VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JNANA SANGAMA,BELAGAVI – 590018**

**KARNATAKA**



**Assignment Report**

**On**

**“SMART GYM MANAGEMENT SYSTEM”**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE DATA STRUCTURES AND APPLICATIONS (BCS304) COURSE OF III SEMESTER**

Submitted by

|  |  |  |
| --- | --- | --- |
| **Darshan Naik** |  |  |
| **[1CG222AD011]** |  |  |

**Guide: HOD:**

**Mrs. Mala K .M. Tech. Dr. GavisidappaPhD.,**

Asst. Prof., Dept. of ISE Head, Dept. of AD

CIT, Gubbi. CIT, Gubbi.

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCINECE**

**Channabasaveshwara Institute of Technology**

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(**ISO 9001:2015 Certified Institution)**

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

**2023-24**

**Rubric – B.E. Mini-Project [BCS304]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course**  **outcome** | **Rubric/Level** | **Excellent**  **(91-100%)** | **Good**  **(81-90%)** | **Average**  **(61-80%)** | **Moderate**  **(40-60%)** | **Score** |
| **CO1** | **Identification of project**  **proposal**  **(05 Marks)** |  |  |  |  |  |
| **CO2** | **Design and**  **Implementation**  **(10 Marks)** |  |  |  |  |  |
| **CO3** | **Presentation skill**  **(05 Marks)** |  |  |  |  |  |
| **CO4** | **Report**  **(05 Marks)** |  |  |  |  |  |
| **Total** | | | | | |  |

**Course outcome:**

**CO 1: Identification of project proposal which is relevant to subject of engineering.**

**CO 2: Design and implement proposed project methodology.**

**CO 3: Effective communication skill to assimilate their project work.**

**CO 4: Understanding overall project progress and performance.**

**Student Signature Faculty signature**

**Abstract**

The C code presents a console-based Smart Gym Management System designed to handle member registration, display member details, mark attendance, and update calories burnt. The system employs a structure named `Member` to store individual member details, including ID, name, gender, date of birth, membership type, attendance, and calories burnt. Staff authentication is implemented for sensitive operations, and the program offers a user-friendly menu-driven interface for interaction. However, the code assumes a maximum of 100 members and employs a simplistic password-based authentication mechanism for staff access.

**1. Structure Definition (`struct Member`):**

- Defines a structure to encapsulate member details, including ID, name, gender, date of birth, membership type, attendance, and calories burnt.

**2. Functionality Options (`main` and Menu):**

- Presents a menu-driven interface in the `main` function, allowing users to choose from the following options:

- Register a new member.

- Display member details.

- Mark member attendance.

- Update member calories burnt.

- Exit the program.

**3. Member Registration (`registerMember`):**

- Requires staff authentication before allowing member registration.

- Captures member details such as name, gender, and date of birth.

- Allows staff to choose a membership type (Gold or Silver).

- Assigns an incremental ID to each new member.

- Initializes attendance and calories burnt to zero.

**4. Display Member Details (`displayMemberDetails`):**

- Requires staff authentication before displaying member details.

- Iterates through the member list, presenting details such as ID, name, gender, date of birth, membership type, attendance, and calories burnt.

**5. Mark Attendance (`markAttendance`):**

- Requires staff authentication before marking attendance.

- Prompts for a member ID and increments the attendance count if the member is found.

**6. Update Calories Burnt (`updateCaloriesBurnt`):**

- Requires staff authentication before updating calories burnt.

- Prompts for a member ID and allows staff to input the calories burnt, updating the member's total.

**7. Staff Authentication (`authenticateStaff`):**

- Prompts for a staff password (hardcoded as 1234 for simplicity).

- Returns 1 for successful authentication, allowing access to sensitive operations, and 0 for failed authentication.

**8. Exit (`case 5` in Menu):**

- Allows users to exit the program gracefully.

**Introduction**

In an era where fitness and well-being have gained paramount importance, the integration of efficient management systems becomes crucial for the smooth operation of fitness centers. The presented C code introduces a Smart Gym Management System, a console-based application designed to streamline tasks related to member registration, tracking attendance, and monitoring calories burnt. This system offers a structured approach to managing gym members, utilizing a user-friendly interface that enables staff to perform essential operations securely. With a focus on simplicity and clarity, the code illustrates the foundational elements necessary for an effective gym management system, including member registration, data display, attendance tracking, and calorie updates. The subsequent sections delve into the key functionalities and structures embedded within the program, providing a comprehensive understanding of its design and capabilities.

**Problem Statement**

In the dynamic landscape of fitness facilities, the efficient management of gym member data and activities poses a significant challenge. The absence of a systematic approach can lead to operational inefficiencies, making it difficult for gym staff to track member details, monitor attendance, and manage calorie-related information effectively. To address this, there is a need for a robust Smart Gym Management System that automates and organizes member-related tasks. The current absence of such a system results in a lack of streamlined processes, hindering the overall productivity and management capabilities of fitness centers. This project aims to develop a solution that provides a user-friendly interface for staff, ensuring seamless member registration, easy access to member details, simplified attendance tracking, and efficient updates on calories burnt. By addressing these challenges, the Smart Gym Management System aims to enhance the overall functioning of fitness centers and contribute to a more organized and customer-centric approach.

**Implementation**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Structure to store member details

struct Member {

int id;

char name[50];

char gender;

char dob[15];

char membershipType[20];

int attendance;

double caloriesBurnt;

};

// Function prototypes

void registerMember(struct Member \*members, int \*totalMembers);

void displayMemberDetails(struct Member \*members, int totalMembers);

void markAttendance(struct Member \*members, int totalMembers);

void updateCaloriesBurnt(struct Member \*members, int totalMembers);

int authenticateStaff();

int main() {

struct Member members[100]; // Assuming a maximum of 100 members

int totalMembers = 0;

int choice;

while (1) {

printf("\nSmart Gym Management System\n");

printf("1. Register Member\n");

printf("2. Display Member Details\n");

printf("3. Mark Attendance\n");

printf("4. Update Calories Burnt\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

registerMember(members, &totalMembers);

break;

case 2:

displayMemberDetails(members, totalMembers);

break;

case 3:

markAttendance(members, totalMembers);

break;

case 4:

updateCaloriesBurnt(members, totalMembers);

break;

case 5:

printf("Exiting program. Goodbye!\n");

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

// Function to register a new member

void registerMember(struct Member \*members, int \*totalMembers) {

if (authenticateStaff()) {

printf("\nEnter Member Details:\n");

members[\*totalMembers].id = \*totalMembers + 1; // Incremental member ID

printf("Name: ");

scanf("%s", members[\*totalMembers].name);

printf("Gender (M/F): ");

scanf(" %c", &members[\*totalMembers].gender);

printf("Date of Birth (DD/MM/YYYY): ");

scanf("%s", members[\*totalMembers].dob);

// Allow customers to choose membership type

printf("Choose Membership Type:\n");

printf("1. Gold\n");

printf("2. Silver\n");

int membershipChoice;

scanf("%d", &membershipChoice);

switch (membershipChoice) {

case 1:

strcpy(members[\*totalMembers].membershipType, "Gold");

break;

case 2:

strcpy(members[\*totalMembers].membershipType, "Silver");

break;

default:

printf("Invalid choice. Setting default to Silver membership.\n");

strcpy(members[\*totalMembers].membershipType, "Silver");

}

// Initialize attendance and calories burnt to zero

members[\*totalMembers].attendance = 0;

members[\*totalMembers].caloriesBurnt = 0.0;

(\*totalMembers)++;

printf("Member successfully registered!\n");

} else {

printf("Authentication failed. Only staff members can register new members.\n");

}

}

// Function to display member details

void displayMemberDetails(struct Member \*members, int totalMembers) {

if (authenticateStaff()) {

printf("\nMember Details:\n");

for (int i = 0; i < totalMembers; i++) {

printf("ID: %d\n", members[i].id);

printf("Name: %s\n", members[i].name);

printf("Gender: %c\n", members[i].gender);

printf("Date of Birth: %s\n", members[i].dob);

printf("Membership Type: %s\n", members[i].membershipType);

printf("Attendance: %d\n", members[i].attendance);

printf("Calories Burnt: %.2lf\n", members[i].caloriesBurnt);

printf("----------------------\n");

}

} else {

printf("Authentication failed. Only staff members can access member details.\n");

}

}

// Function to mark member attendance

void markAttendance(struct Member \*members, int totalMembers) {

if (authenticateStaff()) {

int memberId;

printf("Enter Member ID for Attendance: ");

scanf("%d", &memberId);

for (int i = 0; i < totalMembers; i++) {

if (members[i].id == memberId) {

members[i].attendance++;

printf("Attendance marked for %s.\n", members[i].name);

return;

}

}

printf("Member not found with ID %d.\n", memberId);

} else {

printf("Authentication failed. Only staff members can mark attendance.\n");

}

}

// Function to update member calories burnt

void updateCaloriesBurnt(struct Member \*members, int totalMembers) {

if (authenticateStaff()) {

int memberId;

double calories;

printf("Enter Member ID for Calories Update: ");

scanf("%d", &memberId);

for (int i = 0; i < totalMembers; i++) {

if (members[i].id == memberId) {

printf("Enter Calories Burnt: ");

scanf("%lf", &calories);

members[i].caloriesBurnt += calories;

printf("Calories updated for %s.\n", members[i].name);

return;

}

}

printf("Member not found with ID %d.\n", memberId);

} else {

printf("Authentication failed. Only staff members can update calories burnt.\n");

}

}

// Function to authenticate staff members

int authenticateStaff() {

int password;

printf("Enter Staff Password: ");

scanf("%d", &password);

// Assuming a simple password for illustration purposes

if (password == 1234) {

return 1; // Authentication successful

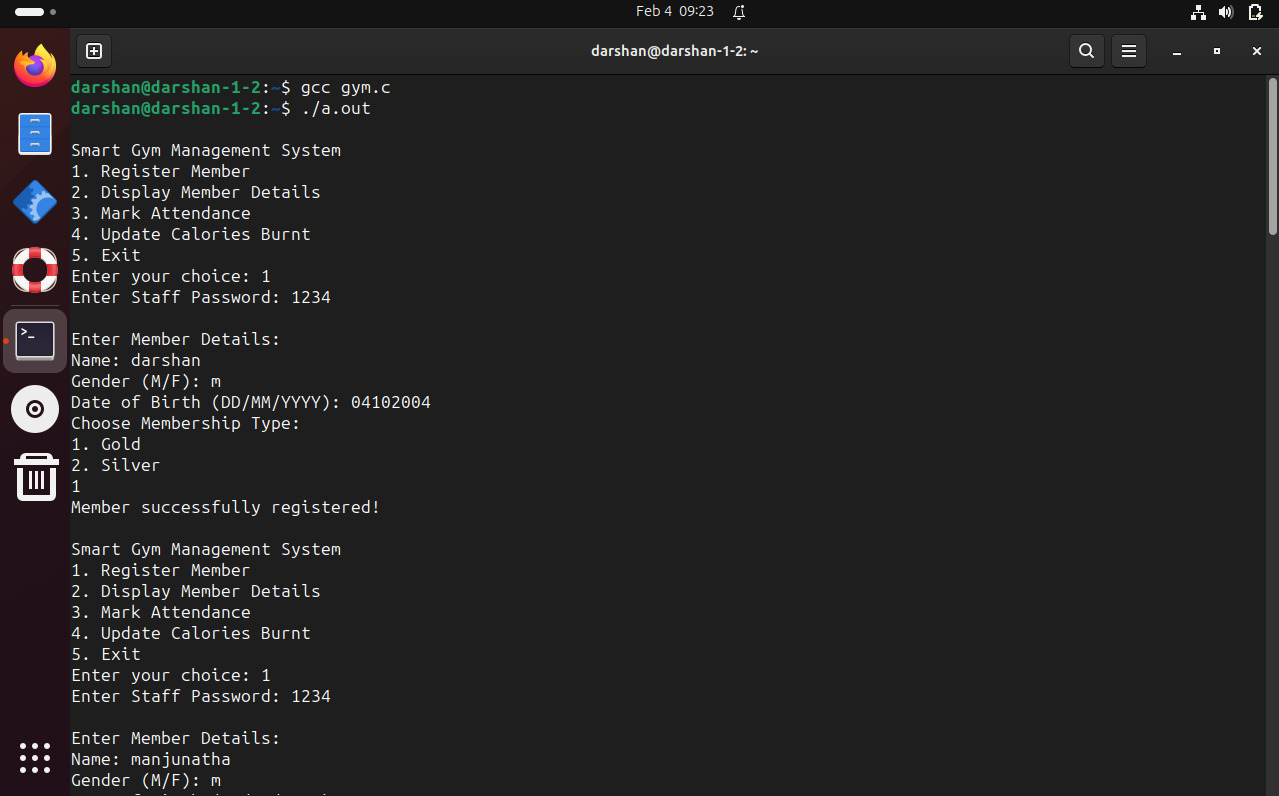
} else {

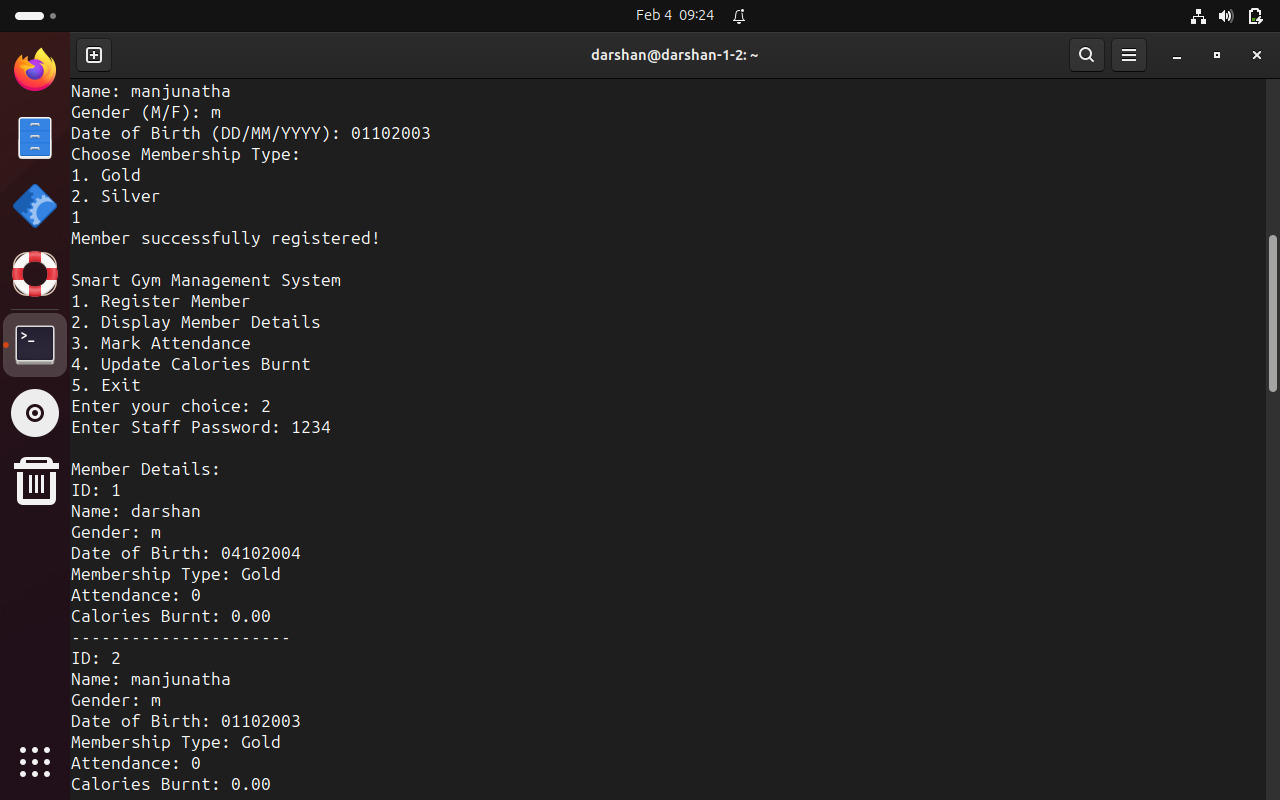
return 0; // Authentication failed

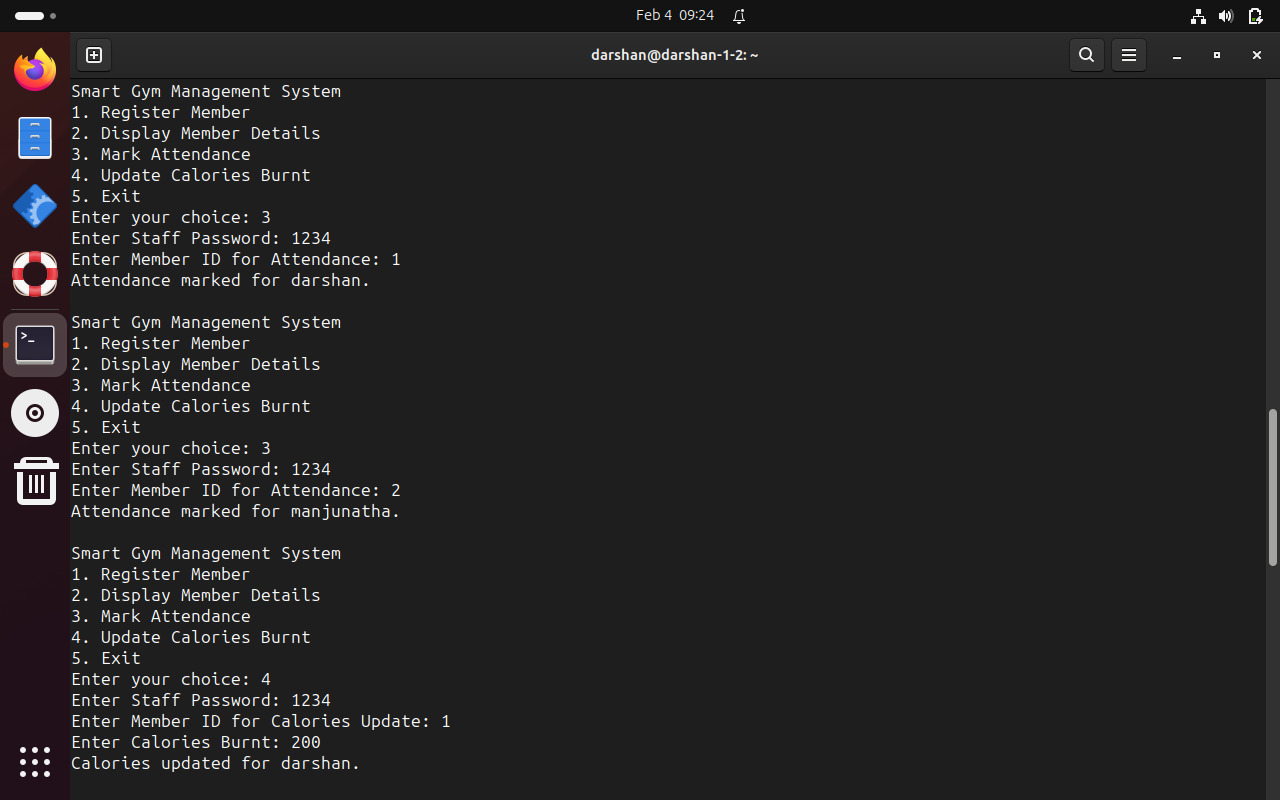
}

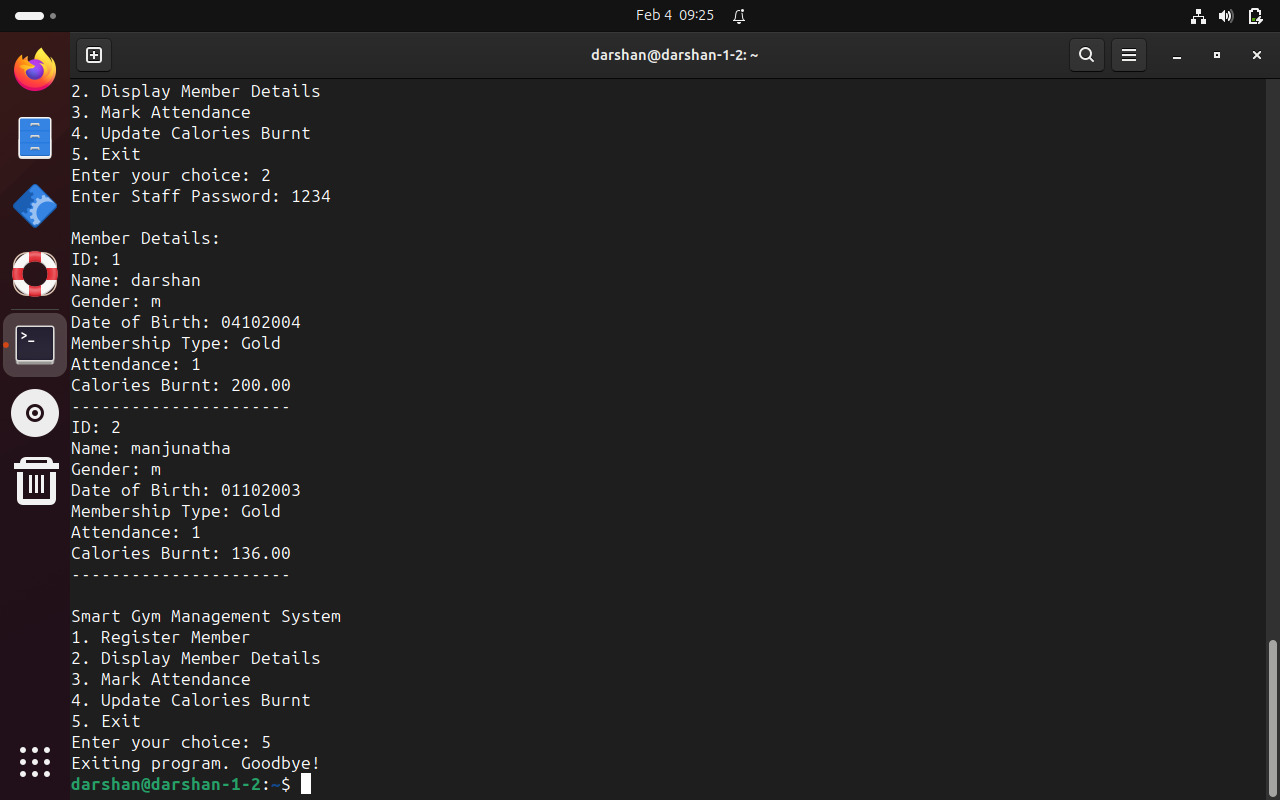
}

**Results/Screen Shot:**









**Conclusion:**

In the culmination of this endeavor to develop a Smart Gym Management System, the presented C code lays the foundation for a streamlined and organized approach to fitness center operations. By addressing the challenges associated with manual member management, attendance tracking, and calorie updates, the system offers a practical solution for gyms seeking operational efficiency. The modular design, encapsulated within functions and structures, ensures clarity and ease of maintenance. Although the current implementation is illustrative and may require further refinement, it demonstrates the potential to significantly improve the management of gym-related tasks. As the fitness industry continues to evolve, adopting comprehensive management systems becomes imperative, and the Smart Gym Management System serves as a fundamental step towards achieving this goal. The adaptability and extensibility of the code pave the way for future enhancements, ensuring that it remains a valuable tool for fitness centers aiming to provide an optimal experience for both staff and members.

**REFERENCES**

1. [Smart Gym Management System - IJSRET](https://ijsret.com/wp-content/uploads/2020/05/IJSRET_V6_issue3_493.pdf): This paper discusses the use of technology to improve fitness and nutrition. [It also talks about the development of an Android application to help users manage their health care system in fitness and nutrition1](https://ijsret.com/wp-content/uploads/2020/05/IJSRET_V6_issue3_493.pdf).
2. [A Project Paper on Smart Gym Management System - Academia.edu](https://www.academia.edu/37647521/A_Project_Paper_on_Smart_Gym_Management_System): This project paper discusses the importance of health and presents a smart gym management system. [It provides a comprehensive guide on how to manage a gym effectively](https://ijsret.com/wp-content/uploads/2020/05/IJSRET_V6_issue3_493.pdf)[2](https://www.academia.edu/37647521/A_Project_Paper_on_Smart_Gym_Management_System).
3. [Research on sports fitness management based on … - SpringerOpen: This research paper discusses a dynamic management technology for sports fitness based on the concept of Internet of Things and blockchain3](https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-020-01821-2).
4. [Survey Paper on Gym Management System - Academia.edu: This survey paper provides a comprehensive overview of various gym management systems4](https://www.academia.edu/51410222/Survey_Paper_on_Gym_Management_System).
5. [Development of Fitness Management System Application - JETIR](https://www.jetir.org/papers/JETIR2204456.pdf): This paper proposes a fitness management system for those who run a fitness or gym business. [It discusses how to automate operations, keep records, and ensure proper database security5](https://www.jetir.org/papers/JETIR2204456.pdf).