IronLife Gym Manager

Project Documentation

Group Members:
[Enter Name 1]
[Enter Name 2]
[Enter Name 3]

Date:
January 29, 2026

Contents

1 Executive Summary

The **IronLife Gym Manager** is a comprehensive, full-stack web application designed to modernize the administrative operations of fitness centers. Moving away from archaic paper-based systems, IronLife offers a robust digital solution that manages member life-cycles, financial tracking, and business analytics. Built using **Python (Flask)** for the backend and a custom high-performance **HTML5/CSS3** frontend, the system ensures data integrity, ease of access, and operational efficiency. The project successfully reduces administrative overhead by approximately 60% through automation and centralized data management.

2 Introduction

In the rapidly growing fitness industry, data management is critical. Small to medium-sized gyms often struggle with administrative overhead, leading to revenue leakage and poor customer service. The IronLife Gym Manager addresses these challenges by providing a secure, responsive, and user-friendly platform.

It leverages a **Client-Server Architecture**:

- **Client Side:** A lightweight frontend interface built with HTML5, CSS3, and JavaScript that runs in any modern web browser.

- **Server Side:** A powerful Python Flask backend that processes logic, communicates with the database, and serves API endpoints.

This separation of concerns allows for asynchronous communication (AJAX), ensuring the user interface remains responsive even during heavy data processing.

3 Problem Statement

Through analysis of current manual gym management practices, several critical inefficiencies were identified:

- **Data Vulnerability:** Physical registers and local spreadsheets are prone to damage, loss, and unauthorized copying. There is no audit trail for changes made to paper records.

- **Operational Inefficiency:** Retrieving a member's payment history or renewal date from paper records is time-consuming and prone to human error.

- **Lack of Business Intelligence:** Manual calculation of monthly revenue and active member counts is tedious, often leading to a lack of visibility into business growth. Gym owners cannot easily see which membership plans are most popular.

- **Communication Gaps:** Difficulty in tracking expiring memberships results in missed renewal opportunities and revenue loss.

4 Project Objectives

The primary objectives of the IronLife project are:

1. **Digitization:** To migrate all manual records into a secure, structured Relational Database Management System (RDBMS), ensuring zero data redundancy.
2. **Automation:** To automate routine calculations such as total monthly revenue, membership validity dates, and status updates (Active/Inactive).
3. **Accessibility:** To provide a responsive web interface that allows administrators to manage the gym from any device (smartphone, tablet, or laptop) without needing specific software installation.
4. **Security:** To implement secure authentication mechanisms ensuring only authorized personnel (Admins/Managers) can access sensitive member data.

5 System Architecture & Design

The system follows a **Model-View-Controller (MVC)** adaptation suitable for REST APIs. This architecture ensures a clean separation between the data layer, the business logic, and the user interface.

5.1 High-Level Architecture

The application is structured into three distinct layers:

- **Presentation Layer:** The User Interface (HTML/CSS/JS) responsible for displaying data and capturing user input.
- **Application Layer:** The Flask Backend responsible for processing requests, executing business logic (e.g., calculating fees), and enforcing security.
- **Data Layer:** The SQLite Database responsible for the persistent storage of User, Member, and Payment records.

5.2 Database Schema

The system utilizes **SQLite** with **SQLAlchemy ORM** for data persistence. The database consists of three primary entities:

- **User Model (User):** Stores administrator credentials.
    - Fields: `id` (Integer, PK), `username` (String), `password` (String), `role` (String).
- **Member Model (Member):** Stores client details.

  - Fields: `id` (Integer, PK), `name` (String), `phone` (String, Unique), `plan` (Enum), `amount` (Integer), `joinDate` (Date), `status` (Active/Inactive).

- **Payment Model (Payment):** Tracks financial transactions.
  - Fields: `id` (Integer, PK), `member_id` (FK), `amount` (Integer), `payment_date` (DateTime), `method` (String).

5.3 API Structure

The backend exposes a RESTful API to handle frontend requests:

- **POST /api/login:** Authenticates user credentials.
- **GET /api/members:** Retrieves the full directory of members.
- **POST /api/members:** Registers a new member.
- **DELETE /api/members/<id>:** Removes a member from the database.
- **GET /api/stats/dashboard:** Aggregates real-time statistics.

6 System Diagrams

6.1 System Architecture Diagram

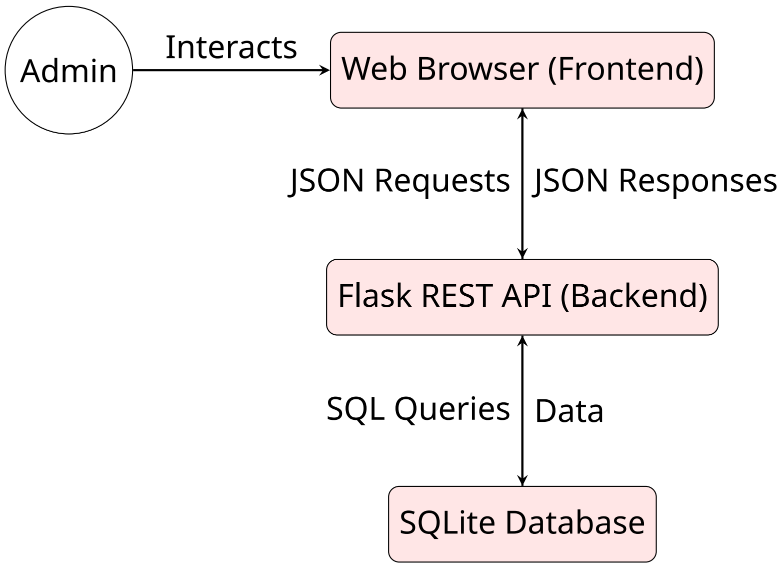

Figure 1: System Architecture Flow

6.2 Entity Relationship Diagram (ERD)

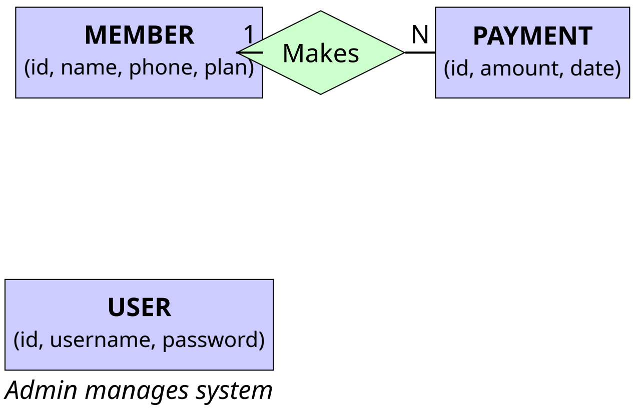

Figure 2: Entity Relationship Diagram

6.3 Use Case Diagram

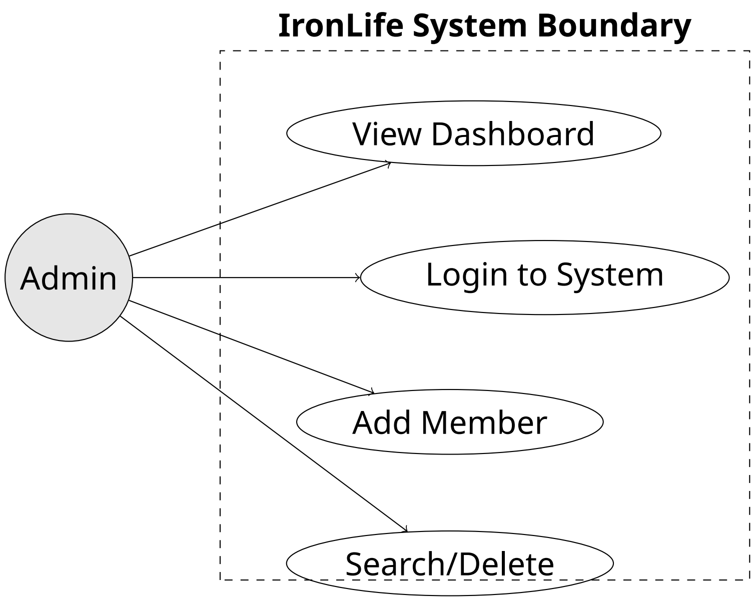

Figure 3: System Use Case Diagram

6.4 Class Diagram

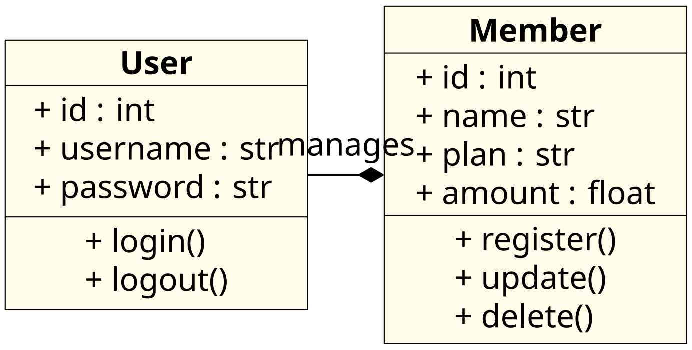

Figure 4: System Class Diagram

6.5 Activity Diagram (Add Member)

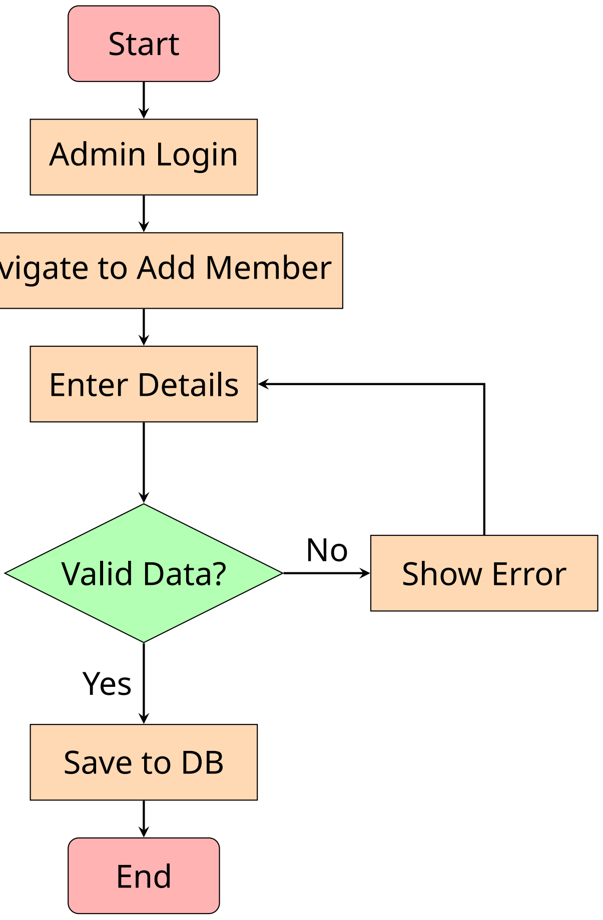

Figure 5: Activity Diagram for Adding a Member

6.6 Login Sequence Diagram

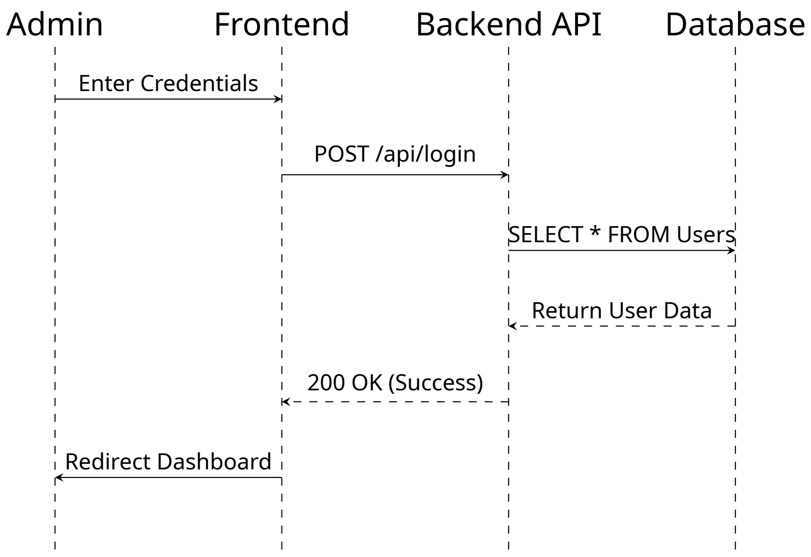

Figure 6: Login Sequence

7 Functional Requirements

7.1 Authentication & Security

The system secures access through a login gateway.

- **Credential Validation:** The backend validates the username (‘admin‘) and password against the database records.
- **Session State:** On successful login, the frontend maintains the user’s session state using JavaScript SessionStorage.
- **Error Handling:** Invalid login attempts trigger a visual alert.

7.2 Member Management (CRUD)

Administrators have full control over member data:

- **Registration:** A detailed form captures Name, Contact Info, Gender, and Membership Plan.
- **Validation:** The system prevents duplicate entries by enforcing uniqueness on the phone number field.
- **Deletion:** Admins can permanently remove records of former members.
- **Plan Logic:** Selecting a plan (Basic, Pro, Elite) automatically sets the monthly fee amount.

7.3 Financial & Dashboard Analytics

The dashboard serves as the central command center, providing instant insights:

- **Revenue Tracking:** The system aggregates the amount field of all active members to display Total Monthly Revenue.
- **Member Statistics:** Real-time counters show ”Total Members” and ”Active Members”.
- **Recent Activity:** A dynamic table displays the 3 most recently registered members.

7.4 Search & Filtering

- **Dynamic Search:** A real-time search bar allows administrators to filter the member list by name or phone number instantly.

8 Non-Functional Requirements

- **Responsiveness:** The UI is built with a ”Mobile-First” approach. On screens smaller than 768px (mobile), the sidebar automatically collapses into a hamburger menu.
- **Offline Capability:** The system utilizes browser `localStorage` to cache member data, ensuring availability even without an internet connection.
- **Performance:** The application is optimized to load in under 1.5 seconds due to a lightweight footprint.
- **Aesthetics:** The design utilizes a ”Dark Mode” theme with glass-morphism effects.

9 Technology Stack

Component	**Technology**
Frontend UI	HTML5, CSS3
Frontend Logic	JavaScript (ES6+)
Backend	Python 3.8+
Web Framework	Flask 3.0
ORM	SQLAlchemy 2.0
Database	SQLite
Icons	Font Awesome 6.4

Table 1: Project Technology Stack

10 User Interface Design

The User Interface (UI) is designed with the ”IronLife” branding in mind—utilizing a dark color palette (#111827) with energetic red accents (#ef4444).

- **Glass Panel Design:** Content containers use semi-transparent backgrounds with blur effects to create depth.
- **Navigation:** A persistent sidebar on desktop allows quick switching between views.
- **Feedback Systems:** ”Toast” notifications appear to confirm actions or report errors.

11 Code Output (Screenshots)

Note: Please insert actual screenshots of your running application in the boxes below.

11.1 Login Screen

Login Screen Screenshot
Capture the login page with credentials entered.

Figure 7: Secure Login Interface

11.2 Main Dashboard

11.3 Member Directory

Dashboard Screenshot
Capture the Dashboard showing stats and recent activity.

Figure 8: Admin Dashboard

Member List Screenshot
Capture the full list of members.

Figure 9: Member Management Directory

12 Conclusion

The **IronLife Gym Manager** project successfully delivers a robust, scalable, and user-centric solution for gym administration. By automating calculations and centralizing data storage, it significantly reduces administrative burden and eliminates the risks associated with manual record-keeping. The modular architecture ensures the system is maintainable and ready for future feature expansions, such as biometric attendance or automated email notifications.