PROJECT REPORT

by

THRISHA MAE P. SABELLINA

A Project Report Submitted in Partial Fulfillment of the Requirements for Information Management and Web Development in the Degree of Bachelor of Science in Computer Science In the Faculty of College of Computer Studies At Kolehiyo ng Lungsod ng Lipa Marawoy Lipa City

January 09, 2026

Supervised by Ms. Arlyn de Villa

Information Management and Web Development Instructor

I. Introduction

GymFlow.PH is a streamlined, web-based management solution designed specifically for local fitness centers in the Philippines. In an era where manual logbooks are becoming obsolete, Gym Flow provides a digital bridge for gym owners and their members.

The primary goal of the project is to simplify the **membership lifecycle**—from initial registration to daily attendance tracking. It addresses the need for financial transparency by displaying membership costs and promoting member consistency through a dedicated "Self-Check-In" portal. By automating attendance logs and membership tier management, Gym Flow reduces administrative overhead and provides members with immediate access to their personal fitness stats.

Technologies Used

The project follows a classic **LAMP-stack inspired** architecture, focusing on high performance and ease of deployment on local servers like XAMPP.

- **Frontend (The View):**
    - **HTML5 & CSS3:** Utilizes modern Flexbox and CSS Grid for a responsive, mobile-first design.
- **JavaScript:** Handles asynchronous API calls (Fetch API) to update the dashboard without reloading the page.
- **Backend (The Controller):**
- **PHP 8.x:** Manages server-side logic, session security, and communication between the frontend and the database.
- **Database (The Model):**

- **MySQL:** Stores relational data across structured tables with optimized constraints to ensure data integrity.
- **Tools:**
- **XAMPP:** Local development environment.
- **MySQL Workbench:** For database modeling and schema management.

Database Design

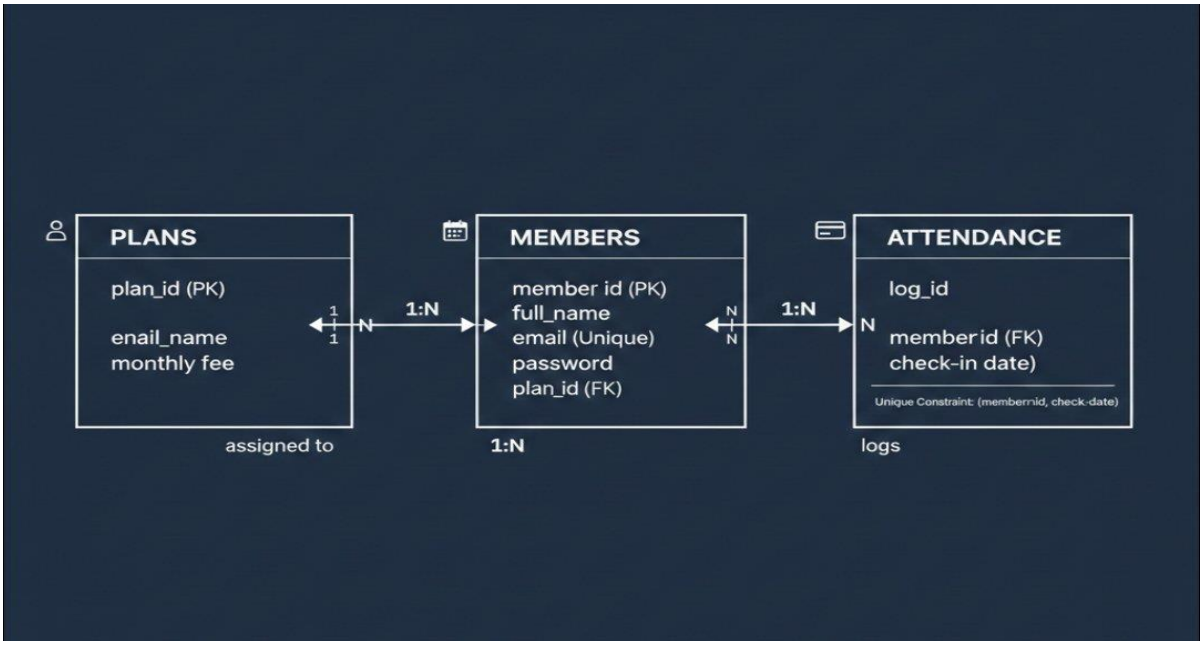
Entities and Relationships

1. **Plans (Membership Tiers):**
    a. **Attributes:** plan_id (PK), plan_name, monthly_fee.
    b. **Relationship:** One-to-Many with **Members**. One plan (e.g., "Student Pass") can be assigned to many different members, but a member can only belong to one plan at a time.

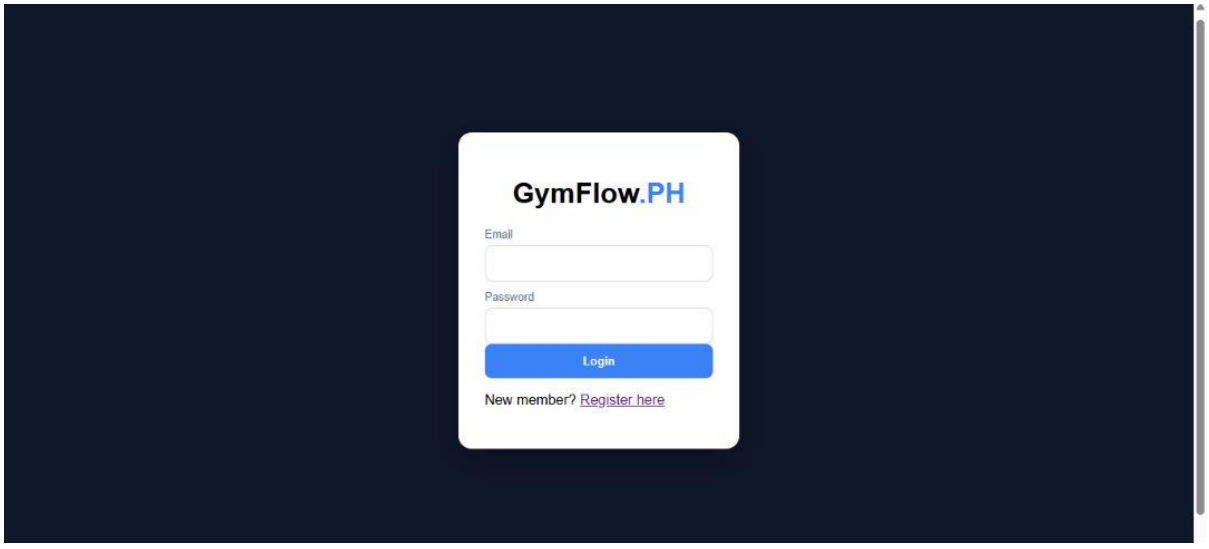
2. **Members (Users):**
    a. **Attributes:** member_id (PK), full_name, email (Unique), password, plan_id (FK).
    b. **Relationship:** The "Heart" of the database. It connects a user to their financial obligations (Plans) and their physical activity (Attendance).

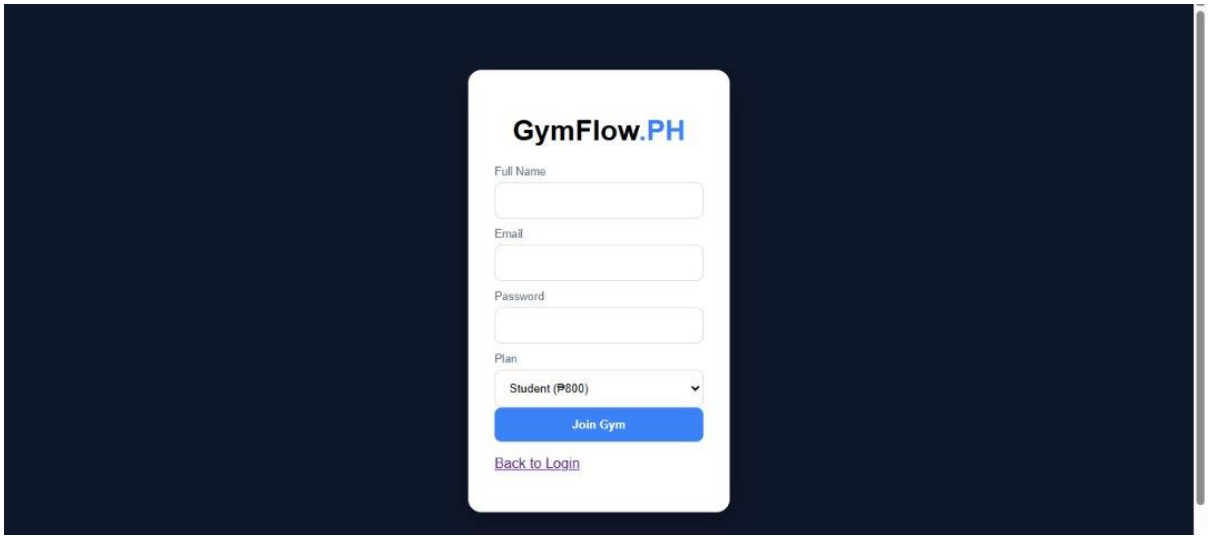
3. **Attendance (Daily Logs):**
    a. **Attributes:** log_id (PK), member_id (FK), check_in_date.
    b. **Relationship:** Many-to-One with **Members**. A single member can have many attendance records (one for every day they visit), but each attendance record belongs to exactly one member.

c. **Unique Constraint:** A composite unique key is applied to (member_id, check_in_date) to ensure a member cannot record two sessions on the same day.



WEB INTERFACE





CHALLENGES AND LEARNING

. Technical Environment & Debugging

- **Hardware Constraints:** Developing on a system with frequent errors meant constant restarts and troubleshooting. This made it difficult to maintain a "flow state" but taught the importance of frequent saving and version control.
- **XAMPP & Port Conflicts:** One major challenge was getting Apache and MySQL to run consistently. Often, other system processes would block ports, requiring deep dives into the BIOS or Task Manager to resolve conflicts.
- **Logic Errors in PHP:** Setting up the "Self Check-In" logic was tricky. Ensuring that the code correctly identifies if a user has *already* checked in for the day required precise SQL queries and session handling.

2. The "Tutorial Trap"

- **Version Mismatch:** Watching YouTube tutorials often led to confusion because many videos use older versions of PHP (like 5.6 or 7). Adapting old code to work with **PHP 8.x** requirements was a significant hurdle.
- **Fragmented Information:** Tutorials often show how to build a login *or* a database, but rarely how to connect them.

Key Learnings

1. Full-Stack Integration

I learned how the "Three Pillars" of web development interact. I now understand how a button click in **HTML/JS** sends a request to **PHP**, which then updates a row in **MySQL**. This "Full Cycle" understanding is much more valuable than just writing code.

2. Database Integrity

Before this project, I didn't realize how important "Unique Keys" were. By adding a unique constraint to the attendance table, I learned how to let the database handle logic (preventing double entries) instead of relying solely on the PHP code.

3. Professional Formatting

Working on the Philippine Peso (₱) integration taught me about **Localization**. Using .toLocaleString('en-PH') showed me how professional apps adapt their data to be user-friendly for a specific target market.

4. Persistence in Troubleshooting

The constant errors on my laptop actually became a "teacher." I learned that 70% of software engineering is not writing code, but **googling the error message** and reading documentation. This project proved that you don't need a perfect computer to build a working system—you just need persistence.