CS 157A Final Project

Application Name/Team Name: Blue Swift

Team Members:
Archana Yadawa – *ayadawa@gmail.com*
Lingfang Gao – *lingfang.gao@gmail.com*
Sandeep Samra – *ssamra1094@gmail.com*
Daksha Divakar – *dakshadivak@gmail.com*
Sridivya Kondapalli – *sridivya20@gmail.com*

Project Overview:

Our application will be a platform for apartment searching. This application will allow users to easily search for apartments according to their requirements/expectations. It will filter out options for users and provide them with the best apartment choices.

Relations:

Suite(suiteId, roomSharing, price, specifications, petsAllowed, numberOfRooms)

Apartment(name, longitude, latitude, city, state)

Tenant(name, phoneNumber, description)

Hobbies(name)

ModeOfTransportation(name, streetAddress)

Bus(modeName, route)

Train(modeName, route)

Stops(name, stopTime, longitude, latitude)

Stores(name, description, longitude, latitude)

SchoolSystems(name, longitude, latitude, rating)

Grade(gradeLevel)

AptHasTrans(aptLongitude, aptLatitude, modeName)

AptHasSchool(aptLongitude, aptLatitude, schoolLongitude, schoolLatitude)

AptHasSuites(aptLongitude, aptLatitude, suiteId)

TenantHasSuite(tenPhoneNum, suiteId)

BusStops(modeName, route, stopName, stopTime, longitude, latitude)

TrainStops(modeName, route, stopName, stopTime, longitude, latitude)

AptHasStore(aptLongitude, aptLatitude, storeLongitude, storeLatitude)

TenantHobbies(tenantPhoneNumber, hobbyName)

SchoolSystemHasGrades(schoolLongitude, schoolLatitude, gradeLevel)

Updated Create Statements:

```
CREATE TABLE Suite(
   suiteId int NOT NULL AUTO_INCREMENT PRIMARY KEY,
   roomSharing boolean,
   price decimal(12,2),
   specifications varchar(255),
   petsAllowed boolean,
   numberOfRooms int
);

CREATE TABLE Apartment(
   name varchar(255),
   longitude double,
   latitude double,
   city varchar(255),
   state varchar(2) CHECK (state = UPPER(state)),
   PRIMARY KEY(longitude,latitude)
);

CREATE TABLE Tenant(
```

```
    name varchar(255),
    phoneNumber varchar(12) PRIMARY KEY,
    description varchar(255)
);

CREATE TABLE Hobbies(
    name varchar(255) PRIMARY KEY
);

CREATE TABLE ModeOfTransportation(
    name varchar(255) PRIMARY KEY,
    streetAddress varchar(255)
);

CREATE TABLE Bus (
    modeName varchar(255),
    route int,
        PRIMARY KEY(modeName, route)
);

CREATE TABLE Train(
    modeName varchar(25),
    route int,
    PRIMARY KEY (modeName, route)
);

CREATE TABLE Stops(
    name varchar(255),
    stopTime varchar(255),
    longitude double,
    latitude double,
    PRIMARY KEY (name, stopTime,longitude,latitude)
);

CREATE TABLE Stores(
    name varchar(255),
    description varchar(255),
    longitude double,
    latitude double,
    PRIMARY KEY(longitude, latitude)
);

CREATE TABLE SchoolSystems (
    name varchar(255),
    longitude double,
```

```
    latitude double,
    rating decimal(4,2),
    PRIMARY KEY(longitude,latitude)
);

CREATE TABLE Grade(
    gradeLevel int PRIMARY KEY
);


CREATE TABLE AptHasTrans(
    aptLongitude double,
    aptLatitude double,
    modeName varchar(255),
    PRIMARY KEY(aptLongitude, aptLatitude, modeName),
    FOREIGN KEY(aptLongitude,aptLatitude) REFERENCES
Apartment(longitude,latitude)ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(modeName) REFERENCES ModeOfTransportation(name) ON
DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE AptHasSchool(
    aptLongitude double,
    aptLatitude double,
    schoolLongitude double,
    schoolLatitude double,
    PRIMARY KEY(aptLongitude, aptLatitude, schoolLongitude, schoolLatitude),
    FOREIGN KEY(aptLongitude,aptLatitude) REFERENCES
Apartment(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(schoolLongitude,schoolLatitude) REFERENCES
SchoolSystems(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE AptHasSuites(
    aptLongitude double,
    aptLatitude double,
    suiteId int,
    PRIMARY KEY(suiteId),
    FOREIGN KEY(aptLongitude,aptLatitude) REFERENCES
Apartment(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (suiteId) REFERENCES Suite(suiteId) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

```
CREATE Table TenantHasSuite(
   tenPhoneNum varchar(255) PRIMARY KEY,
   suiteId int,
   FOREIGN KEY(tenPhoneNum) REFERENCES Tenant(phoneNumber) ON DELETE
CASCADE ON UPDATE CASCADE,
   FOREIGN KEY(suiteId) REFERENCES Suite(suiteId) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE BusStops(
   modeName varchar(255),
   route int,
   stopName varchar(255),
   stopTime varchar(255),
   longitude double,
   latitude double,
   PRIMARY KEY(modeName, route, stopName, stopTime, longitude, latitude) ,
   FOREIGN KEY(modeName,route) REFERENCES Bus(modeName,route) ON
DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY(stopName,stopTime,longitude,latitude) REFERENCES Stops(name,
stopTime,longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE
 );

CREATE TABLE TrainStops(
   modeName varchar(255),
   route int,
   stopName varchar(255),
   stopTime varchar(255),
   longitude double,
   latitude double,
   PRIMARY KEY(modeName, route, stopName, stopTime, longitude, latitude) ,
   FOREIGN KEY(modeName,route) REFERENCES Train(modeName,route) ON
DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY(stopName,stopTime,longitude,latitude) REFERENCES Stops(name,
stopTime,longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE
 );


CREATE TABLE AptHasStore(
   aptLongitude double,
   aptLatitude double,
   storeLongitude double,
   storeLatitude double,
        PRIMARY KEY (aptLongitude , aptLatitude, storeLongitude, storeLatitude),
```

```
        FOREIGN KEY (aptLongitude, aptLatitude) REFERENCES
Apartment(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE ,
        FOREIGN KEY (storeLongitude, storeLatitude) REFERENCES
Stores(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE TenantHobbies(
        tenantPhoneNumber varchar(255),
        hobbyName varchar(255),
        PRIMARY KEY(tenantPhoneNumber, hobbyName),
        FOREIGN KEY (tenantPhoneNumber) REFERENCES Tenant(phoneNumber)
ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (hobbyName) REFERENCES Hobbies(name) ON DELETE
CASCADE ON UPDATE CASCADE
);


CREATE TABLE SchoolSystemHasGrades(
        schoolLongitude double,
   schoolLatitude double,
        gradeLevel int,
        PRIMARY KEY( schoolLongitude, schoolLatitude, gradeLevel),
        FOREIGN KEY(schoolLongitude, schoolLatitude ) REFERENCES
SchoolSystems(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(gradeLevel) REFERENCES Grade(gradeLevel) ON DELETE
CASCADE ON UPDATE CASCADE

);
```

Question 2:

Explain whether or not somebody who is truly working in the domain of your application would use your web-enabled database, or if there is something missing or intrinsically complex that will deter them from using it. (Is your web-enabled database not powerful enough for the real world, or too complicated for the real world?) For example, if you are in the books domain, would a bookseller use your database? It would be ideal if you had access to a real application domain person; you could just ask him/her to visit your web page and give you feedback. Alternately, look at "similar" systems or web pages and see if they are doing something differently and if your approach is better (or worse) and why. Include your answer to this question in your project report.

Answer:

For our project, we worked with apartments. For the most part, because it lacks some functionality such as the ability to save apartment searches, filter by more detailed,complex options and the ability to register a new apartment with the website, the website is less powerful when compared to professional sites such as apartmentsearch.com. However, the current progress on the website, certainly enables it to be a viable option for apartment searchers given a little more development time.

In addition, there were a few things in our application that we did poorly on. One of these was the number of post requests and get requests that were made to the database. Ideally, when the post requests become repetitive, it would be a nice addition to have a cache of some sort to store results of common post requests. If a similar post request was made later on, then the website can directly use the cache to return this to the client. This makes the access time faster and makes the website load faster as well. Of course, this also means that any updates to certain parts of the website, for instance the number of rooms available, means that the cache also needs to be updated.
Also, as it is the current website fails in issues concerning security. Ideally, every input field needs to have a javascript function attached to ensure that sql injection doesn't take place.

In terms of the actual database, the complexity of the design and the amount and variety of data it can hold, fall closely in line with a real world apartment search website. Our database schema as it is, can accommodate more options. We also have a large dataset of both real data and fake data that mimics a real world situation.

Query Constraints:
We have an attribute based check on state in apartment. This check makes sure that all letters are uppercase.

We have referential integrity checks with all our relation tables such as AptHasTrans, AptHasSchool, AptHasStore etc..

Other Notes:
We used angular.js and bootstrap for the front end. PHP for the backend.

Php files can be found in the "config" directory
main app.js can be found in the "app" directory

Currently in the search page:
filtering by School name, rating, grade level, hasNearbyTrans is unsupported

---- Known Bugs: ----
when an apartment is clicked (the apartment information cell has to be clicked), suites and tenant information is displayed, to remove this display, click again on the apartment cell

if you click on the apartment cell of one apartment and another apartment cell of another
apartment without closing the first suite display,
the most recently selected apartment suite information is displayed on all open suite displays
(this is because in angular, I set the clicked apartment as the selected item and
a http post request is made with apt long and lat)

===== QUERIES USED AND WHERE TO FIND THEM ====
Queries for filtering apartments can be found in Apartment.php

OtherQueries.php contains miscellaneous queries such as countNearbyStores(), countNearbyTrans() etc.

Suite.php contains queries for suite information, the queries in this class combine the Suite, AptHasSuites, Tenant,
TenantHasSuite relations

AdHoc Queries are executed through the executeQuery.php file
WARNING -- AdHoc allows any kind of query, including ones for dropping the database, dropping the table etc.

OTHER BUGS:

When using Load DATA INFILE, we encountered an issue with AptHasTrans table's data.

When manually entered or entered using Insert statements, the data inside the AptHasTrans.csv file passed foreign key constraints and no errors were encountered

however, when using LOAD DATA INFILE, mysql began to complain about foreign key constraint errors.
To bypass this error, we used SET FOREIGN_KEY_CHECKS=0 at the beginning of the loading, and reset SET FOREIGN_KEY_CHECKS=1 at the end.

Contributions:

Sridivya Kondapalli: Front-end UI, PHP files to connect to backend, wrote sql queries to access db, report
LingFang Gao: Report, Sql queries to access db
Archana Yadawa: Report, sql queries to access db
Daksha Divakar: Collected all the data for all the tables
Sandeep Samra: Report