COMP 3005 F25 Final Project Report

The ORM used in this project was Prisma, a PostgreSQL database that supports many frameworks and stacks (including Next.js which was used in this project). However, it particularly works best under Node.js/TypeScript environments, making it the best choice for this tech stack.

Prisma was used to map the conceptual entities from the ERD (Member, Trainer, Room, Session, Booking, and HealthMetric) - into concrete database tables in PostgreSQL. Prisma's schema file defines each entity's attributes, primary keys, unique constraints, and the relationships between them, and Prisma automatically generates a fully typed client that the application uses for database interaction. This allowed the application logic to focus on business rules instead of SQL, while still preserving referential integrity through relations such as 1:N (e.g., a Trainer has many Sessions). Indexes and unique constraints defined in the schema (e.g., @@unique([memberId, sessionId])) were used to enforce logical rules like preventing duplicate bookings.

Throughout the project, Prisma was used for all major CRUD operations - from creating new members and trainers to booking sessions and recording health metrics. The ORM's client simplified relational queries, such as fetching sessions with their trainer and room information in a single call. For example:

```
// Creating a new booking (in actions.ts, registerSessions() function)
await prisma.booking.create({
  data: {
    memberId,
```

```
    sessionId,
  },
});

// Fetching all sessions with trainer and room details
//Essentially joining the trainer and room tables with sessions together
const sessions = await prisma.session.findMany({
  include: {
    trainer: true,
    room: true,
  },
});
```