# CS374 – Intro to Database Management

# Application Development Project

# Rubric for Second Deliverable

## Group Member #1: McKenna Galle (The Great Generator)

## Group Member #2: Tersa Almaw (The Awesome Advocate)

## Group Member #3: Vina Le (The Lifeline Leader)

**TEAM G-AL-LE**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Requirements | Points | Awarded |
| Description of Application | * An overview of your application * System requirements (e.g. hardware, DBMS, other software) * A detailed description of your application * Are there features that will not be implemented? What are they, and why won’t you fulfill them? | 10 |  |
| Project Management -Schedule | * Detailed schedule of who will do what part of project, by when | 5 |  |
| Logical Diagram | * Logical diagram in UML or E-R * Discussion of how your data model will satisfy the needs of your application * Discussion of alternative designs that you did not do (and why) | 15 |  |
| Queries Required | * Required queries in English (not SQL) * What entities and/or relationships are required for each query? * How will each query satisfy the needs of your application | 15 |  |
| Grammar, punctuation, syntax, and references | * Follow rules from the Penguin handbook on writing * References as appropriate (e.g. if you are modeling your application after an existing application, make note of that) | 5 |  |

**Project Management – Schedule**

Week 0 (Nov. 9 – 13):

* Do the deliverable 2
* Plan all of this even though we know we’re not going to stick to it

Week 1 (Nov. 14 – 20):

* Schema created and rough outline of the E-R diagram
* Implementing the database:
  + McKenna: Type/Food
  + Tersa: Location and Shopping List
  + Vina: “In the Fridge”

Week 2 and 3 (Nov. 21 – Dec.4):

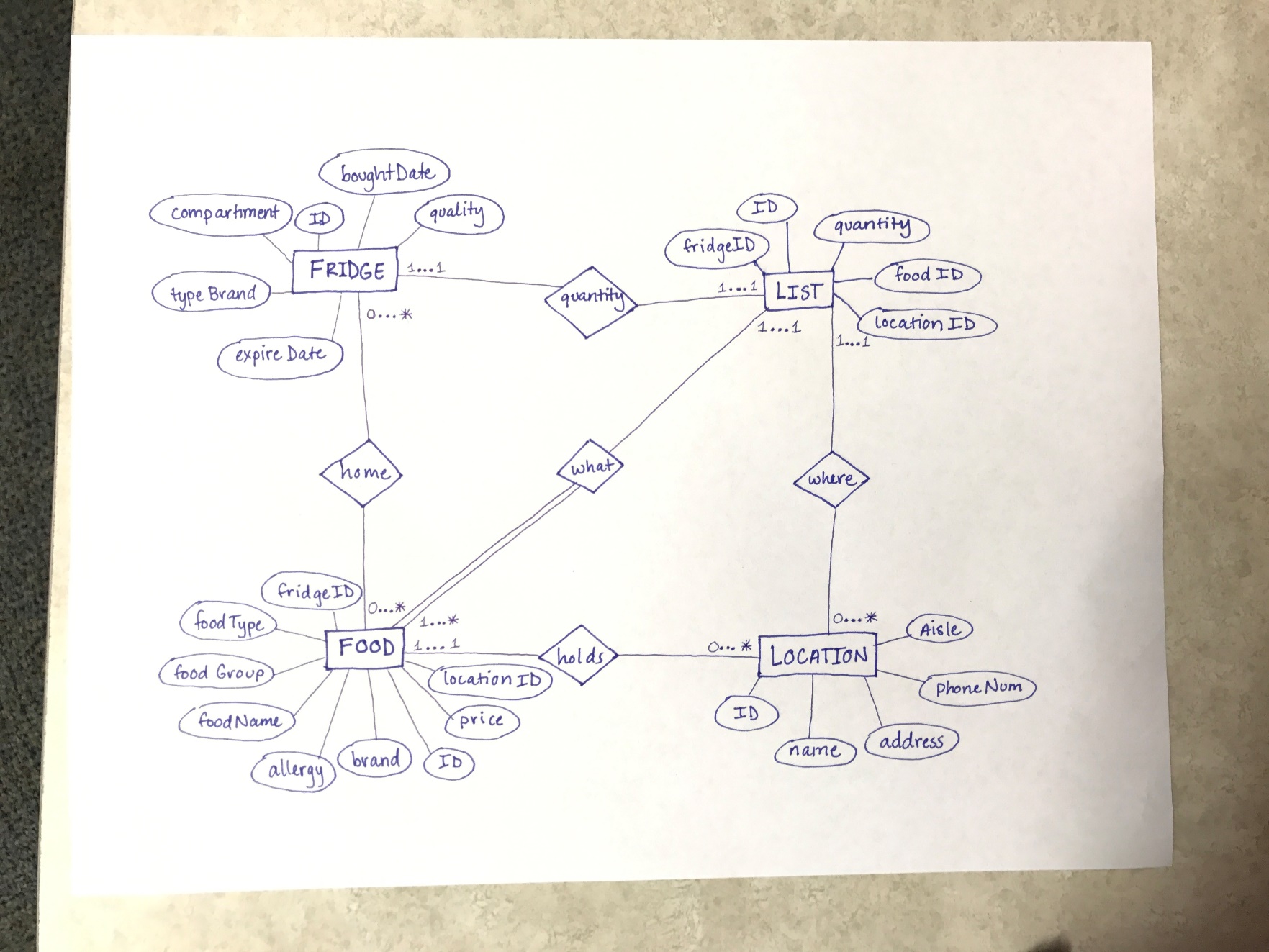
* Complete database
* Start the outline for how the code will be implemented
  + McKenna: Connect the embedded SQL to the Access file
  + Tersa and Vina: Evaluate the queries

Week 4 (Dec. 5 – 11):

* Modify the E-R diagram
* Finish Code! (Really, we have to do it)
* Implement it into an app

**Logical Diagram**

* E-R Diagram that resembles the fridge:



* Schema:
  + Fridge(ID, foodID, expireDate, boughtDate, quantity, quality, typeBrand, compartment)
  + Food(ID, fridgeID, foodType, foodGroup, foodName, allergy, brand, price, locationID)
  + Location(ID, name, address, phoneNum, aisle)
  + List(ID, locationID, fridgeID, foodID, quantity)
* **Queries:**
  + Fridge
    - Has a [1…1] relationship with List because there exist one fridge for the shopping list (buying only for this fridge)
    - Has a [0…\*] relationship with Food because fridge can have zero to infinity types on food currently in fridge.
    - Fridge uses fridgeID to display food items currently in the fridge.
    - Fridge contains the expiration date and the date items were bought to display items that need to be thrown away.
  + Food
    - Has a [0…\*] relationship with Fridge because food still exists even though there is no fridge.
    - Has a [1…1] relationship with Location because there must be at least one food item at the location.
    - Has a [1…\*] relationship with List because there must be at least one item for List to exist.
    - Food contains fridgeID that displays the food’s quantity, and boughtDate, and in which compartment the food is stored at.
    - Food uses LocationID and displays a grocery store’s name and address where the food can be bought.
  + Location
    - Has a [0…\*] relationship with Food because there exist x amount of stores that carries food.
    - Has a [0…\*] relationship with List because the location still exist even if the List is not for that location.
    - where the Food is shopped at using the ID
    - Each location has an aisle that accesses the location of an item in the store.
  + List
    - Has a [1…1] relationship with Fridge because there exist one list for the fridge (we are only shopping for our own fridge)
    - Has a [1…1] relationship with Food because there exist one list for the food item that needs to be bought.
    - Has a [1…1] relationship with Location because there exist one list for that shopping location
    - Uses the ID to display what items need to be bought
    - Uses locationID to display where the food can be bought
    - Uses foodID to get the name of the food
    - Uses fridgeID to display the number of items already in the fridge.