**Final Project Write Up**

***\*To Access Admin Account\****

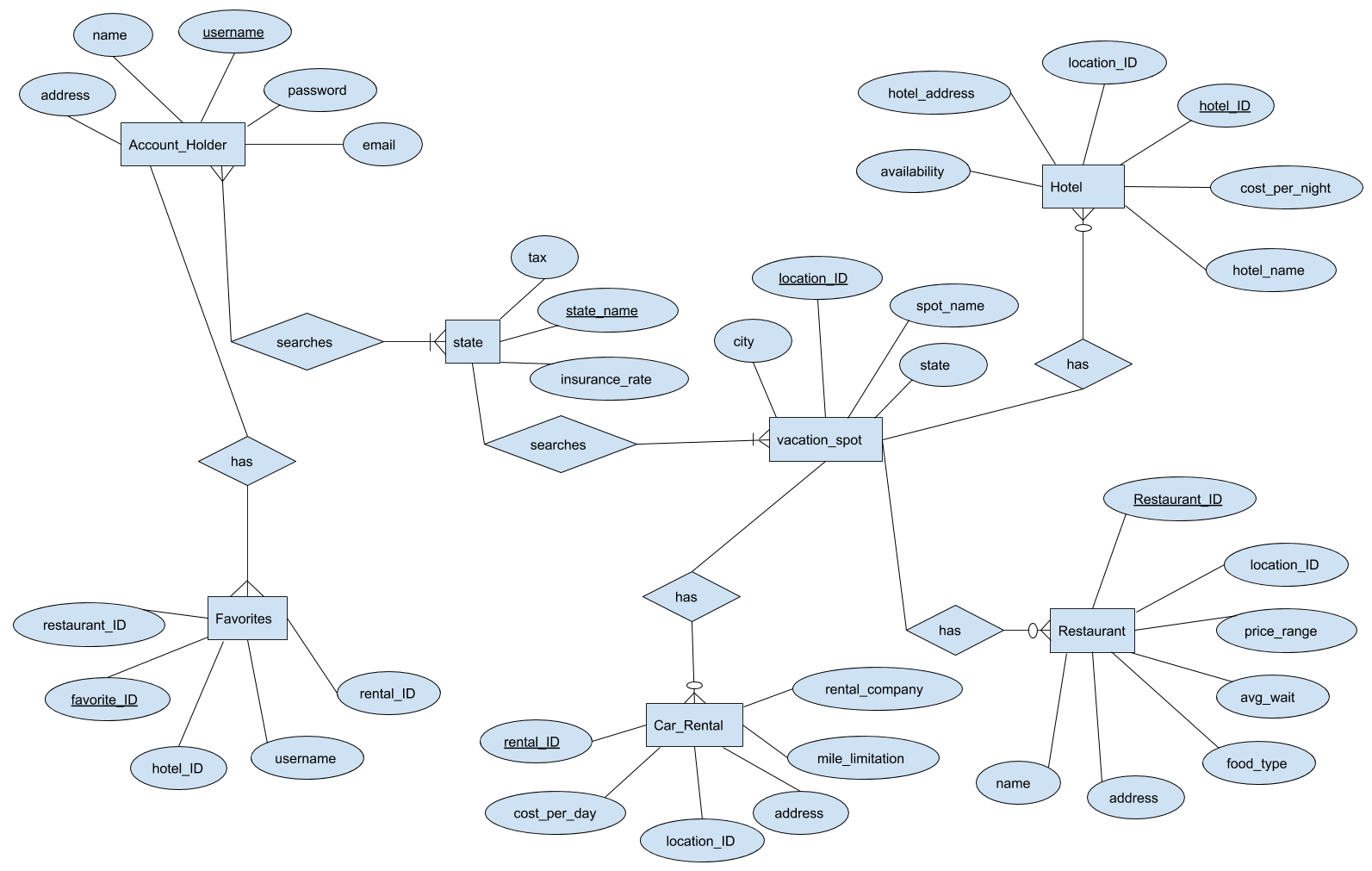
***Username: admin***

***Password: admin***

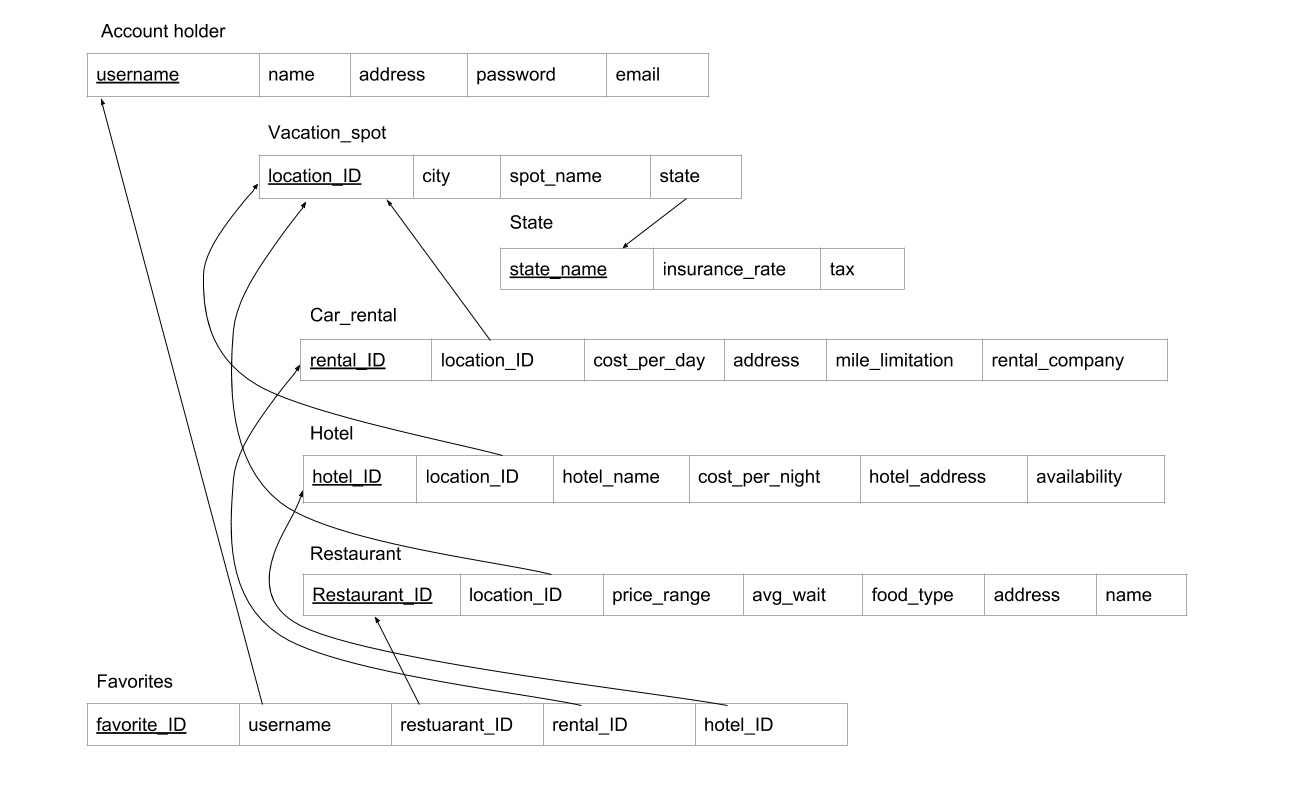
Languages used

JSP, SQL, and HTML

ER Diagram



Relational Schema



Rubric Criteria Explanation

(1) Minimum of 3,000 tuples across all tables.

One can see there are over 3,000 tuples for vacation spots, hotels, restaurants, and car rentals.

(2) The ability to add a tuple to your database from the front end.

Notice that there is a registration page which will create tuples based on the users input so this will add tuples to account holder . Also, we can add tuples from the favorites page.

First we check if the tuple already exists with this query:

SELECT \* FROM Account\_Holder WHERE username ='" + username1 +"'

If not, then we use:

INSERT INTO Account\_Holder(name, address, email,username,password)"

+ " VALUES (?, ?, ?, ?, ?)

To insert it.

Similar process used for favorites.

(3) The ability to query the table that you added the tuple to from the front end.

We have an admin page that allows them to view all users and accounts from the account holder table:

SELECT \* FROM Account\_Holder WHERE username!='" + user+"'

(4) The ability to demonstrate a minimum of 6 unique and complex queries from the front end.

We used complex queries from the admin page to allow them to view certain statistics based on their selection. These include:

# OF ACCOUNTS WITH COMMON EMAIL

"select Sum(t1.num) as cnt from (SELECT distinct email, COUNT(email)as num FROM Account\_Holder GROUP BY email having COUNT(email)>1 and email!='')as t1";

AVERAGE NUMBER OF HOTELS PER STATE

"select round( avg(t1.num),0)as cnt from (SELECT distinct state, COUNT(hotel\_ID)as num FROM hotel natural join vacation\_spot GROUP BY state)as t1";

AVERAGE NUMBER OF CAR RENTAL COMPANIES PER STATE

"select round( avg(t1.num),0)as cnt from (SELECT distinct state, COUNT(rental\_ID)as num FROM car\_rental natural join vacation\_spot GROUP BY state)as t1";

AVERAGE NUMBER OF RESTAURANTS PER STATE

"select round( avg(t1.num),0)as cnt from (SELECT distinct state, COUNT(Restaurant\_ID)as num FROM restaurants natural join vacation\_spot GROUP BY state)as t1";

STATE WITH LOWEST NUMBER OF RESTAURANT

"select st.state as state from (select t1.num as num,state from (SELECT distinct state, COUNT(Restaurant\_ID)as num FROM restaurants natural join vacation\_spot GROUP BY state)as t1) as st ,(select min(t1.num)as m from (SELECT distinct state, COUNT(Restaurant\_ID)as num FROM restaurants natural join vacation\_spot GROUP BY state)as t1) as tmin where tmin.m=st.num";

STATE WITH LOWEST NUMBER OF CAR RENTAL COMPANIES

"select st.state as state, st.num from (select t1.num as num,state from (SELECT distinct state, COUNT(rental\_ID)as num FROM car\_rental natural join vacation\_spot GROUP BY state)as t1) as st ,(select min(t1.num)as m from (SELECT distinct state, COUNT(rental\_ID)as num FROM car\_rental natural join vacation\_spot GROUP BY state)as t1) as tmin where tmin.m=st.num";

STATE WITH LOWEST NUMBER OF HOTELS

"select st.state as state from (select t1.num as num,state from (SELECT distinct state, COUNT(hotel\_ID)as num FROM hotel natural join vacation\_spot GROUP BY state)as t1) as st ,(select min(t1.num)as m from (SELECT distinct state, COUNT(hotel\_ID)as num FROM hotel natural join vacation\_spot GROUP BY state)as t1) as tmin where tmin.m=st.num";

STATE WITH highest NUMBER OF RESTAURANT

"select st.state as state from (select t1.num as num,state from (SELECT distinct state, COUNT(Restaurant\_ID)as num FROM restaurants natural join vacation\_spot GROUP BY state)as t1) as st ,(select max(t1.num)as m from (SELECT distinct state, COUNT(Restaurant\_ID)as num FROM restaurants natural join vacation\_spot GROUP BY state)as t1) as tmin where tmin.m=st.num";

STATE WITH highest NUMBER OF CAR RENTAL COMPANIES

"select st.state as state, st.num from (select t1.num as num,state from (SELECT distinct state, COUNT(rental\_ID)as num FROM car\_rental natural join vacation\_spot GROUP BY state)as t1) as st ,(select max(t1.num)as m from (SELECT distinct state, COUNT(rental\_ID)as num FROM car\_rental natural join vacation\_spot GROUP BY state)as t1) as tmin where tmin.m=st.num";

STATE WITH highest NUMBER OF HOTELS

"select st.state as state from (select t1.num as num,state from (SELECT distinct state, COUNT(hotel\_ID)as num FROM hotel natural join vacation\_spot GROUP BY state)as t1) as st ,(select max(t1.num)as m from (SELECT distinct state, COUNT(hotel\_ID)as num FROM hotel natural join vacation\_spot GROUP BY state)as t1) as tmin where tmin.m=st.num";

USER WITH highest NUMBER OF FAVORITES

"select username from (SELECT distinct username, COUNT(favorite\_ID)as num FROM favorites GROUP BY username )as user,(select max(cc.c) as num from (SELECT COUNT(favorite\_ID)as c FROM favorites GROUP BY username )as cc)as cnt where user.num=cnt.num";

(5) The ability to demonstrate two verifiable patterns within your data from the front end.

We have incorporated an insurance rate and tax rate by state. These rates affect the total cost of renting a hotel or car, by incorporating an added “Tax” or “Insurance Cost” value.

We use:

SELECT \* FROM hotel, state, vacation\_spot WHERE hotel.location\_ID='" + lid+"' AND vacation\_spot.location\_ID='" + lid+"' AND state.state\_name=vacation\_spot.state

Which allows us to determine the percent tax .

We then multiply this value to the cost of the respective hotel’s price per night to determine the cost of the tax.

A similar procedure is done to determine the value for renter’s insurance in each state altering the cost of insurance someone would have to pay in a certain state depending on the cost per day of the rental company..

(6) The ability to demonstrate two integrity constraints in from your front end.

You cannot register with the same username more than once, nor can you add the same favorites multiple times.

We check if the tuple with the same username already exists with this query:

SELECT \* FROM Account\_Holder WHERE username ='" + username1 +"'

If it is we alert the user that the username already exists

For favorites, we use:

SELECT COUNT(\*) as cnt FROM favorites WHERE username ='" + user +"'and restaurant\_ID="+rid;

Which, we use to return the number of tuples corresponding to the favorite we are adding

If this number is 1 that means the user has already added this to favorites, and we alert the user.

(7) The ability to create a functional and easily navigated front end.

Upto your judgement. We believe that our website is easily navigable and the front-end has a pleasant display that attracts consumers.