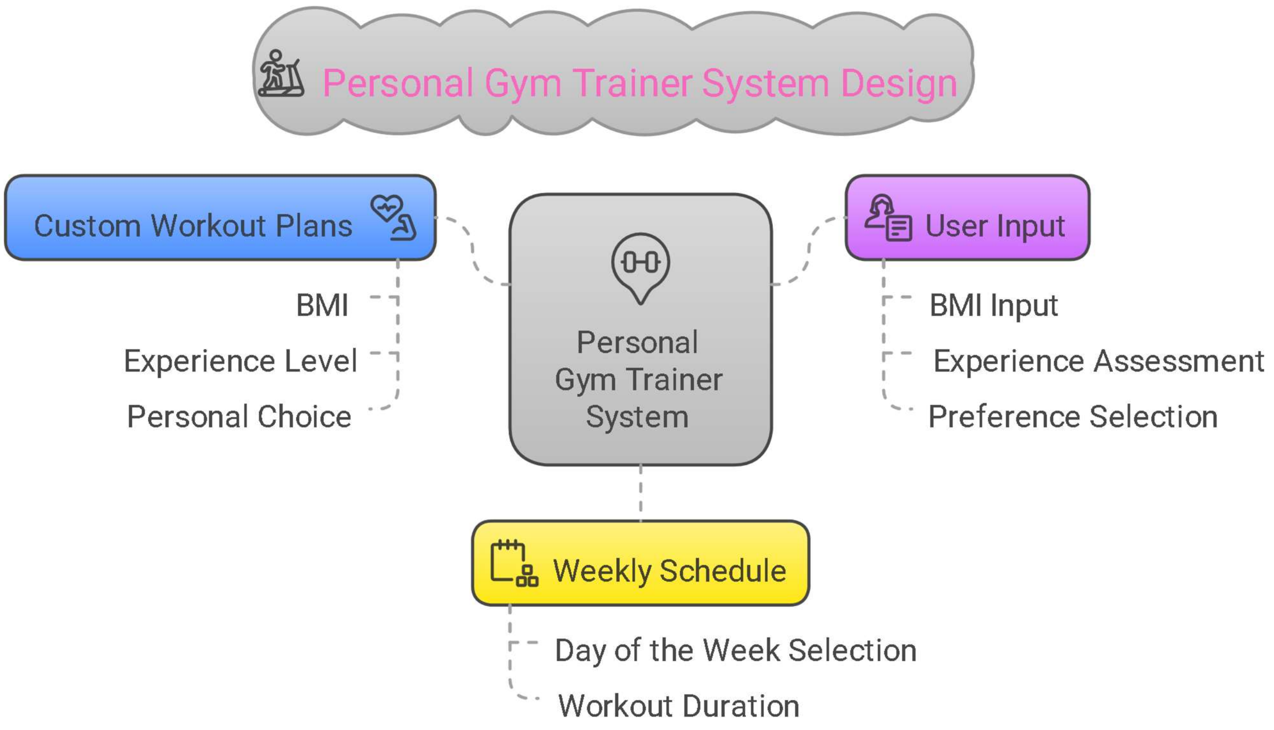
Group Members Description:

National University of Science and Technology (NUST)			
School of Mechanical & Manufacturing Engineering(SMME)			
S.No	Submitted by:	CMS ID:	Section
01	**Frais Asghar**	**516900**	**A**
02	**Hamid Ali Khan**	**501290**	**A**
03	**Danial Haider**	**511042**	**A**
04	**Mujaddid Khawaja**	**521377**	**A**

Abstract:

In this project, we aim to develop a Personal Gym Trainer using C++ which will provide users with custom workout plans depending on their Body Mass Index (BMI), experience with the application, and their personal choice. The system will be designed in a way tailored to be applicable over a week long period, with users selecting the Day of the Week for their workout.

The idea behind making the Gym Trainer dependent on BMI and expertise is to avoid giving the user a workout that may be unsuitable for them, while the inclusion of personal choice is so that the user may pick that which they are comfortable with.

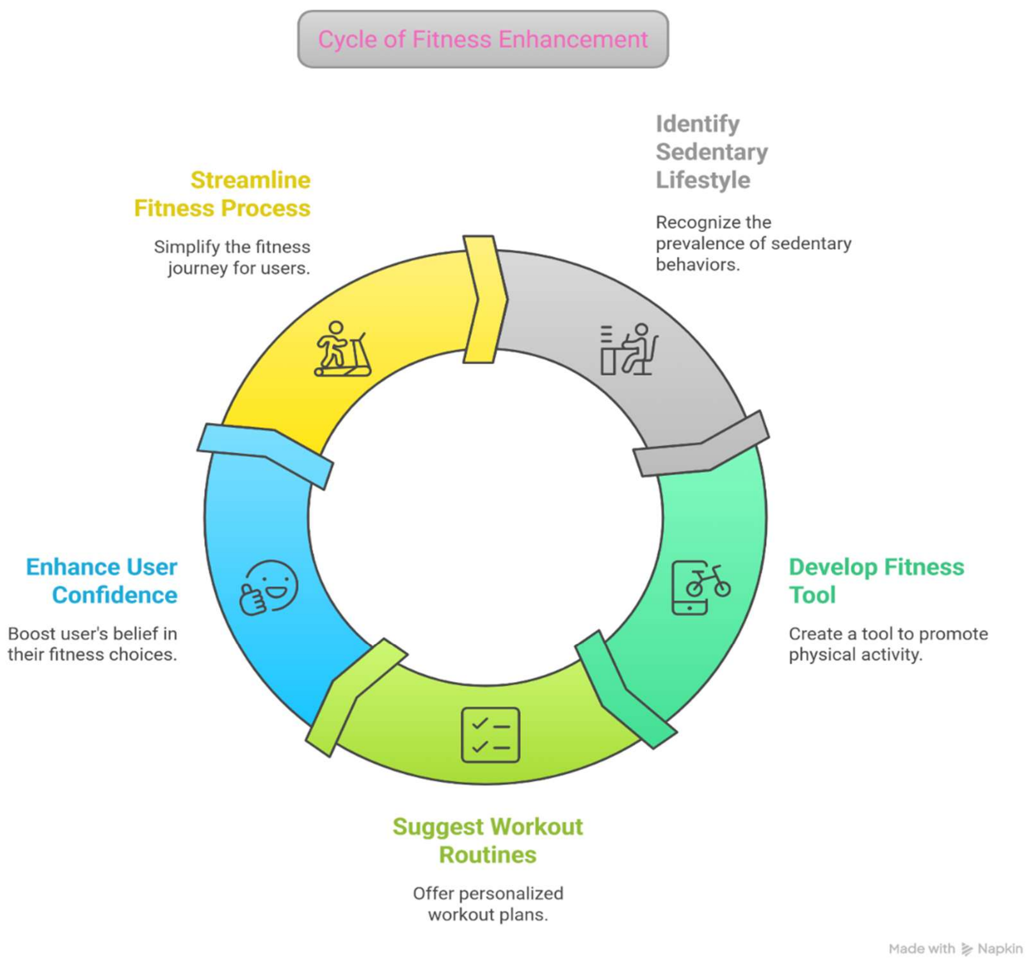


Introduction:

Being Physically Fit is an essential part of good health, guaranteeing a person more energy and better well-being. However, it is also unfortunately something that has become less common nowadays owing to the sedentary lifestyles that we all live.

This has resulted in a demand amongst many people for tools that can contribute to better fitness, especially on the online space, where sedentary behaviors are most encouraged.

Therefore, it can often be beneficial to have a multipurpose tool to recommend and suggest specific workout routines that may help a user to feel more confident in their choices while also streamlining the process for them in their attempts for them becoming physically fit. Programming languages such as C++ in particular, allow for the development of such training applications, primarily on the basis of their conditional statements which allow for several choices, and thus allow for a differentiation of different users depending on category while also allowing for more general choices

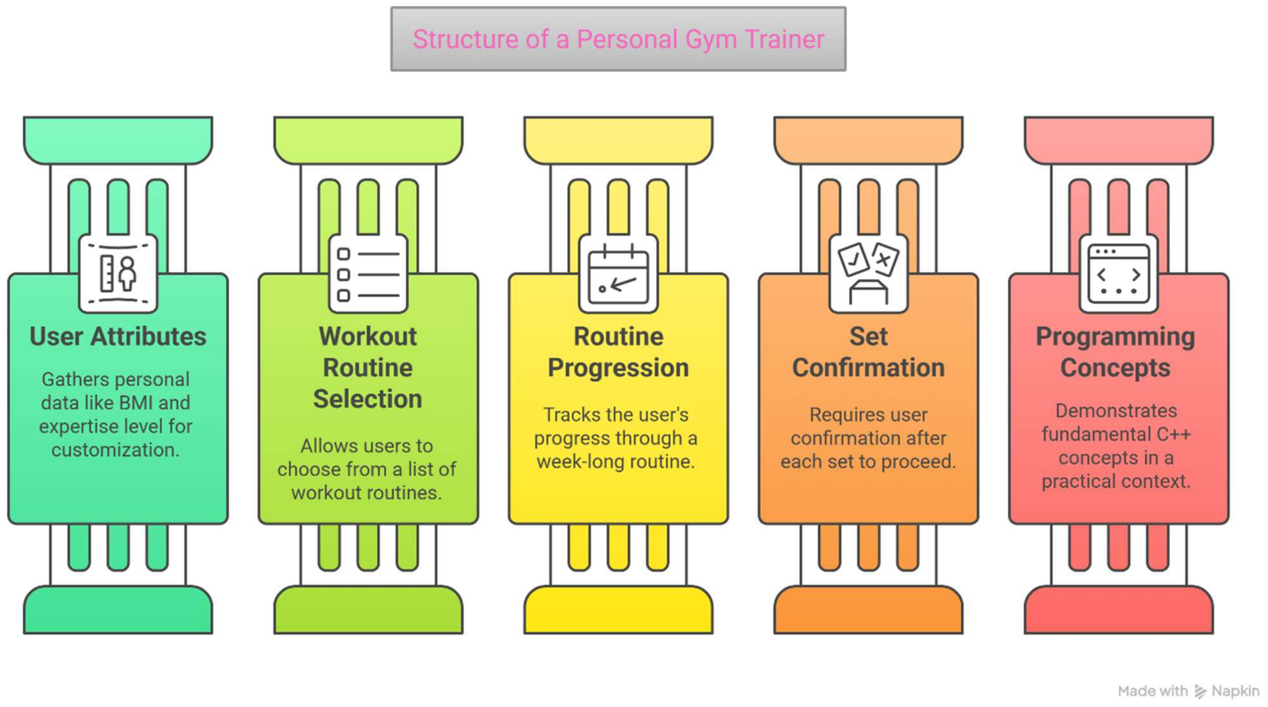

Cycle of Fitness Enhancement
Identify Sedentary Lifestyle
Recognize the prevalence of sedentary behaviors.
Develop Fitness Tool
Create a tool to promote physical activity.
Suggest Workout Routines
Offer personalized workout plans.
Enhance User Confidence
Boost user's belief in their fitness choices.
Streamline Fitness Process
Simplify the fitness journey for users.
Made with Napkin

Objectives:

The Primary goal of this project is to write up code on C++ that will output a Personal Gym Trainer for the User. The Personal Gym Trainer will then provide the user with a customized workout routine based on their personal attributes such as BMI and expertise level. These will be obtained directly from the user.

The program will also ask the user to specify their chosen workout routine from a list, and then which day of the workout routine it is, with the routine lasting a week. Finally, the routine will involve sets, with the user manually confirming whenever they have finished one set so they may be given the next one.

The other main goal of this project is to demonstrate the use of fundamental programming concepts in C++, such as the use of conditional statements, loops, and user inputs. That is to also show how these concepts are then practically applicable for real world scenarios.

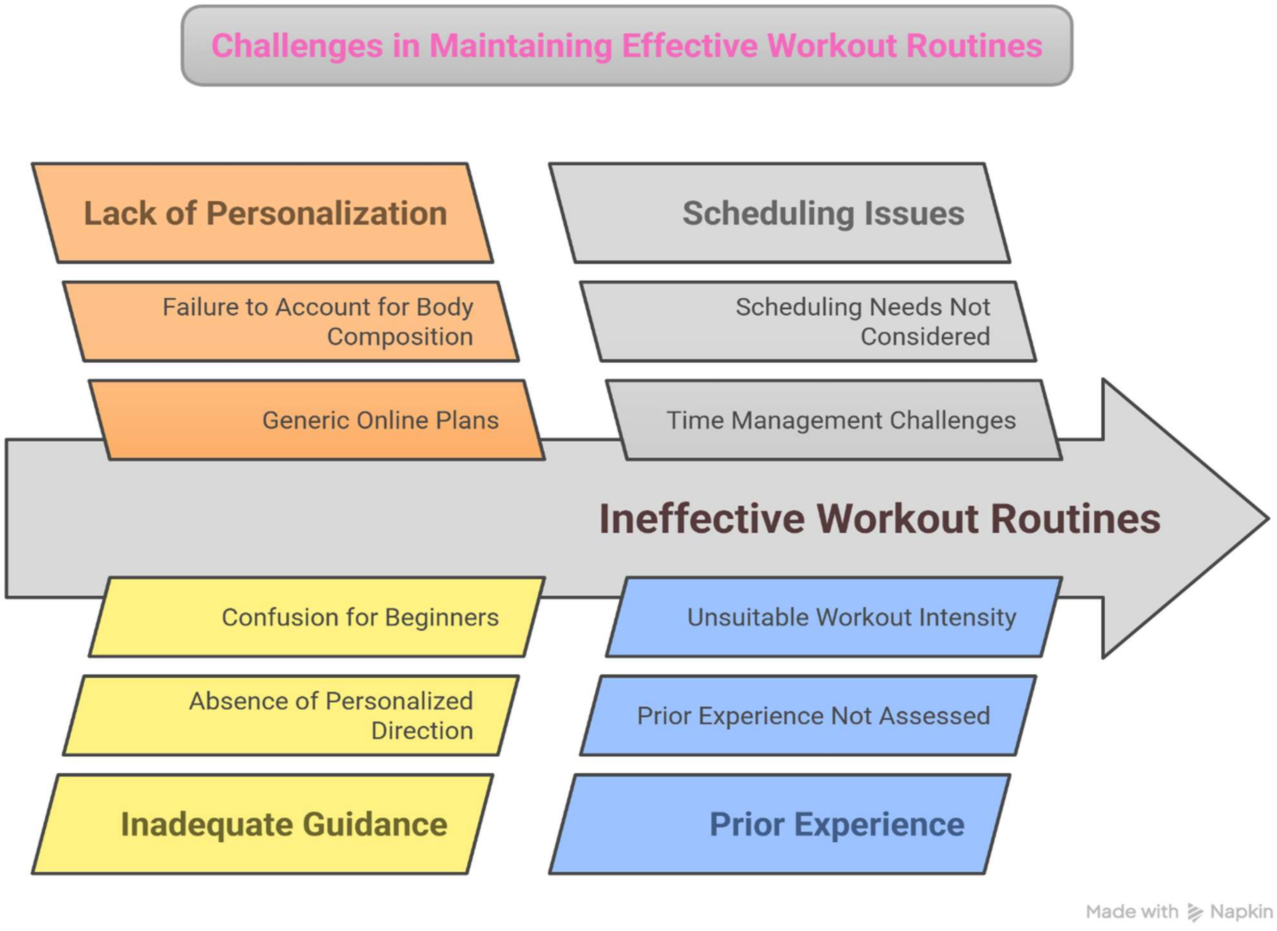

Structure of a Personal Gym Trainer
User Attributes
Gathers personal data like BMI and expertise level for customization.
Workout Routine Selection
Allows users to choose from a list of workout routines.
Routine Progression
Tracks the user's progress through a week-long routine.
Set Confirmation
Requires user confirmation after each set to proceed.
Programming Concepts
Demonstrates fundamental C++ concepts in a practical context.
Made with Napkin

Problem Statement:

Many individuals struggle to maintain an effective workout routine because they lack the guidance to create one tailored to their personal health and fitness levels. Generic online plans often fail to account for body composition, prior experience, or scheduling needs, which can lead to injury, burnout, or a lack of motivation.

The absence of personalized direction can make the journey toward better health confusing and frustrating, especially for beginners who are unsure where to start.

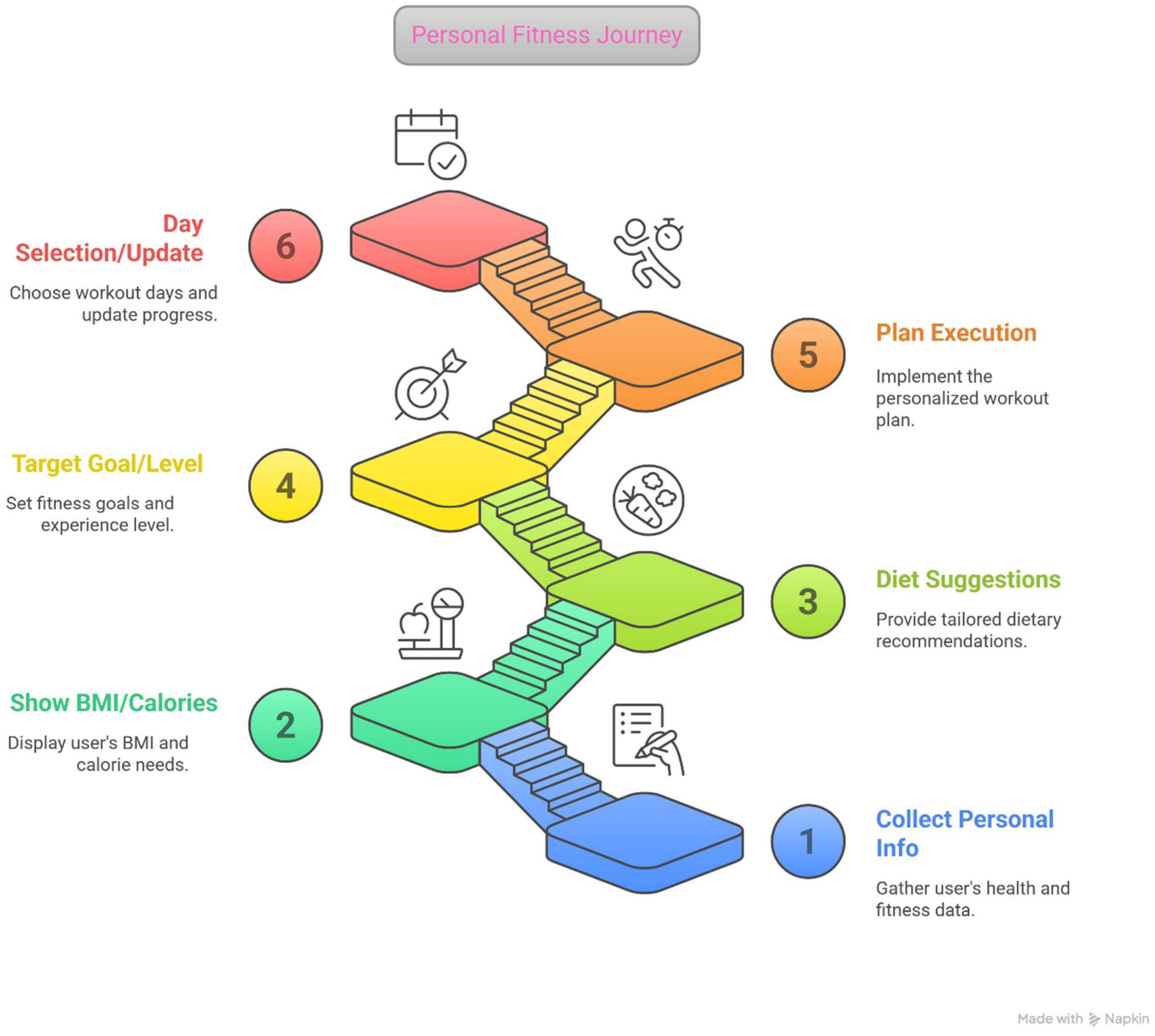
To address this gap, our project proposes a C++ based Personal Gym Trainer program. It serves as an assistant that asks the user for key personal data such as BMI, and experience level, and then returns a set of workout plans suited to those criteria. This not only


Challenges in Maintaining Effective Workout Routines
Lack of Personalization
Failure to Account for Body Composition
Generic Online Plans
Scheduling Issues
Scheduling Needs Not Considered
Time Management Challenges
Ineffective Workout Routines
Confusion for Beginners
Absence of Personalized Direction
Inadequate Guidance
Unsuitable Workout Intensity
Prior Experience Not Assessed
Prior Experience
Made with Napkin

personalizes the fitness experience but also introduces structure into the user’s routine.

Flow Chart:

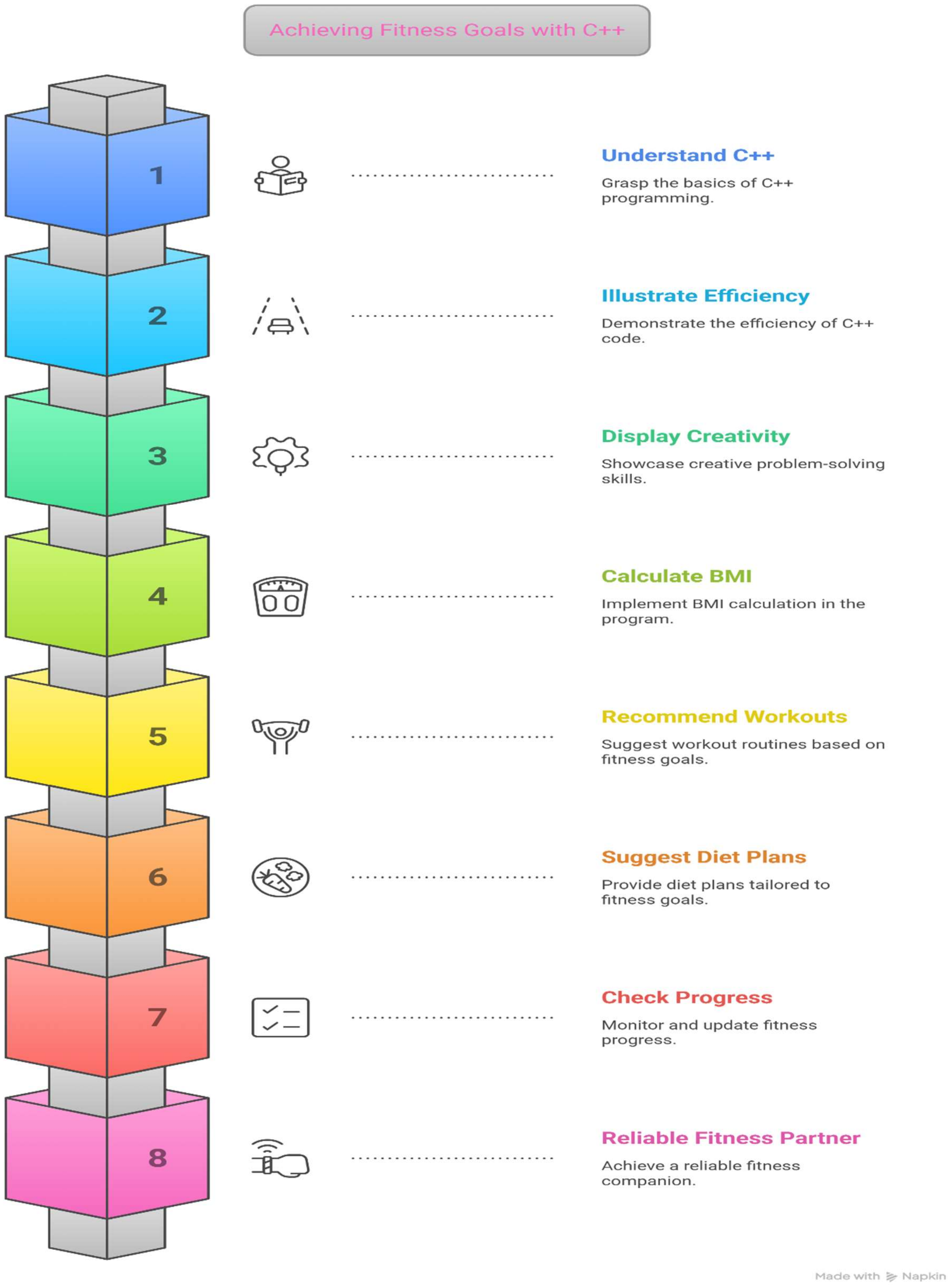
if there is a new user, program will collect personal info, show bmi/calories, diet suggestions, target goal/ level selection, plan execution. And on the next side it allows the user day selection, update.


Personal Fitness Journey
6
Day Selection/Update
Choose workout days and update progress.
5
Plan Execution
Implement the personalized workout plan.
4
Target Goal/Level
Set fitness goals and experience level.
3
Diet Suggestions
Provide tailored dietary recommendations.
2
Show BMI/Calories
Display user's BMI and calorie needs.
1
Collect Personal Info
Gather user's health and fitness data.
Made with Napkin

PROJECT DELIVERABLES:

- Gives a basic and structured understanding of C++.
- The code illustrates the efficiency of C++.
- Displays the creativity and problem solving skill.
- Helpful in both professional and practical life.
- The code calculates and Displays BMI.
- Workout Routines Are Automatically Recommended.
- Diet Plans Are Suggested Based on Fitness Goals.
- Progress is checked and updated.
- Modified programs according to your body type and level.
- Reusable and expendable code with room for additions.
- Provides a reliable partner for your fitness journey.

National University of Science and Technology (NUST)
School of Mechanical & Manufacturing Engineering(SMME)
ME-16: Section – A: Batch 2024


Achieving Fitness Goals with C++
1
Understand C++
Grasp the basics of C++ programming.
2
Illustrate Efficiency
Demonstrate the efficiency of C++ code.
3
Display Creativity
Showcase creative problem-solving skills.
4
Calculate BMI
Implement BMI calculation in the program.
5
Recommend Workouts
Suggest workout routines based on fitness goals.
6
Suggest Diet Plans
Provide diet plans tailored to fitness goals.
7
Check Progress
Monitor and update fitness progress.
8
Reliable Fitness Partner
Achieve a reliable fitness companion.
Made with Napkin

Coding Concepts

Before initializing the process of coding, we made sure to make good use of functions in our code. This was efficient so that we could break down the different days of exercise into chunks of code, and then run them for the specific days that the code applies for depending entirely on user input.

However, before we elaborate on that, it would be beneficial to discuss the variables that are involved in this program.

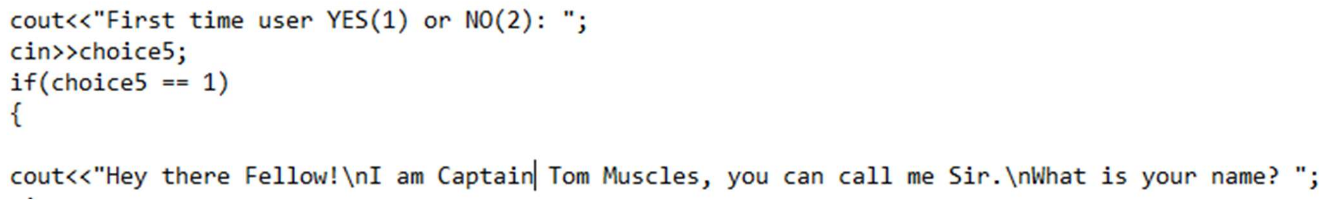
We took the variable for name as a string by using the string library. This was executed as "string name;" That is because the name of the user is unlikely to be a single character, and so we need to account for multiple characters.

The user's weight and choices were all taken as integers. This is because users are unlikely to know their weight to a high degree of accuracy, and it does not majorly impact the results, while for the choices themselves, it is efficient to use numbers to cycle through the decisions that the user makes by asking them to input fairly small numbers instead of a string, a complex digit, or an unnecessary character.

For the gender of the user, we executed the variable as a char, primarily because it is more efficient to have the input be a character (There are many words for either gender, and integer values may cause confusion).

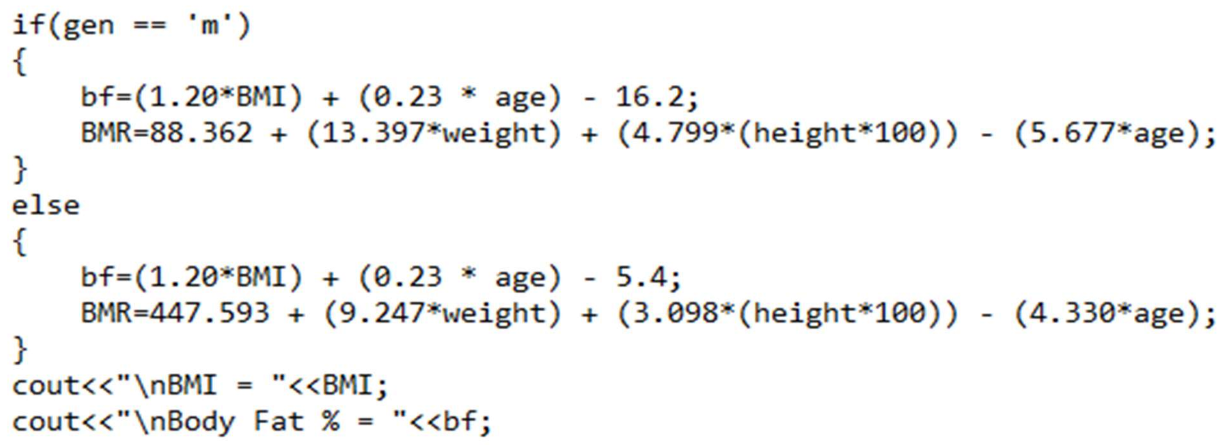
However, for height, BMI, BMR, bf, and calories, we represented them as a double primarily because these are our main values, and so need to have a high degree of accuracy.

The main code itself employed conditional statements itself for the main choices that the user made such as answering whether they had used the program before or not, so as to decide to give them a tutorial or not.



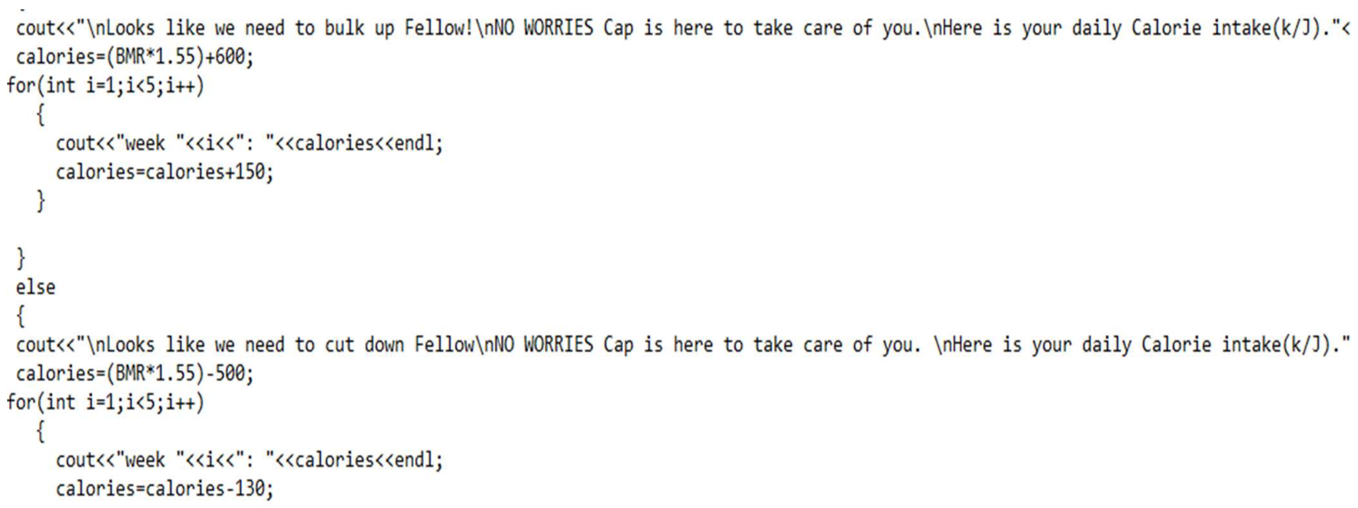
```
cout<<"First time user YES(1) or NO(2): ";
cin>>choice5;
if(choice5 == 1)
{

cout<<"Hey there Fellow!\nI am Captain Tom Muscles, you can call me Sir.\nWhat is your name? ";
```

Conditional statements were also used to make sure the calculations that were performed were accurate for the gender of the user, depending on earlier provided inputs. This made sure that our results would be as accurate as possible.



```
if(gen == 'm')
{
    bf=(1.20*BMI) + (0.23 * age) - 16.2;
    BMR=88.362 + (13.397*weight) + (4.799*(height*100)) - (5.677*age);
}
else
{
    bf=(1.20*BMI) + (0.23 * age) - 5.4;
    BMR=447.593 + (9.247*weight) + (3.098*(height*100)) - (4.330*age);
}
cout<<"\nBMI = "<<BMI;
cout<<"\nBody Fat % = "<<bf;
```

For loops, were primarily used to steadily increase or decrease the caloric intake per week for the user depending on whether they chose to cut down on their food or to bulk up on how much they were eating, with an appropriate output message for either case.

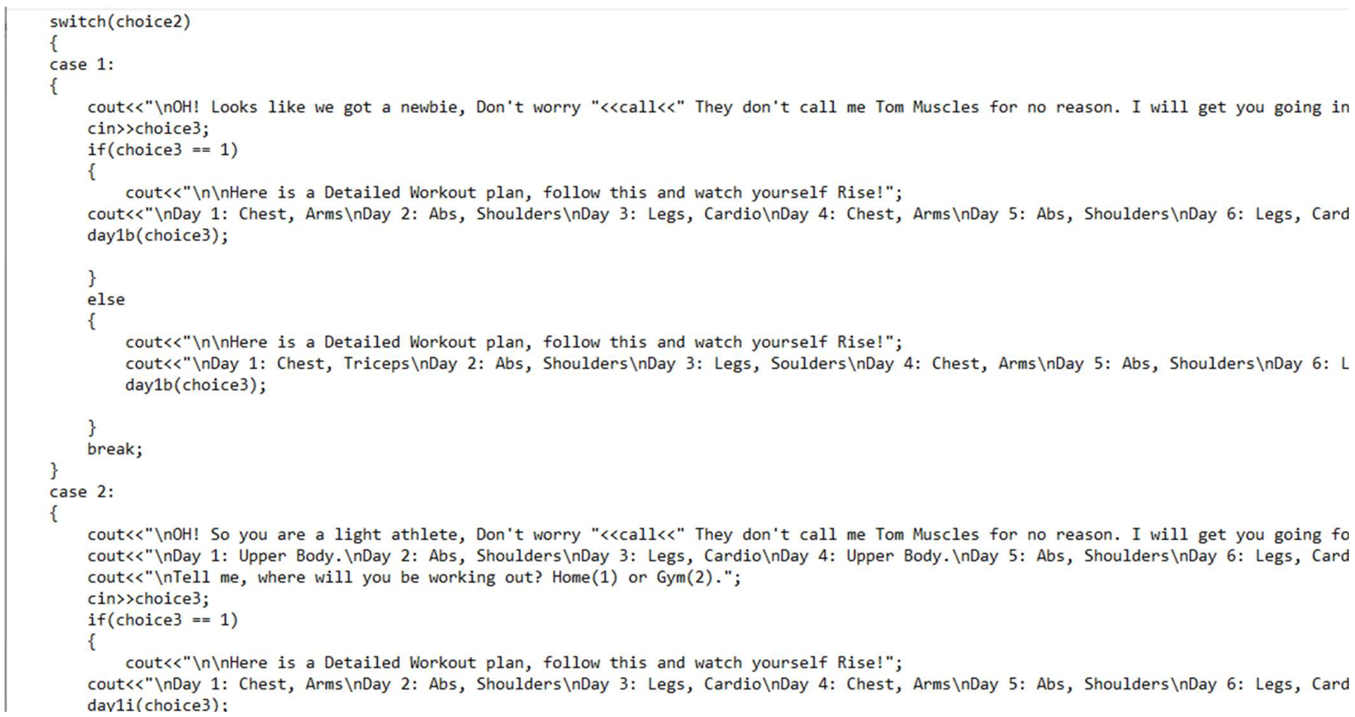


```
 cout<<"\nLooks like we need to bulk up Fellow!\nNO WORRIES Cap is here to take care of you.\nHere is your daily Calorie intake(k/J)."<
 calories=(BMR*1.55)+600;
for(int i=1;i<5;i++)
   {
     cout<<"week "<<i<<": "<<calories<<endl;
     calories=calories+150;
   }

 }
 else
 {
 cout<<"\nLooks like we need to cut down Fellow\nNO WORRIES Cap is here to take care of you. \nHere is your daily Calorie intake(k/J)."
 calories=(BMR*1.55)-500;
for(int i=1;i<5;i++)
   {
     cout<<"week "<<i<<": "<<calories<<endl;
     calories=calories-130;
```

We also used the switch and break statement to apply the cases for the exercise that depended on the user's fitness level specifically, after receiving the necessary input as a choice of the user (1 for beginner, 2 for intermediate, 3 for experienced).

It should be noted that this was done twice, primarily because there were two cases of outputs.

The first case was the situation where the user was using the program for the first time, and thus required a tutorial before they could start the exercise plan, and the other case was the situation where they had used the program before.

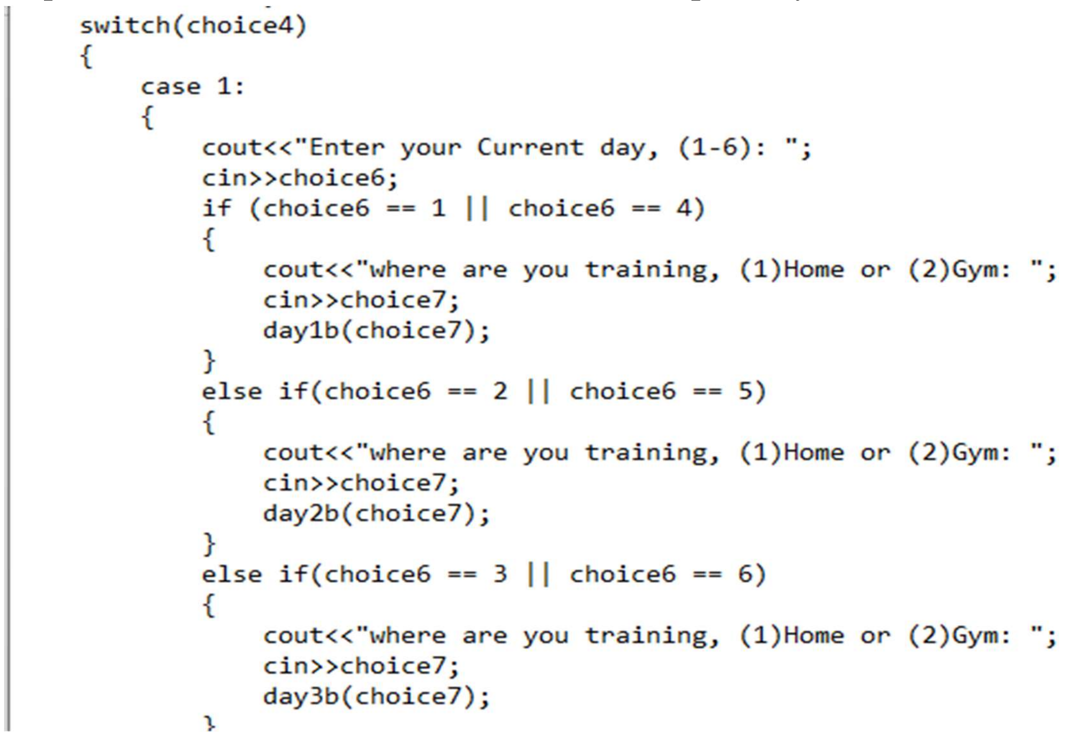


```
switch(choice2)
{
case 1:
{
    cout<<"\nOH! Looks like we got a newbie, Don't worry "<<call<<" They don't call me Tom Muscles for no reason. I will get you going in
    cin>>choice3;
    if(choice3 == 1)
    {
        cout<<"\n\nHere is a Detailed Workout plan, follow this and watch yourself Rise!";
    cout<<"\nDay 1: Chest, Arms\nDay 2: Abs, Shoulders\nDay 3: Legs, Cardio\nDay 4: Chest, Arms\nDay 5: Abs, Shoulders\nDay 6: Legs, Card
    day1b(choice3);

    }
    else
    {
        cout<<"\n\nHere is a Detailed Workout plan, follow this and watch yourself Rise!";
        cout<<"\nDay 1: Chest, Triceps\nDay 2: Abs, Shoulders\nDay 3: Legs, Soulders\nDay 4: Chest, Arms\nDay 5: Abs, Shoulders\nDay 6: L
        day1b(choice3);

    }
    break;
}
case 2:
{
    cout<<"\nOH! So you are a light athlete, Don't worry "<<call<<" They don't call me Tom Muscles for no reason. I will get you going fo
    cout<<"\nDay 1: Upper Body.\nDay 2: Abs, Shoulders\nDay 3: Legs, Cardio\nDay 4: Upper Body.\nDay 5: Abs, Shoulders\nDay 6: Legs, Card
    cout<<"\nTell me, where will you be working out? Home(1) or Gym(2).";
    cin>>choice3;
    if(choice3 == 1)
    {
        cout<<"\n\nHere is a Detailed Workout plan, follow this and watch yourself Rise!";
    cout<<"\nDay 1: Chest, Arms\nDay 2: Abs, Shoulders\nDay 3: Legs, Cardio\nDay 4: Chest, Arms\nDay 5: Abs, Shoulders\nDay 6: Legs, Card
    day1i(choice3);
```

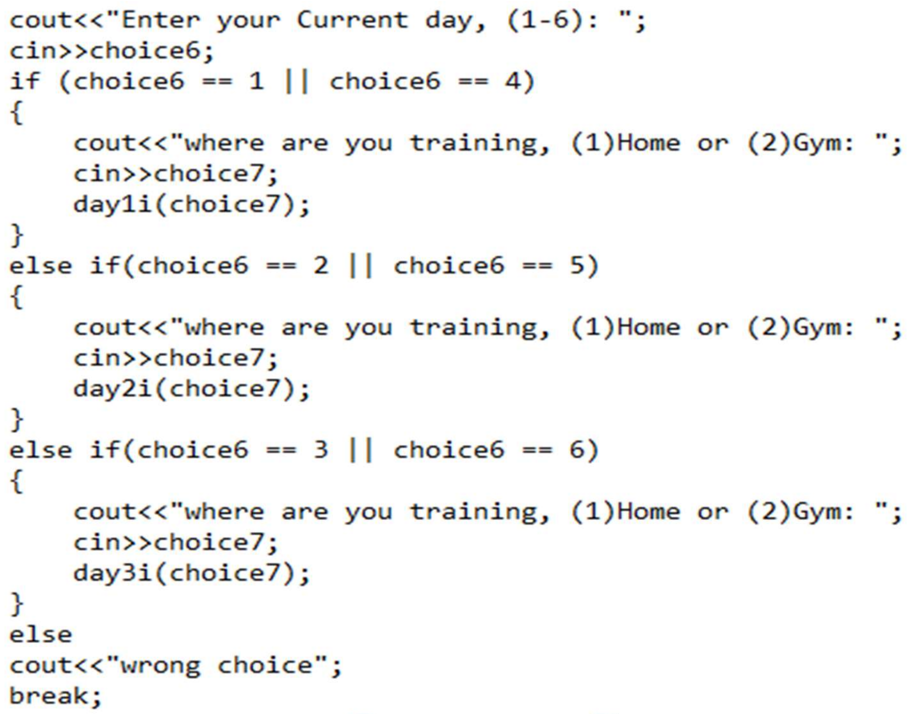
Finally, to bring attention back to the functions, they were primarily used in this aforementioned second case. This is because, in a situation where the user had used the program before, there was a necessity to keep track of which day they were on so that they could continue their exercise routine in a case of closing the program for later use.

This involved a total of 9 functions, rather than 18 functions, and this efficiency was achieved by making it so that the exercises for a week repeated every 3 days, which created a symmetrical structure, not requiring a unique exercise for each day.

The reason for 9 void functions, though, was because a repetition every three days still requires at least 3 exercises which when multiplied by the cases of exercise level

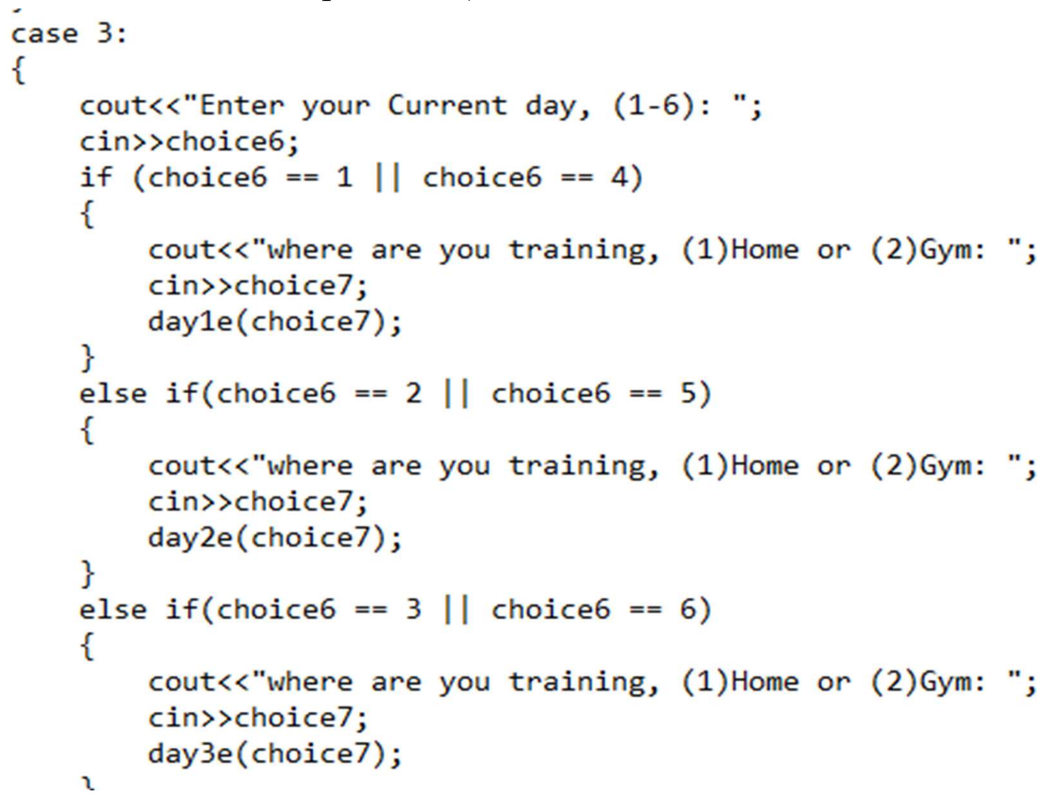


```
    switch(choice4)
    {
        case 1:
        {
            cout<<"Enter your Current day, (1-6): ";
            cin>>choice6;
            if (choice6 == 1 || choice6 == 4)
            {
                cout<<"where are you training, (1)Home or (2)Gym: ";
                cin>>choice7;
                day1b(choice7);
            }
            else if(choice6 == 2 || choice6 == 5)
            {
                cout<<"where are you training, (1)Home or (2)Gym: ";
                cin>>choice7;
                day2b(choice7);
            }
            else if(choice6 == 3 || choice6 == 6)
            {
                cout<<"where are you training, (1)Home or (2)Gym: ";
                cin>>choice7;
                day3b(choice7);
            }
```



```
            cout<<"Enter your Current day, (1-6): ";
            cin>>choice6;
            if (choice6 == 1 || choice6 == 4)
            {
                cout<<"where are you training, (1)Home or (2)Gym: ";
                cin>>choice7;
                day1i(choice7);
            }
            else if(choice6 == 2 || choice6 == 5)
            {
                cout<<"where are you training, (1)Home or (2)Gym: ";
                cin>>choice7;
                day2i(choice7);
            }
            else if(choice6 == 3 || choice6 == 6)
            {
                cout<<"where are you training, (1)Home or (2)Gym: ";
                cin>>choice7;
                day3i(choice7);
            }
            else
            cout<<"wrong choice";
            break;
```

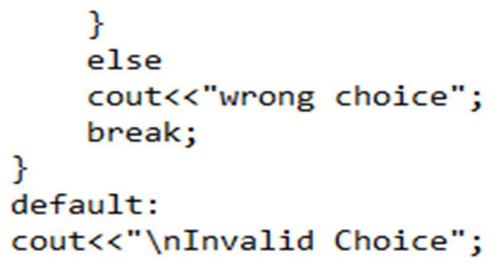
(beginner, intermediate, experienced), leads to nine. These were named 1b, 2b, 3b, 1i, 2i,



```
case 3:
{
    cout<<"Enter your Current day, (1-6): ";
    cin>>choice6;
    if (choice6 == 1 || choice6 == 4)
    {
        cout<<"where are you training, (1)Home or (2)Gym: ";
        cin>>choice7;
        day1e(choice7);
    }
    else if(choice6 == 2 || choice6 == 5)
    {
        cout<<"where are you training, (1)Home or (2)Gym: ";
        cin>>choice7;
        day2e(choice7);
    }
    else if(choice6 == 3 || choice6 == 6)
    {
        cout<<"where are you training, (1)Home or (2)Gym: ";
        cin>>choice7;
        day3e(choice7);
```

3i, 1e, 2e, 3e, specifically.

We should also mention, in the end, that the void function was specifically used because by this point of our program, because no variable was required as the only thing necessary was output for the specific exercises, and we made sure that any invalid inputs would be addressed



```
        }
        else
        cout<<"wrong choice";
        break;
    }
    default:
    cout<<"\nInvalid Choice";
```

Conclusions:

This project proposes the development of a **Personal Gym Trainer** using **C++**, designed to provide users with customized workout plans based on their **Body Mass Index (BMI)**, fitness level, and personal preferences. The program will guide users through a structured weekly routine, incorporating inputs for weight, height, gender, and experience level to generate tailored exercises and diet suggestions. Key programming concepts such as **conditional statements, loops, functions, and user input handling** are employed to ensure accuracy and flexibility. The system aims to address the lack of personalized fitness guidance, offering a practical solution for users to achieve their health goals efficiently. The project also demonstrates the application of fundamental C++ principles in real-world scenarios, emphasizing reusability and scalability for future enhancements.