

# UNI EATS Requirement Analysis

Group 15 Soham Hisabia, Ammaar Khan, Adam Yaagoub, Ivan Torres, Nathen Priyonggo

## Overview

The goal of UNI EATS is to build a simplified restaurant locator and information system, specifically curated for college students by focusing on affordable options in close proximity to major college campuses in the Chicago area. The application's primary function is to allow users to search and filter restaurants efficiently, with a unique emphasis on Campus Proximity.

## Data Requirements

### Restaurants

The central entity for the application. We record important identification information for users to identify a Restaurant.

- For each Restaurant we record its Name, Price Range (<\$5, \$5-\$10, \$10-\$20), estimated Walk Distance (in miles), a Boolean/flag for Take Out and Food Delivery Service availability, and a Thumbnail Link (URL).
- Restaurants are uniquely identified by a system-generated **Restaurant ID**.

### Campus

Major Campus information.

- For each Campus we record Name and Address.
- Campus' are uniquely identified by a system-generated **Campus ID**.
- Indirectly links to **Restaurants** entries via **Campus Proximity** associative table

### Campus Proximity

Junction table modeling relative proximity between Restaurant and Campus.

- Each entry store a **Distance** attribute, measured in miles
- **Restaurant ID** and **Campus ID** maps proximity relations.

### Cuisines

Defines type of cuisine.

- For each Cuisine, we record the Cuisine Name (e.g., Japanese, Korean, Mexican).
- Cuisines are uniquely identified by a system-generated **Cuisine ID**.

### User Ratings

Logging mechanism for unique user feedback, mapped to restaurant.

- For each User Rating we store the Rating Value (e.g., 1-5), an optional Name, and a Review Text.
- User Ratings are uniquely identified by a system-generated **Rating ID**.

## Application Requirements (User Actions)

The application will support the following actions for Public Users viewing the single search/results page.

### Searching and Filtering

- Search by Text: The application must allow the user to input text into a search bar to filter restaurants by matching characters in the Name or Cuisine Type.
- Filter by Campus: The application must allow the user to select one or more Campuses (e.g., from a dropdown/checkbox list) to display only restaurants associated with that campus's proximity.
- Filter by Cuisine: The application must allow the user to filter the results by selecting a specific Cuisine Type.
- Filter by Service: The application must allow the user to filter results based on the availability of Take Out and/or Food Delivery Service.
- View Results: The application must display the filtered list of restaurants, showing all key attributes (Name, Rating, Walk Distance, etc.) in either a row or card format.

### BONUS: User Interaction (If Implemented)

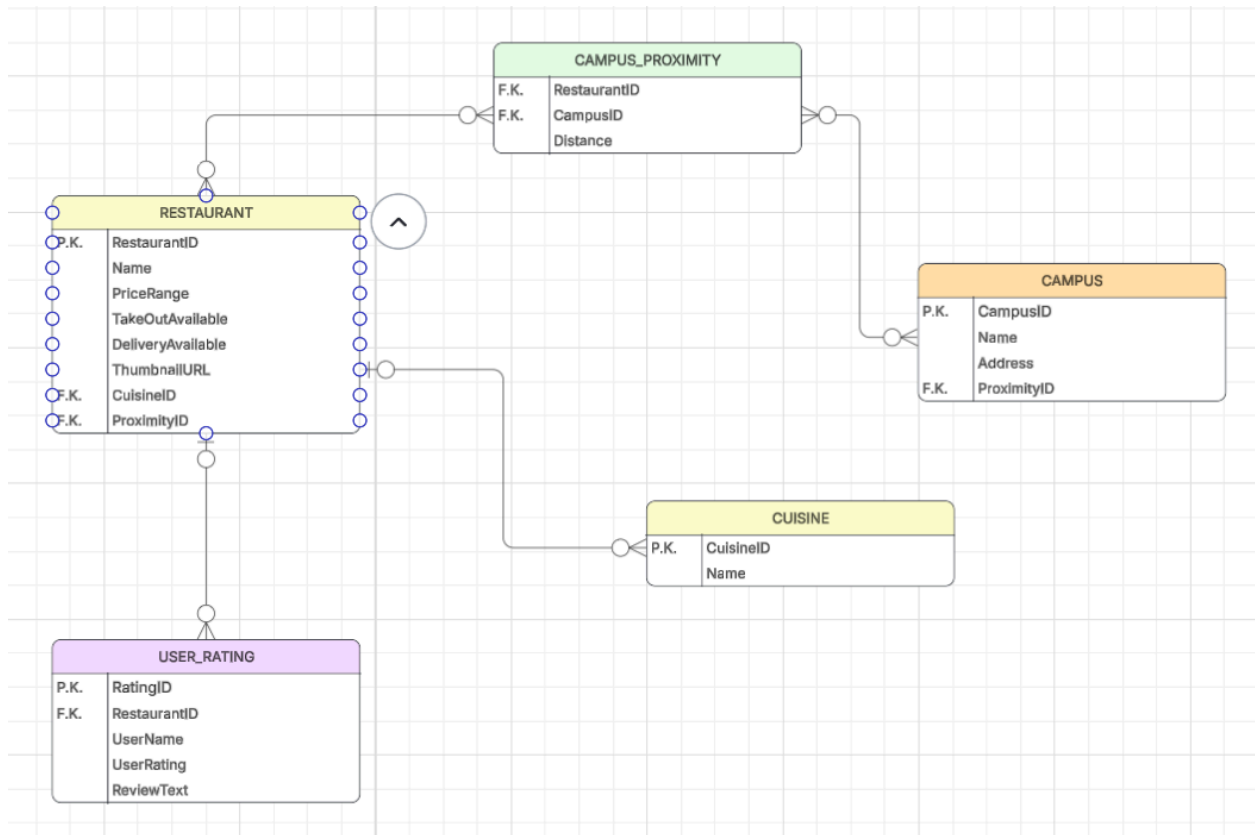
- Submitting Rating: Registered Users should be able to click on a restaurant and submit a User Rating (and optional review text). The application must save this data to the User\_Rating table.

### BONUS: Basic Analytics/Data Reports (To showcase database queries)

To demonstrate advanced querying skills beyond basic filtering, the system should be capable of performing the following database operations (results would likely be visible only in a basic backend admin panel or developer logs):

- Top-Rated Restaurants: Find the top-K restaurants based on the average value of User Ratings (for a given time period, if applicable).
- Campus Popularity: Compute the number of unique Restaurants associated with each Campus to determine which campus area has the most options.
- Service Trends: Compute the percentage of restaurants that offer Take Out versus Food Delivery Service across all data.

# UNI EATS ER-Diagram



## Entities and Attributes

### RESTAURANT (The central entity)

- RestaurantID (PK) - Primary Key
- Name
- PriceRange (<\$5, \$5-\$10, \$10-\$20)
- TakeOutAvailable (Boolean)
- DeliveryAvailable (Boolean)
- ThumbnailURL
- CuisineID (FK to CUISINE)
- ProximityID (FK to CAMPUS PROXIMITY)

### CAMPUS (To facilitate your unique proximity filter)

- CampusID (PK)
- Name (e.g., 'UIC', 'Loyola Chicago')
- Address
- ProximityID (FK to CAMPUS PROXIMITY)

**CAMPUS PROXIMITY (Bridges Restaurant with Campus)**

- RestaurantID (FK)
- CampusID (FK)
- Distance (Miles)

**CUISINE (To allow for easy filtering and consistent data entry)**

- CuisineID (PK)
- Name(e.g., 'Japanese', 'Korean')

**USER\_RATING (The table for user reviews)**

- RatingID (PK)
- RestaurantID (FK to RESTAURANT)
- UserName (String)
- UserRating (e.g., a number 1-5)
- ReviewText (String/NOT NULL)

**Tech Stack: Frontend (React/Node.js) → Backend (Python/Flask)  
→ Database (PostgreSQL)**