## TABLE OF CONTENTS

## Abstract:

As the global emphasis on health and fitness grows, the need for effective digital tools to assist individuals in their personal health journeys has become paramount. The "HealthLogix - Health Assistance" is a digital solution, developed in C++, designed to meet this need. This program offers users a comprehensive platform to register, track workouts, log food intake, and monitor weight changes. Built with a user-friendly interface, it ensures that users can seamlessly navigate through various functionalities, making health tracking a hassle-free experience. The program employs structured data types and modular functions to ensure efficient data management and processing. Furthermore, the capability to export user details ensures data portability, a crucial feature in the modern digital age. In summary, the "HealthLogix - Health Assistance" serves as an essential tool for anyone aiming to keep a close watch on their health metrics, enabling informed decision-making and goal setting.

## Objective:

In today's digital age, where data-driven insights play a pivotal role in decision-making, maintaining and tracking health metrics digitally has become more than just a convenience—it's a necessity. The primary objective of the "HealthLogix - Health Assistance" is to bridge this need by providing a holistic platform that caters to the diverse health tracking needs of individuals.

User-Centric Approach: The program is designed with the user in mind. By offering a seamless registration and login process, it ensures that users can easily access and utilize the platform without any hindrance.

Comprehensive Health Tracking: The system aims to provide a one-stop solution for all health tracking needs—from logging daily workouts and food intake to updating weight metrics regularly. This all-encompassing approach ensures that users have a bird's-eye view of their health journey.

Data Portability: In a world where data sharing is crucial, especially for consultations with health professionals or personal trainers, the ability to export user details becomes paramount. The tracker aims to provide this feature, ensuring users can share their progress whenever necessary.

Enhanced User Experience: By integrating features like a user dashboard, the program seeks to provide an intuitive experience. Users can easily navigate, view recent activities, and get insights, making the health tracking process not just efficient but also enjoyable.

Security and Privacy: Given the personal nature of health data, the system is designed with a strong emphasis on data security. Whether it's user credentials or their health metrics, the program ensures that all data is stored and managed securely.

In essence, the "HealthLogix - Health Assistance" is not just a program—it's a digital companion for anyone embarking on or already on their fitness journey. By providing the tools and insights necessary, it aims to make the journey more informed, structured, and rewarding.

## 1. PROBLEM STATEMENT:

In today's health-conscious era, tracking one's fitness progress and setting health goals has become paramount. Traditionally, people relied on handwritten logs, charts, or memory to keep track of their workouts and dietary habits. As the world shifts to a more digital approach, there arises a need for a comprehensive system to manage these fitness details seamlessly.

With the increasing number of fitness enthusiasts, there's a demand for a system that can not only log daily workouts and food intake but also allow users to monitor their weight changes over time. An efficient, user-friendly system can ensure that users stay motivated, as they can easily access their progress and set new milestones.

In light of this, the client envisioned a digital '**HealthLogix – Health Assistance**' tailored for individuals who wish to take control of their health journey. This platform serves multiple purposes:

1. It allows new users to register and existing users to authenticate their credentials.
2. t provides a comprehensive dashboard for users to log workouts, record food intake, update weight, and view recent activity.
3. With the potential to export user details, it ensures data portability and easy sharing with health professionals or personal trainers.

The proposed system is designed to deliver top-notch health tracking services to its users. It aims to be fast, secure, and intuitive. Catering to both registered and unregistered users, the core modules of this project include:

- User Registration and Login
- Workout Logging
- Food Intake Recording
- Weight Update
- User Dashboard and Data Export

By implementing this system, users can now effortlessly monitor their health metrics, ensuring a more informed and focused approach to their fitness journey.

## 2. MODULES OF PROJECT:

The **"HealthLogix – Health Assistance"** is a comprehensive system, crafted meticulously in C++, aiming to serve fitness enthusiasts and individuals who are keen on maintaining a digital record of their health metrics. The modules are designed to provide users with an intuitive and seamless experience, allowing them to navigate through the functionalities effortlessly.

1. **User Management System:**
   - User Registration: New users can seamlessly register by providing essential details such as username, password, weight, and height.
   - User Login: Ensures that only verified users access the system by verifying their credentials. If a user isn't registered, they are prompted to create an account.

2. **Fitness Tracking:**
   - Workout Logging: Post-login, users can record details of their daily workouts, from the type of exercise to its duration.
   - Food Intake Recording: A dedicated module where users can log their daily food consumption, helping them monitor their dietary habits.
   - Weight Update: As fitness journeys involve progress, this module allows users to update their weight, giving them insights into their progress over time.

3. **User Dashboard:**
   - A centralized space where users can view their recent activities, be it workouts or food logs.
   - Data Export: Ensuring data portability, users can export their health metrics, facilitating sharing with fitness trainers or health professionals.

The primary objective of this system is to offer a digital platform where users can monitor and manage their fitness journey. With the ever-increasing emphasis on health and the shift to digital platforms, this HealthLogix - Health Assistance stands out as an essential tool for modern-day individuals. The easy navigation, coupled with detailed tracking features, ensures that users are always updated about their health metrics. Whether it's about setting a new fitness goal or monitoring daily calorie intake, the system caters to all these needs efficiently.

Our HealthLogix - Health Assistance is a boon for those who believe in a systematic and digital approach to health. Users can access their data anytime, ensuring they are always in tune with their health status. And for those who wish to share their progress with others or require professional insights, the data export feature comes in handy.

## 3. UML DIAGRAMS:

### 3.1) Use Case Diagram:

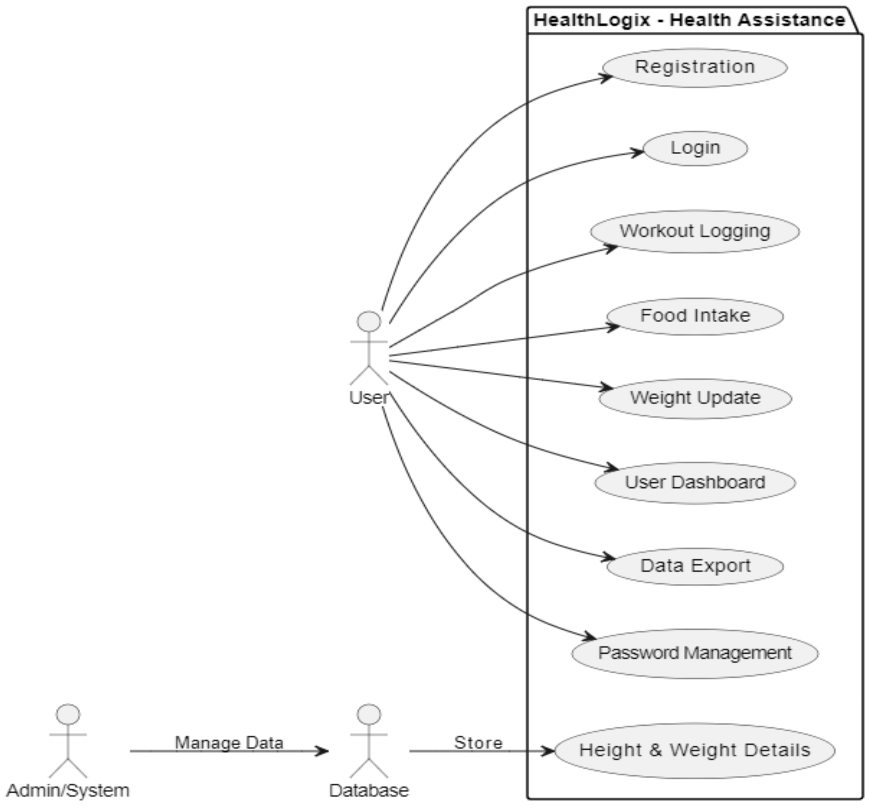

**Fig. 3.1**

In the depicted diagram, there are three primary actors:

1. User: Represents the individual who uses the "HealthLogix - Health Assistance" system.
2. Admin/System: Manages user data and provides the necessary functionalities to the user.
3. Database: Responsible for storing all the information related to users, workouts, food intake, and other associated data.

Diving into the use cases:

1. Registration: The initial point of interaction where new users can sign up to access the system. The user provides essential details such as username, password, height, and weight.
2. Login: A critical use case ensuring that only verified users gain access to the system. The user provides their credentials, and the system validates them.
3. Workout Logging: Once authenticated, users can record details of their daily workouts, from the type of exercise to its duration, ensuring a comprehensive log of their physical activities.
4. Food Intake: A dedicated module that allows users to log their daily food consumption. It aids them in monitoring their dietary habits and ensuring balanced nutrition.
5. Weight Update: Recognizing that fitness journeys involve progress and change, this use case allows users to update their current weight, offering insights into their fitness progress over time.
6. User Dashboard: A central hub where users can view a summary of their recent activities and health metrics. It provides an overview, making it easier for users to assess their health journey at a glance.
7. Data Export: A significant feature ensuring data portability. Users can export their health metrics,

which is especially useful when sharing progress with health professionals or personal trainers.

8. Height & Weight Details: Stores and manages the height and weight metrics of users. While primarily managed by the system and stored in the database, users interact with this use case whenever they register or update their weight.
9. Password Management: With security being paramount, this use case manages user passwords and provides functionalities like password reset to ensure users can securely access the system.

The Admin/System primarily interacts with the Database to manage data, ensuring that all user interactions, from logging workouts to updating weight, are accurately reflected and stored.

In essence, the use-case diagram provides a comprehensive view of the "HealthLogix - Health Assistance", detailing each interaction a user might have with the system and how the system manages and stores this data.

## 3.2) Class Diagram:

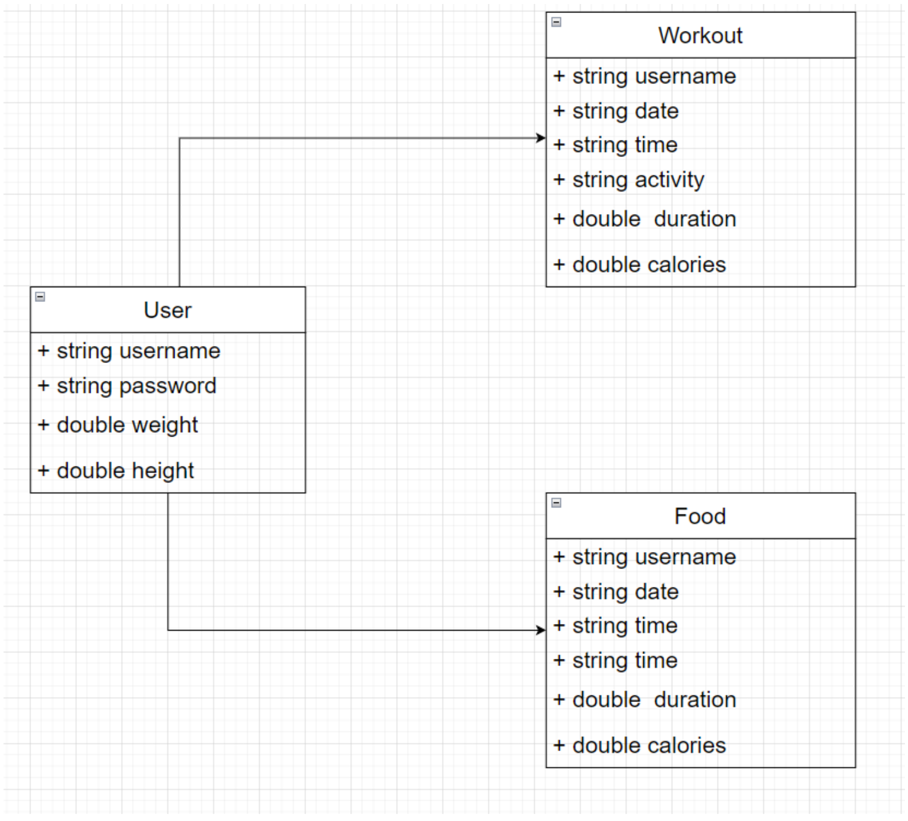

**Fig. 3.2**

In the depicted class diagram:

1. User Class:
   - Represents individual users of the "HealthLogix - Health Assistance" system.
   - Contains attributes like username, password, weight, and height to store essential user details.

2. Workout Class:
   - Represents the logged workouts of users.
   - Contains attributes such as username, date, time, activity, duration, and calories1 to capture comprehensive workout details.

3. Food Class:
   - Represents the food intake logs of users.
   - Contains attributes like username, date, time, item, and calories to record users' dietary habits in detail.

These classes collectively provide a structured way to store and manage the data necessary for the functionalities of the "HealthLogix - Health Assistance".

## 3.4) Collaboration Diagram:

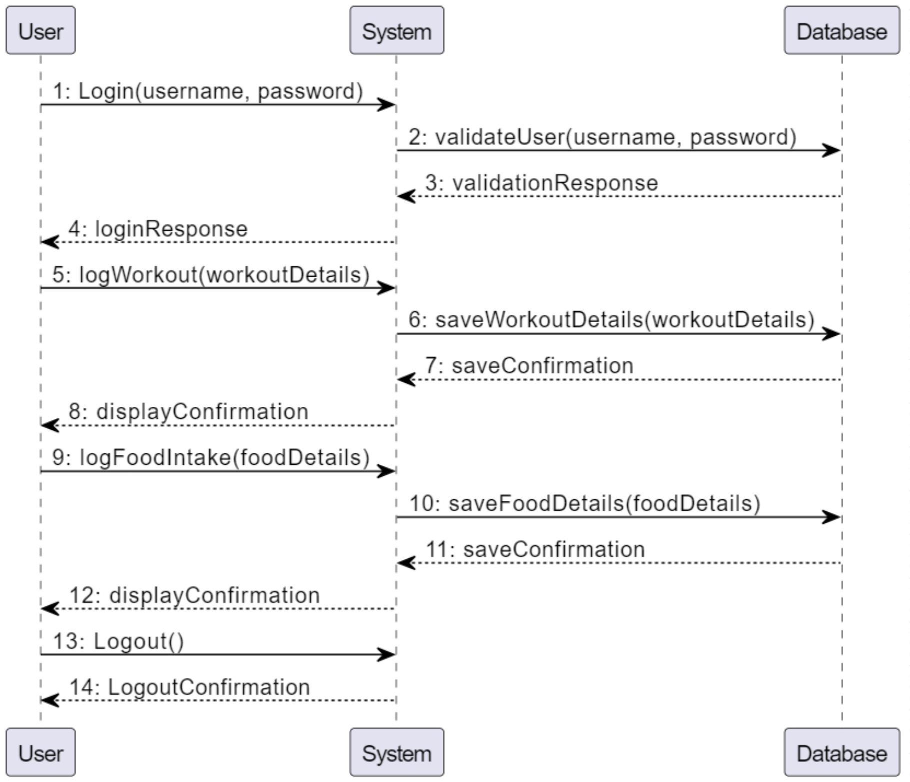

**Fig. 3.4**

In the collaboration diagram:

1. User: Represents the individual using the system.
2. System: Represents the "HealthLogix - Health Assistance" system that provides the necessary functionalities.
3. Database: Represents the backend storage system.

The interactions follow a similar pattern as the sequence diagram:

- The user initiates a login request (interaction 1). The system checks with the database to validate the user's credentials (interactions 2 and 3). The system then provides a response back to the user (interaction 4).
- The user logs a workout (interaction 5). The system processes the details and saves them in the database (interactions 6 and 7). A confirmation is sent back to the user (interaction 8).
- The user logs food intake (interaction 9). The system processes the details and saves them in the database (interactions 10 and 11). Again, a confirmation is sent to the user (interaction 12).
- The user then logs out of the system (interactions 13 and 14).

## 3.6) Activity Diagram:

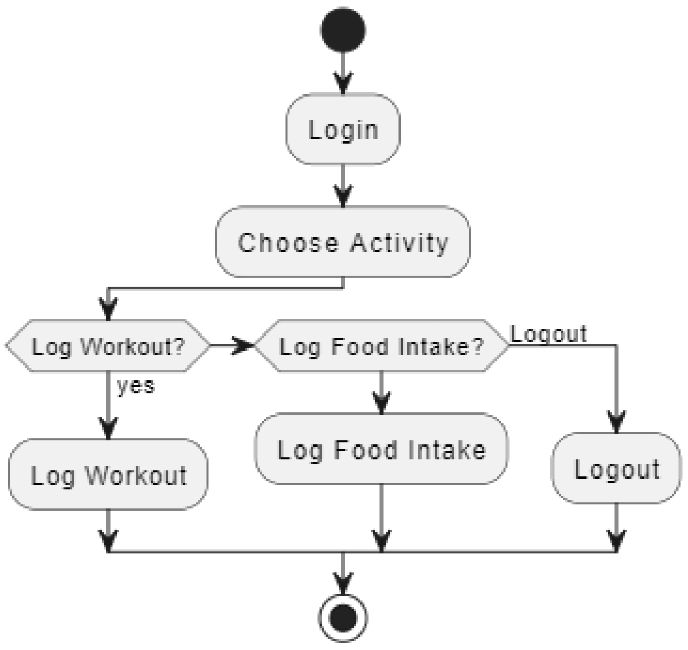

**Fig. 3.6**

The activity diagram describes the flow of actions a user can perform within the "HealthLogix - Health Assistance":

1. Start: Represents the beginning of the user's interaction.
2. Login: The user logs into the system.
3. Choose Activity: After logging in, the user decides their next action.
4. Log Workout: If the user decides to log a workout, this action is performed.
5. Log Food Intake: If the user decides to log food intake, this action is performed.
6. Log Out: The user logs out of the system.
7. End: Represents the conclusion of the user's interaction.

Transitions between actions are guided by decisions, such as choosing to log a workout, log food intake, or log out.

## 4. CODE / OUTPUT SCREENSHOTS

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime>
#include <vector>
#include <sstream> // Add this line to include the stringstream header

using namespace std;

struct User {
    string username;
    string password;
    double weight;
    double height;
};

// Struct to represent a workout entry
struct Workout {
    string username;
    string date;
    string time;
    string activity;
    double duration;
    double calories1;
};

// Struct to represent a food entry
struct Food {
    string username;
    string date;
    string time;
    string item;
    double calories;
};

string getCurrentDateTime() {
    time_t now = time(0);
    tm ltm;
    localtime_s(&ltm, &now);
    char buffer[80];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", &ltm);
    return string(buffer);
}
```

```
bool authenticateUser(const string &username, const string &password) {
    ifstream file("users.txt");
    string u, p;
    double w;

    while (file >> u >> p >> w) {
        if (u == username && p == password) {
            file.close();
            return true;
        }
    }

    file.close();
    return false;
}

void logWorkout(const string &username) {
    ofstream file("workouts.txt", ios::app);
    Workout workout;
    workout.username = username;
    workout.date = getCurrentDateTime();

    cin.ignore();
    cout << "Enter Activity: ";
    getline(cin, workout.activity);
    cout << "Enter Duration (in minutes): ";
    cin >> workout.duration;
    cout << "Enter the Calories burned: ";
    cin >> workout.calories1;

    file << workout.username << " " << workout.date << " " << workout.activity << " " << workout.duration << " "
      << workout.calories1 << endl;
    file.close();

    cout << "Workout logged successfully!" << endl;
}

void logFood(const string &username) {
    ofstream file("foods.txt", ios::app);
    Food food;
    food.username = username;
    food.date = getCurrentDateTime();

    cin.ignore();
    cout << "Enter Food Item: ";
    getline(cin, food.item);
    cout << "Enter Calories: ";
    cin >> food.calories;

    file << food.username << " " << food.date << " " << food.item << " " << food.calories << endl;
    file.close();

    cout << "Food logged successfully!" << endl;
}

void updateWeight(const string &username) {
    ifstream inFile("users.txt");
    ofstream outFile("temp.txt");

    string u, p;
    double w;
    double newWeight;
    while (inFile >> u >> p >> w) {
        if (u == username) {
            cout << "Enter New Weight (in kg): ";
            cin >> newWeight;
            outFile << u << " " << p << " " << newWeight << endl;
        } else {
            outFile << u << " " << p << " " << w << endl;
        }
    }

    inFile.close();
    outFile.close();

    remove("users.txt");
    rename("temp.txt", "users.txt");

    cout << "Weight updated successfully!" << endl;
}

void showRecentDetails(const string &username) {
    ifstream workoutFile("workouts.txt");
    ifstream foodFile("foods.txt");
    string line;
    bool workoutFound = false;
    bool foodFound = false;

    cout << "\n--- Recent Details ---\n";

    // Display recent workouts for the user
    while (getline(workoutFile, line)) {
        istringstream iss(line);
        Workout workout;
        iss >> workout.username >> workout.date >> workout.time >> workout.activity >> workout.duration >>
      workout.calories1;
        if (workout.username == username) {
            cout << "Last Workout:\n";
            cout << "Date: " << workout.date << "\n";
            cout << "Activity: " << workout.activity << "\n";
            cout << "Duration: " << workout.duration << " minutes\n";
            cout << "Calories Burned: " << workout.calories1 << "\n";
            workoutFound = true;
            break;  // Display only the most recent workout
        }
    }
    workoutFile.close();

    // Display recent food intake for the user
    while (getline(foodFile, line)) {
        istringstream iss(line);
        Food food;
        iss >> food.username >> food.date >> food.time >> food.item >> food.calories;
        if (food.username == username) {
            cout << "\nLast Food Intake:\n";
            cout << "Date: " << food.date << "\n";
            cout << "Food Item: " << food.item << "\n";
            cout << "Calories: " << food.calories << "\n";
            foodFound = true;
            break;  // Display only the most recent food entry
        }
    }
    foodFile.close();

    if (!workoutFound) {
        cout << "No workouts logged yet.\n";
    }

    if (!foodFound) {
        cout << "No foods logged yet.\n";
    }

    cout << "-----------------------\n\n";
}

void exportUserDetails(const string &username) {
    ofstream exportFile(username + "_details.txt");
    ifstream workoutFile("workouts.txt");
    ifstream foodFile("foods.txt");
    string line;

    exportFile << "=== User Details for " << username << " ===\n\n";

    // Export workout details
    exportFile << "=== Workout Details ===\n";
    while (getline(workoutFile, line)) {
        istringstream iss(line);
        Workout workout;
        iss >> workout.username >> workout.date >> workout.time >> workout.activity >> workout.duration >>
      workout.calories1;
        if (workout.username == username) {
            exportFile << "Date: " << workout.date << "\n";
            exportFile << "Time: " << workout.time << "\n";
            exportFile << "Activity: " << workout.activity << "\n";
            exportFile << "Duration: " << workout.duration << " minutes\n";
            exportFile << "Calories Burned: " << workout.calories1 << "\n\n";
        }
    }
    exportFile << "\n";

    // Export food details
    exportFile << "=== Food Details ===\n";
    while (getline(foodFile, line)) {
        istringstream iss(line);
        Food food;
        iss >> food.username >> food.date >> food.time >> food.item >> food.calories;
        if (food.username == username) {
            exportFile << "Date: " << food.date << "\n";
            exportFile << "Time: " << food.time << "\n";
            exportFile << "Food Item: " << food.item << "\n";
            exportFile << "Calories: " << food.calories << "\n\n";
        }
    }

    exportFile.close();
    cout << "User details exported successfully to " << username << "_details.txt\n";
}

void userDashboard(const string &username) {
    int choice;

    while (true) {
        cout << "=== " << username << "'s Dashboard ===" << endl;
        cout << "1. Log Workout" << endl;
        cout << "2. Log Food" << endl;
        cout << "3. Update Weight" << endl;
        cout << "4. Show Recent Workout and Food Details" << endl;
        cout << "5. Export Details" << endl;
        cout << "6. Logout" << endl;
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                logWorkout(username);
                break;
            case 2:
                logFood(username);
                break;
            case 3:
                updateWeight(username);
                break;
            case 4:
                showRecentDetails(username);
                break;
            case 5:
                exportUserDetails(username); // Call the exportUserDetails function
                break;
            case 6:
                return;  // Logout and return to the main menu
            default:
                cout << "Invalid choice. Try again." << endl;
        }
    }
}

void registerUser() {
    string username, password;
    double weight;

    cout << "Enter Username: ";
    cin >> username;

    cout << "Enter Password: ";
    cin >> password;

    cout << "Enter Weight (in kg): ";
    cin >> weight;
    ofstream file("users.txt", ios::app);
    file << username << " " << password << " " << weight << endl;
    file.close();

    cout << "Registration successful!" << endl;
}

void loginUser() {
    string username, password;

    cout << "Enter Username: ";
    cin >> username;

    cout << "Enter Password: ";
    cin >> password;

    if (authenticateUser(username, password)) {
        cout << "Login successful!" << endl;
        userDashboard(username);
    } else {
        cout << "Invalid username or password. Try again." << endl;
    }
}

int main() {
    int choice;

    while (true) {
        cout << "=== Main Menu ===" << endl;
        cout << "1. Register" << endl;
        cout << "2. Login" << endl;
        cout << "3. Exit" << endl;
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                loginUser();
                break;
            case 3:
                return 0;  // Exit the program
                default:
                    cout << "Invalid choice. Try again." << endl;
            }
        }

        return 0;
}
```

## 1. CONCLUSION:

The "HealthLogix - Health Assistance" project has been successfully developed and its performance has met our expectations. This application has been intricately designed in C++ to cater to the requirements of users aiming to track their fitness routines and dietary habits. The user interface ensures a seamless experience for those looking to log their workouts and meals, ensuring that their fitness journey is well-documented.

The relationship between various components, such as the user authentication system, workout logger, and food logger, has been built to ensure efficient communication and data retrieval. Through this project, users are presented with a comprehensive tool that aids them in maintaining a healthy lifestyle by keeping tabs on their physical activities and nutrition.

Working on this "HealthLogix - Health Assistance" has been an enlightening journey. It served as an excellent platform for us to delve deep into the world of C++ and object-oriented programming. The system we developed is firmly grounded in C++ and OOP principles, with UML diagrams guiding its architecture. Throughout the development phase, we encountered challenges that pushed us to innovate and research, expanding our knowledge base significantly. These experiences have not only honed our technical skills but also provided clarity on various concepts that will undoubtedly be valuable in our future endeavors.

## 2. REFERENCES

[1] kaur, l., kaur, n., ummat, a., kaur, j., & kaur, n. (2016). research paper on object oriented software engineering. international journal of computer science and technology, 36-38.

[2] Kak, Avinash C. Programming with Objects, A Comparative Presentation of Object-Oriented Programming with C++ and Java, John Wiley, 2003. ISBN 0-471-26852-6.

[3] Lafore, Robert, Object-Oriented Programming in C++, Fourth Edition, Sams Publishing, 2002. ISBN 0-672- 32308-7.

[4] Seed, Graham M., An Introduction to Object-Oriented Programming in C++ with Applications in Computer Graphics, Second Ed., Springer-Verlag, 2001.ISBN 1- 85233-450-9.

[5] Svenk, Goran, Object-Oriented Programming: Using C++ for Engineering and Technology Delmar, 2003. ISBN 0-7668-3894-3.

[6] Yevick, David, A First Course in Computational Physics and Object-Oriented Programming, Cambridge University Press, 2005. ISBN 0-521-82778-7.

- **https://www.stackoverflow.com/**
- **https://www.google.com/**
- **https://www.staruml.io/**
- **https://online.visual-paradigm.com/**
- **https://app.diagrams.net/**
- **https://www.naukri.com/**