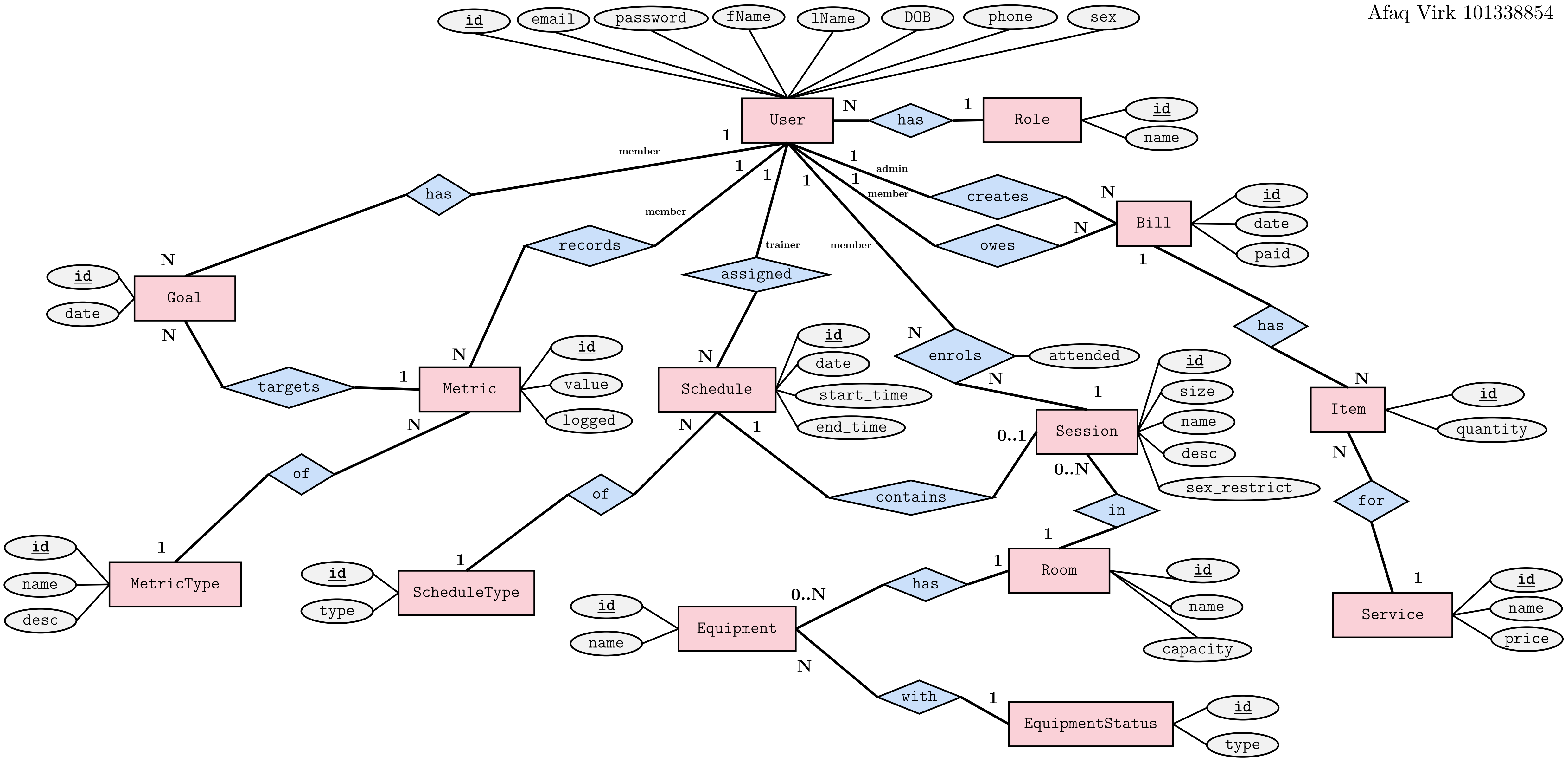
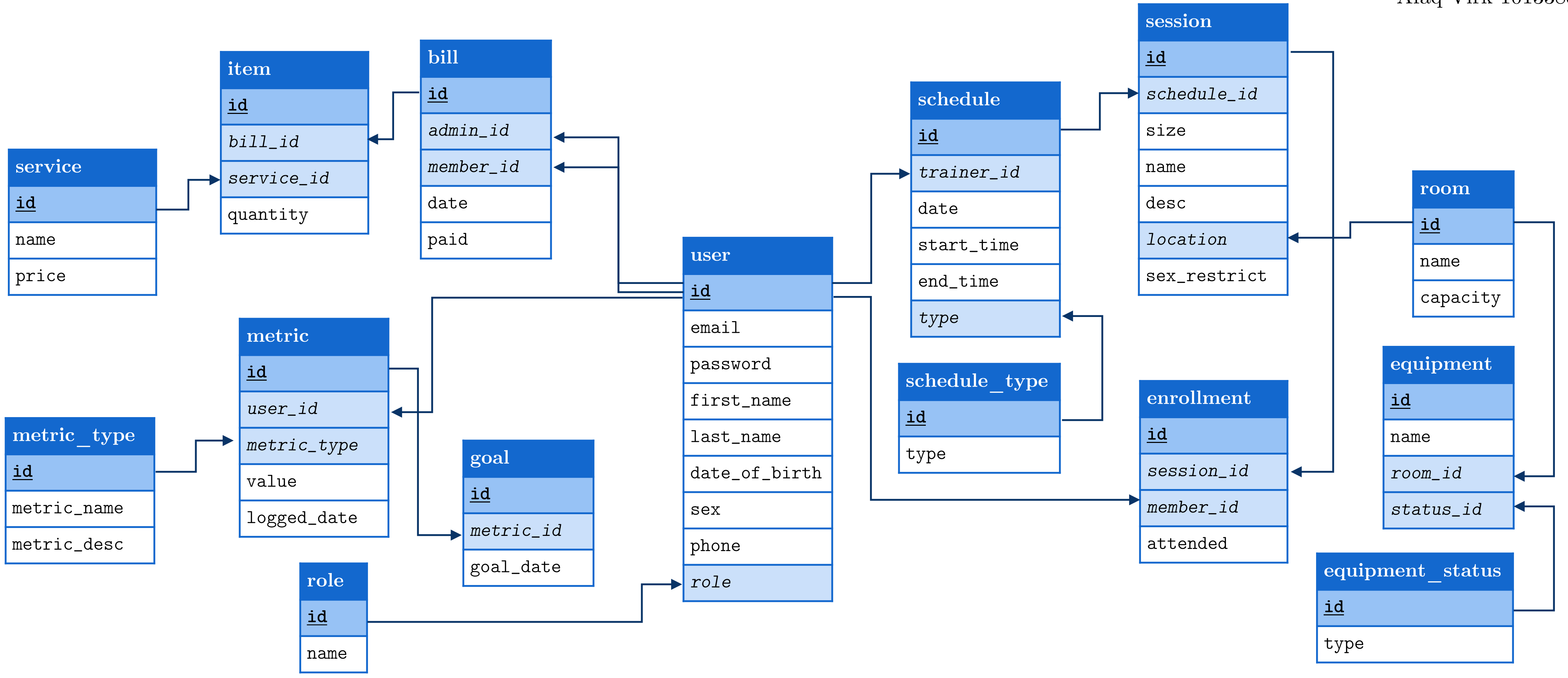
ER Model Design

COMP 3005 - Fall 2025
Database Management Systems
Final Project
Raymond Liu 101264487
Afaq Virk 101338854



Database Schema Diagram

COMP 3005 - Fall 2025
Database Management Systems
Final Project
Raymond Liu 101264487
Afaq Virk 101338854



Mappings (1)

COMP 3005 - Fall 2025
Database Management Systems
Final Project
Raymond Liu 101264487
Afaq Virk 101338854

Requirement	**Assumption**	**Representation in ER Model**
"The application will support three distinct categories of users—members, trainers, and administrative staff—each with specialized access privileges and clearly defined functional responsibilities."	All human actors share the same core attributes (name, contact information, credentials) and differ only by role. Each user holds exactly one role at a time.	Entity `User` with PK `id` and attributes `email`, `password`, `first_name`, `last_name`, `date_of_birth`, `sex`, `phone`. Entity `Role` with values *Member*, *Trainer*, *Admin*; FK `User.role` → `Role.id`.
"A member should be able to register by providing personal information such as name, date of birth, gender, and contact details, and later manage or update these details as needed."	The same personal-information structure applies to all user roles; registration and profile updates operate on a single user record.	Attributes listed above are stored once in `User`. No separate *Member* table is introduced; the distinction is represented by the `role` foreign key.
"Members will have the ability to establish and track personalized fitness goals—such as achieving a target body weight or reducing body fat percentage—and store health metrics like height, weight, heart rate [...]. These metrics should not overwrite previous entries but instead be recorded historically..."	Health metrics are modeled as individual time-stamped measurements. Goals are expressed in terms of these measurable metrics rather than free-form text. The owning user of a goal can be derived from the metric being targeted.	Entity `MetricType` (e.g., Weight, Body Fat %). Entity `Metric` with PK `id`, FK `user_id` → `User.id`, FK `metric_type` → `MetricType.id`, attributes `value`, `logged_date`. Entity `Goal` with PK `id`, FK `metric_id` → `Metric.id`, attribute `goal_date`. Redundant `user_id` is omitted from `Goal` to avoid transitive dependency.
"Members should be able to schedule, reschedule, or cancel personal training sessions [...] and [...] register for group fitness classes, subject to class capacity and schedule constraints. The system must ensure logical enforcement of business rules such as preventing overlapping bookings [...] and verifying trainer availability before confirming any reservation."	Actual classes/sessions are distinct from the availability slots defined by trainers. Sessions have a maximum capacity and may be attended by many members. Each member can enroll in a given session at most once.	Entity `Schedule` (trainer availability) with PK `id`, FK `trainer_id` → `User.id`, attributes `date`, `start_time`, `end_time`, FK `type` → `ScheduleType.id`. Entity `Session` with PK `id`, FK `schedule_id` → `Schedule.id`, attributes `name`, `size`, `desc`, `location`, `sex_restrict`. Associative entity `Enrollment` linking `User` (member) and `Session`, with PK `id`, FKs `session_id`, `member_id`, attribute `attended`, and `UNIQUE(session_id, member_id)`.

Mappings (2)

"Trainers should be able to specify their availability periods—either as recurring weekly slots or as individual time intervals—and update these schedules as needed. The system must prevent overlapping or inconsistent time slots for the same trainer..."	We store concrete time intervals; recurring availability patterns are handled at the application level by generating individual schedule rows. Overlap checks are enforced in application logic rather than as constraints in the ER model.	Entity `Schedule` as above, with each row representing one availability interval for a trainer, linked to `User` (trainer) and `ScheduleType`. No additional recurrence entity is modeled.
"Administrators will have the capability to manage room bookings to ensure that physical spaces—such as studios or training rooms—are properly allocated for classes and personal sessions, while avoiding conflicts in scheduling. They should also be able to oversee equipment management, including tracking the operational status of machines, logging maintenance issues, assigning repair tasks, and updating maintenance records once issues are resolved."	Rooms and equipment are modeled explicitly; each piece of equipment belongs to at most one room and has one current status. Detailed maintenance history and automatic room–session conflict detection are treated as application-level responsibilities in this version.	Entities `Room` (PK `id`, attributes `name`, `capacity`), `EquipmentStatus` (PK `id`, attribute `type`), and `Equipment` (PK `id`, attributes `name`, FK `room_id` → `Room.id`, FK `status_id` → `EquipmentStatus.id`). Session locations are stored as a text attribute `location`.
"The administrative component should also handle billing and payment processes, where the system can generate bills for various services such as membership subscriptions, personal training sessions, or class enrollments. Payments are to be simulated rather than processed through real financial gateways; however, the system should maintain a consistent record of invoices [...] and payment status..."	Bills may contain multiple services, and each service can appear on many bills. Only basic payment state is required (paid / unpaid); no sensitive payment details or external transaction identifiers are stored.	Entity `Service` for billable offerings. Entity `Bill` with PK `id`, FKs `admin_id` and `member_id` referencing `User`, attributes `date`, `paid`. Associative entity `Item` with PK `id`, FKs `bill_id` and `service_id`, and attribute `quantity` representing line items on a bill.