Ain Shams University

GYM& PADEL Management System

Saifullah Adam AbdulMoamen Ahmed	2023170285
Salma elshowehdy abdelfatah mohamed	2023170260
Habiba hatem ahmed desoky	2023170185
Razan Saeed Mohammed Ibrahim	2023170218
Yousef Karam Abdel-fadel Abdel-al	2023170736
Youssef Mohamed Ebrahim Mohamed	2023170783
Khalid abdelHameed hossni abdelHameed	2023170201

Under the Supervision of Dr. Engy Abdallah

Table of Contents:

Introduction

This application is designed to support gyms effectively managing all their daily operations in a smooth and organized way. It covers a wide range of important tasks such as member registration, assigning staff roles, scheduling classes, and handling Padel court reservations. By offering a simple and reliable interface, the system helps automate routine processes that would otherwise require significant manual effort.

Members can easily create personal accounts and choose from various subscription options based on their preferences, including monthly, 3-month, 6-month, or yearly plans. Once registered, they have access to features like class enrollment, court reservations, and workout tracking. Through the system, members can monitor their activity history, receive timely notifications about subscription renewals, and manage their bookings with just a few clicks.

Staff members—including receptionists, coaches, and managers—have tools available to schedule training sessions, assign coaches, handle court availability, and organize waitlists when classes or courts are full. The system automatically manages these waitlists, ensuring fairness by adding users in the order they sign up and promoting them when cancellations occur. In addition, staff can generate monthly reports to track the number of active users and overall revenue, making it easier to monitor the gym’s performance over time.

To improve fairness and user satisfaction, VIP members are given priority when booking courts and enrolling in classes. This ensures they get preferred access during peak times, offering better overall experience for long-term or premium subscribers. The system also includes smart reminders for upcoming subscription renewals and cancellations, reducing the need for staff to manually follow up with members.

Overall Design

The system is structured using a layered design to keep responsibilities clear and the code organized. Each layer handles a specific part of the application's functionality, which helps make the system easier to build, understand, and improve over time.

- **UI Layer (Qt):** This is the part users interact with directly. It includes dialog windows for actions like logging in, registering, booking courts, and scheduling training sessions. Each form is designed to be intuitive and simple to navigate, so members and staff can quickly complete their tasks.
- **Controller Layer:** This layer acts as the bridge between the user interface and the system's core logic. Components like *SearchManager* and *Receptionist* manage user actions, perform lookups, and coordinate operations such as registering users, handling renewals, and managing schedules.
- **Data Layer:** All application data is stored in memory using well-structured C++ containers. Classes like *User, Court,* and *Training* hold real-time information during use. The *FileManager* class manages saving and loading this data to and from text files, making sure everything is preserved between sessions.
- **Business Logic:** This part contains the logic that supports features like reporting, handling VIP booking priority, managing waitlists, and sending reminders. It connects with the other layers to ensure that rules like fair queueing and subscription tracking are consistently applied.

Using a layered approach makes the system more maintainable and scalable. It allows developers to work on separate parts of the project without interfering with each other's work. For example, changes to the UI can be made without affecting how data is stored, and updates to the logic behind booking rules won't require rewriting interface code. This structure also improves readability and makes it easier to track down bugs or make future upgrades. Overall, it supports a clean, organized development process and results in a smooth experience for both users and developers.

Data Structures & Storage

The system relies on C++ containers chosen for their algorithmic guarantees and suitability to specific tasks:

Data Structure Used	Purpose
QList<Court>	A list of Court objects.
map<string, queue<string>>	Maintains ordered waitlists for fair queuing.
priority_queue<pair<int,string>>	Automatically prioritizes VIP members for bookings and waitlist spots.

All application data is stored in structured text files such as *users.txt, courts.txt, classes.txt,* and *notifications.txt*. These files are loaded into memory during application startup and saved back on program exit. This file-based system is simple yet effective for small to medium-sized datasets, and it avoids the overhead of using a full database system. Each data structure was selected not just for performance, but also for ease of use, flexibility, and alignment with real-world requirements like booking priority and historical tracking.

Modules & Classes

FileManager:

- Role: Reads from and writes to text files.
- Key methods: *loadUsers(), saveUsers(), loadCourts(), etc.*

User & Staff:

- Data: Member and staff details (ID, name, DOB, subscription plan, role).
- VIP Flag: Marks priority for booking operations.

Court:

- Data: Court ID, capacity, schedule of booked slots.

Training (Class):

- Data: Class ID, coach ID, date, capacity, enrolled members, waitlist.

WorkoutRecord:

- Data: Workout entries (user ID, date, exercises performed).

Notification:

- Data: Messages for upcoming subscription expirations or booking changes.

SearchManager & Receptionist:

- SearchManager: Handles lookups (available courts/classes) and report generation.
- Receptionist: Manages workflows like member registration, subscription renewals, and cancellations.

UI Forms:

- Qt dialogs for Login, Registration, MainWindow, Booking, and Scheduling.

Covered Requirements

Feature	Implemented In
Member registration & subscription	*Register, User, FileManager*
Staff roles & member lookup	*Receptionist*
Class scheduling & waitlists	*Training, map, queue*
Court booking & rescheduling	*Court, BookCourt, SearchManager*
Workout history	*Main Page, Profile*
Subscription reminders	*Notification, FileManager*
Monthly reporting & revenue tracking	*Main Page, Coach*
VIP booking priority	*User + priority_queue*
GUI implementation	*Qt UI classes*

This table illustrates how every project requirement is linked to a corresponding part of the system. It also acts as a checklist making sure every aspect of the project has been successfully satisfied in accordance with the issued rubric.

Data Loading & Saving

- **Startup***:* *FileManager* loads text files once into in-memory containers.
- **Runtime:** All updates happen in memory.
- **Shutdown:** *FileManager* writes containers back to text files.

This approach ensures the application runs quickly during use while still preserving all updates between sessions. It also minimizes the risk of file corruption, since writing only occurs at the end of the session. The system avoids repetitive read/write operations during runtime, reducing I/O overhead and improving responsiveness.

Inputs/Outputs

This section explains how users interact with the system (inputs) and how the system responds (outputs). Users interact with the program using the graphical interface (GUI), and all changes are saved in files when the program closes.

User Inputs (through the GUI):

1- Login / Register:

- Users enter their username, password, name, date of birth, and subscription type. - The system saves this information in *users.txt* (Figure 1).

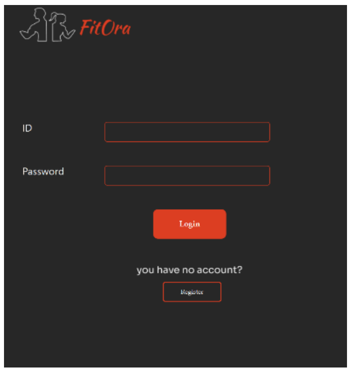
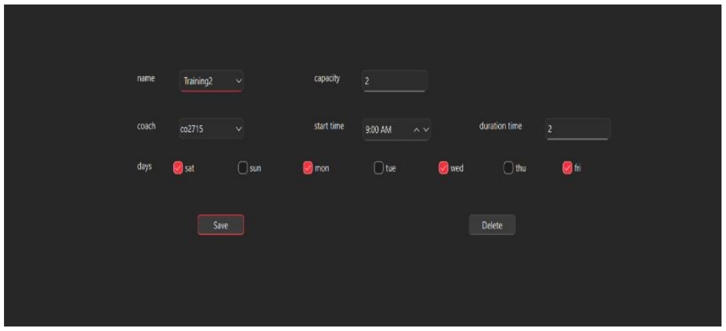


Figure.1: Login Screen

2- Class Scheduling (Staff Only):

- Staff members enter class details like class name, coach ID, date, and number of spots (Figure 2).

- The system stores this in memory and saves it to *classes.txt* at the end.



name: Training2
capacity: 2
coach: co2715
start time: 9:00 AM
duration time: 2
days: [✓ sat] sun [✓ mon] tue [✓ wed] thu [✓ fri]

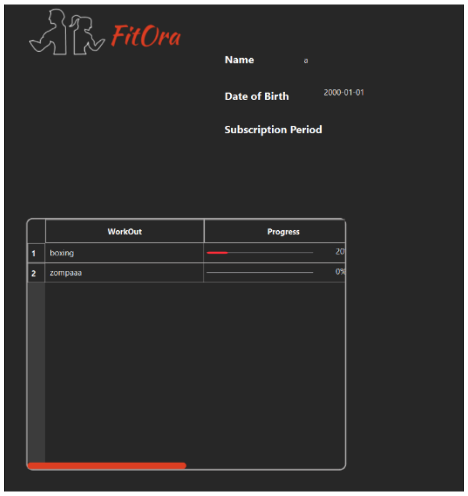
Figure.2: Class Scheduling

3- Booking Padel Courts:

- Members select a date, time, and court to book.
- If the court is available, the booking is confirmed.
- If not, the member is added to a waitlist or shown the next available time.

4- Workout Tracking:

- Members can add their workout details (exercises, date) (Figure 3).
- These are saved to *workouts.txt* when the program ends.



Name a

Date of Birth 2000-01-01

Subscription Period

	WorkOut	Progress
1	boxing	20
2	zompaaa	0%

Figure.3: Workout Tracking

5- Subscription Renewal or Cancellation:

- Users choose to renew or cancel their subscription.
- The system updates their subscription and creates a reminder in *notifications.txt*.

System Outputs:

1- On the Screen (GUI):

- Available classes and courts are shown in real time.
- Users see confirmation messages for bookings and renewals.
- VIP members get a special message when they are given booking priority.
- Notifications show when a subscription is to expire or a waitlist spot opens.

2- Saved to Files:

- *users.txt*: Updated list of users and their subscription info.

- *courts.txt*: Court schedules with all bookings.

- *classes.txt*: Class information and enrolled members.

- *notifications.txt*: Renewal alerts and waitlist updates.

- *workouts.txt*: Workout history for each member.

The system only reads files once at the start and saves everything at the end. This keeps the app fast and avoids issues with repeated file access.

Conclusion

This system combines robust C++ data structures with an intuitive Qt-based interface to deliver a fully functional Gym and Padel Management solution. The software efficiently handles user registrations, staff operations, class scheduling, Padel court bookings, and real-time updates to availability and waitlists. VIP members receive booking advantages through the priority queue, while staff can rely on automated features such as monthly activity reports and renewal notifications.

The project successfully integrates file-based persistence with optimized in-memory operations, ensuring a balance between speed and reliability. The user interface is user-friendly and provides an accessible way for both staff and members to interact with the system. By using appropriate data structures, the program demonstrates good performance and thoughtful problem-solving. This architecture provides a strong foundation for future improvements such as integration with databases, mobile app support, or deeper analytics on member behavior and revenue trends.

References

- *C++ Standard Library - CpPreference.com*. (n.d.). https://en.cppreference.com/w/cpp/standard_library

- QT Documentation | Home. (n.d.). https://doc.qt.io

- Weiss, M. A. (2012). Data Structures & Algorithm Analysis in C++