

# Spring 2017: Database Systems, Project

## 1 Overview

In this project, you will design, implement, and document a database system for an Airbnb-like system. You will carry out this project in a team of two. We call this system “Uotel”.

## 2 Application Specification

A description of the major functions and data items follows. Some functions will be provided to the customers and others to the admin user of the system.

### 2.1 Basic Data

The system should manage information about temporary housing (TH) of various categories such as apartments, SFH, condol; (registered) users and THs they have visited. It should also store information about user opinions and TH ratings. While you are working on the project, keep in mind that these items are a minimal set of requirements.

**TH Data:** A unique ID for each TH, name, address, URL, telephone number, year built, dates available (there may be multiple periods a TH is available), price (an estimated expected price per person per day), keywords, category. You should add additional information if necessary such as one (or more) picture(s) associated with a TH (but not required).

**User Data:** For each registered user, you need to maintain: his/her full name, login name, password, address, phone number, THs he/she has visited and information regarding each visit (such as dates, cost as total \$ spent, and number of person in the party). You should also record the TH(s) he/she has listed.

**Opinions:** Users can provide ‘feedback’ for a TH, as a score (1-10) along with optional short text. Users can also rate other users’ feedback as ‘useless’, ‘useful’, ‘very useful’; finally, they are allowed to declare other users as ‘trusted’ or ‘not-trusted’.

### 2.2 Functionality of the system

The following set of events and queries should be handled by the your system:

- 1) [5pts] **Registration:** Registration: a new user has to provide the appropriate information; he/she can pick a login-name and a password. The login name should be checked for uniqueness.
- 2) [5pts] **Reserve:** After registration, a user can record a reservation to any TH (the same user may reserve the same TH multiple times from different available dates). Each user session (meaning each time after a user has logged into the system) may add one or more researvations, and all reservations added by a user in a user session are reported to him/her for the final review and confirmation, before they are added into the database.
- 3) [5pts] **New TH:** A user may record the details of a new PH, and may update the information regarding an existing PH he/she owns.
- 4) [5pts] **Stays:** A user can record a stay at any TH (the same user may stay at the same TH multiple times). Each user session (meaning each time after a user has logged into the system) may add one or more stays, and all stays added by a user in a user session are reported to him/her for the final review and confirmation, before they are added into the database. Note that a user may only record a stay at a TH during a period he/she has a reservation.
- 5) [5pts] **Favorite recordings:** Users can declare a TH as his/her favoriate place to stay.

- 6) [5pts] **Feedback recordings:** Users can record their feedback for a TH. We should record the date, the numerical score (0= terrible, 10= excellent), and an optional short text. No changes are allowed; only one feedback per user per TH is allowed.
- 7) [5pts] **Usefulness ratings:** Users can assess a feedback record, giving it a numerical score 0, 1, or 2 ('useless', 'useful', 'very useful' respectively). A user should not be allowed to provide a usefulness-rating for his/her own feedbacks.
- 8) [5pts] **Trust recordings:** A user may declare zero or more other users as 'trusted' or 'not-trusted'.
- 9) [20pts] **TH Browsing:** Users may search for THs, by asking conjunctive queries on the price (a range), and/or address (at CITY or State level), and/or name by keywords, and/or category. Your system should allow the user to specify that the results are to be sorted (a) by price, or (b) by the average numerical score of the feedbacks, or (c) by the average numerical score of the trusted user feedbacks.
- 10) [5pts] **Useful feedbacks:** For a given TH, a user could ask for the top  $n$  most 'useful' feedbacks. The value of  $n$  is user-specified (say, 5, or 10). The 'usefulness' of a feedback is its average 'usefulness' score.
- 11) [10pts] **TH suggestions:** Like most e-commerce websites, when a user records his/her reservations to a TH 'A', your system should give a list of other suggested THs. TH 'B' is suggested, if there exist a user 'X' that visited both 'A' and 'B'. The suggested THs should be sorted on decreasing total visit count (i.e., most popular first); count only visits by users like 'X'.
- 12) [10pts] **'Two degrees of separation':** Given two user names (logins), determine their 'degree of separation', defined as follows: Two users 'A' and 'B' are 1-degree away if they have both favorited at least one common TH; they are 2-degrees away if there exists an user 'C' who is 1-degree away from each of 'A' and 'B', AND 'A' and 'B' are not 1-degree away at the same time.
- 13) [10pts] **Statistics:** At any point, a user may want to show
- the list of the  $m$  (say  $m = 5$ ) most popular THs (in terms of total visits) for each category,
  - the list of  $m$  most expensive THs (defined by the average cost per person of all visits to a TH) for each category
  - the list of  $m$  highly rated PHs (defined by the average scores from all feedbacks a TH has received) for each category
- 14) [5pts] **User awards:** At random points of time, the admin user wants to give awards to the 'best' users; thus, the admin user needs to know:
- the top  $m$  most 'trusted' users (the trust score of a user is the count of users 'trusting' him/her, minus the count of users 'not-trusting' him/her)
  - the top  $m$  most 'useful' users (the usefulness score of a user is the average 'usefulness' of all of his/her feedbacks combined)

### 3 Phase 1: Data Modeling with ER and the Conversion to the Relational Model

Initially, you must design and create a database that organizes and stores data about the UTrack system. This includes designing the ER diagram, and then converting your designs to relational tables/schemas to store this data, and designing rules (when necessary) for how these tables relate to each other so that the data they store can be combined in meaningful ways. We DO NOT require you to enforce constraints via either trigger or assertions. Rather, you can enforce any business rules (if any) in your application-level code (e.g., Java or JSP).

When phase 1 of the project is due, you need to submit your ER diagram and your relational schemas (there is no need to show the CREATE TABLE statements).

**Due Data: Mon, Feb 20 in Canvass.**

## 4 Phase 2: A Workstation/Desktop System with Java

In this phase, you should write the SQL queries and stand-alone Java code that implements a (simple) user interface to your database and executes the queries. Your system should provide all the functionality as specified in Section 2.2. Sample codes will be provided to you at the beginning of this phase.

A command-line based interface will be accepted. A GUI is not necessary, but you are welcome to do so (no bonus points for a GUI though if implemented).

**Due Data: Monday, March 20, 5pm.** You should create a single compressed file for your entire working directory with the naming convention yourname-phase2 (e.g., zip, or use tar and gzip as follows: `tar czf yourname-phase2.tgz ./yourname-phase2`)

For submission: please use the handin system as follows. You can choose one of the following options: 1) Log in using your CADE account at: <https://webhandin.eng.utah.edu> (class name is cs5530); or 2) use the command line on the linux machines (from the CADE lab): `handin cs5530 phase2 /path/to/thecompressedfile`.

Your implementation in this phase will be used as APIs to facilitate your implementation in the next phase for the web interface.

## 5 Phase 3: Web-based System with JSP

Finally, you will create a web portal to your database system. Users will connect to the web site and will be able to execute various functionality from the web interface. This will be achieved via JSP+Tomcat. Note that you are not allowed to use servlet, or any other programming frameworks. Only JSP is allowed (since it allows you to call your Java classes in Phase 2 in an extremely simple fashion). You will be provided sample JSP codes to start with, as well as sample initial set up to your web folder under the Tomcat server. Details will be provided on the class website when this phase begins.

**Due Data: Monday, April 24 5pm.** You will simply stop editing any files on your account on our webserver when this phase is due.

## 6 Grading of the Project

Your design (ER+Relational Schemas) carries 20 points. Each subsequent phase carries 40 points. Phase 2 and Phase 3 will be strictly graded based on the points assigned to each functionality as shown above in Section 2.2.