**CS 4750 Database Systems**
**Group name**: Database Group
**Group members**:

Ethan Gumabay (emg4dv)
Amelia Hampford (agh3ft)
Connor Yager (csy7ay)
Faye Park (shp8xb)
Richard Ohr (rfo8sf)

**3NF Proof:**
Here are the functional dependencies for our database:

A -> B
AE -> AE
E -> C D
F -> G
EF-> H I
J -> K
EL-> M N O P Q

A: area_name  B: transit_from_campus

C: address  D: rest_name  E: rid

F: username  G: password

H: text  I: rating

J: accname  K: password

L: item_name  M: desc  N: cost

O: recurs_when  P: start_date
Q: end_date

The only extraneous dependency is AE->AE. To make the canonical cover, all we have to do is get rid of this dependency.

A -> B
E -> C D
F -> G
EF-> H I
J -> K
EL-> M N O P Q

A 3NF version of our database has the following tables:
(area_name, transit_from_campus)

(rid, address, rest_name)
(username, password)
(username, rid, text, rating)
(accname, password)
(rid, item_name, desc, cost, recurs_when, start_date, end_date)

We will only be diverging from this in two ways: first, we'll add the AE table back in (area_name, rid). This is because we want to relate restaurants to areas in a many-to-many relationship (a restaurant can be near multiple areas, and an area can contain multiple restaurants). Even though it doesn't uniquely identify any new information, this relationship is useful for our application.
Second, we will split the table:
(rid, item_name, desc, cost, recurs_when, start_date, end_date)
into 3 separate tables, since we have 3 different entity sets: menuItem, regularDeal, and tempDiscount. Only regularDeals have a "recurs_when" value, and only tempDiscounts have "start_date"s and "end_date"s. If we had all of these represented by one table then there would be many null values, so it makes more sense for our application to have 3 tables for these objects. Our three tables are as follows:
menuItem: (rid, item_name, desc, cost)
regularDeal: (rid, item_name, desc, cost, recurs_when)
tempDiscount: (rid, item_name, desc, cost, start_date, end_date)

**Reporting Functionality:**

Our project is a web-based platform for Charlottesville residents, particularly students, to find deals at various local restaurants. Currently, all of our tables and SQL commands and implemented, along with the first few pages of our website.

The landing page of our website displays all of the restaurants that are currently in the database. Along with the data stored in the restaurant table, like address and name, this landing page shows the number of comments and the average rating for each restaurant. On the landing page, a user can click ‘Submit a place’, which will direct them to a form. This form takes in a restaurant's name,  address, and RID. Submitting this form will add the new restaurant to the database, and this change is reflected on the landing page.

In the next few weeks, we will be working more on finishing the website. We will implement features to add and search deals, and a feature to leave a review. The finished project will not only allow students to find cheaper deals for eating out, but it will also help small businesses flourish by advertising their deals.

The tables we currently have implemented are:

- area(area_name, transit_from_campus)
- comment(username, rid, text, rating)
- isIn(rid, area_name)
- menuItem(item_name, description, cost, rid)
- ownedBy(accname, rid)
- ownerAccount(accname, password)
- regularDeal(item_name, description, cost, rid, recurs_when)
- restaurants(address, name, rid)
- tempDiscount(item_name, description, cost, rid, start_date, end_date)
- users(username, password)

SQL commands implemented:

- Foreign key constraints on relationships “isIn”, “ownedBy” and “comment”
- Weak entity sets “menuItem”, “regularDeal” and “tempDiscount”