COMP3005 Final Project 2 Report

Matthew Khoury, #101188452

Evan Baldwin, #101222276

2024/04/12

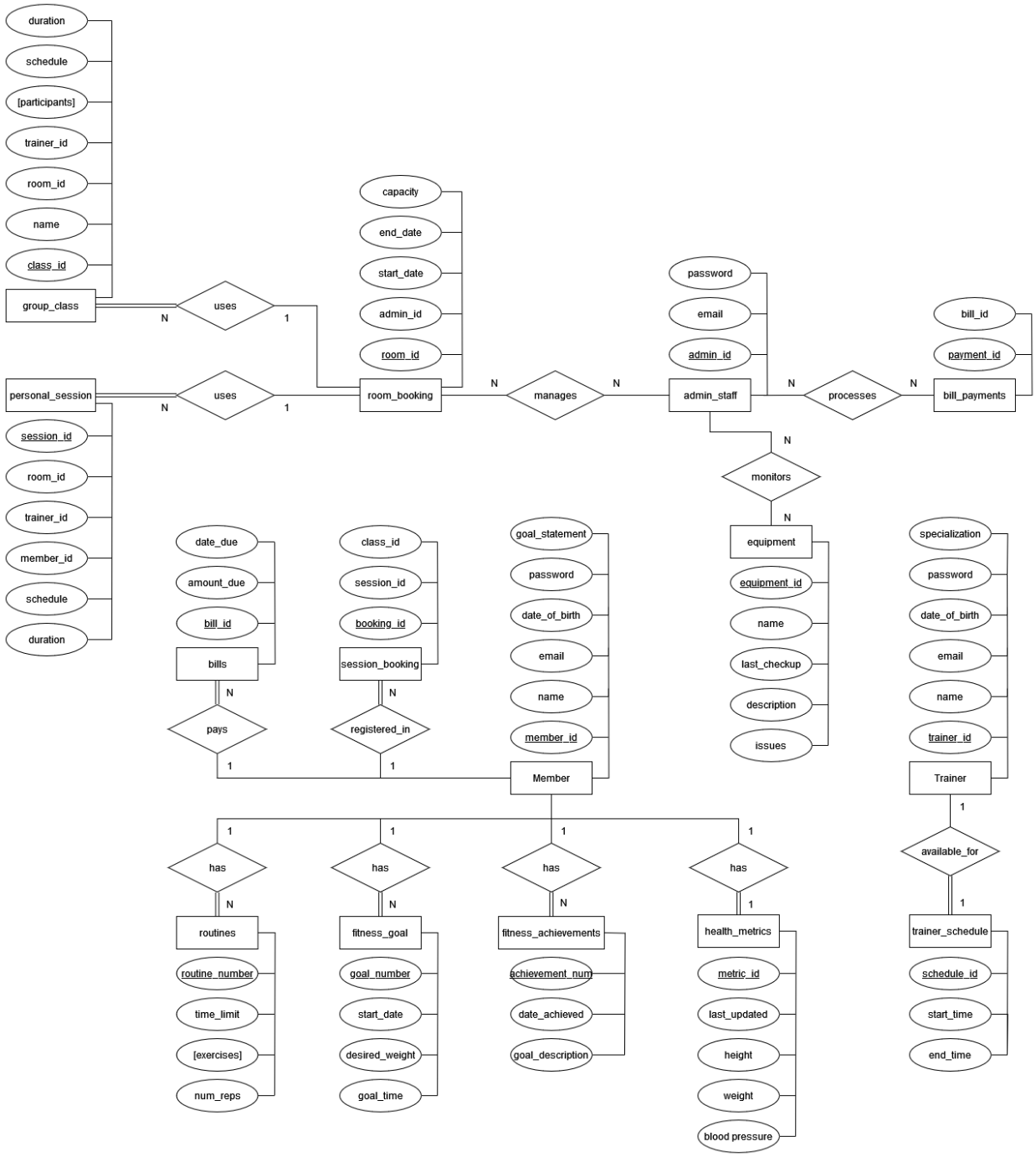
Table of Contents

1. Conceptual Design

This ER diagram models a possible implementation of the requirements outlined for a health and fitness management system. Each individual member is represented by an entity with parameters and corresponding entities representing 0..N routines, fitness goals, fitness achievements, bookings, and bills. Each member has a 1-1 cardinality relationship with a health metric entity which contains stats for the corresponding member.

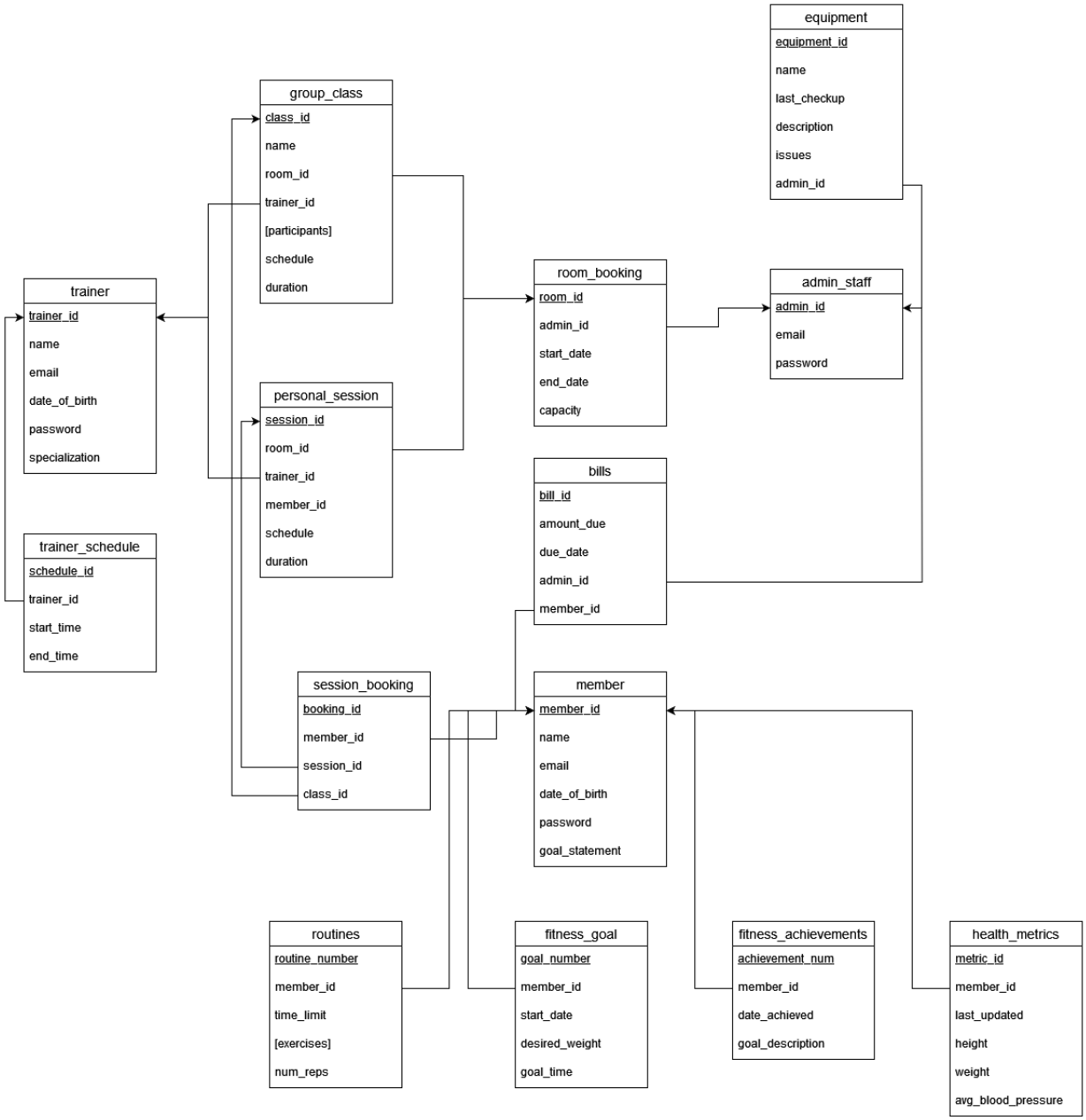
Each trainer has a 1..1 cardinality with a trainer schedule whose participation is dependent on the trainer entity, which is utilized to determine their availability for classes and personal sessions. Administrative staff are represented by an admin staff entity, which possesses N-N relations with both bills and equipment entities, as multiple admin members may regulate multiple bills and equipment in the database. Admin staff additionally have an N-N “manages” relationship with room bookings, which are used by a 1-N relationship for both personal session and group class entities. Because these entities could not exist without a room booking, their participation types are both dependent on the “uses” relationship with the room booking entity.

This conceptual model will be sufficient for implementing all necessary functions for the health and fitness system and is aligned with the requirements outlined in the problem statement. A backup diagram can be found within the GitHub repository (Section 7) under the “diagrams” folder labeled “COMP3005_Project_ER_Diagram”.


duration
schedule
[participants]
trainer_id
room_id
name
class_id
group_class
uses
capacity
end_date
start_date
admin_id
room_id
password
email
admin_id
bill_id
payment_id
personal_session
uses
room_booking
manages
admin_staff
processes
bill_payments
session_id
room_id
trainer_id
member_id
schedule
duration
monitors
date_due
amount_due
bill_id
bills
pays
class_id
session_id
booking_id
session_booking
registered_in
goal_statement
password
date_of_birth
email
name
member_id
Member
equipment
equipment_id
name
last_checkup
description
issues
specialization
password
date_of_birth
email
name
trainer_id
Trainer
available_for
trainer_schedule
schedule_id
start_time
end_time
has
routines
routine_number
time_limit
[exercises]
num_reps
has
fitness_goal
goal_number
start_date
desired_weight
goal_time
has
fitness_achievements
achievement_num
date_achieved
goal_description
has
health_metrics
metric_id
last_updated
height
weight
blood pressure

2. Reduction to Relational Schema

The conceptual ER diagram displayed in the previous section (Section 1) was utilized to construct this relational schema, mapping each entity to a table with logical primary keys and sufficient foreign keys for each function of the system outlined in the requirements. Individual relationships were not implemented as separate tables to avoid redundancy in values, and each entity contains references to the tables necessary for functionality. A backup diagram can be found within the GitHub repository (Section 7) under the "diagrams" folder labeled "COMP3005_Project_Relation_Schema_Diagram".



3. DDL File

The relational schema formed in the previous section (Section 2) was mapped to a series of SQL statements which form all tables necessary for the implementation of the health and fitness system. The .sql file necessary for executing these statements can be found within the GitHub repository (Section 7) under the "SQL" folder labeled "COMP3005_Project_DDL". This report contains a textual backup of the full .sql file which can be utilized in PostgreSQL.

```
CREATE TABLE Members (
   member_id SERIAL PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100) UNIQUE,
   date_of_birth DATE,
            password VARCHAR(20),
            goal_statement TEXT
);

CREATE TABLE Routines (
            routine_number SERIAL PRIMARY KEY,
            member_id INT,
            time_limit TIME,
            exercise_list VARCHAR(255),
            num_reps INT,
            FOREIGN KEY (member_id) REFERENCES Members(member_id)
);

CREATE TABLE Fitness_Goals (
            goal_number SERIAL PRIMARY KEY,
            member_id INT,
            start_date DATE,
            desired_weight DECIMAL,
            goal_time DATE,
            FOREIGN KEY (member_id) REFERENCES Members(member_id)
);

CREATE TABLE Fitness_Achievements (
            achievement_num SERIAL PRIMARY KEY,
            member_id INT,
            date_achieved DATE,
            goal_description VARCHAR(255),
            FOREIGN KEY (member_id) REFERENCES Members(member_id)
);

CREATE TABLE Health_Metrics (
            metric_id SERIAL PRIMARY KEY,
            member_id INT,
            last_updated DATE,
            height DECIMAL,
```

```
        weight DECIMAL,
        avg_blood_pressure DECIMAL,
        FOREIGN KEY (member_id) REFERENCES Members(member_id)
);

CREATE TABLE Trainers (
    trainer_id SERIAL PRIMARY KEY,
    name VARCHAR(100),
        email VARCHAR(100) UNIQUE,
        date_of_birth DATE,
        password VARCHAR(20),
    specialization VARCHAR(100)
);

CREATE TABLE Trainer_Schedules (
        schedule_id SERIAL PRIMARY KEY,
        trainer_id INT,
        start_time TIMESTAMP,
        end_time TIMESTAMP,
        FOREIGN KEY (trainer_id) REFERENCES Trainers(trainer_id)
);

CREATE TABLE Admin_Staff (
        admin_id SERIAL PRIMARY KEY,
        email VARCHAR(100) UNIQUE,
        password VARCHAR(20)
);

CREATE TABLE Equipment (
        equipment_id SERIAL PRIMARY KEY,
        name VARCHAR(100),
        last_checkup DATE,
        description VARCHAR(255),
        issues VARCHAR(255),
        admin_id INT,
        FOREIGN KEY (admin_id) REFERENCES Admin_Staff(admin_id)
);

CREATE TABLE Room_Bookings (
    room_id SERIAL PRIMARY KEY,
    admin_id INT,
        start_date DATE,
        end_date DATE,
    capacity INT,
        FOREIGN KEY (admin_id) REFERENCES Admin_Staff(admin_id)
);

-- Revised Classes table
CREATE TABLE Classes (
    class_id SERIAL PRIMARY KEY,
    name VARCHAR(100),
```

```
    room_id INT,
    trainer_id INT,
            participants INT[], -- Utilizes array of integer values representing member_ids
    schedule TIMESTAMP,
            duration TIME,
    FOREIGN KEY (room_id) REFERENCES Room_Bookings(room_id),
    FOREIGN KEY (trainer_id) REFERENCES Trainers(trainer_id)
);

CREATE TABLE Sessions (
    session_id SERIAL PRIMARY KEY,
            room_id INT,
    member_id INT,
    trainer_id INT,
            schedule TIMESTAMP,
            duration TIME,
            FOREIGN KEY (room_id) REFERENCES Room_Bookings(room_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id),
    FOREIGN KEY (trainer_id) REFERENCES Trainers(trainer_id)
);

-- Revised Bookings table
CREATE TABLE Bookings (
    booking_id SERIAL PRIMARY KEY,
    member_id INT,
    class_id INT NULL, -- Class ID null if private session, otherwise session ID null
            session_id INT NULL,
    FOREIGN KEY (member_id) REFERENCES Members(member_id),
    FOREIGN KEY (class_id) REFERENCES Classes(class_id),
    FOREIGN KEY (session_id) REFERENCES Sessions(session_id)
);
```

4. DML File

A dml.sql file was created to populate tables in the database with information necessary to demo the functions of the system. Although it is possible to utilize the database entirely without this file because members/trainers can register through the GUI, this file aids in demonstrating each requirement of the system with previously existing members, trainers, admins, courses/sessions, and all their corresponding information. The .sql file necessary for executing these statements can be found within the GitHub repository (Section 7) under the “SQL” folder labeled “COMP3005_Project_DML”. This report contains a textual backup of the full .sql file which can be utilized in PostgreSQL once the DDL file has been executed.

```
-- Populate Members Table
INSERT INTO Members (name, email, date_of_birth, password, goal_statement)
VALUES
('John Doe', 'test@email.com', '2001-05-12', 'abcde1', 'I want to get fit!'),
('Mary Jane', 'temporary@outlook.com', '1994-02-27', '12345', 'Trying to bulk up!'),
('Tom Jeremy', 'another@gmail.com', '2000-11-15', 'Password', 'Wanting to lose weight.'),
('Jimmy Johnson', 'jimmymail@email.com', '1972-08-02', 'GymPass', 'Keep in shape.');

-- Populate Health_Metrics Table
INSERT INTO Health_Metrics (member_id, last_updated, height, weight, avg_blood_pressure)
VALUES
(1, CURRENT_DATE, 72, 175, '140/80'),
(2, '2023-11-19', 59, 125, '110/70'),
(3, '2024-01-16', 67, 210, '160/80'),
(4, '2017-03-09', 63, 155, '110/70');

-- Populate Routines Table
INSERT INTO Routines (member_id, time_limit, exercise_list, num_reps)
VALUES
(1, '003000', 'Pullups, Pushups, Squats', 20),
(1, '001500', 'Planks, Lunges, Deadlifts, Burpees', 10),
(2, '004500', 'Deadlifts, Pushups, Lunges, Squats', 15),
(2, '010000', 'Burpees, Planks, Lunges', 10),
(2, '003000', 'Squats, Lunges, Situps, Jumping-Jacks', 20),
(3, '002000', 'Deadlifts, Pushups, Dumbbell-press', 15),
(4, '003000', 'Pushups, Pullups, Situps', 25),
(4, '001000', 'Kettlebell-lifts, Bicep-curls, Deadlifts', 10);

-- Populate Fitness_Goals Table
INSERT INTO Fitness_Goals (member_id, start_date, desired_weight, goal_time)
VALUES
(1, CURRENT_DATE, 160, '2024-06-01'),
(2, '2024-03-15', 130, '2024-05-01'),
(3, '2024-01-20', 180, '2024-09-15'),
(4, '2023-05-11', 150, '2024-05-30');

-- Populate Fitness_Achievements Table
```

```
INSERT INTO Fitness_Achievements (member_id, date_achieved, goal_description)
VALUES
(1, CURRENT_DATE, 'You joined the gym, congrats!'),
(2, '2023-11-19', 'You joined the gym, congrats!'),
(2, '2024-01-23', 'You achieved your first goal, well done!'),
(3, '2024-01-16', 'You joined the gym, congrats!'),
(4, '2017-03-09', 'You joined the gym, congrats!'),
(4, '2019-06-21', 'You achieved your first goal, well done!'),
(4, '2022-02-04', 'You achieved another goal, great job!'),
(4, '2023-05-09', 'You achieved your goal, nice work!');

-- Populate Trainers Table
INSERT INTO Trainers (name, email, date_of_birth, password, specialization)
VALUES
('Dorothy', 'dorothytrainer@email.com', '1992-04-29', 'Apples21', 'Cardio'),
('James', 'jameswork@gmail.com', '1999-02-05', '12345', 'Arms'),
('Amir', 'amirtrainer@outlook.com', '1985-09-10', 'Password123', 'Legs');

-- Populate Trainer_Schedules Table
INSERT INTO Trainer_Schedules (trainer_id, start_time, end_time)
VALUES
(1, '2024-04-10 12:00:00', '2024-06-09 00:00:00'),
(2, '2024-03-07 14:30:00', '2024-04-10 00:00:00'),
(3, '2023-05-21 12:00:00', '2024-09-11 00:00:00');

-- Populate Admin_Staff Table
INSERT INTO Admin_Staff (email, password)
VALUES
('admin1@outlook.com', 'password123'),
('admin2@gmail.com', 'Orange97'),
('admin3@email.com', '12354');

-- Populate Equipment Table
INSERT INTO Equipment (name, last_checkup, description, issues, admin_id)
VALUES
('Treadmill #1', '2024-02-12', 'Treadmill with 12 speeds, inclined', 'No issues', 1),
('Treadmill #2', '2024-03-23', 'Treadmill with 12 speeds, no incline', 'Issue with display', 1),
('Treadmill #3', '2024-03-10', 'Treadmill with 5 speeds, no incline', 'No issues', 1),
('Leg Press #1', '2024-03-10', 'Leg Press with 150 lbs max', 'Frayed wire', 2),
('Leg Press #2', '2024-03-23', 'Leg Press with 100 lbs max', 'No issues', 2),
('Exercise Bike #1', '2024-04-02', 'Exercise Bike with adjustable resistance', 'No issues', 3),
('Exercise Bike #2', '2024-03-23', 'Exercise Bike with adjustable resistance', 'Broken pedal', 3);

-- Populate Room_Bookings Table
INSERT INTO Room_Bookings (admin_id, start_date, end_date, capacity)
VALUES
(1, '2024-02-10', '2024-05-01', 20),
(1, '2024-04-01', '2024-04-20', 30),
(2, '2024-03-15', '2024-05-15', 10),
(2, '2024-04-05', '2024-06-05', 25),
(2, '2024-02-25', '2024-04-25', 20);
```

```
-- Populate Classes Table
INSERT INTO Classes (name, room_id, trainer_id, participants, schedule, duration)
VALUES
('Yoga', 1, 1, '{1, 2, 3}', '2024-05-09', '010000'),
('Lifting', 1, 2, '{1, 2, 4}', '2024-04-01', '003000'),
('Running', 2, 3, '{3, 4}', '2024-06-21', '013000');

-- Populate Bookings Table
INSERT INTO Bookings (member_id, class_id, session_id)
VALUES
(1, 1, NULL),
(2, 1, NULL),
(3, 1, NULL),
(1, 2, NULL),
(2, 2, NULL),
(4, 2, NULL),
(3, 3, NULL),
(4, 3, NULL);

-- Populate Bills Table
INSERT INTO Bills (amount_due, due_date, member_id, admin_id)
VALUES
(54.99, CURRENT_DATE + interval '14 days', 1, 1),  -- New membership fee
(24.99, '2024-05-23', 1, 2),
(24.99, '2024-05-23', 2, 2), -- Yoga class fees
(24.99, '2024-05-23', 3, 2),
(19.99, '2024-04-15', 1, 2),
(19.99, '2024-04-15', 2, 2), -- Lifting class fees
(19.99, '2024-04-15', 4, 2),
(29.99, '2024-07-05', 3, 3), -- Running class fees
(29.99, '2024-07-05', 4, 3);
```

5. Implementation

This project was implemented in Python 3.11 utilizing the tkinter interface to create separate desktop GUI applications for the member, trainer, and admin functions. Each GUI file utilizes a separate python file containing methods necessary to utilize the relational database implemented in PostgreSQL by the DDL and DML files in Section 3 and 4. For instance, when a member opens the application, they will be greeted with a login screen prompting them to enter their name. Once entered, the member GUI will interface with the member file which will query the database for the name of the member and return the ID of the member if they exist. The GUI will then prompt for a password before interfacing with the database through the member file to ensure that the given password is correct.

The admin and trainer GUIs function similarly, prompting the user and then utilizing backend functions through a separate file to connect with the relational database and send queries appropriate to the user input before returning relevant results. These results are displayed on a tkinter frame (dashboard) which updates with relevant labels and buttons depending on the context of the app. These labels and buttons display information retrieved by queries to the database and will update based on the user inputs and subsequent updates to the database.

6. Bonus Features

This application implements a variety of features which were not outlined in the original requirements but aid with user experience and understanding of the contents of the database. One additional feature that was implemented is the ability for a user to check all fitness goals associated with their account, as well as a menu utilized for checking all bills corresponding to that member with respective amounts due and due dates.

Additionally, functionality was added to register new trainers with their corresponding schedules and specialization so they could be added to new group fitness classes or personal training sessions. Although this was not a requirement, it aids with the functionality of the system as trainers would have had to be pre-existing inserts into the relational database.

7. GitHub Repository

A GitHub repository containing all source code necessary to run this system, as well as the .sql files required to create and populate the database, can be found at the following link:
https://github.com/MatthewKhoury/Comp3005Project/tree/main
Additionally, a clone of the project can be made utilizing the following link:
https://github.com/MatthewKhoury/Comp3005Project.git

All python source code can be found within the folder “source_code”, which contains GUIs for each application as well as corresponding backend functions. Diagrams representing the conceptual design of the system can be found within the folder “diagrams”, and .sql files required to construct and populate a PostgreSQL database can be found in the “SQL” folder.
A README.md file is present within the repository which contains a link to an unlisted demonstration video detailing conceptual design decisions and critical code components, along with explanations explaining how to install and run the system.