University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Feb 26, 2024
Group Number: 66

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Brendan Yuen	16212987	u2r4d	brendanyuen@hotmail.com
Roberto Mulliadi	45216298	d4n0y	robertomulliadi@gmail.com
Zehao Guo	13448915	r2k8i	zehaog@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your email address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia.

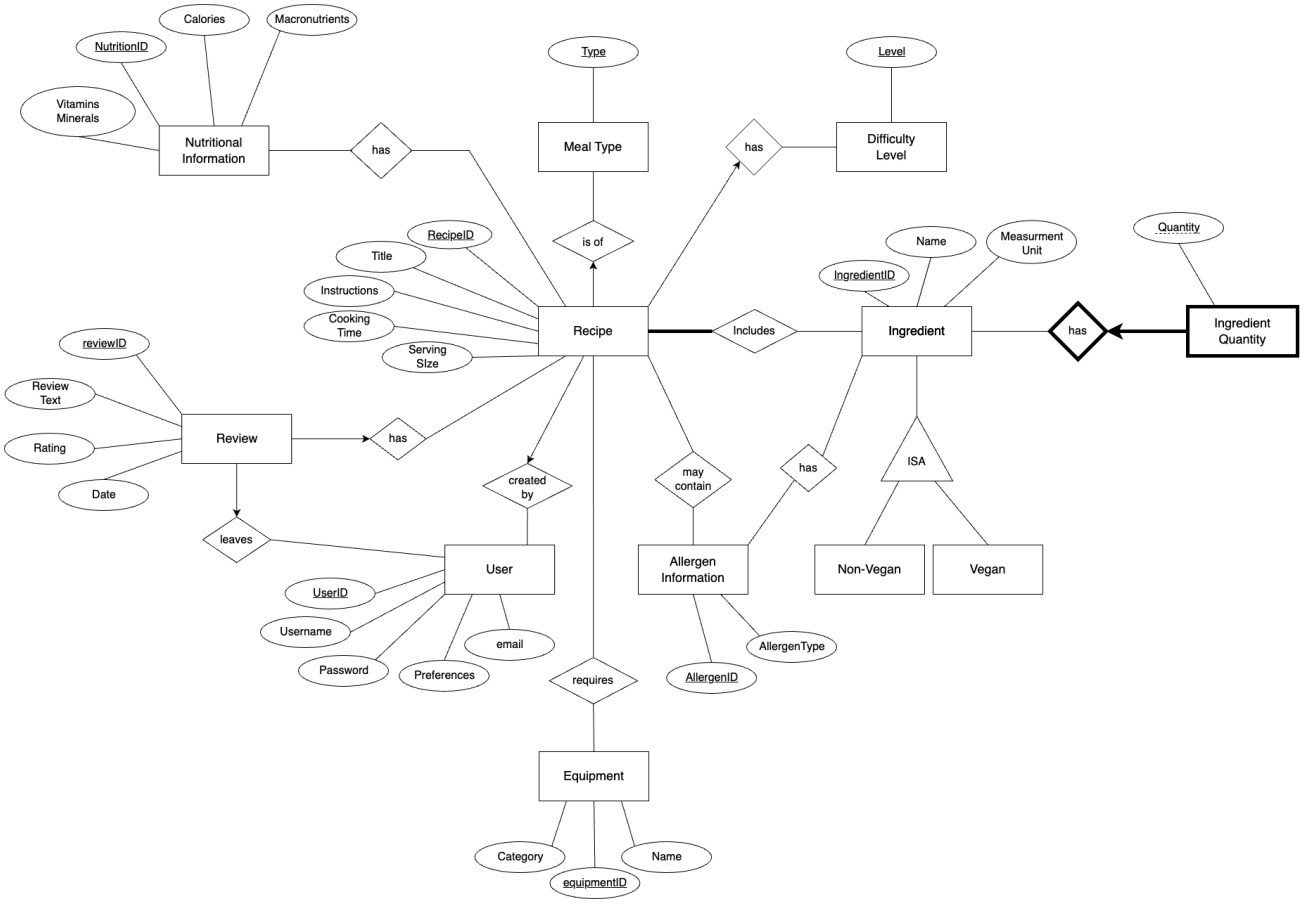
University of British Columbia, Vancouver
Department of Computer Science

Deliverables
Each group must provide the following as a single PDF file:
A completed cover page (template on Canvas)

1. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

The domain of the application is in the culinary area. It will cover entities like recipes, ingredients, and ratings. This database will be for recipes for restaurants and individual use. The app will try to provide a platform for managing and sharing recipes among professional and amateur chefs. Users can search, browse, filter and upload recipes based on dietary restrictions, cuisine type, or difficulty level.

2. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.



We removed unit as an attribute of the weak entity IngredientQuantity, because of redundancy

3. If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. Your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why to better assist the group moving forward.

4. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:
    a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.
    b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

Equipment(equipmentID: INTEGER, Category: CHAR(20), Name: VARCHAR)
Not Null: Name

requires(**equipmentID**: INTEGER, **RecipeID:** INTEGER)

Recipe(RecipeID: INTEGER, Title: VARCHAR, Instructions: VARCHAR, CookingTime: INTEGER, Serving Size INTEGER)
- Not Null: Instructions

isOf(**RecipeID**: INTEGER, **Type:** VARCHAR)

MealType(Type: VARCHAR)

Recipe_has_DifficultyLevel(**RecipeID:** INTEGER, **Level:** INTEGER)

DifficultLevel(Level: INTEGER)

includes(**RecipeID:** INTEGER, **IngredientID:** INTEGER)

Ingredient(IngredientID: INTEGER, Name: VARCHAR, MeasurementUnit: VARCHAR)
    Non-Vegan(**IngredientID:** INTEGER, Name: VARCHAR, MeasurementUnit: VARCHAR)
    Vegan(**IngredientID:** INTEGER, Name: VARCHAR, MeasurementUnit: VARCHAR)
- Not Null: Name, Measurement Unit

IngredientQuantity(Quantity: INTEGER, **IngredientID:** INTEGER)

Ingredient_has_AllergenInformation(**IngredientID:** INTEGER, **AllergenID:** INTEGER)

mayContain(**RecipeID:** INTEGER, **AllergenID:** INTEGER)

AllergenInformation(AllergenID: INTEGER, AllergenType: VARCHAR)
- Not Null: AllergenType

createdBy(**RecipeID:** INTEGER, **UserID:** INTEGER)

User(**UserID:** INTEGER, Username: VARCHAR, Password: VARCHAR, Preferences: VARCHAR, email: VARCHAR)
- Unique: Username, email
- Not Null: Username, Password, email
- Candidate Key: Username, email

leaves(**UserID:** INTEGER, **reviewID:** INTEGER)

Review(**reviewID:** INTEGER, Review Text: VARCHAR, Date: DATE)

Review_has_Recipe(**reviewID:** INTEGER, **RecipeID:** INTEGER)

Recipe_has_NutritionalInformation(**RecipeID:** INTEGER, **NutritionID:** INTEGER)

NutritionalInformation(NutritionalID: INTEGER, Calories: INTEGER, Macronutrients: VARCHAR, Vitamins & Minerals: VARCHAR)

5. Functional Dependencies (FDs)
    a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).
    PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A -> A.

equipmentID -> Category, Name
reviewID -> ReviewText, Rating, Date
NutritionID -> VitaminsMinerals, Calories, Macronutrients
Macronutrients -> Calories
UserID -> Username, Password, Preferences, email
Username -> Username, Password, Preferences, email
email -> Username, Password, Preferences, email
AllergenID -> AllergenType
IngredientID -> Name, MeasurementUnit
ReceiptID -> Title, Instructions, Cooking Time, ServingSize

Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process. You do not need to have a non-PK/CK FD for each relation but be reasonable. If your TA feels that some non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.

6. Normalization
    a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization.

       You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, …). ALL Tables must be listed, not only the ones post normalization.

Equipment:
equipmentID -> Category, Name

Review:
reviewID -> ReviewText, Rating, Date

Table User:
UserID -> Username, Password, Preferences, email
Username -> Username, Password, Preferences, email
email -> Username, Password, Preferences, email

Allergen Information:
AllergenID -> AllergenType

Ingredient:
IngredientID -> Name, MeasurementUnit

Recipe:
RecipeID -> Title, Instructions, CookingTime, ServingSize

Nutritional Information:
NutritionID -> VitaminsMinerals, Calories, Macronutrients
Macronutrients -> Calories

Normalizing Nutritional Information to BCNF:
Step 1 Pick any FD in the set F of FDs that violate BCNF of the form X->b:
Macronutrients-> Calories
Step 2 Decompose R into two relations R1(A-b) & R2(X U b):
R1 ={NutritionID, VitaminsMinerals, Macronutrients}
R2 ={Macronutrients, Calories}

No recursion needed, BCNF decompositions completed.

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE Equipment (
        EquipmentID INTEGER PRIMARY KEY,
        Category VARCHAR(20),
        Name VARCHAR NOT NULL
);

CREATE TABLE Recipe (
        RecipeID INTEGER PRIMARY KEY,
        Title VARCHAR,
        Instructions VARCHAR NOT NULL,
        CookingTime INTEGER,
        ServingSize INTEGER
);

CREATE TABLE MealType (
        Type VARCHAR PRIMARY KEY
);

CREATE TABLE DifficultyLevel (
        Level INTEGER PRIMARY KEY;
);
```

```
CREATE TABLE Ingredient (
        IngredientID INTEGER PRIMARY KEY,
        Name VARCHAR NOT NULL,
        MeasurementUnit VARCHAR NOT NULL
);

CREATE TABLE Non_Vegan (
        IngredientID INTEGER PRIMARY KEY,
        Name VARCHAR NOT NULL,
        MeasurementUnit VARCHAR NOT NULL,
        FOREIGN KEY (IngredientID) REFERENCES Ingredient(IngredientID));

CREATE TABLE Vegan (
        IngredientID INTEGER PRIMARY KEY,
        Name VARCHAR NOT NULL,
        MeasurementUnit VARCHAR NOT NULL,
        FOREIGN KEY (IngredientID) REFERENCES Ingredient(IngredientID)
);

CREATE TABLE IngredientQuantity (
        Quantity INTEGER,
        IngredientID INTEGER,
        PRIMARY KEY (QuantityID, IngredientID)
);

CREATE TABLE AllergenInformation (
        AllergenID INTEGER PRIMARY KEY,
        AllergenType VARCHAR NOT NULL
);

CREATE TABLE User (
        UserID INTEGER PRIMARY KEY,
        Username VARCHAR NOT NULL,
        Password VARCHAR NOT NULL,
        Preferences VARCHAR,
        Email VARCHAR NOT NULL,
        UNIQUE (Username),
        UNIQUE (Email)
)
```

```
CREATE TABLE Review (
        ReviewID INTEGER PRIMARY KEY,
        ReviewText VARCHAR,
        Date DATE
);

CREATE TABLE NutritionalInformation (
        NutritionID INTEGER PRIMARY KEY,
        Calories INTEGER,
        Macronutrients VARCHAR,
        VitaminsMinerals VARCHAR
);


CREATE TABLE Recipe_has_DifficultyLevel (
        RecipeID INTEGER,
        Level INTEGER,
        PRIMARY KEY (RecipeID, Level),
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID),
        FOREIGN KEY (Level) REFERENCES DifficultyLevel(Level)
);

CREATE TABLE includes (
        RecipeID INTEGER,
        IngredientID INTEGER,
        PRIMARY KEY (RecipeID, IngredientID),
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID),
        FOREIGN KEY (IngredientID) REFERENCES Ingredient(IngredientID)
);

CREATE TABLE Ingredient_has_AllergenInformation (
        IngredientID INTEGER,
        AllergenID INTEGER,
        PRIMARY KEY (IngredientID, AllergenID),
        FOREIGN KEY (IngredientID) REFERENCES Ingredient(IngredientID),
        FOREIGN KEY (AllergenID) REFERENCES AllergenInformation(AllergenID)
);
```

```
CREATE TABLE mayContain (
        RecipeID INTEGER,
        AllergenID INTEGER,
        PRIMARY KEY (RecipeID, AllergenID),
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID),
        FOREIGN KEY (AllergenID) REFERENCES AllergenInformation(AllergenID)
);

CREATE TABLE createdBy (
        RecipeID INTEGER,
        UserID INTEGER,
        PRIMARY KEY (RecipeID, UserID),
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID),
        FOREIGN KEY (UserID) REFERENCES User(UserID)
);

CREATE TABLE leaves (
        UserID INTEGER,
        ReviewID INTEGER,
        PRIMARY KEY (UserID, ReviewID),
        FOREIGN KEY (UserID) REFERENCES User(UserID),
        FOREIGN KEY (ReviewID) REFERENCES Review(ReviewID)
);

CREATE TABLE Review_has_Recipe (
        ReviewID INTEGER,
        RecipeID INTEGER,
        PRIMARY KEY (ReviewID, RecipeID),
        FOREIGN KEY (ReviewID) REFERENCES Review(ReviewID),
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID)
);

CREATE TABLE Recipe_has_NutritionalInformation (
        RecipeID INTEGER,
        NutritionID INTEGER,
        PRIMARY KEY (RecipeID, NutritionID),
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID),
        FOREIGN KEY (NutritionID) REFERENCES NutritionalInformation(NutritionID)
);
```

```
CREATE TABLE requires(
        equipmentID INTEGER,
        RecipeID INTEGER
        PRIMARY KEY(equipmentID, RecipeID),
        FOREIGN KEY (equipmentID) REFERENCES Equipment(EquipmentID)
        FOREIGN KEY (RecipeID) REFERENCES Recipe(RecipeID)
);
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
INSERT INTO Equipment VALUES
(1, 'Cutting', 'Paring Knife')
(2, 'Cooking', 'Microwave')
(3, 'Cooking', "Oven')
(4, 'Pots and Pans', 'Frying Pan')
(5, 'Measuring', 'Liquid Measure');

INSERT INTO Recipe(RecipeID, Title, Instructions, CookingTime, Serving Size) VALUES
(1, 'Best Buttermilk Biscuits', 'insert instruction', 50, 4)
(2, 'Oatmeal', 'boil oats', 30, 1)
(3, 'Good soup', 'boil things', 30, 5)
(4, 'Smoothie', 'blend fruits', 10, 4)
(5, 'Apple Juice', 'juice apples', 10, 2);

INSERT INTO MealType (Type) VALUES
('Breakfast'),
('Lunch'),
('Dinner')
('Dessert')
('Beverage');

INSERT INTO DifficultyLevel (Level) VALUES
(1),
(2),
(3),
(4),
(5);
```

```
INSERT INTO User (UserID, Username, Password, Preferences, Email) VALUES
(1, 'John Smith', 'supersecretpassword123', 'Italian cuisine', 'smith_john@gmail.com'),
(2, 'Ben Simmons', 'iamtall4019', 'Healthy eating', 'bensimmons@gmail.com'),
(3, 'Alexander Bell', 'imakeTelephonesringring3303', 'Desserts', bellringer_alex@yahoo.com),
(4, 'Lionel Messi', 'iamthegoat', 'Asian dishes', greatestofall_time@gmail.com),
(5, 'Billy Joel', '\\pianoman\\!', 'Vegetarian', 'billythejoelreal@gmail.com');

INSERT INTO Ingredient (IngredientID, Name, MeasurementUnit) VALUES
(1, 'Eggs', 'unit'),
(2, 'Rice', 'gram'),
(3, 'Chicken Breast', 'gram'),
(4, 'Potato', 'unit'),
(5, 'Cow Milk', 'mL'),
(6, 'Steak', 'gram'),
(7, 'Salmon', 'gram'),
(8, 'Apple', 'unit'),
(9, 'Orange' 'unit'),
(10, 'Flour', 'mL')
(11, 'Peanut', 'unit'),
(12, 'Almond', 'unit');

INSERT INTO Non_Vegan (IngredientID, Name, MeasurementUnit) VALUES
(1, 'Eggs', 'unit'),
(3, 'Chicken Breast', 'gram'),
(5, 'Cow Milk', 'mL')
(6, 'Steak', 'gram')
(7, 'Salmon', 'gram');

INSERT INTO Vegan (IngredientID, Name, MeasurementUnit) VALUES
(2, 'Rice', 'gram'),
(4, 'Potato', 'unit'),
(8, 'apple', 'unit'),
(9, 'Orange' 'unit'),
(10, 'Flour', 'ml'),
(11, 'Peanut', 'unit'),
(12, 'Almond', 'unit');

INSERT INTO IngredientQuantity (IngredientID, Quantity) VALUES
(2, 300),
(2, 400),
(1, 1),
(1, 2),
(1, 3);
```

```
INSERT INTO AllergenInformation (AllergenID, AllergenType) VALUES
(1, 'Gluten'),
(2, 'Dairy'),
(3, 'Nuts'),
(4, 'Eggs'),
(5, 'Soy');

INSERT INTO Recipe_has_DifficultyLevel (RecipeID, Level) VALUES
(1, 4),
(2, 2),
(3, 3),
(4, 2),
(5, 1);

INSERT INTO includes (RecipeID, IngredientID) VALUES
(1, 1),
(1, 10),
(4, 8),
(4, 9),
(5, 8);

INSERT INTO Ingredient_has_AllergenInformation (IngredientID, AllergenID) VALUES
(10, 1),
(1, 4),
(11, 3),
(5, 2),
(12, 3);

INSERT INTO mayContain (RecipeID, AllergenID) VALUES
(1,1),
(1,2),
(1,4),
(1,3),
(2,3);

INSERT INTO createdBy (RecipeID, UserID) VALUES
(1,1),
(2,2),
(3,3),
(4,4),
(5,5);
```

INSERT INTO leaves (UserID, ReviewID) VALUES
(1, 2),
(3, 3),
(4, 1),
(2, 4),
(5, 5);

INSERT INTO Review (ReviewID, ReviewText, Date) VALUES
(1,'Tastes great!',2024-03-01),
(2,'I like the process of doing it.', 2024-03-02),
(3, 'It'd be better if it can taste sweeter', 2024-03-05),
(4, 'Took too long to make it.', 2024-04-02),
(5, 'Nice!',2024-03-01);

INSERT INTO Review_has_Recipe (ReviewID, RecipeID) VALUES
(1, 3),
(2, 4),
(3, 1),
(4, 1),
(5, 5);

INSERT INTO Recipe_has_NutritionalInformation (RecipeID, NutritionID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

INSERT INTO NutritionalInformation (NutritionID, Calories, Macronutrients, VitaminsMinerals) VALUES
(1, 212, '4.2 g Protein, 9.8 g fat, 27 g carbohydrate per 60 g', 'Cholesterol 1.8mg, Sodium 348mg, Calcium 141 mg, Iron 1.7 mg, Potassium 72.6mg per 60 g'),
(2, 95, '5 g Protein, 3 g fat, 27 g Carbohydrates per 81 g', '8.1g fiber per 81 g '),
(3, 80, '5.3g Protein, 2.2g Fat, 29.1g Carbohydrates per 100g', 'Vitamin A 206.4µg, Calcium 48mg per 100g'),
(4, 513, '21 g Protein , 25 g Fat , 56 g Carbohydrates per 100g', 'vitamin C 12.3 mg per 100g'),
(5, 114, '0.2 g Protein, 0.3 g Fat, 28 g Carbohydrates per 200mL', 'vitamin C 13.3 mg , Potassium 142 mg, Iron 0.44 mg , Calcium 8 mg per 200mL');

```
INSERT INTO requires(equipmentID, RecipeID) VALUES
(1, 3),
(2, 2),
(3, 1),
(5, 4),
(5, 5);
```

Note: Be consistent with the names used in your ER diagram, schema, and FDs. Make a note if the name has been intentionally changed.