Conceptual Design:

Our design in centered around the three main components, the member, trainer, and administrator. All relationships with other tables must connect to one of these three main tables. Members have much more partial participation than trainers and administrators since there's nothing making them register for classes, track their exercises, or register for training sessions. Whereas Administrators must be involved in monitoring equipment, overseeing billing, managing room bookings, and updating the class schedule. Classes also must be organized by one or more admins, but classes do not necessarily have to have any members register for them. Trainer also must organize possible training sessions to allow members to pick and choose which times work best for them. Trainer also must decide what their daily availability looks like.

Application's Architecture

Our program is launched from the gymDbGUI class, which contains the general logic for launching the main window. The layout of the main window varies depending on which of the three user types is selected and logged in. For each user type, a variety of functions and features become available after clicking the features button. Upon clicking this button, a QDialog popup, which contains the display and information relevant to that function, opens. This QDialog popup is located in another file and is called by the main window. The logic for each QDialog popup is contained in a file shared with other relevant functions for that component. The popup class manages the input and GUI-related logic, whereas any database requests required by the popup are handled by calling the associated method in the functionImplementation class. This implementation class manages all our SQL code and database requests, also containing an execute_query() method which handles all database connections.

An example of this:

Log in as a Member.

Click 'Make Payment'.

makePaymentsPopup class is called to display in the GUI.

Click 'Pay'.

functionImplementation.makePayment() is called.

makePayment() calls execute_query() with the SQL command 'UPDATE members SET payment_status….'.

The database is updated.

A feedback message is sent.

Bonus Feature:

GUI using python QT designer.

Assumptions:

Requirement	Assumption	Representation in ER Model
Members should be able to register and manage their profiles, establish personal fitness goals (you can determine suitable fitness goals such as weight and time, and members will set the values), and input health metrics. Members should have access to a personalized dashboard that tracks exercise routines, fitness achievements, and health statistics.	Fitness goals are represented as text a user can input. Each member can have N fitness goal. A person's health metrics are their current_weight and height. Fitness achievements are also represented as text a trainer can put for a member; a member can have N achievements.	**Members Entity**: Attributes include member_id (PK), first_name, last_name, current_weight, email, phone_number, height, achievement, fitness_goal
Members can schedule, reschedule, or cancel personal training sessions with certified trainers. Additionally, they should be able to register for group fitness classes.	Members have partial participation with training_sessions because a member does not have to register for any sessions. A training session can have 1 member, but a member can register for N training sessions. Trainers have total participation with training_sessions because they are always active and waiting for a member to join them. A trainer can have N training_sessions but a training_session can only have 1 trainer. A Class can have N members, but it can also have 0 members, so it has partial participation to register. A member can register for N classes, but a member can also not register for any, so it also has partial participation in register.	**Trainers Entity:** Attributes include trainer_id (PK), first_name, last_name, phone_number **Training_Sessions Entity:** Attributes include session_id (PK), day_of_week, session_time, status. **Register Relationship:** No attributes, relationship between Member and Training_Sessions **Organize Relationship:** No attributes, relationship between Trainers and Training_Sessions **Classes Entity:** Attributes include class_id (PK), class_name, class_time **Register Relationship**: No attributes, relationship between members and classes

Trainers should have the ability to manage their schedules and view member profiles.	Each element of the availability must only have 1 trainer and each trainer has N availabilities.	**Training_Sessions Entity:** Attributes include session_id (PK), day_of_week, session_time, status. **Availability Entity:** Attributes include avail_id (PK), start_time, end_time, day_of_week **Has Relationship:** No attributes, relationship between Trainers and Availability
Administrative Staff should be equipped with features to manage room bookings, monitor fitness equipment maintenance, update class schedules, oversee billing, and process payments for membership fees, personal training sessions, and other services.	An admin must monitor equipment, an admin can monitor N equipments. A piece of equipment must be monitored and can be monitored by N admins. A member must have exactly one billing and a billing has exactly one member. An admin must oversee billings and can oversee N billings. A billing must be overseen and can be overseen by N admins. An admin must manage room_bookings and can manage N of them. A room_booking must be managed by an admin and can be manages by N admins. An admin must update a class and can update N classes. A class must be updated by an admin and can be updated by N admins.	**Admin Entity:** Attributes include admin_id (PK), first_name, last_name, phone_number **Equipment Entity:** Attributes include equipment_id (PK), equipment_name, maintenance_status **Monitor Relationship:** No attributes, relationship between admin and equipment **Billing Entity**: Attributes include billing_id (PK), amount, payment_status **Pays Relationship:** No attributes, relationship between billing and members. **Oversees Relationship:** No attributes, relationship between admin and billing. **Room_Bookings Entity:** Attributes include booking_id (PK), room_number, booking_time, purpose, status. **Manages Relationship:** No attributes, relationship between Admin and Room_Bookings

		Updates Relationship: No attributes, relationship between Admin and Classes