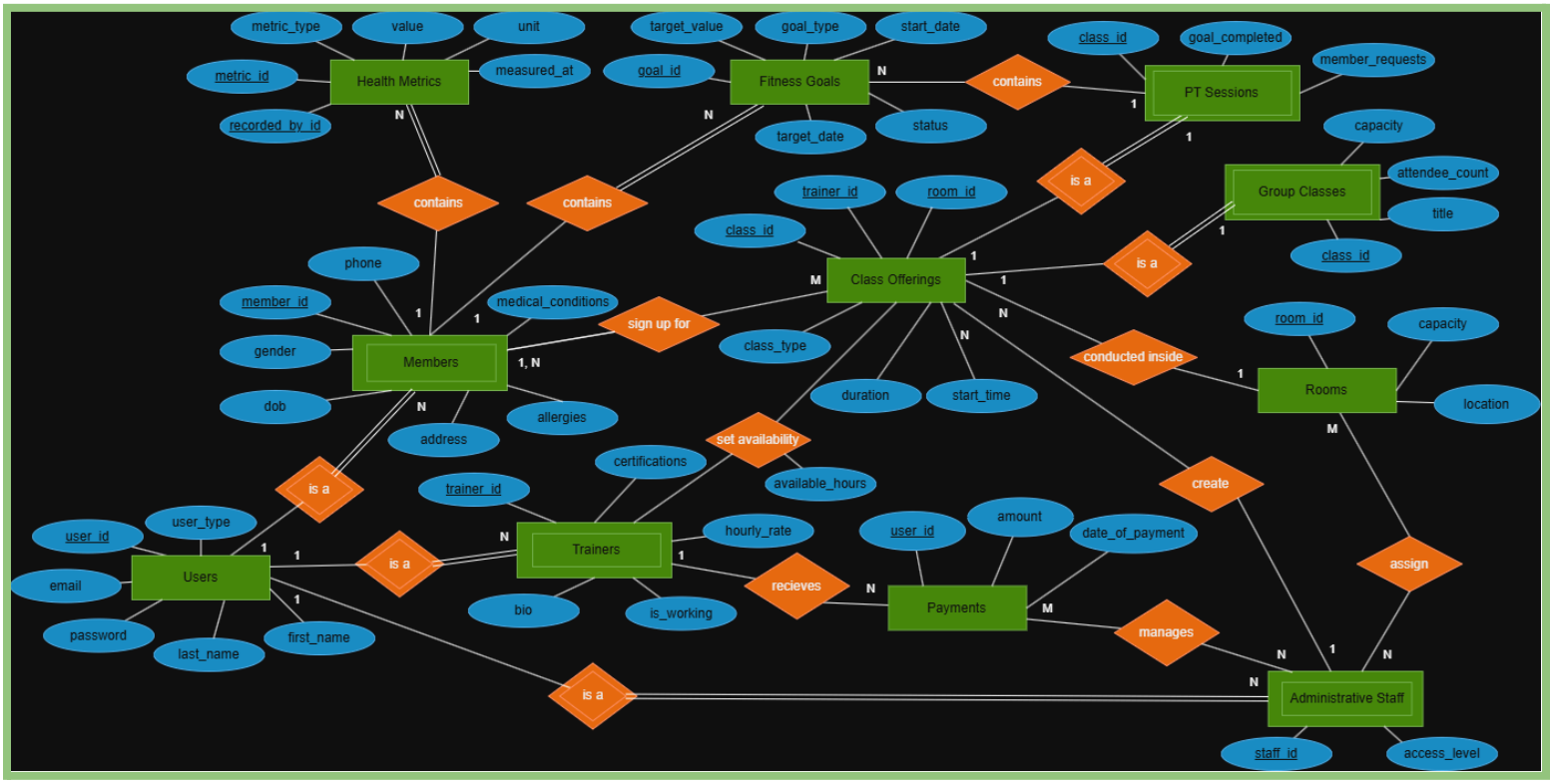
COMP 3005 - Final Project

Group 99
Jessica Hill - 101280293
Stanny Huang - 101272645
Jansen Khoe - 101260040

ER diagram



ORM Explanation

For our database, we utilized Prisma, an extension of Node.js that serves as an ORM for our PostgreSQL database. To generate our ORM, we have created a `schema.prisma` file stored in the project's ./prisma folder. Within this file we have created models to map all the major entities and their relations which includes all users and their derived types (member, trainer, system admin), all classes, all rooms, and all metrics. Furthermore, the ORM replaced the need for raw SQL queries within our application which leads to cleaner and easier to maintain code. these are all neatly stored in the app/actions/auth.ts file which defines all of our available server side actions.

An example would be:

```
export async function updateMemberGoal(
  memberId: number,
  metricType: MetricType,
  goalValue: number
) {
  try {
    const goalField = `${metricType}Goal` as 'heartbeatGoal' |
'caloriesGoal' | 'stepsGoal';

    await prisma.member.update({
      where: { id: memberId },
      data: { [goalField]: goalValue }
    });

    return { success: true };
  } catch (error) {
    console.error('Error updating goal:', error);
    return { success: false, error: 'Failed to update goal' };
  }
}
```

This function is used to update a member's goal based on their member id, the type of metric they are updating and the new value of that metric.

Link to Video:

Link to Github Repo:

https://github.com/JanKhoe/COMP3005-Final-Project/blob/main/README.md

NOTE: The Github README contains setup information for running the project yourself.