CMPUT 291 – File and Database Management (Fall 2018)

Mini-Project 1

CMPUT291 - Fall 2018
Mini Project I
(group project)

***Due**: Nov 5 at 5pm*

Clarifications:

You are responsible for monitoring the course news and discussion forums in eclass and this section of the project specification for more details and clarifications. No clarification will be posted after *5pm on Nov 4*.

- None.

Introduction

The goal of this assignment is twofolds: (1) to teach the use of SQL in a host programming language, and (2) to demonstrate some of the functionalities that result from combining SQL with a host programming language.
Your job in this project is to build a system that keeps the enterprise data in a database and to provide services to users. You will be storing data in a SQLite database and will be writing code in Python (or similarly Java/JDBC, C, etc.) to access it. Your code will implement a simple command line interface. You are free to implement a GUI interface instead but there will be no support nor bonus for doing that. You are also free to write your code in Python, Java, C, C++, Perl or any other language that is suited for the task. If you decide to use any language other than Python, discuss it with the instructor first.

Your project will be evaluated on the basis of 84% of the mark for implementing the functionalities listed in this specification; this component will be assessed in a demo session. Another 12% of the mark will be assigned for both the documentation and the quality of your source code. 4% of the mark is assigned for the quality of your group coordination and the project break-down between partners.

Group work policy

You will be doing this project with *one or two other partners* from the 291 class. Register your group at the group registration page. It is assumed that all group members contribute somewhat equally to the project, hence they would receive the same mark. In case of difficulties within a group and when a partner is not lifting his/her weight, make sure to document all your contributions. If there is a break–up, each group member will get credit only for his/her portion of the work completed.

Database Specification

You are given the following relational schema.

- members(email, name, phone, pwd)
- cars(cno, make, model, year, seats, owner)
- locations(lcode, city, prov, address)
- rides(rno, price, rdate, seats, lugDesc, src, dst, driver, cno)
- bookings(bno, email, rno, cost, seats, pickup, dropoff)
- enroute(rno, lcode)
- requests(rid, email, rdate, pickup, dropoff, amount)
- inbox(email, msgTimestamp, sender, content, rno, seen)

Tables are derived from the specification of Assignment 1 and are identical to those in Assignment 2 except (1) the field *pwd* which is added to *members*, and (2) table *inbox* which is new. The table inbox will keep the messages exchanged between members. For example, the tuple <m1,t,m2,c,r,s> in inbox indicates that member m1 has a message from member m2 with a timestamp t (this is of type date and includes date and time) and content c, and the message is regarding ride r. The last column indicates whether the message is seen or not and can take one of the values 'y' and 'n'. Each time a message is sent, the timestamp is set by your system to current date and time (i.e. date('now') in sqlite), rno is set to the current ride number and seen is set to 'n'. The SQL commands to create the tables of the system are given here (right click to save as a file). Use the given schema in your project and do not change any table/column names.

Login Screen

The first screen of your system should provide options for members to login and for new members to register. Existing members should be able to login using a valid email and password, denoted with *email* and *pwd* in table members. After a login, all unseen messages of the member will be displayed, and the status of the messages will be set to seen (i.e, the seen column is set to 'y'). Unregistered members should be able to sign up by providing a unique email, a name, a phone, and a password. Proper messages should be given if the provided email is not unique. After a successful login or signup, members should be able to perform the subsequent operations (possibly chosen from a menu) as discussed next.

Members should be able to logout and there must be also an option to exit the program.

System Functionalities

Members should be able to perform all of the following tasks. All string matches must be case-insensitive (e.g., edmonton will match Edmonton, EDMONTON, edmontoN and edmonton).

1. *Offer a ride.*The member should be able to offer rides by providing a date, the number of seats offered, the price per seat, a luggage description, a source location, and a destination location. The member should have the option of adding a car number and any set of enroute locations. For locations (including source, destination and enroute), the member should be able to provide a keyword, which can be a location code. If the keyword is not a location code, your system should return all locations that have the keyword as a substring in city, province or address fields. If there are more than 5 matching locations, at most 5 matches will be shown at a time, letting the member select a location or see more matches. If a car number is entered, your system must ensure that the car belongs to the member. Your system should automatically assign a unique ride number (rno) to the ride and set the member as the driver of the ride.
2. *Search for rides.*The member should be able to enter 1–3 location keywords and retrieve all rides that match all keywords. A ride matches a keyword if the keyword matches one of the locations source, destination, or enroute. Also a location matches a keyword if the keyword is either the location code or a substring of the city, the province, or the address fields of the location. For each matching ride, all information about the ride (from the rides table) and car details (if any) will be displayed. If there are more than 5 matches, at most 5 will be shown at a time, and the member is provided an option to see more. The member should be able to select a ride and message the member posting the ride that h/she wants to book seats on that ride.
3. *Book members or cancel bookings.*The member should be able to list all bookings on rides s/he offers and cancel any booking. For any booking that is cancelled (i.e. being deleted from the booking table), a proper message should be sent to the member whose booking is cancelled. Also the member should be able to book other members on the rides they offer. Your system should list all rides the member offers with the number of available seats for each ride (i.e., seats that are not booked). If there are more than 5 matching rides, at most 5 will be shown at a time, and the member will have the option to see more. The member should be able to select a ride and book a member for that ride by entering the member email, the number of seats booked, the cost per seat, and pickup and drop off location codes. Your system should assign a unique booking number (bno) to the booking. Your system should give a warning if a ride is being overbooked (i.e. the number of seats booked exceeds the number of seats offered), but will allow overbooking if the member confirms it. After a successful booking, a proper message should be sent to the other member that s/he is booked on the ride.
4. *Post ride requests.*The member should be able to post a ride request by providing a date, a pick up location code, a drop off location code, and the amount willing to pay per seat. The request rid is set by your system to a unique number and the email is set to the email address of the member.
5. *Search and delete ride requests.*The member should be able to see all his/her ride requests and be able to delete any of them. Also the member should be able to provide a location code or a city and see a listing of all requests with a pickup location matching the location code or the city entered. If there are more than 5 matches, at most 5 matches will be shown

at a time. The member should be able to select a request and message the posting member, for example asking the member to check out a ride.

Groups of size 3 must counter SQL injection attacks and make the password non-visible at the time of typing.

Testing

At development time, you will be testing your programs with your own data sets (but conforming to the project specification). At demo time, we will be creating the database using these SQL statements (right click to save as a file) and will be populating it with our own test data set. Your application will be tested under a TA account.
The demo will be run using the source code submitted and nothing else. It is essential to include every file that is needed to compile and run your code including all source code and any makefile or script that you may use to compile or run your code. You will neither be able to change your code, nor use any file other than those submitted. This policy can be altered only in exceptional cases at the instructor's discretion and for a hefty penalty. As a test drill, you should be able to set up your application under someone else's account (in case of testing, this would be under a TA account) within 3 minutes at most.

Every group will book a time slot convenient to all group members to demo their projects. **At demo time, all group members must be present.**The TA will be using a script to both create and populate the tables. The TA will be asking you to perform various tasks and show how your application is handling each task. A mark will be assigned to your demo immediately after the testing.

Instructions for Submissions

Your submission includes (1) the application source code, (2) README.txt, and (3) your design document *Report.pdf.*

- Create a single gzipped tar file with all your source code, additional files you may need for your demo, README.txt and Report.pdf. Name the file *prjcode.tgz*.
- Submit your project tarfile in the project submission site.
- All partners in a group must submit their own copies (even though the copies may be identical).

The file README.txt is a text file that lists the names and ccids of all group members. This file must also include the names of anyone you collaborated with (as much as it is allowed within the course policy) or a line saying that you did not collaborate with anyone else. This is also the place to acknowledge the use of any source of information besides the course textbook and/or class notes.

Your design document must be type-written and is saved as a PDF and be included in your submission. Your design document cannot exceed 4 pages.

The design document should include (a) a general overview of your system with a small user guide, (b) a detailed design of your software with a focus on the components required to deliver the major functions of your application, (c) your testing strategy, and (d) your group work break-down strategy. The general overview of the system gives a high level introduction and may include a diagram showing the flow of data between different components; this can be useful for both users and developers of your application. The detailed design of your software should describe the responsibility and interface of each primary function or class (not secondary utility functions/classes) and the structure and relationships among them. Depending on the programming language being used, you may have methods, functions or classes. The testing strategy discusses your general strategy for testing, with the scenarios being tested, the coverage of your test cases and (if applicable) some statistics on the number of bugs found and the nature of those bugs. The group work strategy must list the break-down of the work items among partners, both the time spent (an estimate) and the progress made by each partner, and your method of coordination to keep the project on track. The design document should also include a documentation of any decision you have made which is not in the project specification or any coding you have done beyond or different from what is required.

Your design document will not include the source code. However your source code will be inspected for source code quality (whether the code is easy to read and if data processing is all done in SQL and not in the application) and self-documentation (whether the code is properly commented).

Last modified: Friday, 12 October 2018, 10:36 PM

◀ Mini–Project 1 groups

Jump to...

Introduction to Python bsddb3 ▶