Jared, Ruyuan, Randolf
EECS 341
Final Report
12/13/2019

Final Project Report: Snack Tracker

1. Application background

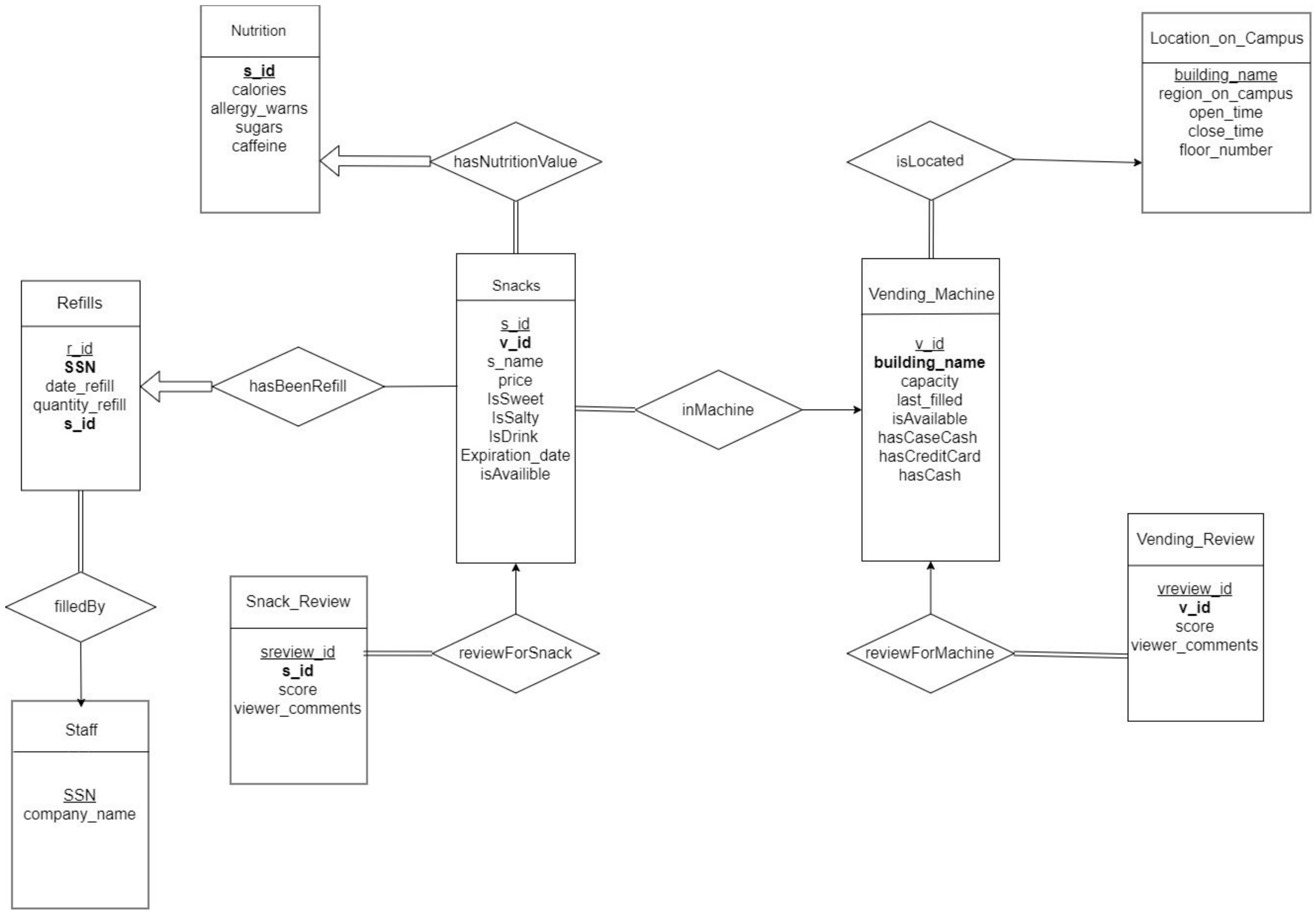
For our project, we decided to create a database for the network of vending machines on the CWRU campus. We thought of this idea after reflecting on all the instances in which we have walked out of our way to vending machines only to find out they ran out of the item(s) we wanted, the machine was broken, the machine only accepted CaseCash, etc. Thus, we came to a consensus that we would love if there were a way to track on-campus vending machine data such that information could be searched for ahead of time. In theory, our database would allow an interface to display searches for specific snacks in vending machines, as well as display information about the specific snacks and vending machines including reviews/comments from users.

2. Data description

In this database, we will be tracking the following main entities: snacks, vending machines, staff. A snack will be associated with nutrition information, snack review(s), refill information, and a specific vending machine. A vending machine will be associated with a location on campus and vending machine review(s). Staff will contain all employed staff members, and all snack refills will be associated with a staff member and a specific snack.

3. ER Diagram

Noted: In the ER diagram, primary keys are <u>underlined</u> and foreign keys are in bold.



4. List of functional dependencies derived from the semantics of our data

Snack(<u>s_id</u>, v_id, s_name, price, isSweet, isSalty, isDrink, Expiration_date, isAvailable)
s_id → v_id, s_name, price, isSweet, isSalty, isDrink, Expeication_date, isAvailable
For each snack, the s_id correlates to only one name, price, expiration date, current availability, and vending machine in which it can be found. Each snack also has a few different attributes describing the snack type (salty, sweet, drink/solid food). s_id is the candidate key. BCNF form.

Vending_Machine(<u>v_id</u>, buiding_name, capacity, last_fill, isAvailable, isCaseCash, isCredit, isCash)
v_id → buiding_name, capacity, last_fill, isAvailable, isCaseCash, isCredit, isCash

For each vending machine, the v_id correlates to only one capacity, last filled date, current availability, and the name of the building where it can be found. Each machine also has a few different attributes describing the accepted payment type (Case Cash, credit/debit, cash). v_id is the candidate key. BCNF form.

Nutrition(s_id, calories, allergy_warns, sugars, caffeine)
s_id → calories, allergy_warns, sugars, caffeine
For each snack's nutrition information, the s_id correlates to only one number of calories, allergy warnings, amount of sugar, and amount of caffeine. s_id is the candidate key. BCNF form.

Location_on_Campus(building_name, region_on_campus, open_time, close_time, floor_number)
building_name → region_on_campus, open_time, close_time, floor_number
For each building location on campus, the building_name correlates to only one region on campus, open time, close time, and the floor number that the building's vending machines are located on. building_name is the candidate key. BCNF form.

Snack_Review(srewiew_id, s_id, score, comments)
srewiew_id → s_name, score, comments
For each snack review, the srewiew_id correlates to only one snack id, score/rating, and actual user comment. srewiew_id is the candidate key. BCNF form.

Vending_Review(vrewiew_id, v_id, score, comments)
vrewiew_id → v_id, score, comments
For each vending machine review, the vrewiew_id correlates to only one machine id, score/rating, and actual user comment. vrewiew_id is the candidate key. BCNF form.

Refills(r_id, s_id, date_refill, quantity_refill, SSN)
r_id → s_id, date_refill, quantity_refill, SSN
For each snack refill, the r_id correlates to only one snack id, refill date, refill quantity, and staff member SSN. r_id is the candidate key. BCNF form.

Staff(SSN, company_name)
SSN → company_name
For each staff member, the SSN correlates to only one company name (who they work for). SSN is the candidate key. BCNF form.

Note: for the dependencies below, an asterisk (*) is used to differentiate attributes from different tables that have the same name(s)

isLocated(building_name, *building_name)
building_name → *building_name

Each vending machine on campus has only one location on campus, uniquely identified by its building_name. BCNF form.

reviewForMachine(v_id, *v_id)
v_id → *v_id
Each vending machine review correlates to only one vending machine on campus, uniquely identified by a v_id. BCNF form.

reviewForSnack(s_id, *s_id)
s_id → *s_id
Each snack review correlates to only one snack, uniquely identified by a s_id. BCNF form.

hasNutritionValue(s_id, *s_id)
s_id → *s_id
Each snack, uniquely identified by an s_id, has only one set of nutrition details, uniquely identified by an s_id. BCNF form.

inMachine(v_id, *v_id)
v_id → *v_id
Each snack correlates to only one vending machine on campus, uniquely identified by a v_id. BCNF form.

hasBeenRefill(s_id, *s_id)
s_id → *s_id
Each snack refill correlates to only one snack, uniquely identified by a s_id. BCNF form.

refillBy(SSN, *SSN)
SSN → *SSN
Each snack refill correlates to only one staff member, uniquely identified by a SSN. BCNF form.

5. The schema of our database, satisfying 3NF (if not BCNF)

Snack(s_id, v_id, s_name, price, isSweet, isSalty, isDrink, Expiration_date, isAvailable)
s_id: id for the snack (per vending machine), primary key, not null
v_id: id for the vending machine, foreign key,
s_name: name of the snack
price: amount of the money the snack costs
isSweet: whether the snack is sweet
isSalty: whether the snack is salty
isDrink: whether the snack is a drink
Expiration_date: the expiration date of the snack
isAvailable: whether the snack is available

Vending_Machine(v_id, buiding_name, capacity, last_fill, isAvailable, isCaseCash, isCredit, isCash)
v_id: id of the vending machine, primary key, not null
buiding_name: the name of the building with the vending machine
capacity: total capacity of the vending machine
last_fill: date the machine was last filled
isAvailable: whether the machine is available/working
isCaseCash: whether Case Cash is accepted as payment
isCredit: whether credit/debit card is accepted as payment
isCash: whether cash is accepted as payment

Nutrition(s_id, calories, allergy_warns, sugars, caffeine)
s_id: id of the snack, primary key, not null
calories: number of calories of the snack
allergy_warns: allergy warnings for the snack
sugars: number of grams of sugar
caffeine: whether the snack is caffeinated or not (Boolean)

Location_on_Campus(building_name, region_on_campus, open_time, close_time, floor_number)
building_name: name of the building, primary key, not null
region_on_campus: region on campus where the building exists
open_time: the time when the building opens
close_time: the time when the building closes
floor_number: the floor number where the building's vending machines exist

Snack_Review(srewiew_id, s_id, score, comments)
srewiew_id: id for the snack's review, primary key, not null
s_id: id for the snack being reviewed
score: the number for the snack's rating out of 5
comments: the written comment for the snack's review

Vending_Review(vrewiew_id, v_id, score, comments)
vrewiew_id: id for the vending machine's review, primary key, not null
v_id: id for the vending machine being reviewed
score: the number for the machine's rating out of 5
comments: the written comment for the machine's review

Staff(SSN, company_name, s_id, date_refill, quantity_refill)
SSN: the social security number of the staff, primary key, not null
company_name: name of the company the staff works for

Refills(r_id, s_id, date_refill, quantity_refill, SSN)

r_id: the id for the unique refill, primary key, not null
s_id: the id for the snack being refilled
date_refill: the date of the refill
quantity_refill: the total number of items added during the refill
SSN: the social security number of the staff who completed the refill

Note: for the dependencies below, an asterisk (*) is used to differentiate attributes from different tables that have the same name(s)

isLocated(<u>building_name</u>, *building_name)
building_name: name of a building on campus, primary key in Location_On_Campus, not null
*building_name: name of a building on campus in Vending_Machine, foreign key references Location_On_Campus

reviewForMachine(<u>v_id</u>, *v_id)
v_id: id for a vending machine on campus, primary key in Vending_Machine, not null
*v_id: id for a vending machine on campus in Vending_Review, foreign key references Vending_Machine

reviewForSnack(<u>s_id</u>, *s_id)
s_id: id for a snack, primary key in Snacks, not null
*s_id: id for a snack in Snack_Review, foreign key references Snacks

hasNutritionValue(<u>s_id</u>, *<u>s_id</u>)
s_id: id for a snack, primary key in Nutrition, foreign key references Snacks, not null
*s_id: id for a snack, primary key in Snacks, foreign key references Nutrition, not null

inMachine(<u>v_id</u>, *v_id)
v_id: id for a vending machine on campus, primary key in Vending_Machine, not null
*v_id: id for a vending machine on campus in Snacks, foreign key references Vending_Machine

hasBeenRefill(<u>s_id</u>, *s_id)
s_id: id for a snack, primary key in Snacks, not null
*s_id: id for a snack in Refills, foreign key references Snacks

refillBy(<u>SSN</u>, *SSN)
SSN: social security number for a staff member, primary key in Staff, not null
*SSN: social security number for a staff member in Refills, foreign key references Staff

6. **Example queries supported by our system. Our queries need to have some queries that use NOT EXISTS, EXCEPT as we practiced in the class**

Select all available or unavailable snacks

```
"select * from Snacks where isavailable='"+isavailable+"'"
```

Select snacks by snacks name

```
"select * from Snacks where s_name='"+s_name+"'"
```

Select snacks by maxprice (using EXCEPT)

```
"( select s_name, price from Snacks )except( select s_name, price from Snacks where
price >= '"+maxprice+"' )";
```

Select snacks by minprice (using NOT EXISTS)

```
"select s_name, price from Snacks where not exists (select s_name, price from Snacks
where price <= '"+minprice+"')"
```

Select all nutrition values by snack name

```
"select Snacks.s_name, Snacks.price, Nutrition.calories, Nutrition.allergy_warns,
Nutrition.sugars, Nutrition.caffeine from Snacks, Nutrition where
Nutrition.s_id=Snacks.s_id and Snacks.s_name = '"+nutritionbyname+"'"
```

Select snacks by calories

```
"select Snacks.s_name, Snacks.price, Nutrition.calories from Snacks, Nutrition where
Nutrition.s_id=Snacks.s_id and Nutrition.calories < '"+calories+"'"
```

Select snacks by allergy_warns

```
"select Snacks.s_name, Snacks.price, Nutrition.allergy_warns from Snacks, Nutrition
where Nutrition.s_id=Snacks.s_id and Nutrition.allergy_warns = '"+allergy_warns+"'"
```

Select snacks by sugars

```
"select Snacks.s_name, Snacks.price, Nutrition.sugars from Snacks, Nutrition where
Nutrition.s_id=Snacks.s_id and Nutrition.sugars < '"+sugars+"'"
```

Select snacks by caffeine

```
"select Snacks.s_name, Snacks.price, Nutrition.caffeine from Snacks, Nutrition where
Nutrition.s_id=Snacks.s_id and Nutrition.caffeine = '"+caffeine+"'"
```

Select snacks by review score

```
"select Snack_Review.s_id, Snacks.s_name, Snack_Review.viewer_comments  from
Snack_Review, Snacks where Snack_Review.score  = '"+score+"' and
Snacks.s_id=Snack_Review.s_id"
```

Select reviews by snack name

```
"select Snacks.s_name, Snack_Review.score, Snack_Review.viewer_comments  from
Snack_Review, Snacks where Snacks.s_name = '"+snackReviewSName+"' and
Snacks.s_id=Snack_Review.s_id"
```

Insert your comments

```
int r = st.executeUpdate("INSERT INTO Snack_Review ( s_id, score,viewer_comments)
SELECT  s_id ,"+insertSnackReviewScore+", '"+insertSnackReviewComments+"' FROM
Snacks WHERE s_name ='"+insertSnackReviewSname+"'");
```

Select snacks by refill date

```
"select Snacks.s_name, Refills.date_refill  from  Refills, Snacks where Refills.s_id =
Snacks.s_id and Refills.date_refill ='"+dateRefill+"'"
```

Select snacks by refill quantity

```
"select  Snacks.s_name, Refills.quantity_refill  from  Refills, Snacks  where Refills.s_id =
Snacks.s_id and Snacks.s_name ='"+quantRefil+"'"
```

Select company name by snack name

"select Snacks.s_name, Staff.company_name from Staff, Snacks, Refills where Snacks.s_id = Refills.s_id and Refills.SSN = Staff.SSN and Snacks.s_name = '"+comName+"'"

Select snacks by calories (using NOT EXISTS)

"select * from Vending_Machine where not exists (select * from Vending_Machine where isavailable<>'"+isavailablev+"')"

Select vending machine by snack name

"select * from Snacks, Vending_Machine where Snacks.v_id = Vending_Machine.v_id AND Snacks.s_name = '"+vbys+"'"

Select locations by building name

"select * from Location_on_Campus where building_name = '"+building_name+"'"

Select vending machine by region on campus (using EXCEPT)

"(select * from Location_on_Campus) except (select * from Location_on_Campus where region_on_campus <> '"+region_on_campus+"')"

Select vending machine by building name

"select v_id, Location_on_Campus.building_name from Location_on_Campus, Vending_Machine where Location_on_Campus.building_name = Vending_Machine.building_name AND Location_on_Campus.building_name = '"+vidbybuildingname+"'"

Select vending machine by review score

"select Vending_Review.v_id, Vending_Review.viewer_comments from Vending_Review where Vending_Review.score = '"+vscore+"'"

Select vending machine above some review score (using NOT EXISTS)

"select vr1.v_id, vr1.viewer_comments from Vending_Review as vr1 where vr1.score >= '"+vscore_above+"' AND NOT EXISTS (select * from Vending_Review as vr2 where vr2.score < '"+vscore_above+"' AND vr1.v_id=vr2.v_id)"

Select vending machine by vending id

"select Vending_Review.score, Vending_Review.viewer_comments from Vending_Review where Vending_Review.v_id = '"+vReviewsName+"'"

Insert your reviews to vending machine

int r = st.executeUpdate("insert into Vending_Review (vreview_id, v_id,score, viewer_comments) Values ("+rand_int2+","+insertvReviewvid+", "+insertvReviewScore+", '"+insertvReviewComments+"')");

7. Implementation

The SQL used to create the tables including all of the data inserted into the database. Please see attached codes final_project.sql, server.jsp and query.jsp for all codes used to make the GUI/run all the queries.

```
create table if not exists Location_on_Campus (
        building_name  character varying,
        region_on_campus character varying,
        open_time TIME,
        close_time TIME,
        floor_number integer,
        primary key(building_name)
);

create table if not exists Vending_Machine (
        v_id SERIAL,
        building_name character varying,
        capacity integer,
        last_filled date,
        isAvailable boolean,
        hasCaseCase boolean,
        hasCreditCard boolean,
        hasCash boolean,
        primary key(v_id),
        foreign key(building_name) REFERENCES Location_on_Campus
);

create table if not exists Snacks(
        s_id SERIAL,
        v_id int,
        s_name character varying NOT NULL,
        Price Money NOT NULL ,
   IsSweet Boolean,
        IsSalty Boolean,
        IsDrink Boolean,
        Expiration_date Date  NOT NULL ,
        isAvailable Boolean  NOT NULL,
        Primary key (s_id),
        foreign key(v_id) REFERENCES Vending_Machine

);

create table if not exists Vending_Review (
        vreview_id SERIAL,
        v_id integer,
        score integer,
        viewer_comments text,
        primary key(vreview_id),
        foreign key(v_id) REFERENCES Vending_Machine
);

create table if not exists Snack_Review (
        sreview_id SERIAL,
        s_id integer,
        score integer,
        viewer_comments text,
        primary key(sreview_id),
        foreign key(s_id) REFERENCES Snacks

);

create table if not exists Nutrition (
        s_id integer,
        calories integer,
        allergy_warns character varying,
        sugars integer,
        caffeine boolean,
        primary key(s_id),
        foreign key(s_id) REFERENCES Snacks

);

create table if not exists Staff (
        SSN CHAR(11),
        company_name character varying,
        primary key(SSN)
);

create table if not exists Refills (
        SSN CHAR(11),
        r_id SERIAL,
        s_id integer,
        date_refill DATE,
        quantity_refill integer,
        primary key(r_id),
        foreign key(SSN) REFERENCES Staff,
        foreign key(s_id) REFERENCES Snacks

);
```

Inserting locations on campus: [building_name, regional_campus, opentime, closetime, floor_number]

```
INSERT INTO Location_on_Campus VALUES ('Crawford Hall', 'Main
Quad','08:00:00','18:00:00','1');
INSERT INTO Location_on_Campus VALUES ('KSL', 'Mather Quad','00:00:00','24:00:00','2');
```

INSERT INTO Location_on_Campus VALUES ('Sears Library', 'Main Quad','08:00:00','19:00:00','3');
INSERT INTO Location_on_Campus VALUES ('Nord Hall', 'Main Quad','08:00:00','19:00:00','3');
INSERT INTO Location_on_Campus VALUES ('Rockefeller Building', 'Main Quad','08:00:00','19:00:00','1');

Inserting vending machines: [vid(serial), building name, capacity, lastfilleddate, is_avilabel, casecash, credit, cash]
INSERT INTO Vending_Machine VALUES (DEFAULT,'Crawford Hall','80','2019-09-11','true','true','true','true') ;
INSERT INTO Vending_Machine VALUES (DEFAULT,'Crawford Hall','80','2019-10-11','true','true','true','true') ;
INSERT INTO Vending_Machine VALUES (DEFAULT,'KSL', '50','2019-10-11','true','true','true','true') ;
INSERT INTO Vending_Machine VALUES (DEFAULT,'KSL', '50','2019-09-11','true','true','true','true') ;
INSERT INTO Vending_Machine VALUES (DEFAULT,'KSL', '50','2019-10-11','true','true','true','true') ;

Inserting snacks: [sid, vid, sname, price, isWeet, isSalty, isDrink, expiration date, isAvailable]
INSERT INTO Snacks VALUES (DEFAULT,'1','Rice Crispy Treat', '2.5','true','false','false','2022-10-10','true') ;
INSERT INTO Snacks VALUES (DEFAULT,'1','Pop Tarts Frosted Strawberry','2.5','true','false','false','2022-10-10','true');
INSERT INTO Snacks VALUES (DEFAULT,'1','Doritos', '2.5','false','true','false','2022-10-10','true');
INSERT INTO Snacks VALUES (DEFAULT,'1','Skittles', '2.5','true','false','false','2022-10-10','false');
INSERT INTO Snacks VALUES (DEFAULT,'2','Rice Crispy Treat', '2.5','true','false','false','2020-11-10','false');
INSERT INTO Snacks VALUES (DEFAULT,'2','Rice Crispy Treat', '2.5','true','false','false','2020-10-10','true') ;
INSERT INTO Snacks VALUES (DEFAULT,'2','Pop Tarts Frosted Strawberry','2.5','true','false','false','2022-10-10','true');
INSERT INTO Snacks VALUES (DEFAULT,'2','Doritos', '2.5','false','true','false','2022-09-10','true');
INSERT INTO Snacks VALUES (DEFAULT,'3','Skittles', '2.5','true','false','false','2022-09-10','true');
INSERT INTO Snacks VALUES (DEFAULT,'4','Starbuck frappuccino Mocha', '2.5','true','false','true','2020-06-15','true');
INSERT INTO Snacks VALUES (DEFAULT,'4','Starbuck frappuccino Coffee', '2.5','true','false','true','2020-06-15','false');

Inserting staff: [SSN, company, refile date, quantities, s_id]

```
INSERT INTO Staff VALUES ('123456789','A&M Vending Machine Sales');
INSERT INTO Staff VALUES ('223456789','A&M Vending Machine Sales');
INSERT INTO Staff VALUES ('323456789','A&M Vending Machine Sales');
INSERT INTO Staff VALUES ('423456789','D & S Vending Inc');
INSERT INTO Staff VALUES ('523456789','D & S Vending Inc.');
INSERT INTO Staff VALUES ('623456789','D & S Vending Inc.');
INSERT INTO Staff VALUES ('723456789','D & S Vending Inc.');
```

Inserting nutrition: [s_id, calories, allergies, sugar, caffeine]

```
INSERT INTO Nutrition VALUES ('1','160','false','18','0');
INSERT INTO Nutrition VALUES ('2','200','false','40','0');
INSERT INTO Nutrition VALUES ('3','250','false','0','0');
INSERT INTO Nutrition VALUES ('4','250','false','30','0');
```

Inserting vending review: [reviewid, vid, score, comment]

```
INSERT INTO Vending_Review VALUES (DEFAULT,'3','5','my favorite vending machine at ksl.
This is the only one that accept debit card');
INSERT INTO Vending_Review VALUES (DEFAULT,'4','1','it eat my money and not giving me
my candie, will never use this one');
INSERT INTO Vending_Review VALUES (DEFAULT,'5','1','my candies all melted ');
INSERT INTO Vending_Review VALUES (DEFAULT,'2','5','this is a good one because nobody
know this machine so their always candy');
INSERT INTO Vending_Review VALUES (DEFAULT,'4','1','they need refill I NEED MY
COFFEE');
INSERT INTO Vending_Review VALUES (DEFAULT,'4','5','the new melt chocolate mocha is
really good');
```

Inserting snack review: [reviewid, s_id score, comments]

```
INSERT INTO Snack_Review ( s_id, score,viewer_comments)
SELECT  s_id ,1, 'too sweet not for me'
FROM Snacks
WHERE s_name ='Rice Crispy Treat';

INSERT INTO Snack_Review ( s_id, score,viewer_comments)
SELECT  s_id ,5, 'loved the strawberry'
FROM Snacks
WHERE s_name ='Pop Tarts Frosted Strawberry';

INSERT INTO Snack_Review ( s_id, score,viewer_comments)
SELECT  s_id ,4, 'rainbow color made me so happy '
FROM Snacks
WHERE s_name ='Skittles';
```

*Note: we want to make sure that each time a viewer leaves a review for a specific snack, we will be able to add their review to all the snacks with the same name. That is why we are inserting snack reviews by select s_ids from snacks where s_name name is equal to the inputting snack name (this is collected from the web page end).

Insert into refill tabel: [SSN; refill id(serial), s_id, data_refill, quantity_refill]

```
INSERT INTO Refills Values ('123456789',DEFAULT,'1','2019-12-11','20');
INSERT INTO Refills Values ('223456789',DEFAULT,'2','2019-12-11','20');
INSERT INTO Refills Values ('323456789',DEFAULT,'1','2019-12-10','20');
```

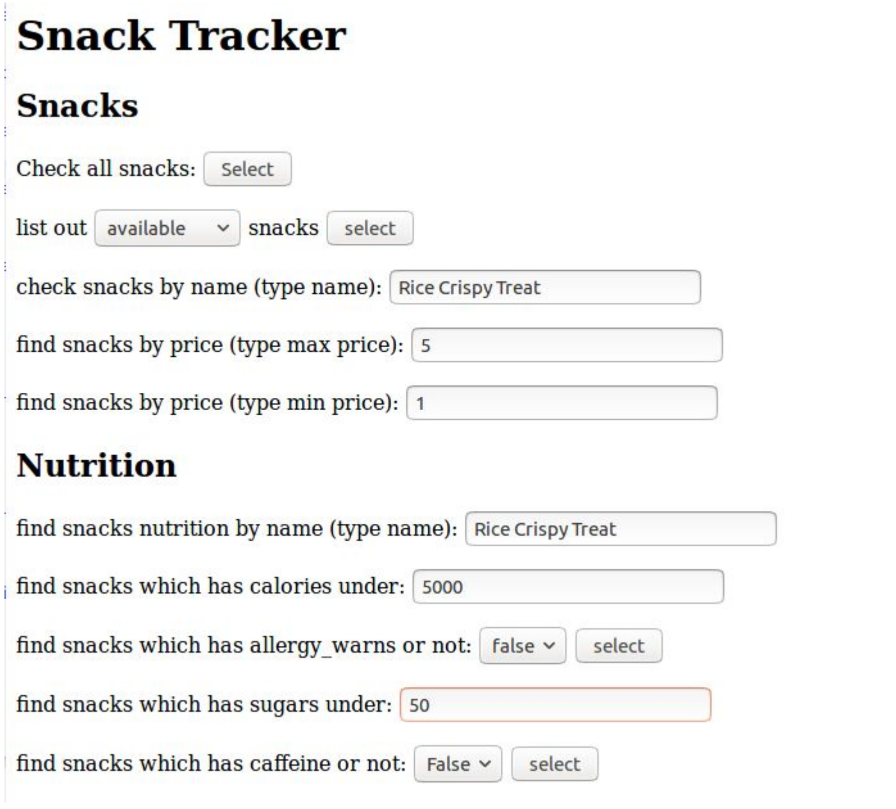
8. The role and contributions of each team member

All team members contributed to the final project. We met numerous times in KSL and talked continuously on GroupMe. Creating and revising the ER diagram as well as forming the SQL queries was a collaborative effort. We split up work after the initial report based on interest and available resources. Randolf took the lead on implementing the database with JSP and testing the queries. Ruyuan took the lead on writing the SQL schemas and composing the presentation slides as well as inserting our data into the database. Jared took the lead on creating and guiding the application topic as well as building out the database design and composing the final report. We all worked together on refactoring for project revisions and proofreading.

9. What you have learned from this project—over and above from what is covered in the course.

Going to lectures, doing practice problems, and studying the textbooks does a great job of providing a theoretical understanding of databases. However, this project forced us to actually apply all of our theoretical knowledge by modeling and buildinging a real-life database from beginning to end. This allowed us to experience the full build process/cycle, which is highly valuable as we may be expected to do something similar once we are working full-time in industry. On a lower level, we specifical learned how to not only design a database, but also how to actually implement it with SQL and Java (JSP) to make a working final product. Additionally, working in a group setting helped us strengthen our interpersonal and communication skills as well as practice working simultaneously on a project without breaking/interfering with each others parts/work.

Appendix of Screenshots



Snack Tracker

Snacks

Check all snacks: Select

list out available snacks select

check snacks by name (type name): Rice Crispy Treat

find snacks by price (type max price): 5

find snacks by price (type min price): 1

Nutrition

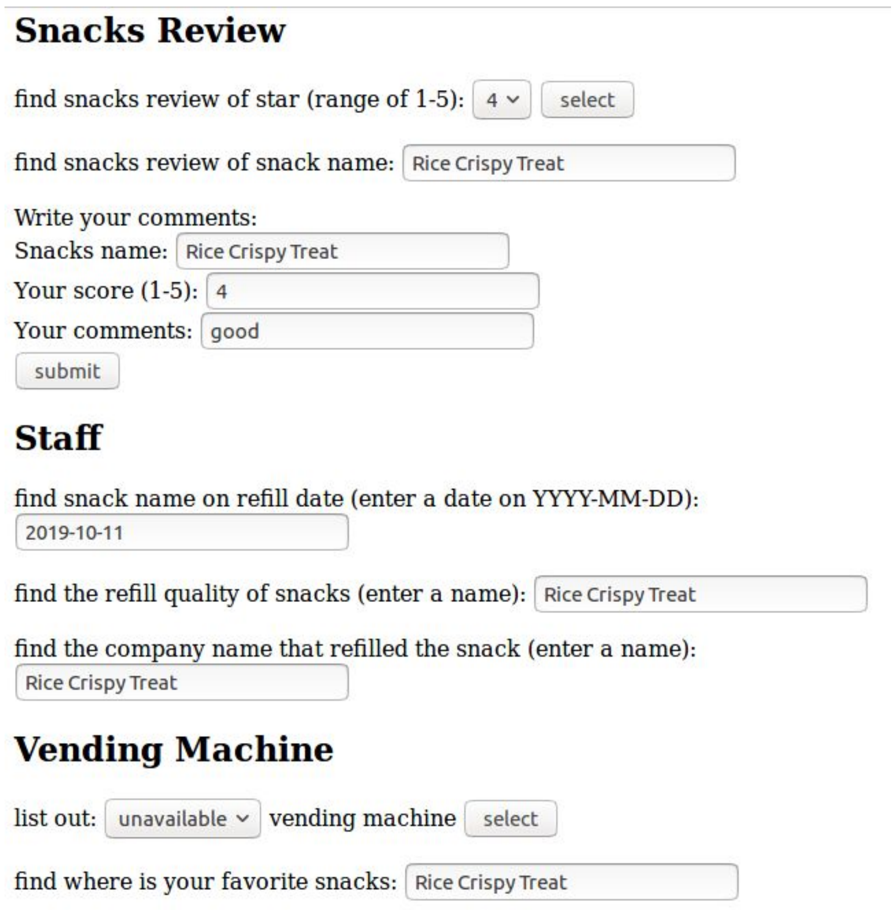
find snacks nutrition by name (type name): Rice Crispy Treat

find snacks which has calories under: 5000

find snacks which has allergy_warns or not: false select

find snacks which has sugars under: 50

find snacks which has caffeine or not: False select



Snacks Review

find snacks review of star (range of 1-5): 4 select

find snacks review of snack name: Rice Crispy Treat

Write your comments:
Snacks name: Rice Crispy Treat
Your score (1-5): 4
Your comments: good
submit

Staff

find snack name on refill date (enter a date on YYYY-MM-DD):
2019-10-11

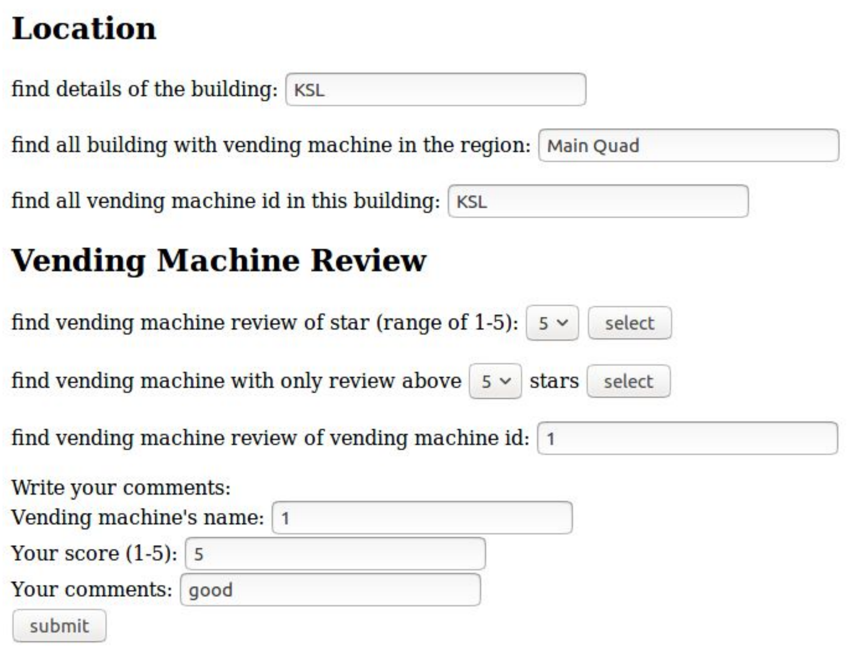
find the refill quality of snacks (enter a name): Rice Crispy Treat

find the company name that refilled the snack (enter a name):
Rice Crispy Treat

Vending Machine

list out: unavailable vending machine select

find where is your favorite snacks: Rice Crispy Treat



Location

find details of the building: KSL

find all building with vending machine in the region: Main Quad

find all vending machine id in this building: KSL

Vending Machine Review

find vending machine review of star (range of 1-5): 5 select

find vending machine with only review above 5 stars select

find vending machine review of vending machine id: 1

Write your comments:
Vending machine's name: 1
Your score (1-5): 5
Your comments: good
submit