University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 2024/03/01

Group Number: 21

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Alex Lee	4422902962 290296	al7031	alexmy31@gmail.com
Erica Buchanan	55077747	erica4	eeobuchanan@gmail.com
Brooklyn Cheng	68614932	bcheng7	brooklyncheng2002@gmail.com

University of British Columbia, Vancouver
Department of Computer Science

Project Description

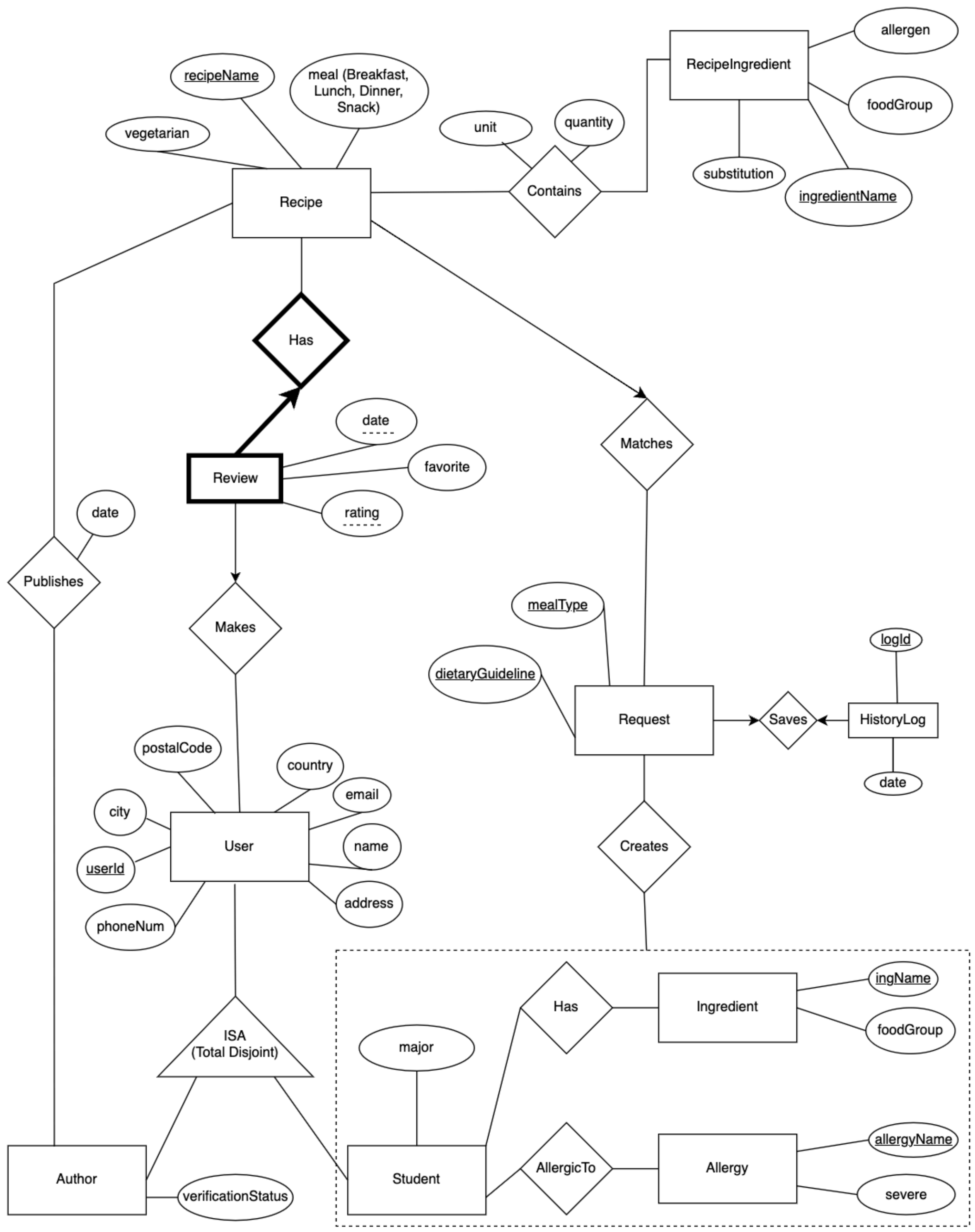
Brief Description

The domain of the application is a need-based cookbook for university students. We will be creating a cooking database for university students with recipes that they can query for, given the ingredients they have. These recipes are also able to be reviewed and have their associated author so that the students can have all of the information they need before deciding on their recipe.

ERD Updates (see diagram below)

- Weak entity changed as recipe ingredient was not a valid weak entity (it could be identified without the parent key, as indicated by the TA)
- Attributes added and altered in User, Request, RecipeIngredient, Author for FDs and to make more sense
- HistoryLog updated to have a one-to-one relationship with the request it is saving
- studentId changed to be major as we already had a way to identify the student uniquely (userId)
- Attributes renamed to be lowercase for relational schema

Updated ER Diagram



Relational Schema

LEGEND: PK underlined, FK bolded, CK italicized
Note: Primary keys are a subset of candidate keys and thus are not italicized. Primary keys are also already NON NULL, so those are not stated.

RecipeMatches(recipeName: VARCHAR(50), vegetarian: BIT, meal: VARCHAR(50), **mealType**: VARCHAR(50), **dietaryGuideline**: VARCHAR(50))
- Combined w/ Matches relationship table

RecipeIngredient(ingredientName: VARCHAR(50), foodGroup: VARCHAR(50), allergen: BIT, substitution: VARCHAR(50))

Contains(**ingredientName**: VARCHAR(50), **recipeName**: VARCHAR(50), unit: VARCHAR(50), quantity: INTEGER)
- unit and quantity are NOT NULL

ReviewMakesHas(date: DATE, rating: INTEGER, **recipeName**: VARCHAR(50), **userId**: VARCHAR(50), favorite: BIT)
- combined with Makes and Has relationship tables

RequestSaves(mealType: VARCHAR(50), dietaryGuideline: VARCHAR(50), logId: VARCHAR(50), date: DATE)
- logId is UNIQUE
- HistoryLog table combined w/ Saves relationship table

User(userId: VARCHAR(50), city: VARCHAR(50), *email*: VARCHAR(50), name: VARCHAR(50), postalCode: VARCHAR(50), address: VARCHAR(50), country: VARCHAR(50), phoneNum: VARCHAR(50))
- email and name are NOT NULL
- email is unique

Author(**userId**: VARCHAR(50)**,** verificationStatus: BIT)

Student(**userId**: VARCHAR(50), major: VARCHAR(50))

Ingredient(ingName: VARCHAR(50), foodGroup: VARCHAR(50))

Allergy(allergyName: VARCHAR(50), severe: BIT)

Publishes(**userId**: VARCHAR(50), **recipeName**: VARCHAR(50), date: DATE)

Has(**userId**: VARCHAR(50), **ingName**: VARCHAR(50))

AllergicTo(**userId**: VARCHAR(50), **allergyName**: VARCHAR(50))

Creates(**userId**: VARCHAR(50), **ingName**: VARCHAR(50), **allergyName**: VARCHAR(50), **mealType**: VARCHAR(50), **dietaryGuideline**: VARCHAR(50))

Functional Dependencies

Contains:

- FDs: ingredientName, recipeName -> unit, quantity
- Keys are ingredientName and recipeName, therefore Contains is in BCNF

ReviewMakesHas:

- FDs: date, rating, recipeName ➝ userId, favorite
- Keys are date, rating, and recipeName therefore ReviewMakesHas in BCNF

RequestSaves:

- FDs: mealType, dietaryGuideline ➝ logId, date
- Keys are mealType and dietaryGuideline therefore RequestSaves in BCNF

Allergy:

- FDs: allergyName ➝ severe
- Key is allergyName, therefore Allergy is in BCNF

Author:

- FDs: userId ➝ verificationStatus
- Key is verificationStatus, therefore Author is in BCNF

Student:

- FDs: userId ➝ major
- Key is userId, therefore Student is in BCNF

Student:

- FDs: userId ➝ major
- Key is userId, therefore Student is in BCNF

Ingredient:

- FDs: ingName ➝ foodGroup
- Key is ingname, therefore Ingredient is in BCNF

Publishes:

- FDs: userId, recipeName ➝ date
- Key is userId, therefore recipeName is in BCNF

Creates, AllergicTo and Has:
Only trivial cases, so we have not included them.

RecipeIngredient:

- FDs:
    - ingredientName -> foodGroup, allergen
    - ingredientName, allergen -> substitution
- Keys:
    - ingredientName$^+$ = {ingredientName, foodGroup, allergen, substitution}
    - ingredientName is the CK (and chosen PK) for this relationship, which means that ingredientName, allergen is a superkey. This relationship is in BCNF.

RecipeMatches:

- FDs:
    - recipeName → vegetarian, meal, mealType, dietaryGuideline
    - mealType → meal
- Keys:
    - recipeName$^+$ = {recipeName, vegetarian, meal, mealType, dietaryGuideline}
    - mealType$^+$ = {mealType, meal}
- The candidate key (and chosen PK) is recipeName.

The following FD violates BCNF, meaning RecipeMatches is not in BCNF:
mealType → meal where mealType$^+$ = {mealType, meal}

User:

- FDs:
    - userId → name, address, phoneNum, email
    - email → userId
    - postalCode → city, country
    - address → postalCode
    - phoneNum → country
- Keys:
    - userId$^+$ = {userId, name, address, phoneNum, country, postalCode, city, email}

- email^+ = {email, userId, name, address, phoneNum, country, postalCode, city}
- The candidate keys are userId and email, but based on our ERD we know userId is the chosen PK.

The following FDs violate 3NF, meaning User is not in 3NF:

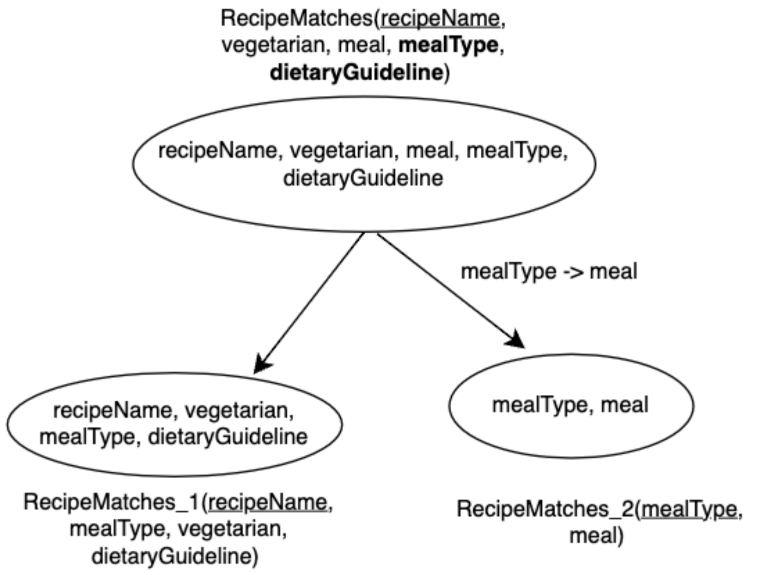
postalCode^+ = {postalCode, city, country}
address^+ = {address, postalCode, city, country}
phoneNum^+ = {phoneNum, country}

Normalization

In the previous step, we determined that RecipeMatches, User and RecipeIngredient are not in BCNF, so we will be normalizing them.

RecipeMatches:
Given schema:
RecipeMatches(recipeName, vegetarian, meal, **mealType**, **dietaryGuideline**)

FDs:
recipeName → vegetarian, meal, mealType, dietaryGuideline
mealType → meal

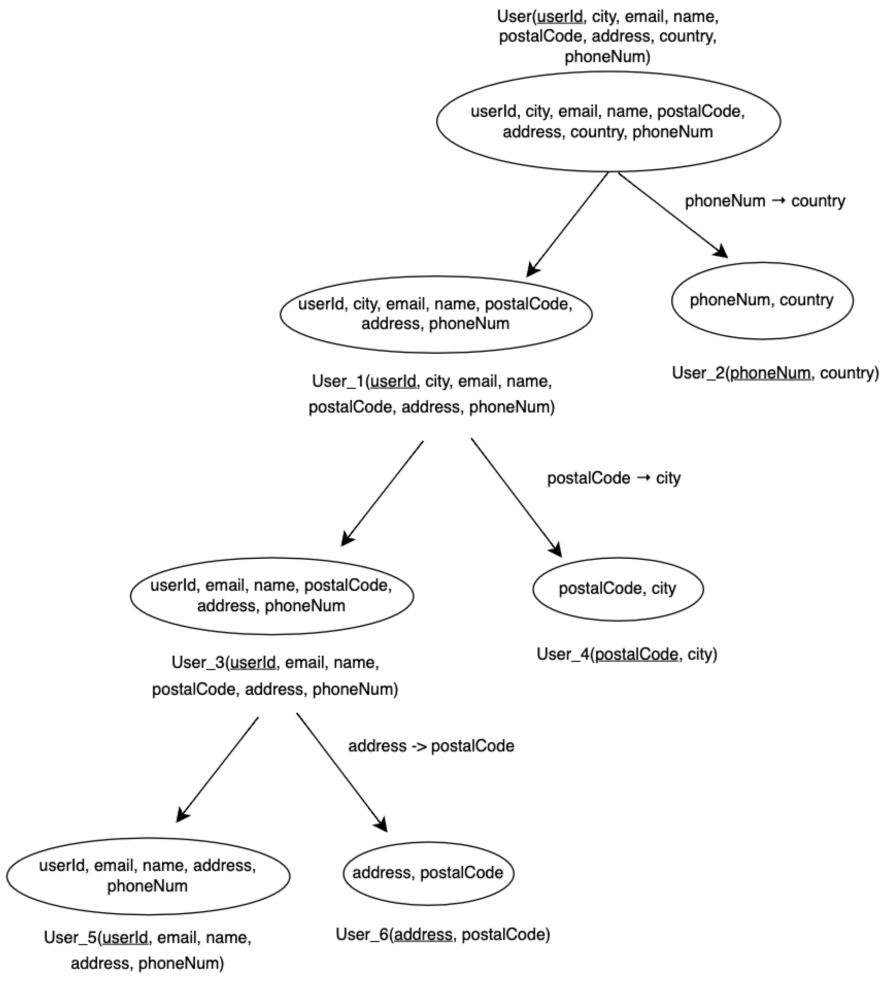

RecipeMatches(recipeName, vegetarian, meal, mealType, dietaryGuideline)
recipeName, vegetarian, meal, mealType, dietaryGuideline
mealType -> meal
recipeName, vegetarian, mealType, dietaryGuideline
mealType, meal
RecipeMatches_1(recipeName, mealType, vegetarian, dietaryGuideline)
RecipeMatches_2(mealType, meal)

BCNF normalized schemas:
RecipeMatches_1(recipeName, vegetarian, **mealType**, **dietaryGuideline**)
RecipeMatches_2(mealType, meal)

User:
Given schema:
User(userId, city, email, name, postalCode, address, country, phoneNum)

FDs:
userId → email, name, address, phoneNum
postalCode → city, country

address → postalCode
phoneNum → country


User(userId, city, email, name, postalCode, address, country, phoneNum)
userId, city, email, name, postalCode, address, country, phoneNum
phoneNum → country
userId, city, email, name, postalCode, address, phoneNum
phoneNum, country
User_1(userId, city, email, name, postalCode, address, phoneNum)
User_2(phoneNum, country)
postalCode → city
userId, email, name, postalCode, address, phoneNum
postalCode, city
User_3(userId, email, name, postalCode, address, phoneNum)
User_4(postalCode, city)
address -> postalCode
userId, email, name, address, phoneNum
address, postalCode
User_5(userId, email, name, address, phoneNum)
User_6(address, postalCode)

BCNF normalized schemas:
User_2(**phoneNum**, country)
User_4(**postalCode**, city)
User_5(userId, email, name, address, phoneNum)
User_6(**address**, postalCode)

SQL DDL CREATE Statements

```
CREATE TABLE RequestSaves (
   mealType                VARCHAR(50),
   dietaryGuideline        VARCHAR(50),
   logId                   VARCHAR(50)    UNIQUE,
   date                    DATE,
   PRIMARY KEY (mealType, dietaryGuideline)
);

CREATE TABLE RecipeMatches_1 (
   recipeName              VARCHAR(50)   PRIMARY KEY,
   vegetarian              BIT,
   mealType                VARCHAR(50),
   dietaryGuideline        VARCHAR(50),
   FOREIGN KEY (mealType, dietaryGuideline) REFERENCES RequestSaves
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE RecipeMatches_2 (
   mealType                VARCHAR(50)    PRIMARY KEY,
   meal                    VARCHAR(50),
   FOREIGN KEY (mealType) REFERENCES RecipeMatches_1
         ON DELETE CASCADE
         ON UPDATE CASCADE
);

CREATE TABLE RecipeIngredient (
   ingredientName          VARCHAR(50)   PRIMARY KEY,
   foodGroup               VARCHAR(50),
   allergen                BIT,
   substitution            VARCHAR(50),
);
```

since contains is a keyword we have to put brackets around it in SQL statements

```
CREATE TABLE [Contains] (
   ingredientName          VARCHAR(50),
   recipeName              VARCHAR(50),
   unit                    VARCHAR(20)    NOT NULL,
   quantity                INTEGER        NOT NULL,
   PRIMARY KEY (ingredientName, recipeName),
   FOREIGN KEY (ingredientName) REFERENCES RecipeIngredient
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (recipeName) REFERENCES RecipeMatches_1
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE User_2 (
   phoneNum        VARCHAR(50)    PRIMARY KEY,
   country         VARCHAR(50)
   FOREIGN KEY (phoneNum) REFERENCES User_5
         ON DELETE CASCADE
         ON UPDATE CASCADE
);

CREATE TABLE User_4 (
   postalCode      VARCHAR(50)    PRIMARY KEY,
   city            VARCHAR(50)
   FOREIGN KEY (postalCode) REFERENCES User_6
         ON DELETE CASCADE
         ON UPDATE CASCADE
);

CREATE TABLE User_5 (
   userId          VARCHAR(50)    PRIMARY KEY,
   email           VARCHAR(50),
   name            VARCHAR(50),
   address         VARCHAR(50),
   phoneNum        VARCHAR(50)
);
```

```
CREATE TABLE User_6 (
   address              VARCHAR(50)                 PRIMARY KEY,
   postalCode           VARCHAR(50)
   FOREIGN KEY (address) REFERENCES User_5
         ON DELETE CASCADE
         ON UPDATE CASCADE
);

CREATE TABLE ReviewMakesHas (
   date                 DATE,
   rating               INTEGER,
   recipeName           VARCHAR(50),
   userId               VARCHAR(50),
   favorite             BIT,
   PRIMARY KEY (date, rating, recipeName),
   FOREIGN KEY (recipeName) REFERENCES RecipeMatches_1
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (userId) REFERENCES User_5
      ON DELETE CASCADE
      ON UPDATE CASCADE
);


CREATE TABLE Author (
   userId                      VARCHAR(50)    PRIMARY KEY,
   verificationStatus          BIT,
   FOREIGN KEY (userId) REFERENCES User_5
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE Student (
   userId               VARCHAR(50)    PRIMARY KEY,
   major                VARCHAR(50),
   FOREIGN KEY (userId) REFERENCES User_5
```

```
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE Ingredient (
   ingName          VARCHAR(50)    PRIMARY KEY,
   foodGroup        VARCHAR(50)
);

CREATE TABLE Allergy (
   allergyName      VARCHAR(50)    PRIMARY KEY,
   severity         INTEGER
);

CREATE TABLE Publishes (
   userId           VARCHAR(50),
   recipeName       VARCHAR(50),
   date             DATE,
   PRIMARY KEY (userId, recipeName),
   FOREIGN KEY (userId) REFERENCES Author
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (recipeName) REFERENCES RecipeMatches_1
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE Creates (
   userId                 VARCHAR(50),
   ingName                VARCHAR(50),
   allergyName            VARCHAR(50),
   mealType               VARCHAR(50),
   dietaryGuideline       VARCHAR(50),
   PRIMARY KEY (userId, mealType, dietaryGuideline, ingName, allergyName),
   FOREIGN KEY (userId) REFERENCES Student
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (mealType, dietaryGuideline) REFERENCES RequestSaves
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (ingName) REFERENCES Ingredient
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (allergyName) REFERENCES Allergy
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE Has (
   userId                    VARCHAR(50),
   ingName                   VARCHAR(50),
   PRIMARY KEY (ingName, userId),
   FOREIGN KEY (ingName) REFERENCES Ingredient
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (userId) REFERENCES Student
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE AllergicTo (
   userId                    VARCHAR(50),
   allergyName               VARCHAR(50),
   PRIMARY KEY (allergyName, userId),
   FOREIGN KEY (allergyName) REFERENCES Allergy
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (userId) REFERENCES Student
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

University of British Columbia, Vancouver
Department of Computer Science

SQL DDL INSERT Statements

1.

```
INSERT INTO RecipeMatches_1 VALUES ('Cheesecake', 1, 'Snack', 'Vegetarian');

INSERT INTO RecipeMatches_2 VALUES ('Snack', 'Snack);

INSERT INTO RecipeIngredient VALUES ('Philadelphia Original Brick Cream Cheese',
'Dairy', 1, '(Vegan) Nut-based Cream Cheese');

INSERT INTO ReviewMakesHas VALUES ('2024/02/29', 8, 'Cheesecake',
'stuas4df656a4f6as4f', 1);

INSERT INTO RequestSaves VALUES ('Snack', 'Vegetarian', '24df23sdaf4565fds3',
'2024/02/29');

INSERT INTO User_2 VALUES ('+1 (123) 456-7890', 'Canada');

INSERT INTO User_4 VALUES ('V8A6E5', 'Powell River');

INSERT INTO User_5 VALUES ('auth15sad1fa415df1', 'john-doe@gmail.com', 'John
Doe', '169 Smoky Hollow Ave', '+1 (123) 456-7890');

INSERT INTO User_6 VALUES ('169 Smoky Hollow Ave', 'V8A6E5');

INSERT INTO User_2 VALUES ('+1 (555) 123-4567', 'USA');

INSERT INTO User_4 VALUES ('12345', 'Springfield');

INSERT INTO User_5 VALUES ('stuas4df656a4f6as4f', 'john@example.com', 'John
Smith');

INSERT INTO User_6 VALUES ('170 Smoky Hollow Ave', '12345');

INSERT INTO Author VALUES (auth15sad1fa415df1', 1);
INSERT INTO Student VALUES (‘stuas4df656a4f6as4f', 'Computer Science');

INSERT INTO Ingredient VALUES ('Lactose-free Milk', 'Dairy');

INSERT INTO Allergy VALUES ('Lactose-Intolerance', 0);

INSERT INTO Publishes VALUES (auth15sad1fa415df1', 'Cheesecake', '2024/02/29');

INSERT INTO Creates VALUES ('stuas4df656a4f6as4f', 'Lactose-free Milk',
'Lactose-Intolerance', 'Snack', 'Vegetarian');

INSERT INTO Has VALUES ('stuas4df656a4f6as4f', 'Lactose-free Milk');

INSERT INTO AllergicTo VALUES ('Lactose-Intolerance', 'stuas4df656a4f6as4f');
```

2.

```
INSERT INTO RecipeMatches_1 VALUES ('Pizza', 0, 'Dinner', 'Meat');

INSERT INTO RecipeMatches_2 VALUES ('Dinner', 'Dinner');

INSERT INTO RecipeIngredient VALUES ('Pepperoni', 'Meat', 0, NULL);

INSERT INTO [Contains] VALUES ('Pepperoni', 'Pizza', 'Ounce', 6);

INSERT INTO ReviewMakesHas VALUES ('2024/03/01', 7, 'Pizza',
'stu2666653a45ec23', 0);

INSERT INTO RequestSaves VALUES ('Dinner', 'Meat', '25d563sdaf4565fds3',
'2024/02/29');

INSERT INTO User_2 VALUES ('+1 (123) 845-7890', 'United States');

INSERT INTO User_4 VALUES ('75115', 'Everett');

INSERT INTO User_5 VALUES ('stu2666653a45ec23', 'jane-doe@gmail.com', ‘Jane
Doe’, ‘249 W. Hawthorne Court Desoto’, ‘+1 (123) 845-7890’);
```

```
INSERT INTO User_6 VALUES ('249 W. Hawthorne Court Desoto', '75115');

INSERT INTO User_2 VALUES ('+1 (123) 845-8890', 'United States');

INSERT INTO User_4 VALUES ('75135', 'Everett');

INSERT INTO User_5 VALUES ('auth2666653a45ec23', 'jane-doe_2@gmail.com', ‘Jane
Doe’, ‘249 W. Hawthorne Court Desoto’, ‘+1 (123) 845-8890’);

INSERT INTO User_6 VALUES ('249 W. Hawthorne Court Desoto', '75135');

INSERT INTO Author VALUES ('auth2666653a45ec23', 0);

INSERT INTO Student VALUES ('stu2666553a45ec23', 'Statistics');

INSERT INTO Ingredient VALUES ('Cheese', 'Dairy');

INSERT INTO Allergy VALUES ('Lactose-Intolerance', 2);

INSERT INTO Publishes VALUES ('auth2666653a45ec23', 'Pizza', '2024/03/01');

INSERT INTO Creates VALUES ('stu2666553a45ec23', 'Pepperoni',
'Lactose-Intolerance', 'Dinner', 'Meat');

INSERT INTO Has VALUES ('stu2666553a45ec23', 'Cheese');

INSERT INTO AllergicTo VALUES ('Lactose-Intolerance', 'stu2666553a45ec23');
```

3.

```
INSERT INTO RecipeMatches_1 VALUES ('Ham Sandwich', 0, 'Lunch', 'Meat');

INSERT INTO RecipeMatches_2 VALUES ('Lunch', 'Lunch');

INSERT INTO RecipeIngredient VALUES ('Ham', 'Meat', 1, 'Veggie ham');

INSERT INTO [Contains] VALUES ('Ham', 'Ham Sandwich', 'Slices', 12);
```

```sql
INSERT INTO ReviewMakesHas VALUES ('2023/12/15', 6, 'Ham Sandwich',
'stu2666653a45ae23', 0);

INSERT INTO RequestSaves VALUES ('Lunch', 'Meat', '25d589sdaf4565fds3',
'2023/12/15');

INSERT INTO User_2 VALUES ('+1 (123) 845-7660', 'United States');

INSERT INTO User_4 VALUES ('98105', 'Seattle');

INSERT INTO User_5 VALUES ('stu2666653a45ae23', 'alan-wake@gmail.com', ‘Alan
Wake’, ‘4264 Union Street’, '+1 (123) 845-7660');

INSERT INTO User_6 VALUES (‘4264 Union Street’, '98105');

INSERT INTO User_2 VALUES ('+1 (123) 844-7660', 'United States');

INSERT INTO User_4 VALUES ('98505', 'Seattle');

INSERT INTO User_5 VALUES ('auth2666653a45ae23', 'alan-wake_2@gmail.com',
‘Alan Wake’, ‘4364 Union Street’, '+1 (123) 844-7660');

INSERT INTO User_6 VALUES (‘4364 Union Street’, '98505');

INSERT INTO Author VALUES ('auth2666653a45ae23', 1);

INSERT INTO Student VALUES ('stu2666653a45ae23', 'Creative Writing');

INSERT INTO Ingredient VALUES ('Bread', 'Grains');

INSERT INTO Allergy VALUES ('Eggs', 5);

INSERT INTO Publishes VALUES ('auth2666653a45ae23', 'Ham Sandwich',
'2024/12/15');

INSERT INTO Creates VALUES ('stu2666653a45ae23', 'Ham', 'Eggs', 'Lunch', 'Meat');
```

INSERT INTO Has VALUES ('stu2666653a45ae23', 'Bread');

INSERT INTO AllergicTo VALUES ('Eggs', 'stu2666653a45ae23');

4.
INSERT INTO RecipeMatches_1 VALUES ('Taiyaki', 1, 'Snack', 'Vegetarian');

INSERT INTO RecipeMatches_2 VALUES ('Snack', 'Snack');

INSERT INTO RecipeIngredient VALUES ('Red beans', 'Legumes’, 1, NULL);

INSERT INTO [Contains] VALUES ('Red bean Paste', 'Taiyaki', 'Grams', 250);

INSERT INTO ReviewMakesHas VALUES ('2023/11/14', 9, 'Taiyaki', 'stu2666653a45ae23', 1);

INSERT INTO RequestSaves VALUES ('Snack', 'Vegetarian', '25d579sdaf4565fds3', '2023/11/14');

INSERT INTO User_2 VALUES (‘+1 (123) 845-7930’, 'United States');

INSERT INTO User_4 VALUES ('20019', 'Washington');

INSERT INTO User_5 VALUES ('stu2666863a45ec52', 'leon-s-kennedy@gmail.com', ‘Leon Scott Kennedy’, '427, Chaplin Street Southeast', ‘+1 (123) 845-7930’);

INSERT INTO User_6 VALUES ('427, Chaplin Street Southeast', '20019');

INSERT INTO User_2 VALUES (‘+1 (123) 849-7930’, 'United States');

INSERT INTO User_4 VALUES ('20119', 'Washington');

INSERT INTO User_5 VALUES ('auth2666863a45ec52', 'leon-s-kennedy@gmail.com', ‘Leon Scott Kennedy’, '427, Chaplin Street Southeast', ‘+1 (123) 849-7930’);

INSERT INTO User_6 VALUES ('427, Chaplin Street Southeast', '20119');

```
INSERT INTO Author VALUES ('auth2666863a45ec52', 1);

INSERT INTO Student VALUES ('stu2666863a45ec52', 'Criminology');

INSERT INTO Ingredient VALUES ('Sugar', 'Carbohydrate');

INSERT INTO Allergy VALUES ('Shellfish', 8);

INSERT INTO Publishes VALUES ('auth2666863a45ec52', 'Taiyaki', '2024/11/14');

INSERT INTO Creates VALUES ('stu2666863a45ec52', 'Red beans', 'Shellfish', 'Snack',
'Vegetarian');

INSERT INTO Has VALUES ('stu2666863a45ec52', 'Sugar');

INSERT INTO AllergicTo VALUES ('Shellfish', 'stu2666863a45ec52');
```

5.

```
INSERT INTO RecipeMatches_1 VALUES ('Gimbap', 1, 'Lunch', 'Vegetarian');

INSERT INTO RecipeMatches_2 VALUES ('Lunch', 'Lunch');

INSERT INTO RecipeIngredient VALUES ('Rice', 'Grains', 0, NULL);

INSERT INTO [Contains] VALUES ('Rice', 'Gimbap', 'Cups', 4);

INSERT INTO ReviewMakesHas VALUES ('2023/10/14', 7, 'Taiyaki',
'stu2666863a45ec52', 1);

INSERT INTO RequestSaves VALUES ('Snack', 'Vegetarian', '32d579sdaf4565fds3',
'2023/10/14');

INSERT INTO User_2 VALUES ('+1 (123) 845-7030', 'Canada');

INSERT INTO User_4 VALUES (‘V6B 3K9’, 'Vancouver');
INSERT INTO User_5 VALUES (auth18sad1fa4056d2', ‘cloud-strife@gmail.com’, 'Cloud
Strife', '2102 Robson St', '+1 (123) 845-7030');

INSERT INTO User_6 VALUES ('2102 Robson St', 'V6B 3K9');

INSERT INTO User_2 VALUES ('+1 (123) 845-7030', 'Canada');

INSERT INTO User_4 VALUES (‘V6B 3K9’, 'Vancouver');

INSERT INTO User_5 VALUES ('stuas4df677a4f6ba7f', ‘cloud-strife2@gmail.com’,
'Cloud Strife', '2102 Robson St', '+1 (123) 846-7030');

INSERT INTO User_6 VALUES ('2102 Robson St', 'V6B 3K9');

INSERT INTO Author VALUES ('auth18sad1fa4056d2', 1);

INSERT INTO Student VALUES ('stuas4df677a4f6ba7f', 'Astronomy');

INSERT INTO Ingredient VALUES ('Gim', 'Sea Vegetable');

INSERT INTO Allergy VALUES ('Peanut', 6);

INSERT INTO Publishes VALUES ('auth15s8d9ea4166d2', 'Gimbap', '2024/11/19');

INSERT INTO Creates VALUES ('stuas4df677a4f6b4vf', 'Gimbap', 'Peanut', 'Lunch',
'Vegetarian');

INSERT INTO Has VALUES ('stuas4df677a4f6b4vf', 'Gim');

INSERT INTO AllergicTo VALUES ('Peanut', 'stuas4df677a4f6b4vf');
```