# CPSC 304 Project Cover Page

Milestone #: 2

Date: October 20, 2022

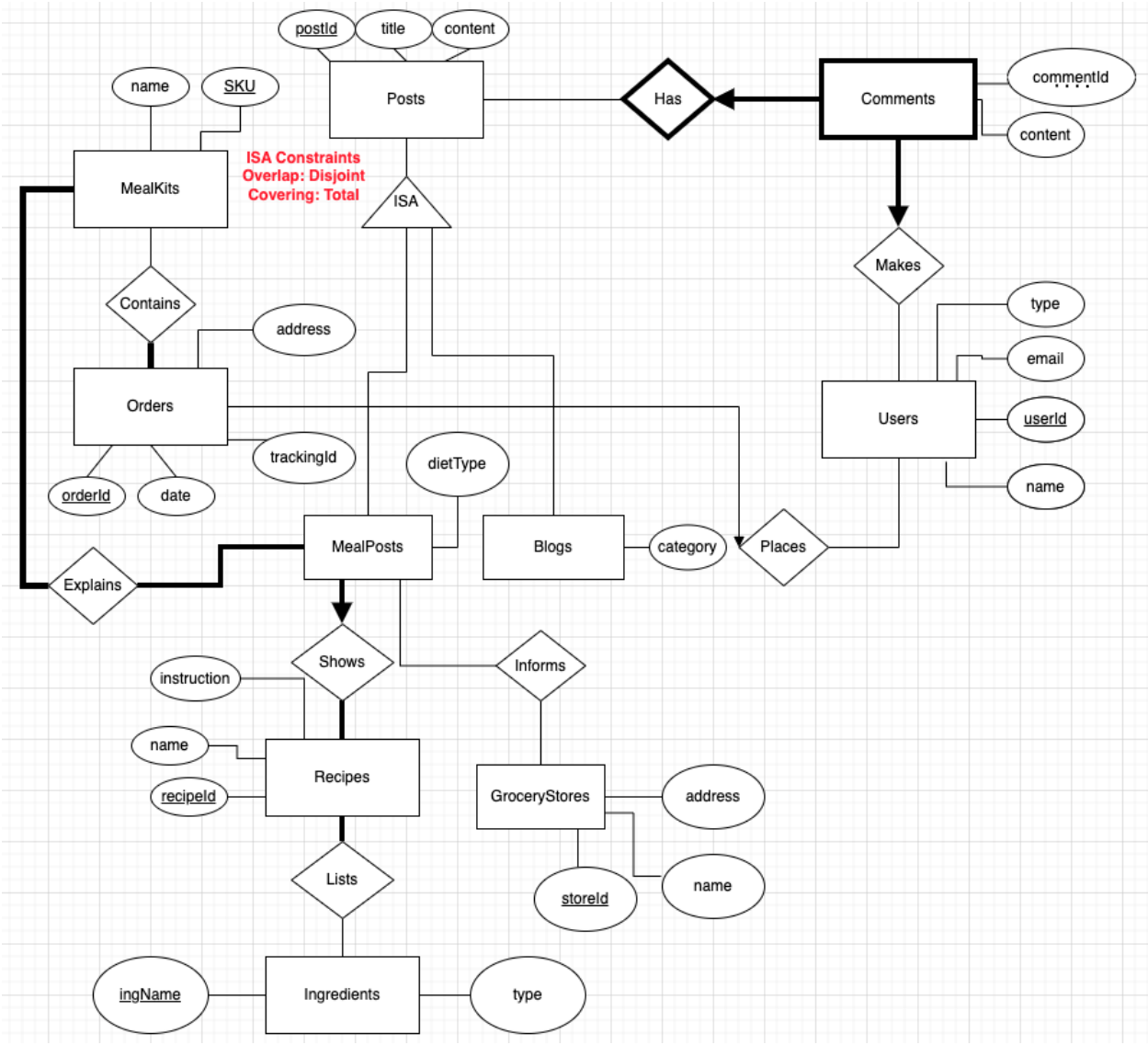
Group Number: 25

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Heeseop An | 78513967 | t3r1y | heeseop.an@gmail.com |
| Jeremy Hay | 31764715 | l7a3b | hayjer000@gmail.com |
| Dabin Im | 20780946 | b7c7l | ldb1216@gmail.com |

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. ER diagram and changes




- Key constraint(Ingredients to Recipes) changed from many-to-one to many-to-many
  - This allows the same Ingredient to be in multiple Recipes, which makes more sense than each Ingredient only being able to be in one Recipe
- Key constraint(MealPosts to Recipes) changed from one-to-one to many-to-one
  - This allows each Recipe to be featured in multiple MealPosts. There is not any requirement that each Recipe only be featured in one MealPost, so this is fine
- Key constraint(MealKits to Orders) changed from many-to-one to many-to-many
  - Before, each Order could have multiple MealKits, but each MealKit could only be contained in one Order. We changed this so that the same MealKit (corresponding to the same 'product') can be ordered multiple times.
- Entity name changed from OffineStores to GroceryStores
  - (For clarity)
- Relationship(searches) removed between Users and Posts

- This relationship didn't have a clear purpose, so we removed it
- Attribute(instruction) moved from MealKits to Recipes
  - The instructions were intended to be associated with a specific Recipe, and so moving this attribute makes sense.
- Attributes(email and type) added to Users
  - Users now have an email associated with their account, as well as a type (business or personal)
- Attributes(trackingId and address) added to Orders
  - Each Order is now associated with a specific trackingId, which determines the delivery address
- Attribute name changed: (Comments)ID => commentId, (Users)ID => userId, (Posts)ID => postId, (Orders)ID => orderId, (Recipes)ID => recipeId, (Ingredients)name => ingName
  - Changed for FD and general clarity

## 3. Schema

- Users(<u>userId</u>: INT, name: CHAR(100), type: CHAR(100), email: CHAR(100))
  - CK: userId
- Comments(**<u>postId</u>**: INT, <u>commentId</u>: INT, content: CHAR(2000), **userId**: INT)
  - CK: {postId, commentId}
  - **userId is NOT NULL**
- Posts(<u>postId</u>: INT, title: CHAR(200), content: CHAR(2000))
  - CK: postId
  - ***Constraint: MealPosts and Blog are disjoint. Require assertions to implement this constraint.**
- MealPosts(**<u>postId</u>**: INT, dietType: CHAR(100), **recipeId**: INT)
  - CK: postId
  - **recipeId is NOT NULL**
- Blogs(**<u>postId</u>**: INT, category: CHAR(100))
  - CK: postId
- MealKits(<u>SKU</u>: INT, name: CHAR(100))
  - CK: SKU, name
- Explains(**<u>SKU</u>**: INT, **<u>postId</u>**: INT)
  - CK: SKU, postId
  - ***Constraint: many-to-many in total participation. Require assertions to implement this constraint.**
- Contains(**<u>SKU</u>**: INT, **<u>orderId</u>**: INT)
  - CK: SKU, orderId
- Orders(<u>orderId</u>: INT, date: DATE, address: CHAR(100), trackingId: CHAR(100), **userId**: INT)
  - CK: orderId
- Recipes(<u>recipeId</u>: INT, name: CHAR(100), instruction: CHAR(2000))
  - CK: recipeId
- Lists(**<u>recipeId</u>**: INT, **<u>ingName</u>**: CHAR(100))
  - CK: recipeId
- Ingredients(<u>ingName</u>: CHAR(100), type: CHAR(100))

- CK: ingName
- GroceryStores(storeId: INT, name: CHAR(100), address: CHAR(100))
  - CK: store Id, address
- Informs(**postId**: INT, **storeId**: INT)
  - CK: postId, storeId

## 4. Functional Dependencies

- Users(userId: INT, name: CHAR(100), type: CHAR(100), email: CHAR(100))
  - userId -> name, type, email
  - email -> type
- Comments(**postId**: INT, commentId: INT, content: CHAR(2000), **userId**: INT)
  - {postId, commentId} -> content, userId
- Posts(postId: INT, title: CHAR(200), content: CHAR(2000))
  - postId -> title, content
- MealPosts(**postId**: INT, dietType: CHAR(100), **recipeId**: INT)
  - postId -> dietType, recipeId
- Blogs(**postId**: INT, category: CHAR(100))
  - postId -> category
- MealKits(SKU: INT, name: CHAR(100))
  - SKU -> name
  - Name -> SKU
- Explains(**SKU**: INT, **postId**: INT)
  - SKU -> postId
  - postId -> SKU
- Contains(**SKU**: INT, **orderId**: INT)
  - SKU -> orderId
  - orderId -> SKU
- Orders(orderId: INT, date: DATE, address: CHAR(100), trackingId: CHAR(100), **userId**: INT)
  - orderId -> date, userId, address, trackingId
  - trackingId -> address
- Recipes(recipeId: INT, name: CHAR(100), instruction: CHAR(2000))
  - recipeId -> name, instruction
- Informs(**postId**: INT, **storeId**: INT)
  - postId -> storeId
  - storeId -> postId
- Lists(**recipeId**: INT, **ingName**: CHAR(100))
  - recipeId -> ingName
- Ingredients(ingName: CHAR(100), type: CHAR(100))
  - ingName -> type
- GroceryStores(storeId: INT, name: CHAR(100), address: CHAR(100))
  - storeId -> name, address

- address -> storeId, name

## 5. Normalization

- Users(<u>userId</u>, name, type, email)
  - FD:
    - userId -> name, type, email
    - email -> type
  - In FD email -> type, email is not a superkey, so we decompose to Users1(<u>email</u>, type) and Users2(<u>userId</u>, name, email)
  - email is a key and CK for Users1, thus it is in BCNF
  - userId is a key and CK for Users2, thus it is in BCNF
  - final answer: Users1(<u>email</u>, type), Users2(<u>userId</u>, name, email)
- Orders(<u>orderId</u>, date, address, trackingId, **userId**)
  - FD:
    - orderId -> date, userId, address, trackingId
    - trackingId -> address
  - In FD trackingId -> address, trackingId is not a superkey, so we decompose to Orders1(<u>trackingId</u>, address) and Orders2(<u>orderId</u>, date, trackingId, **userId**)
  - trackingId is a key and CK for Orders1, thus it is in BCNF
  - orderId is a key and CK for Orders2, thus it is in BCNF
  - final answer: Orders1(<u>trackingId</u>, address), Orders2(<u>orderId</u>, date, trackingId, **userId**)
- All other relations do not need normalization since they are already in BCNF(or at least 3NF)

## 6. The SQL DDL statements

```
CREATE TABLE User1
        (email   CHAR(100)       PRIMARY KEY,
         type    CHAR(100));

CREATE TABLE User2
        (userId  INT      PRIMARY KEY,
         name    CHAR(100),
         email   CHAR(100));

CREATE TABLE Order1
        (trackingId      CHAR(100)      PRIMARY KEY,
         address         CHAR(100));

CREATE TABLE Ingredient
        (ingName         CHAR(100)      PRIMARY KEY,
         type            CHAR(100));
```

```
CREATE TABLE GroceryStore
        (storeId        INT             PRIMARY KEY,
         name           CHAR(100),
         address        CHAR(100));

CREATE TABLE Post
        (postId         INT             PRIMARY KEY,
         title          CHAR(200),
         content        TEXT);

CREATE TABLE Blog
        (postId         INT     PRIMARY KEY,
         title          CHAR(200),
         content        TEXT,
         category       CHAR(100));

CREATE TABLE MealKit
        (SKU    INT     PRIMARY KEY,
         name   CHAR(100));

CREATE TABLE Recipe
        (recipeId       INT             PRIMARY KEY,
         name           CHAR(100),
         instruction    TEXT);


CREATE TABLE Order2
        (orderId        INT             PRIMARY KEY,
         trackingId     CHAR(100)       UNIQUE,
         userId         INT             UNIQUE,
         date           DATE,
         FOREIGN KEY (userId) REFERENCES User2(userId)
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE Comments
        (postId         INT,
         commentId      INT,
         content        TEXT,
         userId         INT     NOT NULL,
         PRIMARY KEY (postId, commentId),
```

```
        FOREIGN KEY (userId) REFERENCES User2(userId)
                ON UPDATE CASCADE);

CREATE TABLE MealPosts
        (postId         INT             PRIMARY KEY,
         title          CHAR(200),
         content        TEXT,
         dietType       CHAR(100),
         recipeId       INT NOT NULL,
         FOREIGN KEY (recipeId) REFERENCES Recipe(recipeId)
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE Explains
        (SKU    INT,
         postId INT,
         PRIMARY KEY (SKU, postId),
         FOREIGN KEY (SKU) REFERENCES MealKit(SKU)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (postId) REFERENCES MealPosts(postId)
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE Contain
        (SKU            INT,
         orderId        INT,
         PRIMARY KEY (SKU, orderId),
         FOREIGN KEY (SKU) REFERENCES MealKit(SKU)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (orderId) REFERENCES Order2(orderId)
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE List
        (recipeId       INT,
         ingName        CHAR(100),
         PRIMARY KEY (recipeId, ingName),
         FOREIGN KEY (recipeId) REFERENCES Recipe(recipeId)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
```

```
        FOREIGN KEY (ingName) REFERENCES Ingredient(ingName)
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE Inform
        (postId  INT,
         storeId INT,
         PRIMARY KEY (postId, storeId),
         FOREIGN KEY (postId) REFERENCES MealPosts(postId)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (storeId) REFERENCES GroceryStore(storeId)
                ON DELETE CASCADE
                ON UPDATE CASCADE);
```

## 7. INSERT statements

```
INSERT
INTO User1 (email, type)
VALUES
('aaa@ubc.ca', 'business')
, ('bbb@ubc.ca', 'business')
, ('ccc@ubc.ca', 'business')
, ('ddd@gmail.com', 'personal')
, ('eee@ubc.ca', 'business');

INSERT
INTO User2 (userId, name, email)
VALUES
(1, 'Bob', 'aaa@ubc.ca')
, (2, 'Juan', 'bbb@ubc.ca')
, (3, 'John', 'ccc@ubc.ca')
, (4, 'Dave', 'ddd@gmail.com')
, (5, 'Jenny', 'eee@ubc.ca');

INSERT
INTO Order1 (trackingId, address)
VALUES
('T10222022115657AAA', '123 Granvile Street, Vancouver, BC, Canada ')
, ('T10222022115700BBB', '4057 Kingsway St, Burnarby, BC, Canada')
, ('T10222022120711CCC', '958 Robson St, Vancouver, BC,Canada')
, ('T10222022115800DDD', '1159 Knight St, Vancouver, BC, Canada')
```

```
, ('T10222022131742EEE', '9 Genie St, Vancouver, BC, Canada');



INSERT
INTO Ingredient(ingName, type)
VALUES
('Onion', 'vegetable')
, ('Potato', 'vegetable')
, ('Chicken', 'meat')
, ('Salmon', 'Seafood')
, ('Broccoli', 'vegetable')
, ('Banana', 'fruit');

INSERT
INTO GroceryStore(storeId, name, address)
VALUES
(1, 'Walmart', 'Downtown')
, (2, 'Super Store', 'Richmond')
, (3, 'Costco', 'Downtown')
, (4, 'Walmart', 'Burnaby')
, (5, 'NoFrills', 'West Point Grey')
, (6, 'K-Mart', 'University St');

INSERT
INTO Post(postId, title, content)
VALUES
(1, 'Korean Chicken', 'Soooo delicious!')
, (2, 'Korean Soup', 'Amazing taste!')
, (3, 'Salad', 'So Sweet!')
, (4, 'Japanese Ramen', 'Real Japanese Ramen!')
, (5,  'Pizza', 'You will be addicted!')
, (6,  'Happy Thanksgiving Day!', 'We announce you that we have big bargain sale!')
, (7, 'Tasting Korean Ramen', 'Go to Downtown!')
, (8, 'Amazing Korean Seafood noodle soup', 'Go to burnaby!')
, (9, 'Fresh Apple Salad', 'Go to any market!')
, (10, 'Go to Vietnam Noodle!', 'Go to main street!')
, (11,  'Cabbage Pancake!', 'Go to grocery store!');

INSERT
INTO Blog(postId, title, content, category)
VALUES
```

```
(7, 'Tasting Korean Ramen', 'Go to Downtown!', 'fun')
, (8, 'Amzaing Korean Seafood noodle soup', 'Go to burnaby!', 'road trip')
, (9, 'Fresh Apple Salad', 'Go to any market!', 'fun')
, (10, 'Go to Vietnam Noodle!', 'Go to main street!', 'fun')
, (11,  'Cabbage Pancake!', 'Go to grocery store!', 'trip');

INSERT
INTO MealKit(SKU, name)
VALUES
(1, 'Korean Chicken Original')
, (2, 'Amazing Korean Soup Special')
, (3, 'Addicted Fresh Salad')
, (4, 'Traditional Japanese Ramen')
, (5, 'Super Delicious Pizza');

INSERT
INTO Recipe(recipeId, name, instruction)
VALUES
(1, 'Korean Spicy Chicken', '1. adds chicken brast with coconut milk…')
, (2, 'Korean Pork Kimchi soup', '1. adds water into a pot…')
, (3, 'Banana Salad', '1. fills out banana…')
, (4, 'Expert Soy Ramen', '1. adds water into a pot…')
, (5, 'Health Food Boiled Broccoli', '1. adds water into a pot…')
, (6, 'Korean Honey butter Chicken', '1. adds chicken brast with coconut milk…');

INSERT
INTO Order2 (orderId, trackingId, userId, date)
VALUES
(1, 'P10222022115657AAA', 1, '2022-10-20')
, (2, 'P10222022115700BBB', 2, '2022-10-20')
, (3, 'P10222022120711CCC', 3, '2022-10-20')
, (4, 'P10222022115800DDD', 4, '2022-10-20')
, (5, 'P10222022131742EEE', 5, '2022-10-20');

INSERT
INTO Comments (postId, commentId, content, userId)
VALUES
(1, 1,  'GOOOD', 1)
, (2, 1, 'Delicious', 1)
, (3, 1, 'So so', 1)
, (3, 2, 'Terrifying', 2)
, (2, 2,  'Awesome', 5);
```

```
INSERT
INTO MealPosts(postId, title, content, dietType, recipeId)
VALUES
(1, 'Korean Chicken', 'Soooo delicious!', 'flextarian', 1)
, (2, 'Korean Soup', 'Amzing taste!', 'flextarian', 2)
, (3, 'Salad', 'So Sweet!', 'vegan', 3)
, (4, 'Japanese Ramen', 'Real Japanese Ramen!', 'flextarian', 4)
, (5,  'Pizza', 'You will be addicted!', 'flextarian', 5);

INSERT
INTO Explains(SKU, postId)
VALUES
(1, 1)
, (2, 2)
, (3, 3)
, (4, 4)
, (5, 5);

INSERT
INTO Contain(SKU, orderId)
VALUES
(1, 1)
, (2, 1)
, (3, 2)
, (4, 3)
, (5, 4);

INSERT
INTO List(recipeId, ingName)
VALUES
(1, 'Onion')
, (2, 'Onion')
, (3, 'Banana')
, (4, 'Broccoli')
, (5, 'Onion');

INSERT
INTO Inform(postId, storeId)
VALUES
(1, 1)
, (2, 2)
```

, (3, 3)
, (4, 4)
, (5, 5);