BIG BOSS MAN FITNESS CLUB

Final Project Report V2

COMP 3005

Database Management Systems

Ahmed El-Roby and Abdelghny Orogat

Katherine Maclean 101284088
Adilet Ishenbov 101270126
Simon Amable 101267294

April 13, 2024

GitHub Link: https://github.com/macleankatherine/COMP_3005_Final_Py

Video Demonstration: https://www.youtube.com/watch?v=Hdhr0buNn60

Problem Statement

Design and implement an application for a Health and Fitness Club Management System. This system will serve as a comprehensive platform catering to the diverse needs of club members, trainers, and administrative staff.

2.1 Conceptual Design

Explain the conceptual design of the database: the ER-diagram for the Health and Fitness Club, and assumptions made regarding cardinalities and participation types. Ensure that assumptions align with the problem statement in Section 1.

Entities their attributes, cardinality, design choices, assumptions

Members: Represents individuals who are part of the Big Boss Man health and fitness club.

- **Attributes**: member_id, first_name, last_name, email, password, phone_number, weight, height, bodyfat_percent
- **Design**: Responsible for updating member information, scheduling/cancelling group/personal training, viewing goals, achievements, and registered training sessions.

Assumptions:

- Each member must have a unique email address for account registration.
- All members may not have values for weight, height, and bodyfat_percent initially, hence they are nullable.

Exercise Routines: Represents the customized exercise routines created for members by trainers.

- **Attributes**: routine_id, member_id (foreign key referencing Members), routine_name, routine_description
- **Cardinality**: Members can have many different personalized exercise routines this was done so that you can create a balanced workout plan.
  ex: a member could have an exercise routine for Leg Day, Upper Body, and Mobility training.
  Design: Can only be created by trainers.

Fitness goals: Represents the goals that members aim to achieve through their fitness journey.

- **Attributes:** goal_id, member_id (foreign key referencing Members), goal_name, goal_description
- **Cardinality**: Members must input Fitness Goals and they may have multiple. This gives a lot of flexibility to personalize training sessions and exercise routines to achieve short and long-term goals.
  Ex: a member could have goals like running 5km, bench pressing 65 lbs or just coming to the gym every week

  Design: Can only be created by and managed by members.

Fitness Achievements: Represents the tracking and management of member progress and accomplishments.

- **Attributes:** achievement_id, , member_id (foreign key referencing Members), achievement_name, achievement_description
- **Cardinality**: Members must input an achievement, and they may have multiple Fitness achievements so that they can track progress on whatever they're working to achieve.
  Ex: a member could have achievements like "Bench Lift: 65lbs for 3 reps and 3 sets", "Ran 5km: With a 6:30 pace", "Lost weight: Lost 10lbs"

  Design: Can only be created by and managed by members.

Group Training Classes: Represents the group fitness classes organized for multiple members.

Attributes: class_id, trainer_id (foreign key referencing Trainers), name, booking_id (foreign key referencing Room_Bookings), details

- **Cardinality**: Members can register in multiple Group fitness classes so that they can participate in a variety of forms of exercise.
  Ex: a member could be registered in yoga, HIT class, and Karate

  Design: Must have a unique room booking associated with the group class, can only be created by administrators

Group Training Class Members: Helper table to establish the many-to-many relationship between group training classes and members.

- **Attributes:** class_id (foreign key referencing Group_training_classes), member_id (foreign key referencing Members)

- **Cardinality**: Group classes can have multiple members enrolled.

Ex: a group class can have a member list of Emma, Joe, and Tim associated with it

Personal Training Classes: Represents the personalized training sessions conducted for individual members by trainers.

Attributes: class_id, trainer_id (foreign key referencing Trainers), member_id (foreign key referencing Members), booking_id (foreign key referencing Room_Bookings), details

- **Cardinality**: Members can register for multiple personal training sessions with different trainers depending on trainer availability and room availability.
  Ex: a member can have personal training with Brad to work on Leg strength and a session with Ally to work on mobility

  Design: Must have a unique room booking associated with the personal training session, which can be created by members and administrators.

Room Bookings: Represents the bookings made for using rooms for various activities.

Attributes: booking_id, room_id (foreign key referencing Rooms), day_of_week, start_time, end_time, recurrence

- **Cardinality**: A group class has a unique room booking associated with it so each group class must book an empty room to run the session in.
  Ex: a group class is booked into studio space 1.

Assumption: each room can only have one activity (Group or personal training session) happening at once in it.

Rooms: Represents the physical spaces available for conducting fitness classes and training sessions.

- **Attributes:** room_id, room_name, capacity
- **Cardinality**: A room can have many room bookings to it as long as the times don't overlap, this makes sure there are no errors with double booking a space.
  Ex: the studio space 1 has the bookings: Yoga (8am-10am), Karate (11am-12:20pm)

Trainers: Represents fitness trainers who provide guidance and training to club members.

- **Attributes:** trainer_id, first_name, last_name, password, phone_number
- **Cardinality**: Trainers must conduct some sort of personal training classes and they also have the option to conduct group lessons. This is because some trainers may not be

qualified for group sessions, but since they are a trainer, they need to have some sort of personal sessions.
Ex: Trainer Amy runs groups yoga, mobility class and does personal training with Joe

- **Design**: Can view all classes scheduled with the trainer, view a member profile, create custom exercise routines for members, update availability and update trainer information.

Assumptions:

- Trainers are solely responsible for creating custom exercise routines for members and updating their availability.
  Ex: Leg Day: 3*5 Barbell squats 100lbs, 2*8 Leg press 180lbs, 3*12 calf raises 400lbs

Trainer Availability: Represents the availability schedule of trainers.

- **Attributes:** availability_id, trainer_id (foreign key referencing Trainers), day_of_week, start_time, end_time
- **Cardinality**: Trainers can have multiple availabilities which allows them to customize the days and times they are available. Trainers must have availabilities, even if they are "00" values indicating they are unavailable on that day.
  Ex: Joe is available Monday 8am-9pm, Tuesday 2pm-5pm, Wednesday 8am-11am 6pm-9pm, Thursday 0am-0pm (unavailable), etc

  Design: Upon an insertion of a trainer to the database, instantly every trainer has 7 time slots for availabilities corresponding from Monday-Sunday, it is up to the trainer to then update each day on their availability. A day with start and end times of "00:00:00" indicates the trainer is unavailable on this day

Administrators: Represents administrative staff responsible for managing the club's operations.

- **Attributes**: admin_id, first_name, last_name, phone_number
  Cardinality: Administrators can view, edit, add, and delete data from various tables. One root Administrators is created in the DML with id = 1 and password = password, the root admin is responsible for creating for administrators.

  Design: Responsible for creating/deleting group classes, viewing, creating and deleting trainers, initially loaded in a root administrator that can create and delete other administrators. Can view, create and remove both maintenance requests and rooms. Handles all the billing functionality such as viewing Due or Completed bills and Deleting or editing bills. Billing is automatically created

based on personal training session length when registering for a personal training class.

- Ex. ROOT ADMIN adds new admin bruce lee, or admin edits Billing with ID 1

Assumptions:

- Administrators are solely responsible for room booking management, equipment maintenance management, class schedule management, Billing management, Administrator account management, Trainer account management.

Equipment: Represents fitness equipment available at the club.

Attributes: equipment_id, equipment_name, equipment_description, room_id (foreign key referencing Room)

- **Cardinality**: Equipment must be stored in a room, but not every room is designated for equipment
- Ex. The Bruce sprint treadmill is in room 1, The Black bicycle is in room 1 .

Assumptions:

- Each Equipment represents a unique item of equipment with a name and desc, also associated with a room, participates in the equipment maintenance table

Equipment Maintenance: Represents maintenance requests for fitness equipment.

Attributes: request_id, request_name, request_details, equipment_name, equipment_id (foreign key referencing Equipment), request_date

- **Cardinality**: Administrators can create many maintenance requests.
- Ex. Equipment maintenance ticket was created to service equipment ID : 1 , Treadmill will need new grease on 1992-12-12

Assumptions:

- Each equipment maintenance request is connected to a piece of equipment, equipment maintenance requests only contain the equipment in question along with a name, details, and a date for the request.

Billing: Represents the billing process for training sessions.

Attributes: billing_id, status, amount, billing_class_is, member_id

Cardinality: Administrators can create billings for members or personal training sessions.
- Ex. Bill # for $30 for ZUUMA personal training class is due Adik

Assumptions:

- Each bill is connected to a member or personal training session. Bills can be Due or Completed based on their status. If we integrated with a payment processor, we would redirect the user to a Payment API like stripe after a bill is generated. Once the bill is paid we would update out bill status to True for completion.

2.2 ER Diagram

- Look at GitHub in diagrams for a better image.

2.3 DDL File

```
CREATE TABLE IF NOT EXISTS Members(
    member_id SERIAL PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    phone_number VARCHAR(255) NOT NULL,
    weight VARCHAR(255),
    height VARCHAR(255),
    bodyfat_percent VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS fitness_goals(
    goal_id SERIAL PRIMARY KEY,
    member_id INT REFERENCES Members(member_id),
    goal_name VARCHAR(255) NOT NULL,
    goal_description VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS fitness_achievements(
    achievement_id SERIAL PRIMARY KEY,
    member_id INT REFERENCES Members(member_id),
    achievement_name VARCHAR(255) NOT NULL,
    achievement_description VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS exercise_routines(
    routine_id SERIAL PRIMARY KEY,
    member_id INT REFERENCES Members(member_id),
    routine_name VARCHAR(255) NOT NULL,
    routine_description VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS Trainers(
    trainer_id SERIAL PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone_number VARCHAR(255) NOT NULL
);
```

```sql
CREATE TABLE IF NOT EXISTS trainer_availability (
    availability_id SERIAL PRIMARY KEY,
    trainer_id INT REFERENCES Trainers(trainer_id),
    day_of_week VARCHAR(10) NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL
);

CREATE TABLE IF NOT EXISTS Administrators(
    admin_id SERIAL PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    phone_number VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS Rooms (
    room_id SERIAL PRIMARY KEY,
    room_name VARCHAR(255) UNIQUE NOT NULL,
    capacity INT NOT NULL
);

CREATE TABLE IF NOT EXISTS room_bookings (
    booking_id SERIAL PRIMARY KEY,
    room_id INT REFERENCES Rooms(room_id),
    day_of_week VARCHAR(10) NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    recurrence VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS personal_training_classes(
    class_id SERIAL PRIMARY KEY,
    trainer_id INT REFERENCES Trainers(trainer_id),
    member_id INT REFERENCES Members(member_id),
    booking_id INT UNIQUE REFERENCES room_bookings(booking_id),
    details VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS group_training_classes(
    class_id SERIAL PRIMARY KEY,
    trainer_id INT REFERENCES Trainers(trainer_id),
    name VARCHAR(255) NOT NULL,
    booking_id INT UNIQUE REFERENCES room_bookings(booking_id),
```

```
    details VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS group_training_class_members (
    class_id INT REFERENCES group_training_classes(class_id),
    member_id INT REFERENCES Members(member_id),
    PRIMARY KEY (class_id, member_id)
);

CREATE TABLE IF NOT EXISTS Equipment(
    equipment_id SERIAL PRIMARY KEY,
    equipment_name VARCHAR(255) NOT NULL,
    equipment_description VARCHAR(255) NOT NULL,
    room_id INT REFERENCES Rooms(room_id)
);

CREATE TABLE IF NOT EXISTS equipment_maintentence(
    request_id SERIAL PRIMARY KEY,
    request_name VARCHAR(255) NOT NULL,
    request_details VARCHAR(255) NOT NULL,
    equipment_id INT REFERENCES Equipment(equipment_id),
    request_date DATE NOT NULL
);

CREATE TABLE IF NOT EXISTS Billing(
    billing_id SERIAL PRIMARY KEY,
    status BOOLEAN NOT NULL,
    amount FLOAT NOT NULL,
    billing_class_id INT REFERENCES personal_training_classes(class_id),
    member_id INT REFERENCES Members(member_id)
);
```

2.4 DML File

```
INSERT INTO Rooms (room_name, capacity)
VALUES
    ('Studio 1', 20),
    ('East studio ', 15),
    ('Main gym', 50);

INSERT INTO Members (first_name, last_name, email, password, phone_number,
weight, height, bodyfat_percent)
VALUES
    ('Emily', 'Jones', 'emily@example.com', 'password123', '123-456-7890', '150',
'65', '20'),
```

```
    ('Michael', 'Brown', 'michael@example.com', 'securepass', '987-654-3210',
'180', '72', '15'),
    ('Sarah', 'Davis', 'sarah@example.com', 'mypassword', '555-555-5555', '130',
'60', '25');

INSERT INTO fitness_goals (member_id, goal_name, goal_description) VALUES
    (1, 'Weight Loss', 'To lose 10 pounds by the end of the month through a
combination of diet and exercise.'),
    (1, 'Muscle Gain', 'To increase muscle mass and strength by following a
structured weightlifting program.'),
    (2, 'Flexibility Improvement', 'To improve flexibility and mobility through
regular stretching and yoga sessions.'),
    (3, 'Cardio Endurance', 'To improve cardiovascular endurance by running 5
kilometers without stopping.'),
    (2, 'Body Composition Change', 'To decrease body fat percentage and increase
lean muscle mass.'),
    (3, 'Overall Health and Wellness', 'To adopt a healthier lifestyle by
incorporating regular exercise and balanced nutrition.');

INSERT INTO fitness_achievements (member_id, achievement_name,
achievement_description) VALUES
    (1, '5K Run', 'Completed a 5-kilometer run in under 30 minutes.'),
    (1, 'Weight Loss Milestone', 'Lost 10 pounds and reached the first weight
loss milestone.'),
    (2, 'Muscle Gain Progress', 'Increased muscle mass and strength by 10% over
the past three months.'),
    (3, 'Improved Flexibility', 'Achieved full splits and improved overall
flexibility by attending regular yoga classes.'),
    (2, 'Fitness Competition Win', 'Won first place in a local fitness
competition.'),
    (1, 'Consistent Gym Attendance', 'Maintained consistent gym attendance for
six months straight.');

INSERT INTO Trainers (first_name, last_name, password, phone_number)
VALUES
    ('John', 'Doe', 'password123', '123-456-7890'),
    ('Alice', 'Smith', 'securepass', '987-654-3210'),
    ('Mike', 'Johnson', 'mypassword', '555-555-5555');

INSERT INTO room_bookings (room_id, day_of_week, start_time, end_time,
recurrence)
VALUES
    (1, 'Monday', '09:00:00', '10:00:00', 'Weekly'),
    (2, 'Tuesday', '14:00:00', '15:00:00', 'Bi-weekly'),
    (3, 'Wednesday', '10:30:00', '11:30:00', 'Monthly');
```

```sql
INSERT INTO group_training_classes (trainer_id, name, booking_id, details)
VALUES
    (1, 'Yoga Class', 1, 'Beginner-friendly yoga session'),
    (2, 'HIIT Workout', 2, 'High-intensity interval training for advanced fitness
enthusiasts'),
    (3, 'Zumba Dance Party', 3, 'Fun and energetic dance workout');

INSERT INTO group_training_class_members (class_id, member_id)
VALUES
    (1, 1),
    (2, 2),
    (3, 3);

INSERT INTO trainer_availability (trainer_id, day_of_week, start_time, end_time)
VALUES
    (1, 'Monday', '08:00', '22:00'),
    (1, 'Tuesday', '08:00', '22:00'),
    (1, 'Wednesday', '08:00', '22:00'),
    (1, 'Thursday', '08:00', '22:00'),
    (1, 'Friday', '08:00', '22:00'),
    (1, 'Saturday', '08:00', '22:00'),
    (1, 'Sunday', '08:00', '22:00'),
    (2, 'Tuesday', '13:00', '17:00'),
    (3, 'Friday', '11:00', '15:00');

INSERT INTO exercise_routines (member_id, routine_name, routine_description)
VALUES
    (1, 'Morning Workout', 'Morning Workout: Cardio: 20 min running, Stretching:
10 min full-body'),
    (1, 'Strength Training', 'Strength Training: Squats: 5*5, Bench Press: 4*6,
Deadlifts: 3*8'),
    (2, 'Yoga Flow', 'Yoga Flow: Sun Salutation: 5 cycles, Downward Dog: 3*30s
hold, Warrior Pose: 3*each'),
    (3, 'High-Intensity Interval Training (HIIT)', 'HIIT: Sprint: 30s, Rest: 1
min, Repeat 5 times'),
    (3, 'Endurance Running', 'Endurance Running: Long Run: 10km at moderate
pace'),
    (2, 'Bodyweight Circuit', 'Bodyweight Circuit: Push-ups: 3*15, Squats: 4*12,
Lunges: 3*each leg'),
    (3, 'Flexibility Routine', 'Flexibility Routine: Hamstring Stretch: 3*30s,
Shoulder Stretch: 3*each side');
```

2.5 Implementation

Our application architecture is structured around a Command-Line Interface (CLI) paradigm, and programmed in python.

Main Menu:

Functionality: The main menu serves as the central hub of our application, providing options for different types of users: new users, returning members, trainers, and administrators.
User Interaction: Upon launching the application, users are presented with a menu where they can select their role or action, such as registering as a new user, logging in as a returning member, accessing trainer functionalities, or entering the admin panel.
Navigation: Based on the user's choice, the application navigates to the corresponding module to fulfill the user's needs.

Registration and Profile Management:

Functionality: These modules handle user registration, login, and profile management tasks. New users can register by providing necessary details, while returning members can log in to access their profiles.
Profile Updates: Users have the option to update their personal information, health metrics, and health goals through the profile management functionality.

Trainers Menu:

Functionality: Trainers can log in to access functionalities such as managing personal training sessions, creating exercise routines, and updating their availability.
Availability: Trainers can update their availability to reflect their schedule accurately, ensuring efficient session planning and management.

Admin Panel:

Functionality: Administrators have access to administrative functionalities, including registering new administrators and managing equipment maintenance requests.
Equipment Management: Admins can oversee equipment maintenance requests, ensuring timely maintenance to keep the facility running smoothly.
User Management: Admins can manage user accounts, including registering new administrators and handling administrative tasks.

Dashboard:

Personalized Information: The dashboard provides users with personalized information such as fitness achievements, upcoming sessions, and group class enrollments.
Visual Representation: Users can visualize their progress, goals and upcoming activities.

Scheduling:

Booking System: Users can schedule personal training sessions and enroll in group classes through dedicated menus.
Cancellation and Rescheduling: Users have the flexibility to cancel or reschedule sessions/classes as needed.

Fitness Achievements Management:

Track Progress: Users can add new fitness achievements, delete existing ones, and view their fitness achievements history.
Motivation: Tracking achievements provides users with motivation and a sense of accomplishment as they progress towards their fitness goals.

Administrative Staff Functions:

All functions in admin.py are responsible for doing their own prompting for input, error checking, and querying. The functions from admin.py are then used in the admin_menu.py file to allow easy terminal navigation and use for all the administrative functions.

Administrative Staff Functions:

Room Booking Management:

Functionality: Handles the management of rooms within the fitness club, including adding, deleting, and viewing room information.
Features: Print available rooms, Add a new room, Delete an existing room.

Equipment Maintenance Monitoring:

Functionality: Manages maintenance requests for fitness equipment to ensure proper upkeep and functionality.
Features: Print maintenance requests, Add a new maintenance request, Delete a maintenance request.

Class Schedule Management:

Functionality: Manages the scheduling of group training classes and personal training sessions.

Features: Print group training class schedule, Print personal training class schedule, Create a new group training class, Delete a group training class.

Billing:

Functionality: Handles billing and financial transactions related to room bookings and training sessions.

Features: Print billing records, Print due bills, Print completed bills, Delete a bill, Add a new bill.

- If this billing system was real after creating a bill by enrolling in a class the user should be redirected to a payment API like stripe, and then validate a members bills in the database if we need to restrict access before payment. We did not take real payments

EXTRA/BONUS: Administrator & Trainer Management + Administrator Login:

Functionality: Facilitates the management of administrators and trainers within the system.

Features:

- Create a new admin.
- Delete an existing admin.
- Create a new trainer.
- Delete an existing trainer.
- Alter admin details.
- Login as administrator.

Design:

All menus for the admin just run a while loop which redirect to admin.py functions to do the hard work or other menus based on user choice in the respective menu. Each menu operates within a loop until the user chooses to go back with “0”.

Trainer Menu:

Functionality: The Trainer Menu provides trainers with a range of functionalities to manage their tasks and interactions within the fitness club system.

Features:

- **View Class Schedule**: Allows trainers to view their class schedule, including details such as class name, timing, and location.
- **View Member Profile**: Enables trainers to search for and view the profiles of club members to better understand their training needs and goals.
- **Update Availability:** Allows trainers to update their availability, ensuring accurate scheduling of training sessions and classes.
- **Create Custom Exercise Routines:** Empowers trainers to create custom exercise routines tailored to the specific needs and goals of individual members.

- **Update Profile**: Provides trainers with the capability to update their profile information, including contact details and specialization areas.
- **Exit**: Allows trainers to exit the Trainer Menu and return to the main menu or log out of the system.