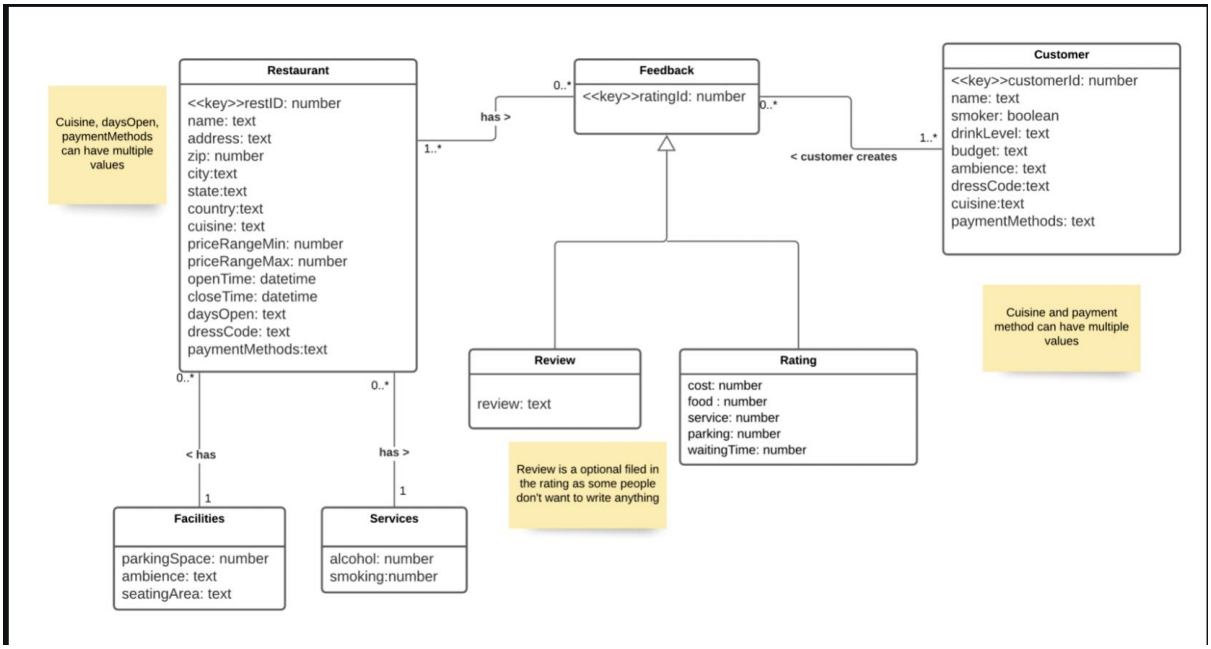
# PROJECT REQUIREMENTS:

# UML:




# PROBLEM REQUIREMENTS:

## Project proposal:

We're going to build a restaurant review management system that lets users submit reviews for a restaurant and lets the restaurant/user manage these reviews.
The application will support:

- CRUD operations of the reviewers
- CRUD operations of the restaurants
- CRUD operations of the reviews
- Query/filter reviews based on cost, service, parking, waiting time, and overall rating
- Query/filter restaurants based on cuisine
- Leaderboard for top 20 restaurants that have the most reviews
- Leaderboard for top 20 customers that have given the most reviews

## Requirements of the problem:

In today's age **restaurants** lean on services like yelp to help them advertise their **restaurant**. These services try to summarize the **customer's** experience by allowing them to *rate and review* the restaurants. Restaurants use the services to build credibility by *asking* customers for positive reviews when they feel that the customer is happy. The Project consists of the following parts: The ability to add reviews, customer and restaurants.
The customer should be able to *create* an **account** by providing the following details: Information on if they are a **smoker**, if they are **heavy drinkers**, the **budget**, the **ambiance** the customer prefers, the preferred **dress code**, what **cuisine** they like and what **payment methods** they have available.
After creating an account, a customer can *write* reviews for an already registered restaurant. A restaurant has the following information provided: The **address**, **city, state, country**, **price range**, the **cuisine** they are serving. The restaurant also provides certain **facilities** like **parking space** available, **ambience, seating area** and **services** like **alcohol, dress code, smoking area.** They have a set of available **payment methods** and **working hours.**
When *creating* a review, the customer *provides* a **rating** of 0-5 for the **cost**, **food**, **service**, **parking**, **waiting time**, and if they want a written **review**.

### Nouns in bold and verbs in Italic

Nouns:

**Restaurants**
- Name
- Address
- Price Range
- Cuisine

**Facilities**
- Ambience
- Seating Area
- Parking

**Services**
- Alcohol
- dress code
- smoking area
- payment methods
- working hours

**Reviews**
- Cost
- Food
- Service
- Parking
- waiting time

**Account/customer**
- smoker
- heavy drinkers
- Budget
- Ambiance
- dress code
- Cuisine
- payment methods
- name

**Verbs:**
- Asking – restaurants ask customers to review them
- Creates - customer creates reviews on restaurant
- Provides – restaurants provide certain facilities and services

## Business Rules:

1. Facilities can be offered by many restaurants while each restaurant provides a set of facilities.
2. Services can be offered by many restaurants while each restaurant provides a set of services.
3. A restaurant can get none to many feedbacks while feedback has 1 or more restaurants
4. Customer can create 0 or more feedbacks while feedbacks have 1 or more customers
5. Cuisine, Days Open and Payment methods can have multiple values
6. There are about 19 cuisines in total to choose from
7. There are 5 total dress code types
8. There are 5 different payment methods
9. 7 working days

# FUNCTIONALITY IN REDIS:

# Restaurants:

1. Leaderboard for Users:
   Sorted sets were used to store the user and the number of times they have given a rating/review as score to create a leaderboard for the top 20 users.

2. Leaderboard for Restaurants
   Sorted sets have been used to record the number of times a restaurant has been reviewed to display the top 20 restaurants that have the highest reviews.

3. Create operation on Restaurant:
   Hashes have been used to create a new restaurant with the key as rest:restID and value as the property of the restaurant.

4. Read operation of Restaurants:
   A list is used to store all the restaurant ids which can we read to get all restaurant details.

5. Update operation on Restaurant:
   Like create operation.

6. Delete
   Removes the restid from the restaurant list and removes the following hash

7. List of Cuisines

An unsorted set is used to keep track of all the available cuisines

8. Query According to cuisine
   An unsorted set is used to store all the restaurant ids that come under a particular cuisine with the key as cuisine:American and value as all the restids.

**CUSTOMERS:**

1. **Cache Customers**
   To set a hash in Redis we use the hSet Function with the key as the first parameter and the Json object as the second parameter.

2. Cache Rating