CS 2300 Database Project – Phase III: Gym Management System

Isaac Stephens
Junior, Computer Science, CS Department
Missouri University of Science and Technology
Rolla, MO 65409
issq3r@mst.edu

I. Introduction

The goal of this phase is to fully implement the Gym Management System (GymMan) proposed in Phases I and II using a relational database management system and an application interface. This phase focuses on transforming the logical and physical database design into a functional software system, population it with realistic data, and validating all operations through structured testing.

The system is implemented using MariaDB as the relational database and the Python framework Flask as the application framework. All database tables, constrains, and relationships are instated using a single SQL deployment script `gymman_demo_source.sql`. The application implements full transactional support for membership management, exercise logging, staff and trainer operations, and financial tracking.

II. Database Implementation

A. Database Management System Selection

The GymMan system is implemented using MariaDB, a fully relational, open source SQL database system compatible with MySQL syntax and tooling. MariaDB was selected because it provides:

- Full support for relational constraints and transactions
- Wide platform compatibility
- Strong Python integration through `mysql-connector-python`
- Production-grade performance for web applications.

The database is created and initialized using a single SQL schema deployment script: `gymman_demo_source.sql`.

This script:

- Drops all existing tables if they exist
- Recreates all relational tables in correct dependency order
- Enforces all primary keys, foreign keys, and referential integrity rules
- Implements all subtype and specialization relationships
- Enables cascade deletion where required

B. Table Implementation Summary

All major entities and relationships defined in the conceptual and logical designs are implemented using normalized relational tables. Core tables include:

- Members, PhoneNumbers, EmergencyContacts, Payments, Checkins
- Staff, Trainers, Managers, Maintenance, Contractors, Hourly_Employees, Salary_Employees
- TrainerCertifications, TrainerClients
- Exercises, Strength_ Exercises, Cardio_Exercises, Runs, Bike_Rides
- users, roles

Each table uses a surrogate integer primary key. Referential integrity is enforced using foreign keys with cascade deletion where relationships are dependent. Aggregated values such as total revenue or average RPE are computed dynamically through SQL queries rather than stored directly.

III. Conceptual Database Design

The conceptual design for GymMan models the real-world entities involved in gym operations and their interrelationships. Compared to Phase I, the conceptual design has been refined to explicitly include application-level authentication entities and attendance tracking.

A. Core Entities

1. **Members**: Represents individuals registered at the gym. Members store personal and membership information and participate in most system operations.
2. **PhoneNumbers**: Represents one or more phone numbers associated with a member.

3. **EmergencyContacts**: Weak entity dependent on Members, storing emergency contact information.
4. **Payments**: Tracks all financial transactions between members and the gym.
5. **Staff (Super-Type)**: Represents all gym employees. Subtypes include Trainers, Managers, Maintenance, Contractors, Hourly_Employees, and Salary_Employees. Each staff member belongs to up to 2 subtypes: one for their payment type, and one for their role.
6. **TrainerCertifications**: Stores professional certifications held by trainers.
7. **TrainerClients (Associative Entity)**: Resolves the many-to-many relationship between Trainers and Members, tracking client engagement periods.
8. **Exercises (Super-Type)**: Represents all workouts performed by members. Subtypes include Strength_Exercises and Cardio_Exercises. Cardio further specialized into Runs and Bike_Rides.
9. **Checkins**: Tracks member attendance by logging check-in timestamps.
10. **Users & Roles**: Supports authentication and role-based access control for Owners, Staff, Trainers, and Members.

B. Relationship Summary

- Member 1–N PhoneNumbers
- Member 1–N EmergencyContacts
- Member 0–N Payments
- Member 1–N Exercises
- Member 1–N Checkins
- Trainer 1–N TrainerClients
- Member 0–N TrainerClients
- Exercises 0–1 Strength OR 0–1 Cardio
- Cardio 0–1 Runs OR 0–1 Bike_Rides
- Staff 1-2 Subtype

Derived metrics such as total revenue, average RPE, and maximum lifted weight are computed dynamically through SQL aggregation and are not stored as attributes.

IV. Logical Database Design

The conceptual schema is mapped to a fully normalized relational design implemented in MariaDB. All relations satisfy Third Normal Form (3NF) or higher. Many-to-many relationships are resolved using associative tables, and specialization is implemented via 1–1 foreign key mappings.

Each relation follows the structure:

- Primary Key (PK) enforces entity uniqueness
- Foreign Keys (FK) enforce referential integrity
- Check constraints enforce domain rules
- Unique constraints prevent duplication of natural keys (email, SSN, usernames, etc.)

All logical tables match the full specification defined in the Phase II design and are implemented in `gymman_demo_source.sql`.

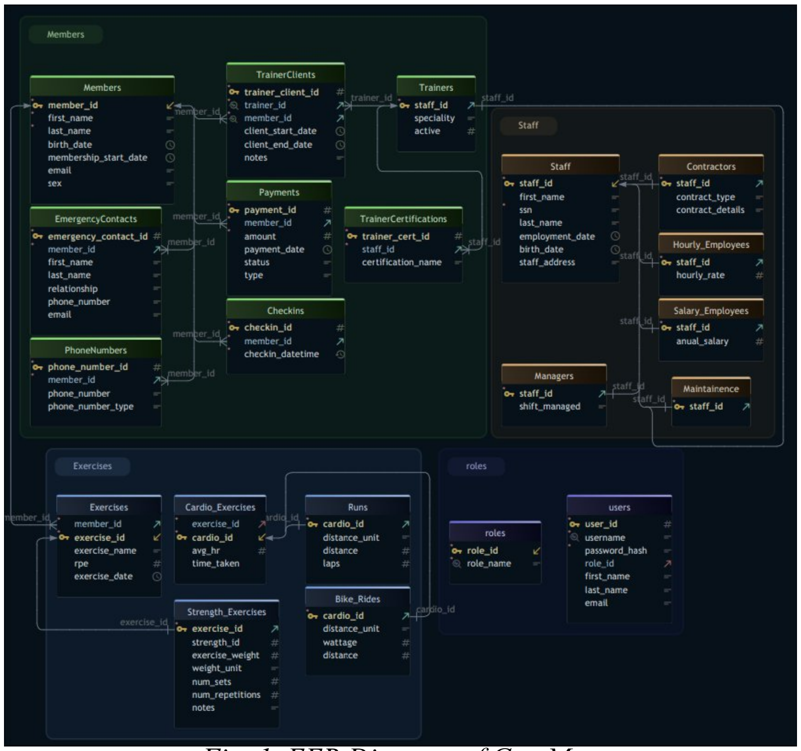


Fig. 1. EER Diagram of GymMan

V. Application Program Design

The implemented system follows a three-layer architecture:

1. **Presentation Layer**: Flask routes and Jinja2 HTML templates provide the user interface.
2. **Application Logic Layer (`auth.py`)**: Handles session management, role-based routing, form validation, and transaction control.
3. **Data Access Layer: (`models.py`)**: Encapsulates all SQL operations using parameterized queries.

A. Core Operations Implemented

- Member registration, modification, and deletion
- Emergency contact and phone number management
- Member check-in tracking
- Staff and trainer registration
- Trainer-client assignment
- Payment creation and revenue aggregation
- Exercise logging for strength and cardio workouts
- Exercise modification and deletion
- Trainer client lists
- Member exercise histories
- System-wide aggregations (average RPE, max weight, total revenue, average run distance)

All multi-step operations are executed inside SQL transactions with rollback on failure. Parameterized queries prevent SQL injection.

VI. Installation Instructions

A. Intended Operating System

The GymMan system is designed for systems using Debian 12 “Bookworm” or newer, or Ubuntu 22.04 LTS or newer.

B. System Dependencies

System Packages (via apt):

- mariadb-server
- libmariadb-dev-compat
- python3
- python3-venv
- python3-pip

Python Packages (throught venv):

- flask
- mysql-connector-python
- python-dotenv
- gunicorn

C. Installation Steps

A full guide on how to setup the system can be found at: https://github.com/Isaac-Stephens/Gym-Manager/blob/main/documentation/BUILD.md. Here are the basic steps:

Clone the repository:

```
git clone
https://github.com/<username>/Gym-
Manager.git
cd Gym-Manager
```

Run the installation script:

```
chmod +x gymman_install_DEBIAN.sh
./gymman_install_DEBIAN.sh
```

Secure MariaDB:

```
sudo mysql_secure_installation
```

Create the database and user:

```
sudo mysql
CREATE DATABASE gymman;
CREATE USER 'gymman_user'@'localhost'
IDENTIFIED BY 'strong_password_here';
GRANT ALL PRIVILEGES ON gymman.* TO
'gymman_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Import schema and demo data:

```
mysql -u gymman_user -p gymman <
scripts/create_gymman_tables.sql
mysql -u gymman_user -p gymman <
scripts/gymman_demo_source.sql
```

Configure environment variables in `gymman-demo/db.env.

Start the application:

```
cd gymman-demo
source venv/bin/activate
python3 main.py
```

The application runs locally at: http://127.0.0.1:5000.

VII. User Manual

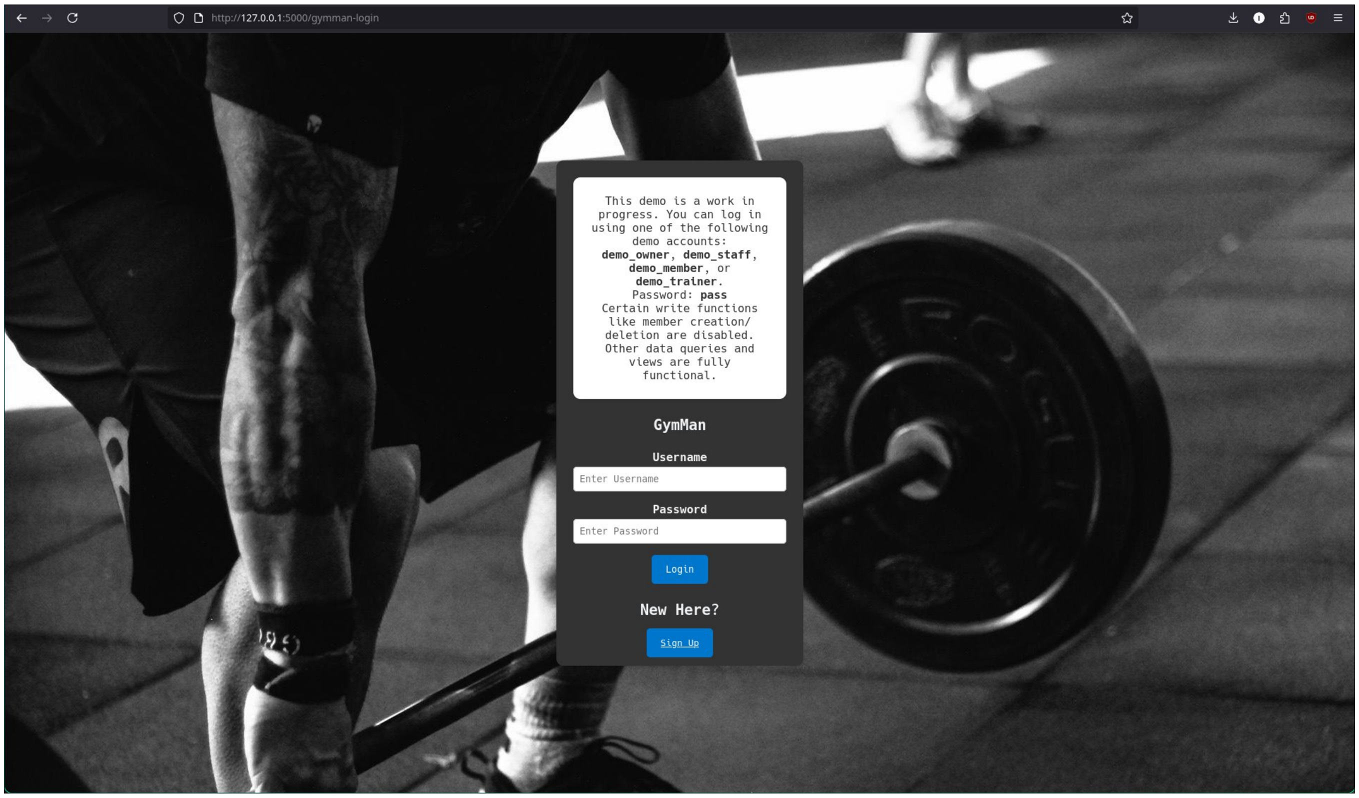
This section is written for end-users who may not have database or programming experience. It explains how to use the GymMan system step by step, with references to screenshots and sample outputs. All screenshots should be captured from a running instance of the application and inserted where indicated.

At the time of this report, the Owner view is the only role with a fully implemented and complete user interface. All core administrative workflows including member management, check-in tracking, trainer assignment, payment management, exercise logging, and system-wide metrics are fully accessible and functional through the Owner dashboard and its associated pages. The Staff, Trainer, and Member interfaces currently exist as functional route endpoints with partial page layouts, but most of their interactive features are still under active development. However, all underlying database operations and permission controls for these roles are fully implemented in the backend and verified through the Owner interface. This ensures that while the graphical interfaces for non-owner roles are incomplete at this stage, the system's core logic, security model, and database transactions are already fully operational and correctly enforced. Future development will complete these remaining interfaces and extend the system into a fully unified multi-user production application.

For the examples below, it is assumed that the web application is running at: http://127.0.0.1:5000 or at the deployed URL provided by the administrator.

A. Accessing the System

1. Open a web browser (Chrome, Firefox, or Edge).
2. In the address bar, enter the provided URL for the GymMan system, for example: http://127.0.0.1:5000 or https://isaacstephens.com/gymman-login.
3. Press Enter. You will see the GymMan Login Page.



This demo is a work in progress. You can log in using one of the following demo accounts: **demo_owner**, **demo_staff**, **demo_member**, or **demo_trainer**. Password: **pass** Certain write functions like member creation/ deletion are disabled. Other data queries and views are fully functional.

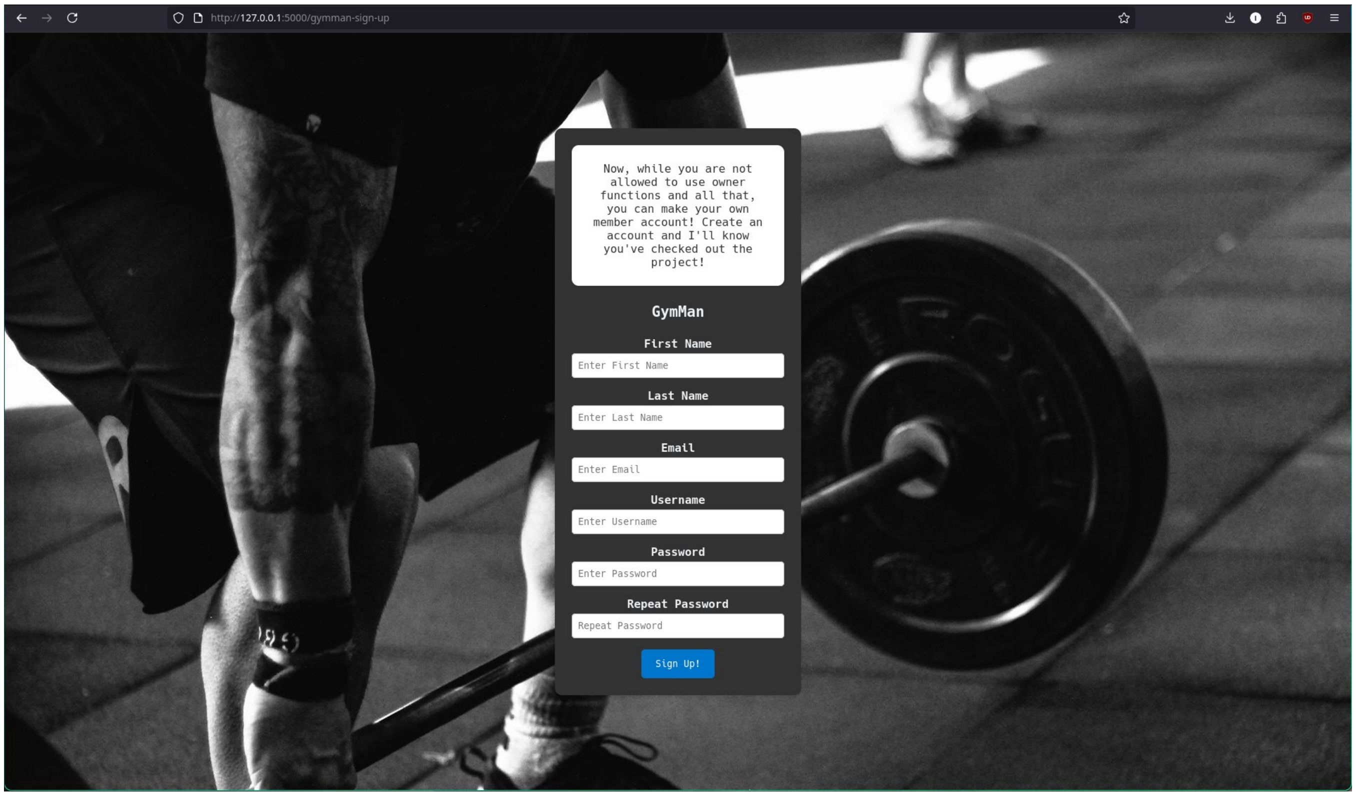
Fig. 2. GymMan Login Page

B. Creating a Member Account (Sign Up)

If you are a new member and do not yet have an account:

1. From the login page, click the "Sign Up" button.
2. You will be taken to the Member Sign Up form.

    a. First Name
    b. Last Name
    c. Email
    d. Username
    e. Password
    f. Repeat Password
3. Fill out all field carefully.
    a. The email and username must be unique.
    b. The two passwords must match.
4. Click the “Sign Up!” button at the bottom of the form.
5. If the information is valid, you will see a success message.



http://127.0.0.1:5000/gymman-sign-up

Now, while you are not allowed to use owner functions and all that, you can make your own member account! Create an account and I'll know you've checked out the project!

GymMan

First Name: Enter First Name

Last Name: Enter Last Name

Email: Enter Email

Username: Enter Username

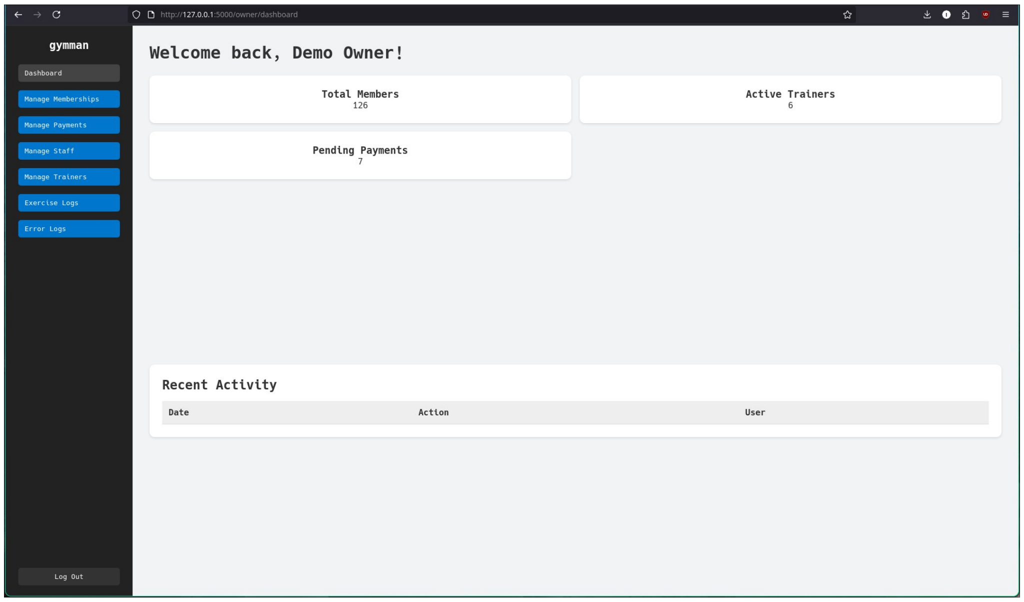
Password: Enter Password

Repeat Password: Repeat Password

Fig. 3. GymMan Sign Up page

C. Logging In

1. On the login page, enter your Username and Password.
2. Click the “Log In” button.
3. If your credentials are correct, you will be redirected to a dashboard based on your role:
    a. Owner Dashboard
    b. Staff Dashboard
    c. Trainer Dashboard
    d. Member Dashboard



Welcome back, Demo Owner!

Total Members
126

Active Trainers
6

Pending Payments
7

Recent Activity

Date	Action	User

Fig. 4. Example Dashboard after Login (Owner View)

If the username or password is incorrect, the system will display a clear error message.

D. Owner Workflows

The owner role has access to the highest level of management features. After login, an owner sees the Owner Dashboard.

1. Viewing the Owner Dashboard

The dashboard shows high-level metrics such as:

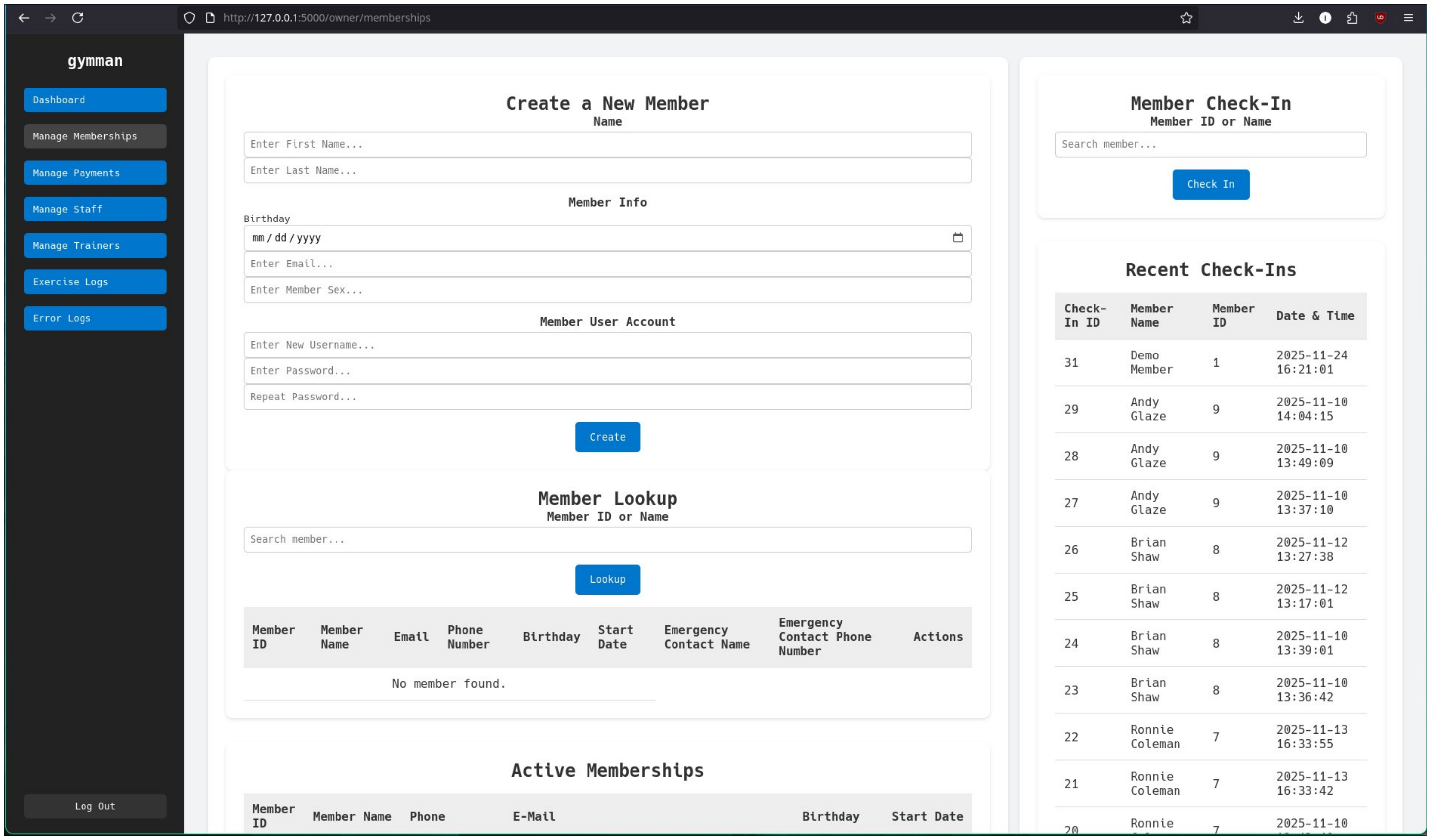
- Total number of members
- Number of pending payments
- Number of active trainers

These values are displayed near the top of the page, as shown in Figure 4.

2. Managing Memberships

To manage member accounts:

1. From the owner sidebar navigation menu, click "Memberships".
2. The Owner Membership page displays:
    a. A form to add a new member.
    b. A table listing all members (name, email, phone, dates)
    c. A panel of recent member check-ins
    d. A search bar to look up specific members



Create a New Member

Name

Enter First Name...

Enter Last Name...

Member Info

Birthday

mm / dd / yyyy

Enter Email...

Enter Member Sex...

Member User Account

Enter New Username...

Enter Password...

Repeat Password...

Create

Member Lookup

Member ID or Name

Search member...

Lookup

Member ID	Member Name	Email	Phone Number	Birthday	Start Date	Emergency Contact Name	Emergency Contact Phone Number	Actions
		No member found.						

Active Memberships

Member ID	Member Name	Phone	E-Mail	Birthday	Start Date

Member Check-In

Member ID or Name

Search member...

Check In

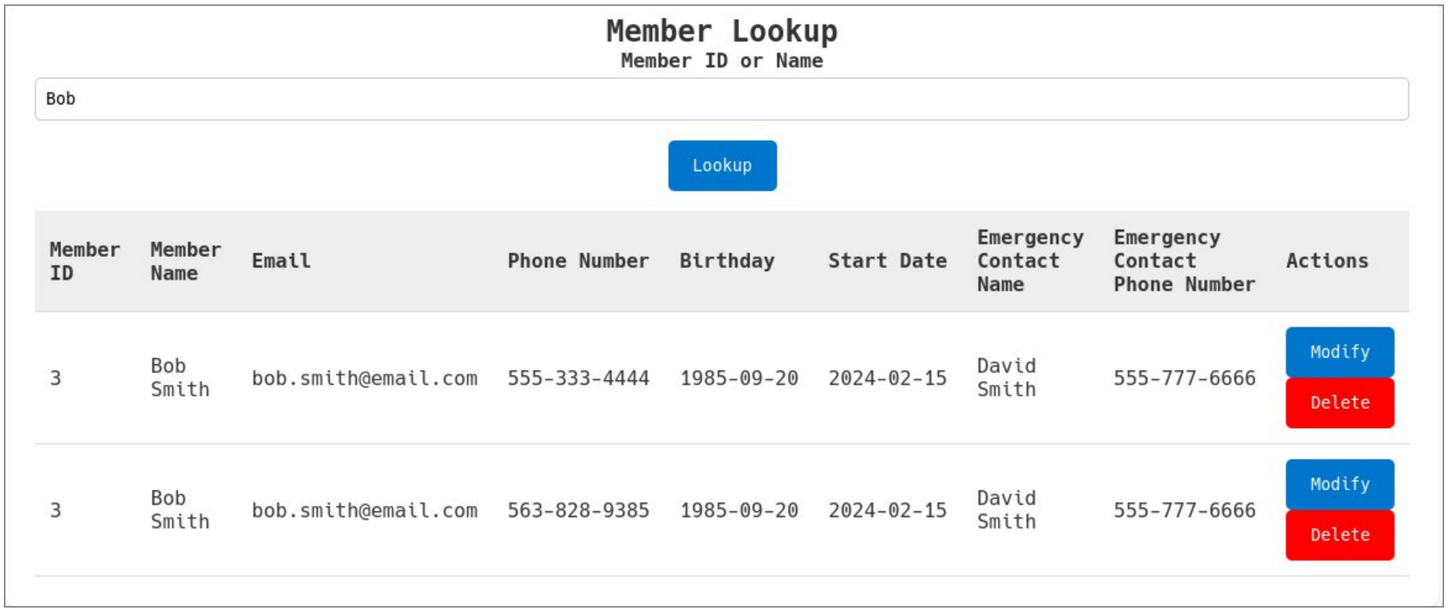
Recent Check-Ins

Check-In ID	Member Name	Member ID	Date & Time
31	Demo Member	1	2025-11-24 16:21:01
29	Andy Glaze	9	2025-11-10 14:04:15
28	Andy Glaze	9	2025-11-10 13:49:09
27	Andy Glaze	9	2025-11-10 13:37:10
26	Brian Shaw	8	2025-11-12 13:27:38
25	Brian Shaw	8	2025-11-12 13:17:01
24	Brian Shaw	8	2025-11-10 13:39:01
23	Brian Shaw	8	2025-11-10 13:36:42
22	Ronnie Coleman	7	2025-11-13 16:33:55
21	Ronnie Coleman	7	2025-11-13 16:33:42
20	Ronnie	7	2025-11-10

Fig. 5. Owner memberships management view

In order to add a new member, use the "Create a New Member" form to input all required fields. Click the "Create" button. If the information is valid and the username and email are unique, the new member will appear in the member list and a confirmation message will be shown.

In order to modify an existing member, use the "Member Lookup" search panel to search for the member you want to edit. Once you've found the member, click the "Modify" button next to their name.



Member Lookup

Member ID or Name

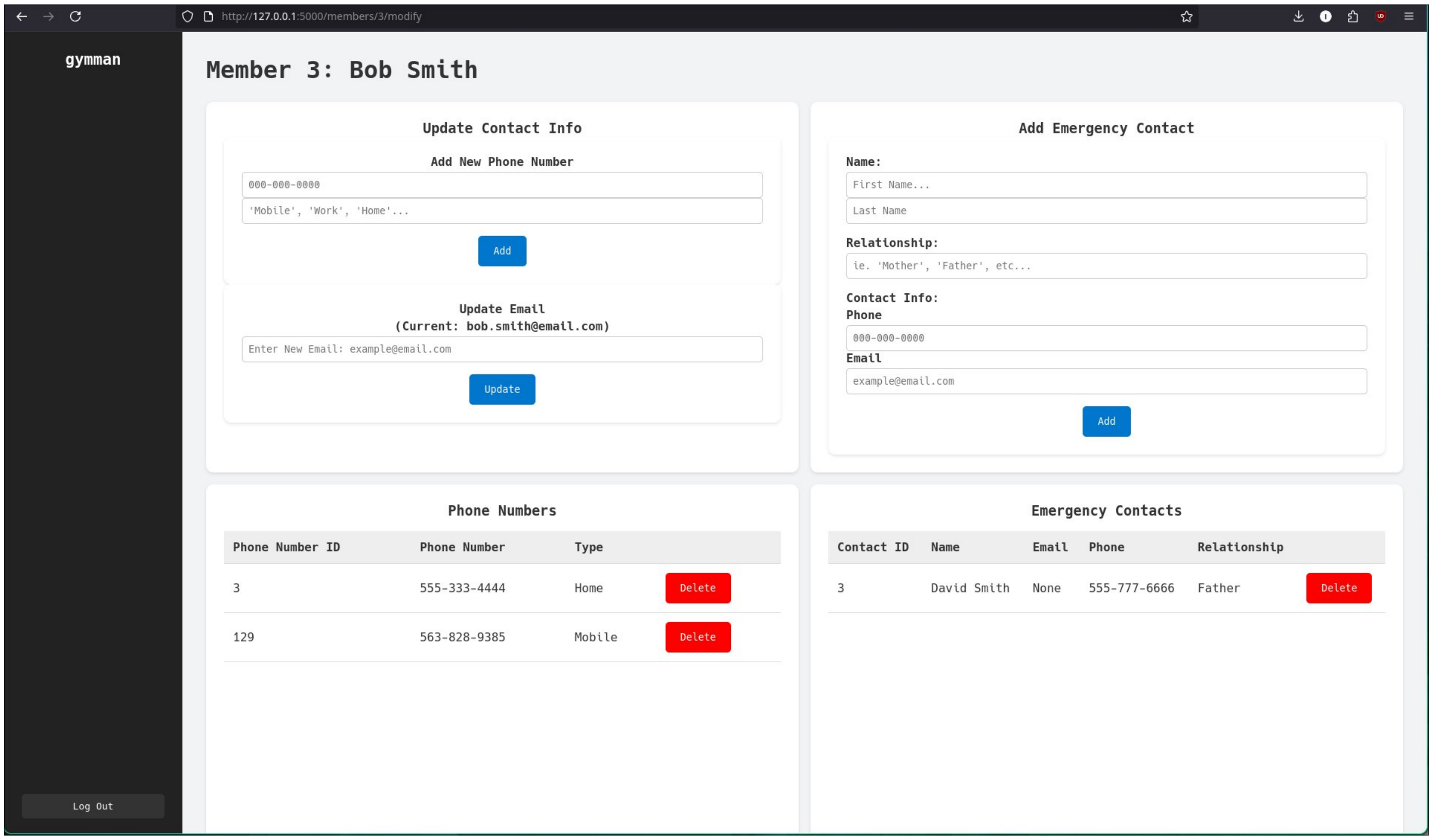
Bob

Lookup

Member ID	Member Name	Email	Phone Number	Birthday	Start Date	Emergency Contact Name	Emergency Contact Phone Number	Actions
3	Bob Smith	bob.smith@email.com	555-333-4444	1985-09-20	2024-02-15	David Smith	555-777-6666	Modify Delete
3	Bob Smith	bob.smith@email.com	563-828-9385	1985-09-20	2024-02-15	David Smith	555-777-6666	Modify Delete

Fig. 6. Member Lookup Form

The Modify Member page shows basic member information such as phone numbers and emergency contacts.



Member 3: Bob Smith

Update Contact Info

Add New Phone Number

000-000-0000

'Mobile', 'Work', 'Home'...

Add

Update Email
(Current: bob.smith@email.com)

Enter New Email: example@email.com

Update

Add Emergency Contact

Name:

First Name...

Last Name

Relationship:

ie. 'Mother', 'Father', etc...

Contact Info:

Phone

000-000-0000

Email

example@email.com

Add

Phone Numbers

Phone Number ID	Phone Number	Type	
3	555-333-4444	Home	Delete
129	563-828-9385	Mobile	Delete

Emergency Contacts

Contact ID	Name	Email	Phone	Relationship	
3	David Smith	None	555-777-6666	Father	Delete

Fig. 7. Modify Member Page

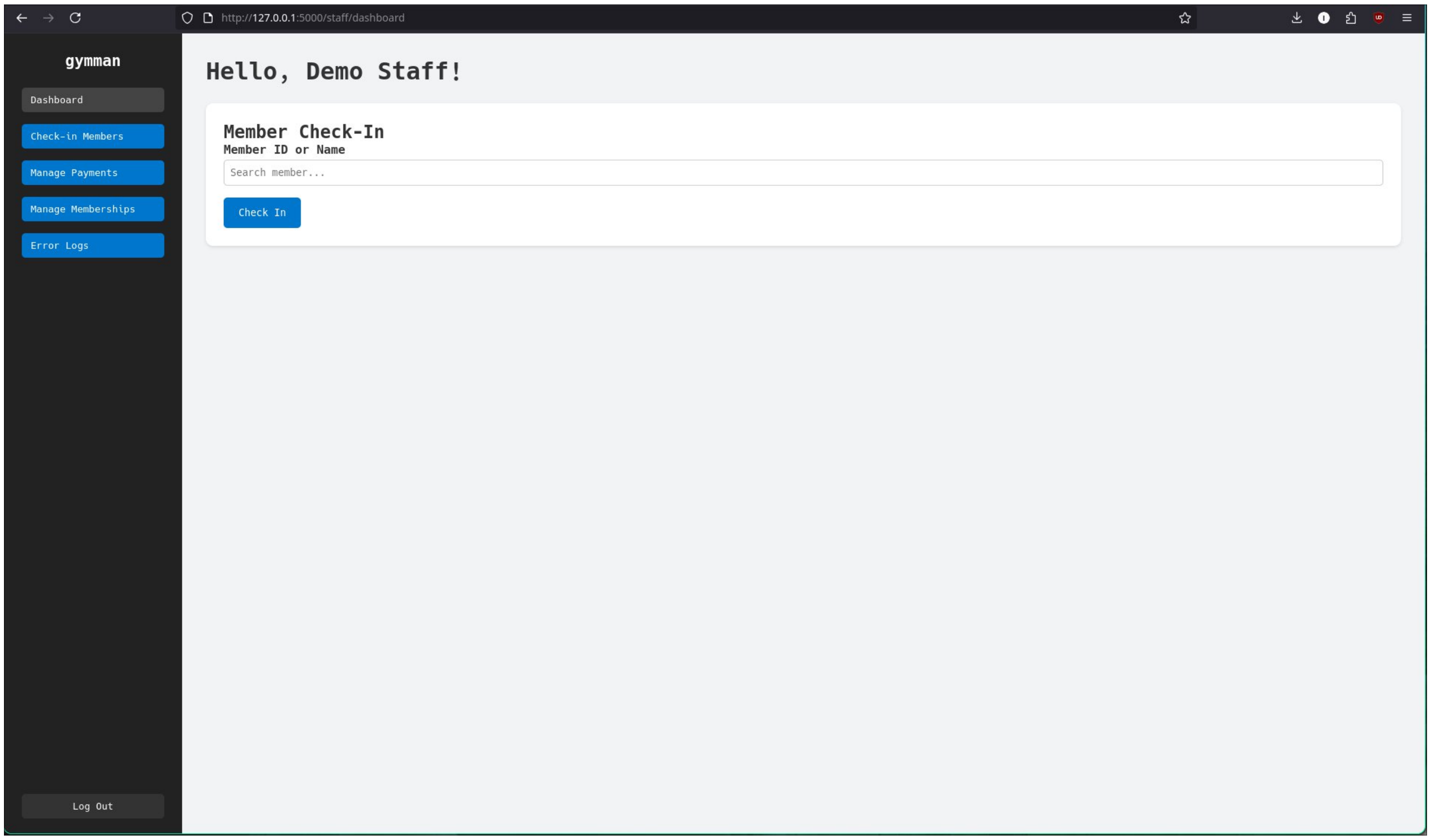
Deleting a member can be done from the owner view can be done through the “Member Lookup” form by pressing the “Delete” button for that member.

Owners can check in a member directly from the Memberships page by searching the member’s member ID, or name in the “Member Check-In” form, shown in Figure 5, and then clicking the “Check In” button.

E. Staff Workflows

The Staff role has similar membership and check-in capabilities as the Owner, but without full administrative control.

After logging in as Staff, the user is taken to the Staff Dashboard, which contains quick access to check-in tools and a link to the Checkins page.



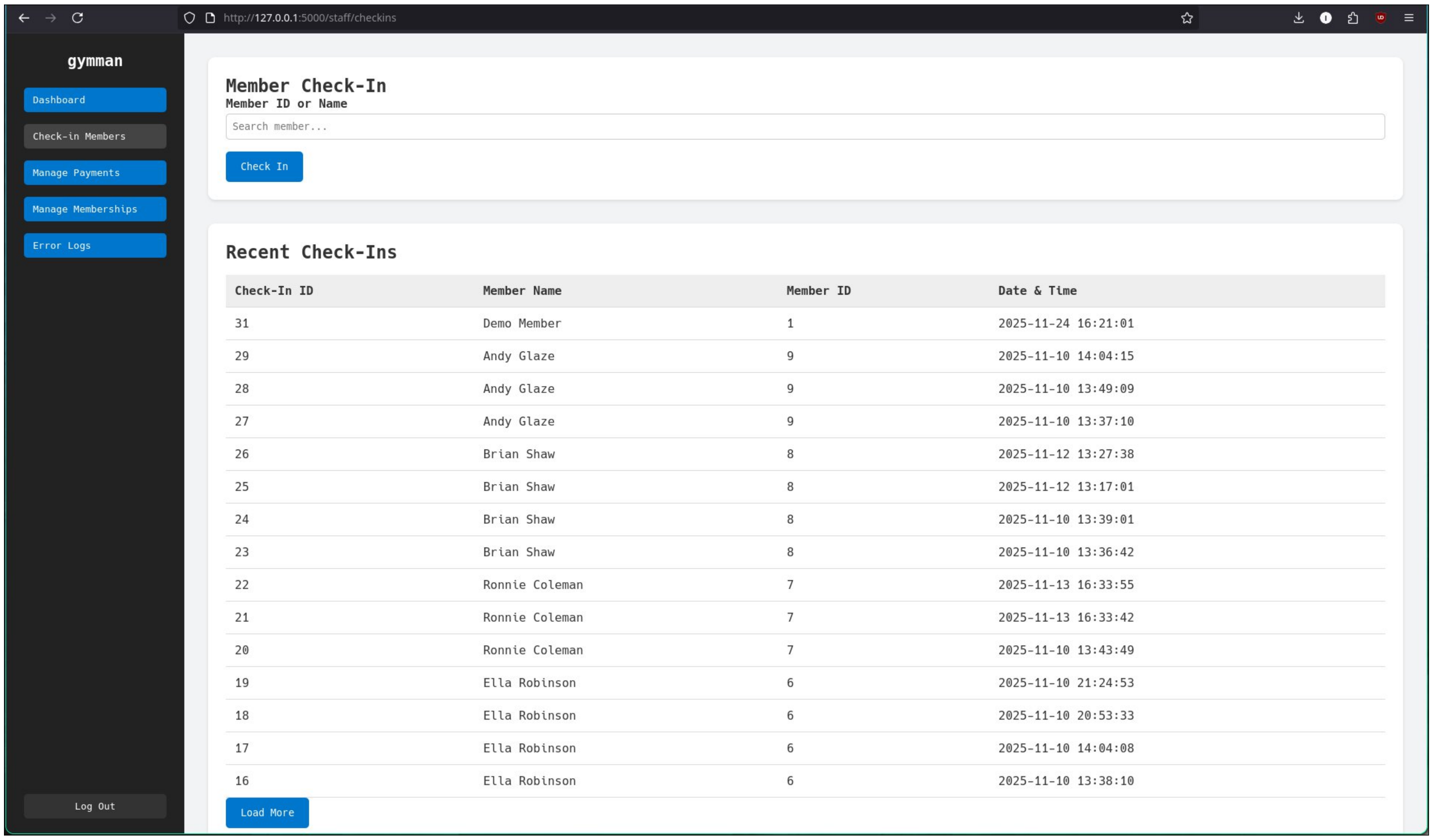
Hello, Demo Staff!

Member Check-In

Member ID or Name

Search member...

Fig. 8. Staff Dashboard



Member Check-In

Member ID or Name

Search member...

Recent Check-Ins

Check-In ID	Member Name	Member ID	Date & Time
31	Demo Member	1	2025-11-24 16:21:01
29	Andy Glaze	9	2025-11-10 14:04:15
28	Andy Glaze	9	2025-11-10 13:49:09
27	Andy Glaze	9	2025-11-10 13:37:10
26	Brian Shaw	8	2025-11-12 13:27:38
25	Brian Shaw	8	2025-11-12 13:17:01
24	Brian Shaw	8	2025-11-10 13:39:01
23	Brian Shaw	8	2025-11-10 13:36:42
22	Ronnie Coleman	7	2025-11-13 16:33:55
21	Ronnie Coleman	7	2025-11-13 16:33:42
20	Ronnie Coleman	7	2025-11-10 13:43:49
19	Ella Robinson	6	2025-11-10 21:24:53
18	Ella Robinson	6	2025-11-10 20:53:33
17	Ella Robinson	6	2025-11-10 14:04:08
16	Ella Robinson	6	2025-11-10 13:38:10

Fig. 9. Staff Check-Ins View