**S19 CSE305 Final Project: Final Report**

Team GiveUsA: Project Eatogether

Sean (Sang-Uk) Lee

Jihye Woo

Banseok Koo

1. **General Description**
   1. Goal: Creating a web application that serve needs for small-sized food delivery. Current food delivery services require a minimum order price. Our true goal is to lower the entry barrier for split customers by offering a match system to combine their orders into one.
   2. Required Technologies:
      1. Target System: Web
      2. Front-end: HTML, PHP, and CSS
      3. Back-end: MySQL and XAMPP
2. **Use Cases**
   1. Sign up

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User creates an account with hashed password and store to database |
| **Precondition** | User is not signed up or wish to create additional account |
| **Trigger** | User clicks ‘Sign up’ on login screen |
| **Scenario** | 1. User enters desired username and password 2. User clicks ‘Register’ button 3. Application redirects the user to main screen |

* 1. Login

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User logs in and become able to interact with application |
| **Precondition** | User is not logged in |
| **Trigger** | User launches application |
| **Scenario** | 1. User gets redirected to login screen 2. User enters username and password 3. User clicks LOGIN button 4. Application redirects the user to main screen |

* 1. Logout

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User logs out and become unable to interact with application |
| **Precondition** | User is logged in |
| **Trigger** | User clicks LOGOUT button on menu bar |
| **Scenario** | 1. Application redirects the user to login screen |

(Home)

* 1. Home

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User is able to navigate through home screen |
| **Precondition** | User is logged in |
| **Trigger** | User clicks HOME button on menu bar |
| **Scenario** | 1. User navigates through home screen |

(About Us)

* 1. About Us

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User is able to view PR information |
| **Precondition** | User is logged in |
| **Trigger** | User clicks ABOUT US button on menu bar or LEARN MORE button on any screens that it is present |
| **Scenario** | 1. User navigates through PR info screen |

(Order)

* 1. Create New Waitlist

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User creates a new waitlist that other users can join |
| **Precondition** | User is logged in |
| **Trigger** | User clicks GROUP ORDER button on ORDER screen |
| **Scenario** | 1. User clicks CREATE NEW GROUP ORDER button 2. User selects a restaurant from local restaurant list 3. User clicks SELECT THE MENU button 4. User selects a menu from selected restaurant 5. User clicks ADD button 6. User repeats 4-5 7. User clicks PAYMENT button 8. User selects a location to receive food 9. User clicks PROCEED button 10. (Optional) User selects available coupon 11. (Optional) User clicks SELECT COUPON button 12. User clicks PAY button 13. Application redirects to success screen |

* 1. View Active Waitlist

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User is able to view current active waitlist waiting for user to join |
| **Precondition** | User is logged in |
| **Trigger** | User clicks ORDER button on menu bar |
| **Scenario** | 1. User navigates through a list of active waitlists if they exist |

* 1. Join Waitlist

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User joins to current active waitlist |
| **Precondition** | User is logged in |
| **Trigger** | User clicks ORDER button on menu bar |
| **Scenario** | 1. User navigates through a list of active waitlists if they exist 2. User selects a waitlist 3. User clicks SELECT button 4. User clicks GO TO NEXT STEP button 5. Application shows information about selected waitlist 6. User input quantity of food he wish to get delivered 7. User clicks PROCEED button 8. (Optional) User selects available coupon 9. (Optional) User clicks SELECT COUPON button 10. User clicks PAY button 11. Application redirects to success screen |

(My Page)

* 1. View My Info

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User is able to view username and current coin he occupies |
| **Precondition** | User is logged in |
| **Trigger** | User clicks MY PAGE button on menu bar |
| **Scenario** | 1. User navigates through a list of user information |

* 1. Charge Coin

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User charges coin |
| **Precondition** | User is logged in |
| **Trigger** | User clicks MY PAGE button on menu bar |
| **Scenario** | 1. User clicks CHARGE button |

* 1. View Coupon Info

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User is able to view discount price and expire date of coupons he has |
| **Precondition** | User is logged in |
| **Trigger** | User clicks MY PAGE button on menu bar |
| **Scenario** | 1. User navigates through a list of coupons |

* 1. View Order Info

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User is able to view order information he previously ordered |
| **Precondition** | User is logged in |
| **Trigger** | User clicks MY PAGE button on menu bar |
| **Scenario** | 1. User navigates through a list of orders |

(Admin)

* 1. View Coupon List

|  |  |
| --- | --- |
| **Primary Actor** | Admin |
| **Goal** | Admin is able to view which coupon is occupied by whom |
| **Precondition** | User is logged in as admin |
| **Trigger** | Admin clicks ADMIN button on menu bar |
| **Scenario** | 1. Admin navigates through a list of information of coupons |

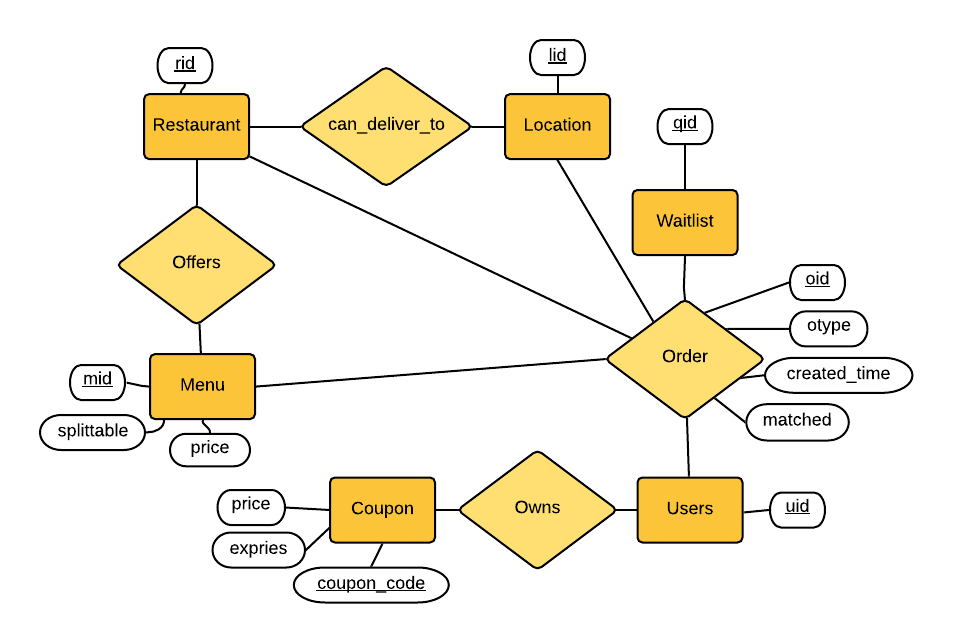
* 1. View User List

|  |  |
| --- | --- |
| **Primary Actor** | Admin |
| **Goal** | Admin is able to view which user has which amount of coin |
| **Precondition** | User is logged in as admin |
| **Trigger** | Admin clicks ADMIN button on menu bar |
| **Scenario** | 1. Admin navigates through a list of information of users |

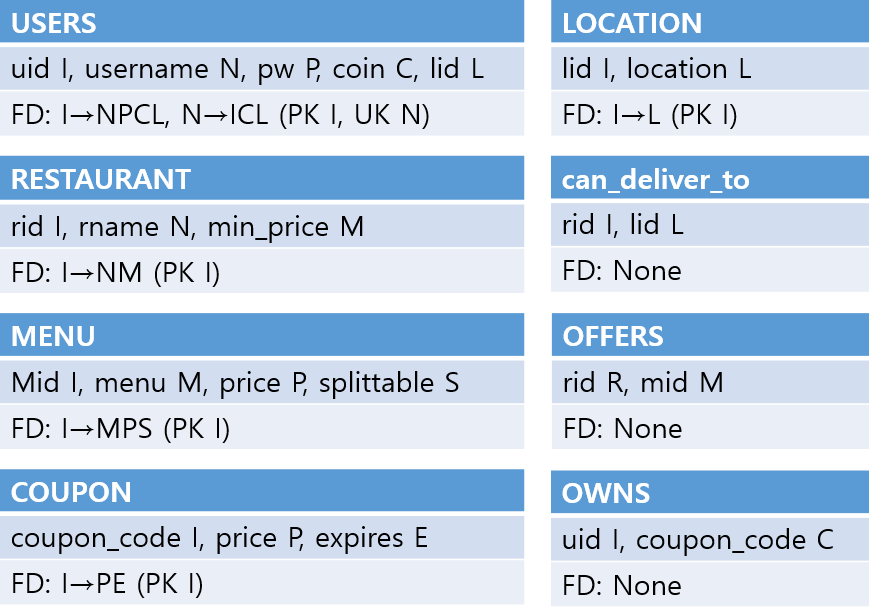
* 1. Add Coupon

|  |  |
| --- | --- |
| **Primary Actor** | Admin |
| **Goal** | Admin adds selected discount amount coupon to selected user |
| **Precondition** | User is logged in as admin |
| **Trigger** | Admin clicks ADMIN button on menu bar |
| **Scenario** | 1. Admin selects discount amount 2. Admin enters username 3. Admin clicks ADD COUPON button 4. Application redirects to ADMIN screen |

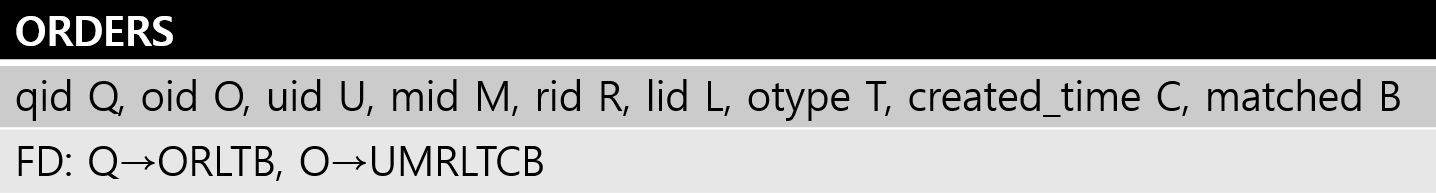
1. **E-R Diagram**



1. **Database Normalization**
   1. Pre-normalization
      1. In *Final Project Progress Report 1* and *Final Project Progress Report 2*, we have gone through numerous changes in our relations. We however are aiming to illustrate our changes in database not from a certain point at past progress reports, but from the point where we decided to confidently start applying normalization techniques.
      2. Prior to proceed our initial step, our relations were full of semantic redundancy. Several columns, or even relations that we considered to be necessary were not logically appropriate after deeper thoughts; relations representing necessary information were not properly arranged.
      3. Prior to normalization, we decided to make every relations except for ORDERS, to be reasonably simplified; every tables should have a unique identifier (as a primary key) and its contents. Below are ORDERS table with example tuples (almost similar with the one in *Report 1*) and other tables that were fixed accordingly.



[Fig. 1] Simplified Tables Except for ORDERS



[Fig. 2] ORDERS Table



[Fig. 3] Example Tuples Populated (Some not described as integer id’s for visualization)

* 1. Post-normalization
     1. In Fig. 1, all tables had been designed to have a unique identifier. Those unique identifiers are superkey of its relation. If a table is in BCNF, every determinant of its FD is a candidate key. For LOCATION, RESTAURANT, MENU, and COUPON, every functional dependencies have superkey as its determinant. So every one of them happened to be in BCNF because of our attempt to reduce semantic redundancy.
     2. For USERS, the merged minimal cover of functional dependencies of **I→NPCL**, **N→ICL** is **I→N, N→PCLI**. This can be reduced, by using inference rule, to **I→NPCL**. As I is a candidate key, USERS is in BCNF.
     3. In Fig. 2, ORDERS table is a little bulky, although every columns are needed in our service; it must be decomposed somehow. **Q→ORLTB** means, knowing a unique identifier of a waitlist will let know: id’s of the orders matched in a waitlist, id of the restaurant the orders are taken, id of the location users wish to receive, which type (split or group) of orders the waitlist is for, and if the waitlist is matched or not. **O→UMRLTCB** means, knowing a unique identifier of an order will let know: id of the user who made the order, id of the menu the user wanted, id of the restaurant the user put the order to, which type (split or group) of order it is, the time the order is created, and if the order is matched with other orders or not. As the determinant of **O→UMRLTCB** violates BCNF because O is not a superkey, ORDERS is not in BCNF. The merged minimal cover of FD’s contain **Q→O** and **O→UMRLTCB**. This still violates BCNF due to same reason. Using above two, ORDERS can be decomposed into **QO** and **OUMRLTCB**.
     4. Decomposition in above normalization [{QOUMRLTCB} being decomposed into {{QO}, {OUMRLTCB}}] is a lossless join decomposition, because
        1. {QO} ∪ {OUMRLTCB} = {QOUMRLTCB}
        2. {QO} ∩ {OUMRLTCB} = {O} ≠ Ø
        3. O (from #2)→UMRLTCB (Given)
     5. Since **Q→O** and **O→UMRLTCB** are part of **QO** and **QUMRLTCB** respectively, decomposition in (iii) is dependency-preserving.

1. **User’s Manual**
   1. Download "[XAMPP for Windows 7.2.2 (PHP 7.2.2)](https://www.apachefriends.org/index.html)" (or the latest version) into a folder of your choice. Adjust in the rest of this page if you are using a different version. You will download xampp-win32-7.2.2-0-VC15-installer.exe.
   2. Execute the installer.
   3. Start XAMPP and initialize.
   4. Start Apache and MYSQL Module.
   5. Click shell command-line utility. Use the following command:
      1. mysql -h localhost -u root -- To start a client from shell.
      2. if above does not work, try mysql -h localhost -u root -p and enter 'root' as the password
   6. Copy all codes in CSE305\_Final\_Project\_Database2.sql file and paste to shell after you see MariaDB [(none)]>.
      1. If above does not work, try with CSE305\_Final\_Project\_Database1.sql
   7. Find where you downloaded XAMPP and unzip CSE305\_Final\_Project\_Eattogether.zip into XAMPP/htdocs
   8. Open an internet browser (perferably Chrome) and type localhost/'(Name of folder containing unzipped contents)' as the URL.
      1. If above does not work, try localhost:(port)
      2. Make sure to put every files directly above ../htdocs/
   9. Test
      1. As automatically redirected to login.php, register yourself by clicking ‘Sign Up’.
      2. Sign up with the id 'admin' with any password. You should be able to perform admin rights (in admin page) with this account.
2. **Post-Mortem Analysis (Pun Intended)**
   1. Our original goals that met:
      1. To finish (our code got perfectly compiled!!!)
      2. To at least actualize a startup idea
      3. Implementing expected number (10+) of use cases
      4. Making every tables to be in BCNF after normalization
      5. Being as communicative as possible during development
   2. That did not met:
      1. Below are the derivation of the goals that were not met.
   3. Possible improvements
      1. Documenting everything (even trivial development approaches)

* Why did not? We all eagerly wanted to code rather than to take our time and write ideas down.
  + 1. Better application programming interface; not in a sense of the entire environment, but at least several procedures in the SQL.
* Why did not? As deadline gets near, we gradually started to hardcode.
  + 1. Fixing our database structure to be more intuitively and concisely represent our service
* Why did not? We used a top-down approach to start creating the database. Being bottom-up would have been more concise.
  + 1. Better automation of our database, such as adding more triggers, checks, and so on.
* Why did not? We valued PHP to have more authority, but it was a wrong belief.
  + 1. Adding richer sample tuples for our simple-data tables such as more restaurants with different menus.
* Why did not? To reduce complexity during testing.
  + 1. Simpler, more interactive, and beautiful UI.
* Why did not? Grader valued back-end more. But beautiful things are beautiful!
  1. Additional features
     1. Concurrency control features on *joining waitlist* as concurrent requests or accesses on a same waitlist might respond with a disturbing UX.
     2. Order statistics for not only individual customers, but also for the entire system.
     3. Google Map API usage for setting a location where user wish to receive delivery.

1. **Code: Included along this Word document in a zip file. Below is the list of files included.**
   1. CSE305\_Final\_Project-Final\_Report\_TeamGiveUsA.docx
   2. CSE305\_Final\_Project\_Database1.sql
   3. CSE305\_Final\_Project\_Database2.sql
   4. CSE305\_Final\_Project\_Eatogether.zip
   5. CSE305\_Final\_Project\_Screenshots.zip
2. **Other comments**
   1. Would our program be stable, reliable, and robust in real life situation?
      1. Our application is designed under the assumption of running on a single server.
      2. Although our queries are dirty, there is not much time complexity issues. There does not exist bulky interactions between user and server.
      3. The service should be stable enough to respond to the number of users that a single processor could handle.