CPSC 304 Project Cover Page

Milestone #: 2

Date: 2024.10.15

Group Number: 55

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jiakang Huang	66382623	t4h5d	jhuang74@student.ubc.ca
Luocheng Shi	64544398	n8h0x	lshi10@ubc.student.ca
kexin (Cosine) feng	44781680	o2q5v	kfeng11@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 e-mail address, and then let us assign you to a TA for your project supervisor.)

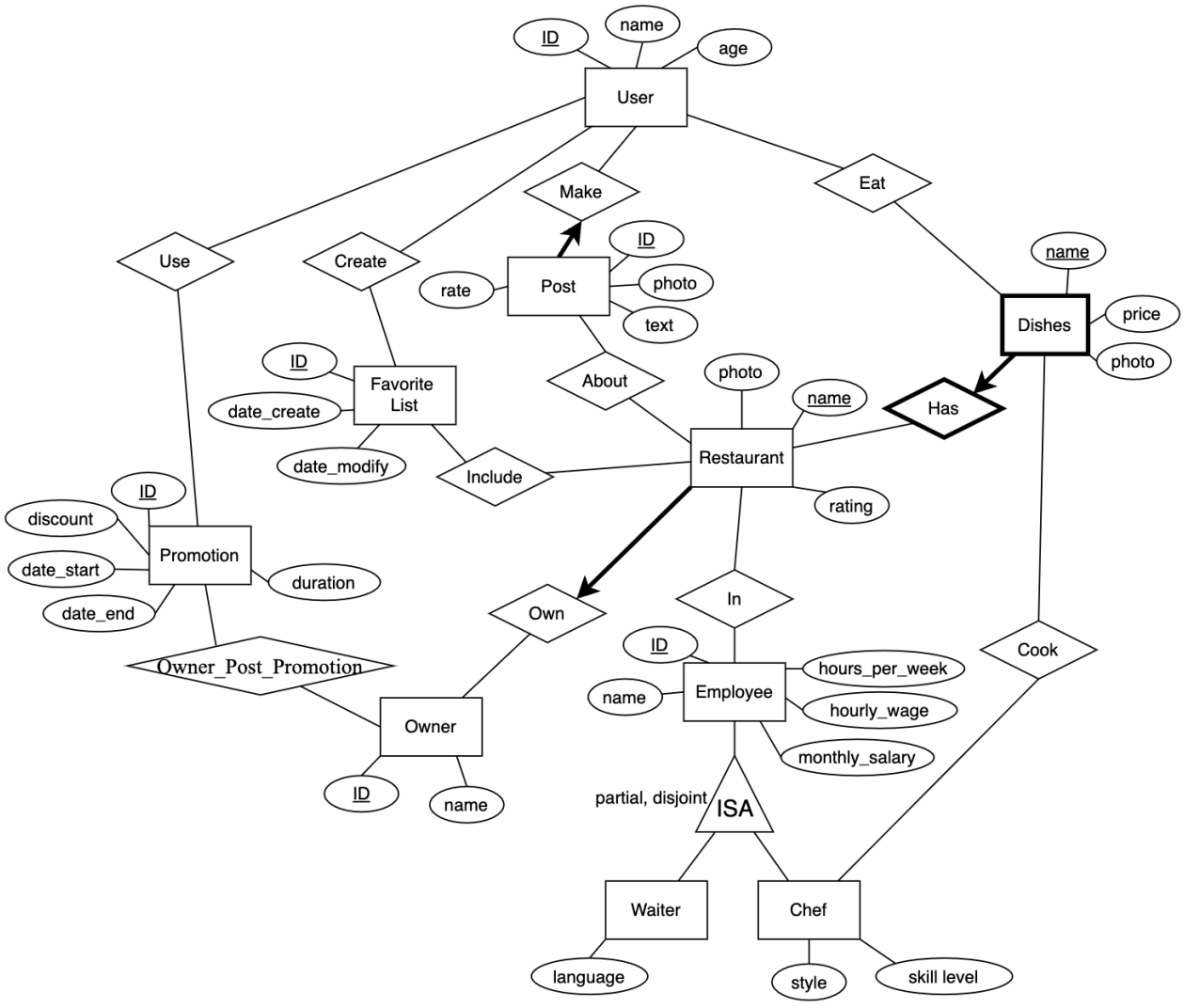
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Milestone 2

Step 2: Summary:

Our project is a restaurant rating app designed to enhance customer engagement by allowing users to post reviews, rate restaurants and dishes, and manage personalized favorite lists. It also enables restaurants to manage staff roles, update dish details, and let owner post promotions.

Step 3: ER diagram:



Changes:

1. Employee's ISA relationship:

   add partial, disjoint constrain to the ISA relationship

2. for entity promotion:

   add date_start and date_end and duration

   Now, user know when is the deadline for using this promotion.

3. for entity restaurant:

   remove ID, set name to PK

   ID is kind of redundant for restaurant, since two restaurant cannot have same name. So name can be the primary key.

4. for entity employee:

   add hourly_wage, monthly_salary, hours_per_week

5. for relationship Post:

   Rename to Owner_Post_Promotion, to make it different with the name of Post entity.

Step 4: The schema derived from your ER diagram:

Restaurant(name: varchar(50), **owner_ID**: integer, photo: image, rating: decimal(3,2))

(owner_ID and name should be unique and not null, rating should be between 0 and 5)

Dishes(**restaurantName**: varchar(50), name: varchar(50), price: decimal(5,2), photo: image)

(name, restaurantName should be unique and not null as a pair, price should be positive)

User(ID: integer, name: varchar(50), age: integer)

(ID should be unique and not null)

Promotion(ID: integer, discount: decimal(5,2), date_start: date, date_end: date, duration: integer)

(ID should be unique and not null, discount should be positive, date_start should be earlier than date_end)

Favorite_List(ID: integer, date_create: date, date_modified: date)

(ID should be unique and not null, date_create should be earlier than date_modified)

Post(ID: integer, **user_ID**: integer, photo: image, text: varchar(999), rate: integer)

(ID should be unique and not null, user_ID should be unique and not null)

(CK: photo is a candidate key)

Employee(ID: integer, name: varchar(50), hours_per_week: integer, hourly_wage: decimal(10,2), monthly_salary: decimal(10,2))

(ID should be unique and not null)

Waiter(ID: integer, language: varchar(50))

(ID should be unique and not null)

Chef(ID: integer, style: varchar(50), skill_level: integer)

(ID should be unique and not null, skill_level should be greater than 0)

Owner(ID: integer, name: varchar(50))

(ID should be unique and not null)

Create(**userID**: integer, **favoriteListID**: varchar(50))

(userID, favoriteListName should be unique as a pair)

In(**restaurantName**: integer, **employeeID**: integer)

(restaurantName, employeeID should be unique as a pair)

Owner_Post_Promotion(**ownerID**: integer, **promotionID**: integer)

(ownerID, promotionID should be unique as a pair)

Cook (**restaurantName**: varchar(50), **name**: varchar(50), **chefID**: integer)

(restaurantName, name, chefID should be unique as a pair)

Eat (**restaurantName**: varchar(50), **name**: varchar(50), **userID**: integer)

(restaurantName, name, userID should be unique as a pair)

About (**postID**: integer, **restaurantName**: integer)

(postID, restaurantName should be unique as a pair)

Use (**promotionID**: integer, **userID**: integer)

(promotionID, userID should be unique as a pair)

Include (**favoriteListID**: integer, **restaurantName**: integer)

(favoriteListID, restaurantName should be unique as a pair)

Step 5: Functional Dependencies:

Restaurant	name -> owner_ID, photo, rating
Dishes	restaurantName, name -> price, photo photo -> name
User	ID -> name, age
Promotion	ID -> discount, date_start, date_end, duration date_start, date_end -> duration
Favorite_List	ID -> date_create, date_modified
Post	ID -> user_ID, photo, text, rate photo -> user_ID, ID, text, rate (photo is CK)
Employee	ID -> name, hours_per_week, hourly_wage, monthly_salary hourly_wage, hours_per_week -> monthly_salary
Waiter	ID -> language
Chef	ID -> style, skill_level
Owner	ID -> name
Create	userID, favoriteListID
In	restaurantName, employeeID
Owner_Post_Promotion	restaurantName, promotionID
Cook	restaurantName, name, chefID
Eat	restaurantName, name, userID
About	postID, restaurantName
Use	promotionID, userID
Include	favoriteListID, restaurantName

Step 6: Normalization, 3NF:

Tables that are highlighted yellow in previous step might violate 3NF, since they have extra FDs other than PK or CK's FD. So we need to check and normalize them below.

Employee:

Employee(ID, name, hours_per_week, hourly_wage, monthly_salary)

key: ID.

① find minimal cover:

a) Let RHS be single.

ID → name ; ID → hourly_wage; ID → monthly_salary;

ID → hours_per_week ; hourly_wage, monthly_salary → hours_per_week

b) Reduce LHS:

Done.

c) Remove redundant:

Remove ID → hours_per_week

② hourly_wage, monthly_salary → hours_per_week violate 3NF, decompose

Emp1(ID, name, hours_per_week, hourly_wage)

Emp2(hours_per_week, hourly_wage, hours_per_week)

③ All FDs hold. This is the final answer.

Emp1(ID: integer, name: varchar(50), hours_per_week: integer, hourly_wage: decimal(10,2))

Emp2(hours_per_week: integer, hourly_wage: decimal(10,2), monthly_salary: decimal(10,2))

Promotion:

Promotion (ID, discount, date_start, date_end, duration)

Key : ID.

① Find minimal cover:

a) Let RHS be single.

ID → discount; ID → date_start ; ID → date_end ;

ID → duration; date_start, date_end → duration

b) Reduce LHS :

Done.

c) Remove redundant:

Remove ID → duration.

② date_start, date_end → duration violate 3NF. decompose:

Promotion1(ID, discount, date_start, date_end)

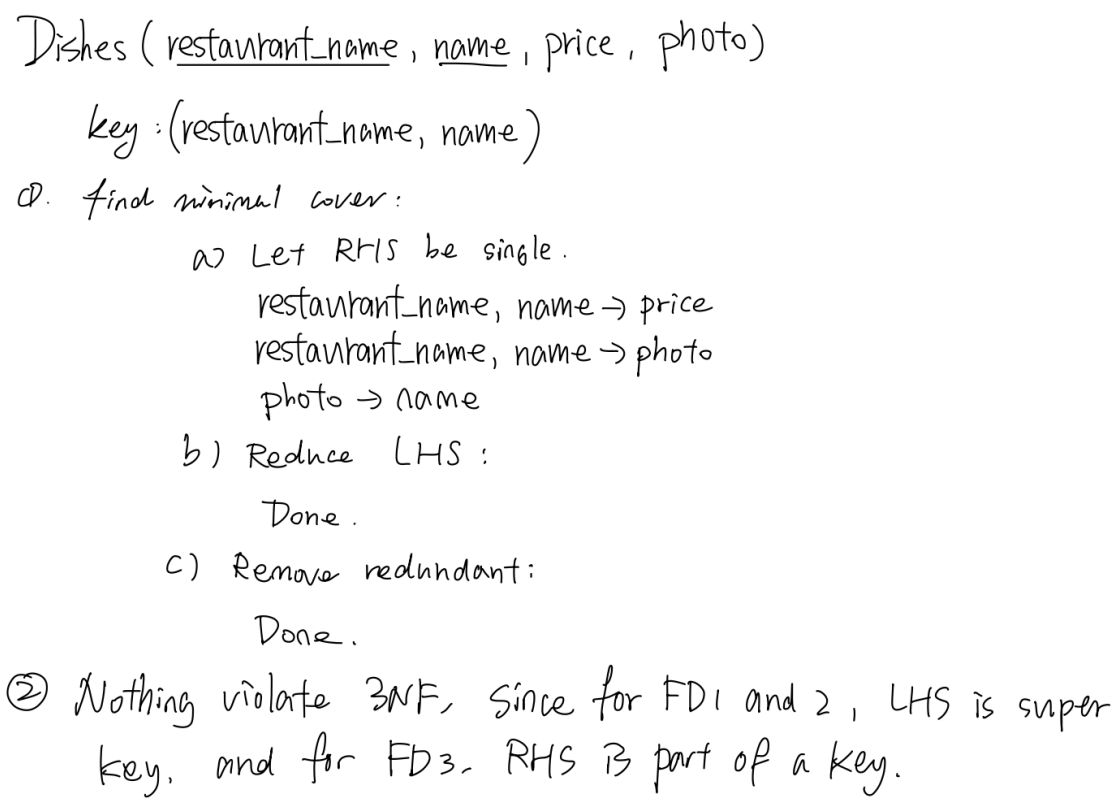
Promotion2(date_start, date_end, duration)

③ All FDs hold. This is the final answer.

Promotion1(ID: integer, discount: decimal(5,2), date_start: date, date_end: date)

Promotion2(date_start: date, date_end: date, duration: integer)

Dishes:



Dishes(**restaurantName**: varchar(50), name: varchar(50), price: decimal(5,2), photo: image)

Step 7 and 8: SQL DDL for tables, and insert statements examples:

```
-- Create Restaurant table
CREATE TABLE Restaurant (
    name VARCHAR(50) PRIMARY KEY,
    owner_ID INT NOT NULL,
    photo image,
    rating DECIMAL(3,2),
    CHECK (rating BETWEEN 0 AND 5),
    FOREIGN KEY (owner_ID) REFERENCES Owner(ID),
);
INSERT INTO Restaurant (name, owner_ID, photo, rating) VALUES
('Chipotle', 301, NULL, 4.5),
('McDonalds', 302, NULL, 3.8),
('KFC', 303, NULL, 4.0),
('Burger King', 304, NULL, 3.9),
('Big Way', 305, NULL, 3.6);

-- Create Dishes table
CREATE TABLE Dishes (
    restaurant_name VARCHAR(50),
    name VARCHAR(50),
```

```
    price DECIMAL(5,2),
    photo image,
    PRIMARY KEY (restaurant_name, name),
    CHECK (price > 0)
);
INSERT INTO Dishes (restaurant_name, name, price, photo) VALUES
('Burger King', 'Burger', 6.99, NULL),
('Big Way', 'Hot Pot', 12.50, NULL),
('McDonalds', 'Burger', 5.99, NULL),
('McDonalds', 'Fries', 3.50, NULL),
('KFC', 'Chicken Wings', 9.99, NULL);

-- Create Cook table
CREATE TABLE Cook (
    restaurant_name VARCHAR(50),
    dish_name VARCHAR(50),
    chef_id INT,
    PRIMARY KEY (restaurant_name, dish_name, chef_id),
    FOREIGN KEY (restaurant_name) REFERENCES Dishes(restaurant_name),
    FOREIGN KEY (dish_name) REFERENCES Dishes(name),
    FOREIGN KEY (chef_id) REFERENCES Chef(ID)
);
INSERT INTO Cook (restaurant_name, dish_name, chef_id) VALUES
('Burger King', 'Burger', 101),
('Big Way', 'Hot Pot', 102),
('McDonalds', 'Burger', 103),
('McDonalds', 'Fries', 103),
('KFC', 'Chicken Wings', 104);

-- Create Chef table
CREATE TABLE Chef (
    ID INT PRIMARY KEY,
    style VARCHAR(50),
    skill_level INT,
    CHECK (skill_level > 0)
);
INSERT INTO Chef (ID, name, style, skill_level) VALUES
(101, 'Italian', 5),
(102, 'Fast Food', 4),
(103, 'American', 6),
(104, 'Mexican', 5),
(105, 'French', 7);

-- Create Waiter table
CREATE TABLE Waiter (
    ID INT PRIMARY KEY,
```

```
    language VARCHAR(50),
    FOREIGN KEY (ID) REFERENCES Employee(ID)
);
INSERT INTO Waiter (ID, language) VALUES
(201, 'English'),
(202, 'Spanish'),
(203, 'French'),
(204, 'Chinese'),
(205, 'German');

-- Create Emp1 table
CREATE TABLE Emp1 (
    ID INT PRIMARY KEY,
    name VARCHAR(50),
    hours_per_week INT,
    hourly_wage DECIMAL(10,2)
);
INSERT INTO Emp1 (ID, name, hours_per_week, hourly_wage) VALUES
(201, 'Mike Johnson', 40, 15.00),
(202, 'Sarah Lee', 35, 12.50),
(203, 'Alex Brown', 20, 18.00),
(204, 'Chris Green', 25, 14.00),
(205, 'Kim White', 30, 16.00);

-- Create Emp2 table
CREATE TABLE Emp2 (
    hours_per_week INT,
    hourly_wage DECIMAL(10,2),
    monthly_salary DECIMAL(10,2),
    PRIMARY KEY (hours_per_week, hourly_wage)
);
INSERT INTO Emp2 (hours_per_week, hourly_wage, monthly_salary) VALUES
(40, 15.00, 2400.00),
(35, 12.50, 1750.00),
(20, 18.00, 1440.00),
(25, 14.00, 1400.00),
(30, 16.00, 1920.00);

-- Create Owner table
CREATE TABLE Owner (
    ID INT PRIMARY KEY,
    name VARCHAR(50)
);
INSERT INTO Owner (ID, name) VALUES
(301, 'Richard Roe'),
(302, 'Anna Wilson'),
```

```
(303, 'David King'),
(304, 'Sophia Taylor'),
(305, 'George Harris');

-- Create In table
CREATE TABLE In (
    restaurant_name VARCHAR(50),
    employee_ID INT,
    PRIMARY KEY (restaurant_name, employee_ID),
    FOREIGN KEY (restaurant_name) REFERENCES Restaurant(name),
    FOREIGN KEY (employee_ID) REFERENCES Emp1(ID)
);
INSERT INTO In (restaurant_name, employee_ID) VALUES
('Chipotle', 301, NULL, 4.5),
('McDonalds', 302, NULL, 3.8),
('KFC', 303, NULL, 4.0),
('Burger King', 304, NULL, 3.9),
('Big Way', 305, NULL, 3.6);


-- Create User table to store user information
CREATE TABLE User (
    ID INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    age INT
);
INSERT INTO User (ID, name, age) VALUES
(1, 'Ali', 25),
(2, 'Ben', 20),
(3, 'Charlie', 28),
(4, 'David', NULL),
(5, 'Eva', 22),
(6, 'Frank', 18),
(7, 'Gabe', 19);

-- Create Use table to store promotion usage by users
CREATE TABLE Use (
    promotion1_id INT,
    user_id INT,
    PRIMARY KEY (promotion1_id, user_id),
    FOREIGN KEY (user_id) REFERENCES User(ID),
    FOREIGN KEY (promotion1_id) REFERENCES Promotion1(ID)
);
INSERT INTO Use (promotion1_id, user_id) VALUES
(1, 1),
(2, 2),
```

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

-- Create Favorite_List table to store favorite lists
CREATE TABLE Favorite_List (
    ID INT PRIMARY KEY,
    date_create DATE,
    date_modify DATE,
);
INSERT INTO FavoriteList (ID, date_create, date_modify) VALUES
(1, '2024-01-01', '2024-01-05'),
(2, '2024-02-10', '2024-02-15'),
(3, '2024-03-20', '2024-03-25'),
(4, '2024-04-05', '2024-04-10'),
(5, '2024-05-15', '2024-05-20'),
(6, '2024-06-25', '2024-06-30');

-- Create Create table to track favorite list creation by users
CREATE TABLE Create (
    user_id INT,
    Favorite_List_id INT,
    PRIMARY KEY (user_id, Favorite_List_id),
    FOREIGN KEY (user_id) REFERENCES User(ID),
    FOREIGN KEY (Favorite_List_id) REFERENCES Favorite_List(ID)
);
INSERT INTO Create (user_id, Favorite_List_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

-- Create Promotion1 table to store promotion details
CREATE TABLE Promotion1 (
    ID INT PRIMARY KEY,
    discount DECIMAL(5, 2),
    date_start DATE,
    date_end DATE
);
INSERT INTO Promotion1 (ID, discount, date_start, date_end) VALUES
(1, 10.00, '2024-07-01', '2024-07-15'),
(2, 15.00, '2024-08-01', '2024-08-20'),
(3, 20.00, '2024-09-01', '2024-09-25'),
(4, 5.00, '2024-10-01', '2024-10-10'),
(5, 14.00, '2024-11-01', '2024-11-15'),
```

```
(6, 9.99, '2024-12-01', '2024-12-10');

-- Create Promotion2 table to store calculated promotion durations
CREATE TABLE Promotion2 (
    date_start DATE,
    date_end DATE,
    duration INT,
    PRIMARY KEY (date_start, date_end)
);
INSERT INTO Promotion2 (date_start, date_end, duration) VALUES
('2024-07-01', '2024-07-15', 14),
('2024-08-01', '2024-08-20', 19),
('2024-09-01', '2024-09-25', 24),
('2024-10-01', '2024-10-10', 9),
('2024-11-01', '2024-11-15', 14),
('2024-12-01', '2024-12-10', 9);

-- Create About table to track posts about restaurants
CREATE TABLE About (
    post_id INT PRIMARY KEY,
    restaurant_name VARCHAR(50) PRIMARY KEY,
    FOREIGN KEY (post_id) REFERENCES Post(ID),
    FOREIGN KEY (restsurant_name) REFERENCES Restaurant(ID)
);
INSERT INTO About (post_id, restaurant_name) VALUES
(1, 'Chipotle'),
(2, 'McDonalds'),
(3, 'KFC'),
(4, 'Burger King'),
(5, 'Big Way');

-- Create Post table to store post information
CREATE TABLE Post (
                    ID INT PRIMARY KEY,
                    user_id INT NOT NULL,
                    photo image,
                    text VARCHAR(999),
                    FOREIGN KEY (user_id) REFERENCES User(ID)
);
INSERT INTO Post (ID, user_id, photo, text) VALUES
(1, 1, NULL, 'Great food and amazing service at Chipotle!'),
(2, 2, NULL, 'McDonalds fries are unbeatable!'),
(3, 3, NULL, 'KFC chicken wings are the best!'),
(4, 4, NULL, 'Had a delicious burger at Burger King.'),
(5, 5, NULL, 'Big Way offers the best hot pot in town.');
```

```sql
-- Create Eat table to track which dishes users have eaten
CREATE TABLE Eat (
    dish_name VARCHAR(50) PRIMARY KEY,
    user_id INT PRIMARY KEY,
    FOREIGN KEY (dish_name) REFERENCES Dishes(name),
    FOREIGN KEY (user_id) REFERENCES User(ID)
);
INSERT INTO Eat (dish_name, user_id) VALUES
('Burger', 1),
('Hot Pot', 2),
('Fries', 3),
('Chicken Wings', 4),
('Burger', 5);

-- Create Owner_Post_Promotion table to track promotion posts by restaurant
owners
CREATE TABLE Owner_Post_Promotion (
    ownerID INT PRIMARY KEY,
    promotionID INT PRIMARY KEY,
    FOREIGN KEY (ownerID) REFERENCES Owner(ID),
    FOREIGN KEY (promotionID) REFERENCES Promotion1(ID)
);
INSERT INTO Owner_Post_Promotion (ownerID, promotionID) VALUES
(301, 1),
(302, 2),
(303, 3),
(304, 4),
(305, 5);

-- Create Include table to track which restaurants are included in favorite
lists
CREATE TABLE Include (
    favoriteListId INT PRIMARY KEY,
    restaurantName VARCHAR(50) PRIMARY KEY,
    FOREIGN KEY (favoriteListId) REFERENCES Favorite_List(ID),
    FOREIGN KEY (restaurantName) REFERENCES Restaurant(restaurantName)
);
INSERT INTO Include (favoriteListId, restaurantName) VALUES
(1, 'Chipotle'),
(2, 'McDonalds'),
(3, 'KFC'),
(4, 'Burger King'),
(5, 'Big Way');
```