University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct 15th 2024

Group Number: 40

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Alex Dart	93792588	g1r1c	adart075@gmail.com
Griffin Velichko	74979287	w0n1r	griffin.velichko@gmail.com
Anna Friesen	33401860	w5h0a	annafriesen@shaw.ca

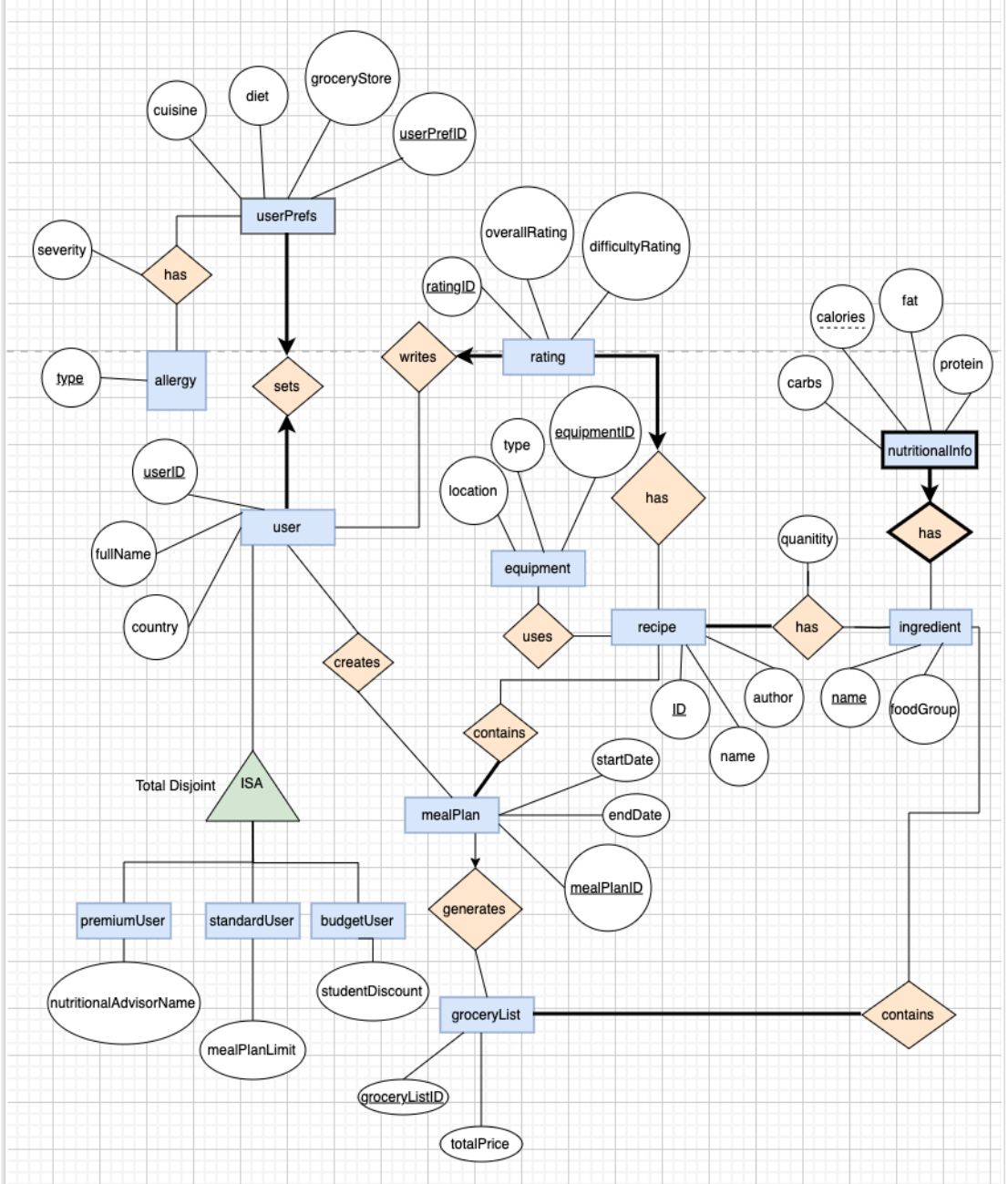
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address,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

2. A brief (~2-3 sentences) summary of your project.

The application allows users to plan meals, create grocery lists, and track their nutritional intake. Users can select from a database of recipes, customize their meal plans for a week, and generate corresponding grocery lists. The system will track recipes, ingredients, user preferences (like allergies and a preferred grocery store), and meal schedules.

3. The ER diagram you are basing your item #3 (below) on.



4. The schema derived from your ER diagram (above).

underlined = PK, **bold** = FK

allergy(
type: *VARCHAR* PRIMARY KEY)

userHasAllergy(
allergyType: *VARCHAR,*
userID*: integer*,
severity: *VARCHAR,*
FOREIGN KEY allergyType REFERENCES Allergy, (allergyType),
FOREIGN KEY userID REFERENCES User(userID))

user(
userID: *integer* PRIMARY KEY,
fullName: *VARCHAR*,
country: *VARCHAR*,
userPrefID: *integer* UNIQUE,
cuisine: *VARCHAR*,
diet: *VARCHAR*,
groceryStore: *VARCHAR*)

premiumUser(
userID: *integer*,
nutritionalAdvisorName: *VARCHAR,*
FOREIGN KEY userID REFERENCES User(userID))

standardUser(
userID: *integer*,
mealPlanLimit: *integer,*
FOREIGN KEY userID REFERENCES User(userID))

budgetUser(
userID: *integer*,
studentDiscount: *float,*
FOREIGN KEY userID REFERENCES User(userID))

userCreatesMealPlan(
userID: *integer,*
mealPlanID: *integer,*
FOREIGN KEY userID REFERENCES User(userID),
FOREIGN KEY mealPlanID REFERENCES MealPlan(mealPlanID))

rating(
ratingID: *integer* PRIMARY KEY,
overallRating: *integer*,
difficultlyRating: *integer,*
userID: *integer* NOT NULL,
recipeID: *integer* NOT NULL,
FOREIGN KEY userID REFERENCES User(userID),
FOREIGN KEY recipeID REFERENCES Recipe(ID))

equipment(
equipmentID: *integer* PRIMARY KEY,
type: *VARCHAR,*
location: *VARCHAR*)

recipe(
ID: *integer* PRIMARY KEY,
name: *VARCHAR*,
author: *VARCHAR*)

recipeHasIngredient(
recipeID: *integer*,
ingredientName: *VARCHAR,*
quantity: *integer,*
FOREIGN KEY recipeID REFERENCES Recipe(ID)
FOREIGN KEY ingredientName REFERENCES Ingredient(name))

mealPlanContainsRecipe(
mealPlanID: *integer,*
recipeID: *integer,*
FOREIGN KEY mealPlanID REFERENCES MealPlan(mealPlanID),
FOREIGN KEY recipeID REFERENCES Recipe(ID))

ingredient(
name: *VARCHAR* PRIMARY KEY,
foodGroup: *VARCHAR*)

ingredientNutritionalInfo(
name: *VARCHAR,*
calories: *integer,*
fat: *integer,*
protein: *integer,*
FOREIGN KEY name REFERENCES Ingredient(name))

mealPlan(
mealPlanID: *integer* PRIMARY KEY,
endDate: *date*,
startDate: *date* NOT NULL,
groceryListID: *integer,*
FOREIGN KEY groceryListID REFERENCES groceryList(groceryListID))

groceryList(
groceryListID: *integer* PRIMARY KEY,
totalPrice: *integer*)

groceryListContainsIngredient(
groceryListID: *integer,*
ingredientName: *VARCHAR,*
FOREIGN KEY groceryListID REFERENCES GroceryList(groceryListID),
FOREIGN KEY ingredientName REFERENCES Ingredient(name))

We cannot represent the total participation constraints for groceryList needing at least one ingredient, mealPlan needing at least one recipe, or each recipe needing at least ingredient as that requires an assertion.

5. Functional Dependencies (FDs)

bold = causes a table to need normalization

userID → fullName, country, userPrefsID

userPrefID → groceryStore, diet, cuisine

userPrefID, allergyType → severity

ratingID → overallRating, difficultyRating, userID, recipeID

mealPlanID → startDate, endDate, groceryListID

equipmentID → type, location

groceryListID → totalPrice

recipeID → name, author

recipeID, ingredientName → quantity

ingredientName → foodGroup

ingredientName, nutritionalInfoCalories → carbs, fat, protein

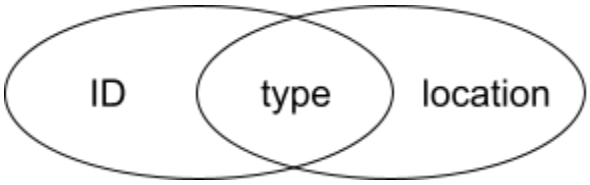
equipmentType → **equipmentLocation** (this is the additional FD added that is not already a PK)

6. Normalization

Normalizing:

equipment(<u>equipmentID</u>, equipmentType, equipmentLocation)
FD that violates: **equipmentType → equipmentLocation**

decompose:



giving us:

equipment(
    <u>equipmentID</u>: *INTEGER*,
    <u>equipmentType</u>: *VARCHAR*)

equipmentLocations(
    <u>equipmentType</u>: *VARCHAR*,
    equipmentLocation: *VARCHAR)*
Note: Primary Keys are underlined

Everything Else:

Every other relationship does not need normalization as they satisfy BCNF. These relationships are copied below:

allergy(
    <u>type</u>: *VARCHAR* PRIMARY KEY)

userHasAllergy(
    <u>allergyType</u>: *VARCHAR,*
    <u>userID</u>*: integer*,
    severity: *VARCHAR,*
    FOREIGN KEY allergyType REFERENCES Allergy(allergyType),
    FOREIGN KEY userID REFERENCES User(userID))

user(
    <u>userID</u>: *integer* PRIMARY KEY,
    fullName: *VARCHAR*,
    country: *VARCHAR*,
    userPrefID: *integer* UNIQUE,
    cuisine: *VARCHAR*,

diet: *VARCHAR*,
groceryStore: *VARCHAR*)

premiumUser(
userID: *integer*,
nutritionalAdvisorName: *VARCHAR,*
FOREIGN KEY userID REFERENCES User(userID))

standardUser(
userID: *integer*,
mealPlanLimit: *integer,*
FOREIGN KEY userID REFERENCES User(userID))

budgetUser(
userID: *integer*,
studentDiscount: *float,*
FOREIGN KEY userID REFERENCES User(userID))

userCreatesMealPlan(
userID: *integer,*
mealPlanID: *integer,*
FOREIGN KEY userID REFERENCES User(userID),
FOREIGN KEY mealPlanID REFERENCES MealPlan(mealPlanID))

rating(
ratingID: *integer* PRIMARY KEY,
overallRating: *integer*,
difficultlyRating: *integer,*
userID: *integer* NOT NULL,
recipeID: *integer* NOT NULL,
FOREIGN KEY userID REFERENCES User(userID),
FOREIGN KEY recipeID REFERENCES Recipe(ID))

recipe(
ID: *integer* PRIMARY KEY,
name: *VARCHAR*,
author: *VARCHAR*)

recipeHasIngredient(
recipeID: *integer*,
ingredientName: *VARCHAR,*
quantity: *integer,*
FOREIGN KEY recipeID REFERENCES Recipe(ID)
FOREIGN KEY ingredientName REFERENCES Ingredient(name))

mealPlanContainsRecipe(
mealPlanID: *integer,*
recipeID: *integer,*
FOREIGN KEY mealPlanID REFERENCES MealPlan(mealPlanID),
FOREIGN KEY recipeID REFERENCES Recipe(ID))

ingredient(
name: *VARCHAR* PRIMARY KEY,
foodGroup: *VARCHAR*)

ingredientNutritionalInfo(
name: *VARCHAR,*
calories*: integer,*
fat: *integer,*
protein: *integer,*
FOREIGN KEY name REFERENCES Ingredient(name))

mealPlan(
mealPlanID: *integer* PRIMARY KEY,
endDate: *date*,
startDate: *date* NOT NULL,
groceryListID: *integer,*
FOREIGN KEY groceryListID REFERENCES groceryList(groceryListID))

groceryList(
groceryListID: *integer* PRIMARY KEY,
totalPrice: *integer*)

groceryListContainsIngredient(
groceryListID: *integer,*
ingredientName: *VARCHAR,*
FOREIGN KEY groceryListID REFERENCES GroceryList(groceryListID),
FOREIGN KEY ingredientName REFERENCES Ingredient(name))

7. The SQL DDL statements required to create all the tables from item #6.

```
CREATE TABLE Allergy (type VARCHAR PRIMARY KEY)

CREATE TABLE UserHasAllergy
        (allergyType VARCHAR,
        userID INTEGER,
        severity VARCHAR,
        PRIMARY KEY (allergyType, userID),
        FOREIGN KEY allergyType REFERENCES Allergy(allergyType)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY userID REFERENCES User(userID))

CREATE TABLE User
        (userID INTEGER PRIMARY KEY,
        fullName VARCHAR,
        country VARCHAR,
        userPrefID INTEGER UNIQUE,
        cuisine VARCHAR,
        diet VARCHAR,
        groceryStore VARCHAR)

CREATE TABLE PremiumUser
        (userID INTEGER PRIMARY KEY,
        nutritionalAdvisorName VARCHAR,
        FOREIGN KEY userID REFERENCES User(userID))

CREATE TABLE StandardUser
        (userID INTEGER PRIMARY KEY,
        mealPlanLimit INTEGER,
        FOREIGN KEY userID REFERENCES User(userID)))

CREATE TABLE BudgetUser
        (userID INTEGER PRIMARY KEY,
        studentDiscount FLOAT,
        FOREIGN KEY userID REFERENCES User(userID)))

CREATE TABLE UserCreatesMealPlan
        (userID INTEGER,
        mealPlanID INTEGER,
        PRIMARY KEY (userID, mealPlanID),
        FOREIGN KEY userID REFERENCES User(userID),
        FOREIGN KEY mealPlanID REFERENCES MealPlan(mealPlanID)
```

```
            ON DELETE SET NULL
            ON UPDATE CASCADE)

CREATE TABLE Rating
        (ratingID INTEGER PRIMARY KEY,
        overallRating INTEGER,
        difficultlyRating INTEGER,
        userID INTEGER NOT NULL,
        recipeID INTEGER NOT NULL,
        FOREIGN KEY userID REFERENCES User(userID),
        FOREIGN KEY recipeID REFERENCES Recipe(ID))

CREATE TABLE Equipment
        (equipmentID INTEGER,
        equipmentType VARCHAR,
        PRIMARY KEY (equipmentID, equipmentType))

CREATE TABLE EquipmentLocations
        (equipmentType VARCHAR PRIMARY KEY,
        equipmentLocation VARCHAR)

CREATE TABLE Recipe
        (ID INTEGER PRIMARY KEY,
        name VARCHAR,
        author VARCHAR)

CREATE TABLE RecipeHasIngredient
        (recipeID INTEGER,
        ingredientName VARCHAR,
        quantity INTEGER,
        PRIMARY KEY (recipeID, ingredientName),
        FOREIGN KEY recipeID REFERENCES Recipe(ID),
        FOREIGN KEY ingredientName REFERENCES Ingredient(name)
            ON DELETE CASCADE
            ON UPDATE CASCADE)

CREATE TABLE MealPlanContainsRecipe
        (mealPlanID INTEGER,
        recipeID INTEGER,
        PRIMARY KEY (mealPlanID, recipeID),
        FOREIGN KEY mealPlanID REFERENCES MealPlan(mealPlanID),
        FOREIGN KEY recipeID REFERENCES Recipe(ID)
            ON DELETE CASCADE
            ON UPDATE CASCADE)
```

```
CREATE TABLE Ingredient
        (name VARCHAR PRIMARY KEY,
        foodGroup VARCHAR)

CREATE TABLE IngredientNutritionalInfo
        (name VARCHAR,
        calories INTEGER,
        fat INTEGER,
        protein INTEGER,
        PRIMARY KEY (name, calories),
        FOREIGN KEY name REFERENCES Ingredient(name))

CREATE TABLE MealPlan
        (mealPlanID INTEGER PRIMARY KEY,
        endDate DATE,
        startDate DATE NOT NULL,
        groceryListID INTEGER,
        FOREIGN KEY groceryListID REFERENCES groceryList(groceryListID))

CREATE TABLE GroceryList
        (groceryListID INTEGER PRIMARY KEY,
        totalPrice INTEGER)

CREATE TABLE GroceryListContainsIngredient
        (groceryListID INTEGER,
        ingredientName VARCHAR,
        PRIMARY KEY (groceryListID, ingredientName),
        FOREIGN KEY groceryListID REFERENCES GroceryList(groceryListID),
        FOREIGN KEY ingredientName REFERENCES Ingredient(name)
                ON DELETE CASCADE
                ON UPDATE CASCADE)
```

8. INSERT statements to populate each table with at least 5 tuples.

```
INSERT INTO
        Allergy (“type”)
VALUES
        (“Strawberry”),
        (“Tree Nuts”),
        (“Dairy”),
        (“Shellfish”),
        (“Eggs”);

INSERT INTO
        UserHasAllergy (allergyType, userID, severity)
VALUES
        (“Dairy”, 4, “Severe”),
        (“Strawberry”, 5, “Mild”),
        (“Eggs”, 4, “Mild”),
        (“Dairy”, 3, “Severe”),
        (“Shellfish”, 1, “Mild”);

INSERT INTO
        User (userID, fullName, country, userPrefID, cuisine, diet, groceryStore)
VALUES
        (1, “Alex Dart”, “Canada”, 1, “Standard”, NULL, “Save-On Kerrisdale”),
        (2, “Griffin Velichko”, “Canada”, 2, “Standard”, NULL, “Save-On Dunbar”),
        (3, “Anna Friesen”, “Canada”, 3, “Standard”, NULL, “Save-On Wesbrook”),
        (4, “LeBron James”, “USA”, 4, “Standard”, “Vegan”, “Save-On Kerrisdale”),
        (5, “Grant Sanderson”, “USA”, 5, “Standard”, NULL, “Save-On Dunbar”);

INSERT INTO
        PremiumUser (userID, nutritionalAdvisorName)
VALUES
        (1, “Gordon Ramsay”),
        (2, “Gordon Ramsay”),
        (3, “Gordon Ramsay”);

INSERT INTO
        StandardUser (userID, mealPlanLimit)
VALUES
        (5, 10);
INSERT INTO
        BudgetUser (userID, studentDiscount)
VALUES
        (4, 0.15);
```

```
INSERT INTO
	userCreatesMealPlan (userID, mealPlanID)
VALUES
	(1, 1),
	(4, 2),
	(2, 3),
	(4, 4),
	(3, 5);

INSERT INTO
	Rating (ratingID, overallRating, difficultyRating, userID, recipeID)
VALUES
	(1, 4, 3, 1, 2),
	(2, 5, 2, 3, 1),
	(3, 5, 3, 2, 1),
	(4, 3, 4, 5, 4),
	(5, 2, 5, 2, 3);

INSERT INTO
	Equipment (equipmentID, equipmentType)
VALUES
	(1, “Large Knife”),
	(2, “Frying Pan”),
	(3, “Oven”),
	(4, “Spoon”),
	(5, “Mixing Bowl”);

INSERT INTO
	EquipmentLocations (equipmentType, equipmentLocation)
VALUES
	(“Large Knife”, “Knife Block”),
	(“Frying Pan”, “Cabinet”),
	(“Oven”, “Kitchen”),
	(“Spoon”, “Drawer”),
	(“Mixing Bowl”, “Cabinet”);

INSERT INTO
	Recipe (ID, name, author)
VALUES
	(1, “Fried Chicken”, “Anthony Bourdain”),
	(2, “Pancakes”, “Aunt Jemimah”),
	(3, “Ratatouille”, “Remy from Ratatouille”),
	(4, “Shrimp Fried Rice”, “A Shrimp”),
```

```
        (5, “Chili”, “Alex Dart”);

INSERT INTO
        RecipeHasIngredient (recipeID, ingredientName, quantity)
VALUES
        (1, “Chicken”, 5),
        (2, “Pancake Mix”, 3),
        (3, “Onion”, 2),
        (4, “Shrimp”, 3),
        (5, “Ground Beef”, 5),
        (1, “Oil”, 10),
        (2, “Water”, 3),
        (3, “Tomato”, 2),
        (4, “Rice”, 5),
        (5, “Diced Tomato”, 7);

INSERT INTO
        MealPlanContainsRecipe (mealPlanID, recipeID)
VALUES
        (1, 1),
        (1, 2),
        (1, 3),
        (2, 2),
        (3, 5);

INSERT INTO
        Ingredient (name, foodGroup)
VALUES
        (“Chicken”, “Meat”),
        (“Pancake Mix”, “Assorted”),
        (“Onion”, “Vegetable”),
        (“Shrimp”, “Seafood”),
        (“Ground Beef”, “Meat”),
        (“Oil”, “Liquids”),
        (“Water”, “Liquids”),
        (“Tomato”, “Vegetable”),
        (“Rice”, “Starches”),
        (“Diced Tomato”, “Vegetable”);

INSERT INTO
        IngredientNutritionalInfo (name, calories, fat, protein)
VALUES
        (“Chicken”, 100, 12, 15),
        (“Pancake Mix”, 150, 22, 2),
```

```
        (“Onion”, 75, 0, 1),
        (“Shrimp”, 125, 17, 13),
        (“Ground Beef”, 135, 10, 22),
        (“Oil”, 430, 26, 1),
        (“Water”, 0, 0, 0),
        (“Tomato”, 55, 1, 0),
        (“Rice”, 175, 13, 2),
        (“Diced Tomato”, 55, 1, 0);

INSERT INTO
        MealPlan (mealPlanID, endDate, startDate, groceryListID)
VALUES
        (1, 2024-10-14, 2024-10-21, 1),
        (2, 2024-10-14, 2024-10-21, 2),
        (3, 2024-10-14, 2024-10-21, 3),
        (4, 2024-10-17, 2024-10-24, 4),
        (5, 2024-10-19, 2024-10-26, 5);

INSERT INTO
        GroceryList (groceryListID, totalPrice)
VALUES
        (1, 21),
        (2, 33),
        (3, 24),
        (4, 56),
        (5, 13);

INSERT INTO
        GroceryListContainsIngredient (groceryListID, ingredientName)
VALUES
        (1, “Chicken”),
        (1, “Oil”),
        (2, “Pancake Mix”),
        (2, “Water”),
        (3, “Onion”),
        (3, “Tomato”),
        (4, “Shrimp”),
        (4, “Rice”),
        (5, “Ground Beef”);
```