CS-GY 6083 section INET
Spring 2024

Project Part 2 and Draft of Part 3

Part 2: Due Apr 7, 2024

Shortly after Part 1 is due, the TAs will post an ER diagram, schema diagram, and table definitions for Roomio that you should use for the rest of the project. (Previously, I indicated that you would create the schema and table definitions in Part 2, but we have decided to give them to you, so you can begin writing application code (Part 3) sooner.)

Part 2 is a milestone to make sure you're working on the project and making progress.
Write INSERT statements to insert data into a few of the tables.

Review the general description of the project in Part 1 and read the requirements for Part 3.

1. Each team member should write application code for at least one feature. This will not be graded for correctness, but you should test it
2. Insert some data into relevant tables and test your code
3. Hand in evidence that you've done this (e.g. GitHub link; more details TBD.) We will not check this carefully for correctness but will give it a quick look to see that you're on track. *If you are having trouble you should seek help well before the part 2 deadline.*

Part 3 (Draft) *Stay tuned to EdStem in case any clarifications or changes are posted.*
Due Apr 21, 2024

In part 3 of the project, you will use the table definitions we will post to implement application code for Roomio as a web-based application. You may use Python, PHP, Java, node.js, or Go. If you'd like to use some other language, check with the TAs by 03/28/2023. *You must use prepared statements if your application language supports them.*

I have supplied Python code for a sample application that includes most of the constructs you'll need in order to do this assignment, along with videos that explain the code in detail. If you do not have prior experience with web and/or database application programming, you should study these and you may modify the provided sample code to implement Roomio.

You may continue to work alone or with the one or two others with whom you worked on parts 1 and/or 2 or you may form a team (two or three students, total): Any new teams or additions to a team must be reported to the TAs (by e-mail or on Ed Discussion by March 29.) Everyone on a team is expected to understand all the code.

In part 3, you'll write application code to implement Roomio as a web application.

Roomio should support the following use cases:

Required Features:

1. **Login & User Session Handle:** Users should be able to **register a new account**. Besides, only registered users are allowed to **log in** to the website. Roomio should add salt to the hashed password whenever a password is required at the database level. On login, if the salted and hashed password matched the record in the database, the user should be then redirected to the main page. If not, the user should be informed of the failure and no session will be created. Your application should support **user sessions** which means the relevant data should be stored in the user session on a successful login.
2. **Search Certain Apartment Units:** The user should be able to search certain apartment units based on the following options:
    a. Given the exact building name and company name, the application should return a list of units for rent and their basic information (monthly rent, square footage, available date for move-in, xbxb etc.)
    b. Based on the registered information of the pet of user, the system shows whether whether the pet is allowed.
3. **Register Pet:** The user should be able to **register** pets associated with their accounts. Besides, they should also be able to **edit** registered pets' information.
4. **Post and View Interests:** When viewing a specific apartment unit, the user should be able to **view** others' interests so that the user can join the interest (You are not required to implement the join feature) or **post** their interest to the unit.
5. **Display Unit and Building Info:** There should be some way for a user to search for a specific building or unit and get information about the building/unit: Roomio should display detailed information including address, year built, a summary of amenities and number of available units for rent (for building), monthly rent, square footage, available date for move-in, xbxb (for unit).
6. **Necessary Security Mechanisms:** You should have the necessary mechanism to prevent SQL Injection and XSS (cross-site script) attacks.

Additional Features: In addition to the required features, you must implement implement 2* numberOfGroupMember additional features. (That is to say, 2 additional features if you're working individually; 4 for a group of two; and 6 for a group of three). Here are some possible additional features. Some of them will require that you design extra tables with additional attribute; those are marked with *. (You may choose other features instead, but they should have similar complexity to these; please check with the prof or TAs if you're not sure.)

7. **More Advanced Search of Units:** The user provides information like expected monthly rent or requirements for specific amenities like an in-door washing machine (unit amenities) or gym (public amenities), and the application returns units that meet the requirements. If there is no match, the application should tell the user to modify the search requirements.
8. **Search Interest:** The user should be able to search for an interest in a certain unit based on the move-in date and roommate count attributes. The user should also be able

to have a look at the information of the initiator to decide on whether to contact for renting together.

9. **Estimate Monthly Rent:** The user can select a zipcode and the number of rooms (xbxb) requirement. The application should calculate the average monthly rent that suits the requirements of the user.
10. **Favorite*:** The user can add certain units to their **favorite** to quickly see the details of the unit through a specific entry.
11. **Extra View on the rent price:** For a unit, together with the price, display the average price of similar footage (footage less than 10% in difference) within the same city.
12. **Recommend System:** For a unit, the system would recommend similar units within and outside the same buildings. (You have the freedom to design what is “similar”)
13. **Comment System*:** The user can leave a comment and rating for certain units.
14. **Join the Interest*:** The user can join a certain interest group. (No need to simulate the mechanism of accepting the interest in the initiator end. Just assume automatic join. But the initiator should be able to see someone join his or her interest.)