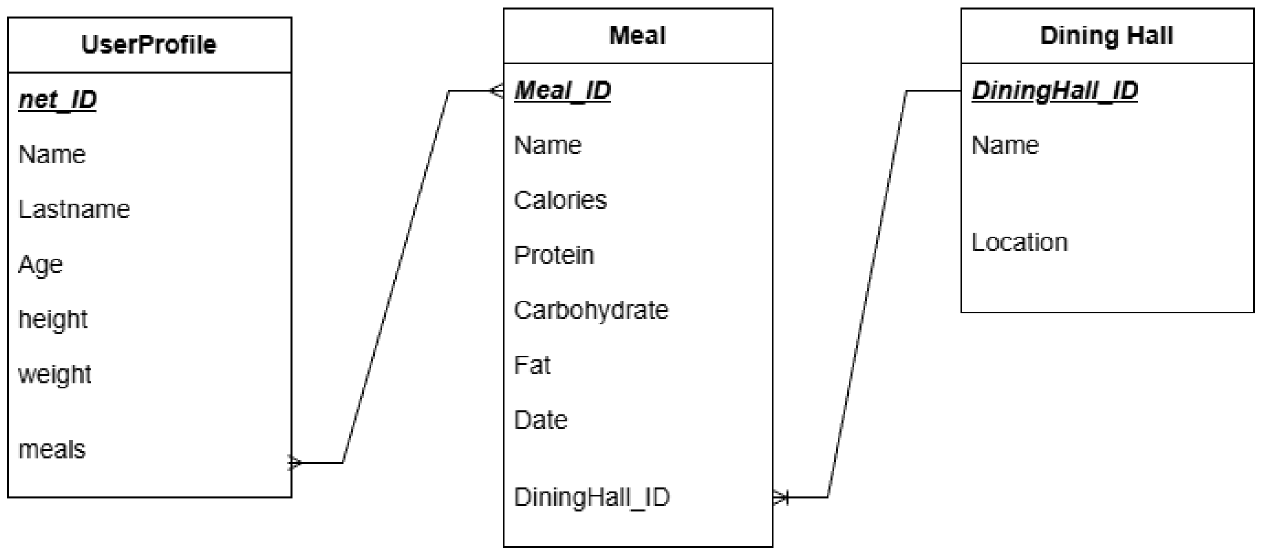

Note: In models.py, primary keys for Meal and DiningHall are not defined explicitly because Django generates an automatic id field for them. However, netID is explicitly defined as the primary key for UserProfile. For the purposes of our ER diagram, net_ID, Meal_ID, and DiningHall_ID represent these primary keys to maintain a normalized PostgreSQL structure.

Dining Hall: This represents dining halls in university. It is a **One-to-Many** relationship with the Meal model, ensuring that every menu item is accurately categorized by its location. This model exists to organize menu items by their specific location, allowing students to filter and compare nutritional options across dining halls university

Meal: Represents an individual food item or dish served within a dining hall. This model is the core of the system's analytics; it stores AI-enriched macronutrient data (protein, carbs, fats) and calorie counts to eliminate the "information gap" students face when making dietary choices. It also includes a date field for tracking each student's macronutrient intake per day.

UserProfile: Represents a student user of the SmartEats application. Ist exists to store personalized health data (height, weight, age) and track consumed meals, enabling the application to calculate cumulative daily intake and provide progress tracking toward health goals. ManyToManyField connects UserProfile to Meal. This choice allows students to log multiple meals over time while keeping the meal data reusable for the entire student body.

Testing

We inserted **5–10 records** into the models to simulate a real environment. This included creating multiple DiningHall entries (representing UIUC dining halls) and several Meal entries with specific macronutrient values for protein, carbohydrates, and fats.

Foreign Key Relationship Test

We verified the One-to-Many relationship between dining halls and meals. We linked multiple meals to a single dining hall (e.g., assigning “Grilled Chicken” and “Brown Rice” to the Ikenberry Dining Hall). This confirmed that each dining hall can have many meals, and each meal belongs to only one specific dining hall.

Uniqueness Constraint Validation

To ensure there are no duplicates, we tested the UniqueConstraint on the Meal and DiningHall models. We attempted to insert a second meal with the exact same name for the same dining hall and also tried to insert a dining hall with the same name. As a result, the database blocked the duplicate entries and threw an error. This proves that our database schema prevents duplicate data.

on_delete Behavior Validation

We tested the CASCADE setting on our foreign keys to ensure data consistency. We deleted a DiningHall record, and Django automatically removed all Meal records associated with that hall. This confirms that the system maintains a clean state and prevents “orphaned” menu items that no longer exist on campus.

Many-to-Many Relationship Test

We validated the “Meal Planning Interface” and “Basic Nutritional Tracking” features. We created a UserProfile using a student netID and added multiple Meal objects to that user’s profile. The test confirmed that a single student can have multiple meals without affecting the availability of those meals for other students.