410 Term Project Description: The "Hotel Reservation Service"

Release: Part I – September 9th, 2004; Part II – October 20th, 2004
Updated: October 29th, 2004
Submission due: November 29th, 2004
Demo times: November 29th- December 2nd, 2004

Brief Introduction: This is a hypothetical situation where hotel reservations are made by city and each hotel reservation service is specialized in the hotels of one and only one city. Students will team up in groups of four students. Each group will deal with one hotel reservation service for one city. There will be ten (10) different cities in total: Edmonton, Calgary, Vancouver, Victoria, Saskatoon, Winnipeg, Ottawa, Toronto, Montreal and Quebec. Hotels are selected (created) by the students, and their descriptions are hypothetical but consistent among hotels and cities. The hotel characteristics and reservation constraints are described latter in this document.

Project Description: You are assigned the task of building a web-based application to serve as a hotel reservation service. The service will offer options to on-line users and will provide activity summaries to management as decision support for a given business. In other words, the web-based application has two parts: one for the reservation seen by the customers inquiring about hotels (reservation module); and one for the manager to track the reservation business activity (management module).

1- Reservation Module

This module provides a user with tools to make hotel reservations in different cities. While each hotel reservation service specializes in one city and keeps information in a local database only about hotels in that city, the tools should provide information about hotels in other cities and allow reservation in these other cities too. How this information about other cities will be accessed is explained later in this document. The tools that this module should provide are:

a- A list of reachable cities
b- A list of available hotels per city
c- A search capability to search for hotels given some constraints.
d- A recommendation capability to recommend hotels.
e- A possibility to make a reservation
f- A possibility to view ones own reservation list

2- Management Module

The management module is solely for administration purposes. It should allow a decision maker see some summaries about the reservation activities. The module should provide simple reporting tools such as:

a- A list of customers per hotel for a given time period with aggregated night counts and amounts.
b- A list of hotels in the concerned city for a given time period with aggregated night counts and amounts.
c- A list of cities for a given time period with aggregated night counts and amounts.
d- Other decision support reports that you see fit: example, top n hotels for a given time period, etc.

Notice that reports a, b, and c can be consolidated into one interactive report that allows drilling down and rolling up from customers to hotels to cities.

Customers, reservations and information about hotels should be stored in a database on ORACLE. The database should be designed and an E-R model is required. The database should be populated with simulated data. Each group is responsible to add data in their database. No data will be provided. A list of the content of the database should be provided to the TAs just before the demo of the project.

Information about hotels:
Each hotel is described using the following features. ***These do not have to be real.***

- Hotel name
- Hotel address
- Hotel picture
- Star level (✩)(✩✩)(✩✩✩)(✩✩✩✩)(✩✩✩✩✩)
- Price per night
- Availability of a swimming pool
- Number of available restaurants
- Availability of Internet access
- Smoking / non-smoking rooms
- Size of bed(s)
- Distance to the city center
- Distance to airport
- Other information such as malls/metro/golf etc.

You can either make the assumption that all rooms are identical in a hotel or that a hotel can present different types of rooms (with regard to smoking/non-smoking and bed type).

Search capability:
A user should be able to search for a hotel room in a given city by specifying constraints. The constraints that could be expressed are:

- city
- star level
- price range
- existence of swimming pool
- smoking/non-smoking
- Internet access availability

The constraint "city" is always mandatory, but all other constraints are optional. If specified, a constraint has to be satisfied by the listed hotels. Otherwise it means that the customer doesn't care. For example, if a customer specifies "with swimming pool", it means that only hotels with swimming pools should be listed. If nothing is specified then all hotels should be displayed regardless of whether there is a pool or not. It is not the negation that applies here.

Hotel recommendation:
The system should be able to recommend hotel / hotel rooms to a user. There are many possibilities for a recommendation agent in this case. You need to implement at least one of them. Here is a list of possible approaches ordered by importance (i.e. mark values).

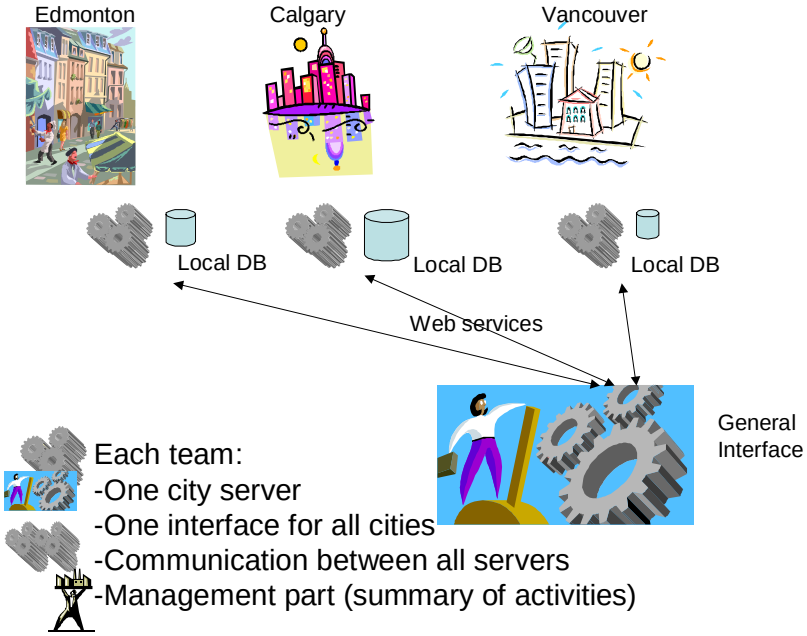
1- a recommender agent that recommends hotels based on a failed constraint-based search. Suppose you send a search query specifying constraints such as price range, star level, swimming pool availability, etc., and that there is no hotel that satisfies the set of

constraints. The recommender agent could "relax" some constraints and provide a list recommending other hotels that "almost" satisfy the specified constraints. The relaxation of constraints could be done in sequence based on some heuristics or based on preferences (i.e. weights) the user provides on each constraint. Think of a customer profile.

2- A recommender agent that recommends hotels based on ratings. If users have the possibility to rate hotels they visit, the agent could recommend hotels to a given user A based on the ratings A previously provided and the matched ratings of other users. This recommendation system relies on the collaborative filtering algorithm.

3- A recomender agent that recommends hotels based on frequency of reservations. Suppose a user A reserved in the past hotel 1 and 2 and other people who reserved 1 and 2 also reserved a room in hotel 3, the agent could recommend hotel 3 to user A. This recommendation system relies on association rule mining.

Communication between hotel reservation services of different cities:
While each group deals with one city, the interface should allow the reservation in different cities. In other words, each group stores information about hotels in their own city but provide the service to access information about hotels in other cities. To access the information from other cities, you should use web services. A registry will be provided by the TAs (check the TA web site for more details and schedule). Basically, we will have 3 available services: *GetHotelList* will query for the hotel list in a given city with or without some specifications; *ReserveRoom* will ask to reserve a hotel room for a given customer; *CheckReservation* will check the reservation already done for a customer. Each group that starts running their server should register their service with the registry. To know the available services (i.e. cities), a simple request to the registry would provide the list of services alive. That same list would provide the address and communication port for each service. To communicate with a given web service, a communication protocol will be specified using a 410 standard WSDL (web service definition language) file. This file will be provided by the TAs and discussed on the newsgroup for final adjustments. More information about the services, the parameters, the data transferred, the format and the WSDL per se will be provided shortly (check course and TA web pages).



Groups: The project must be solved in groups of 4 students. Exception from this rule (i.e. 1-3 student group) is accepted if the number of students is odd, and only applies to one group. Check the web page for the interface to create groups. Cities will be assigned to groups by the Tas.

Note about the management module:
There is no need for web services when dealing with the management module. The management module will only access the local database for its own city and its own customers.
Suppose you are dealing with Edmonton. When a customer comes to you to reserve a hotel in Vancouver, you will transparently use the web service from Vancouver to do the reservation. In your local database you will put the information about the reservation in Vancouver even though you are dealing with Edmonton. This is your customer after all and this customer pays you. The same thing happens in Vancouver. They will put the transaction in their database too. Remember, these services are middlemen. The hotel charges the rooms per night, the reservation service of the city will get a %. If the customer comes from you, you will get a % from the reservation service of the city. The hotels see one supplier and the customer sees one service.

So the management module is about the activity of your hotel reservation service, not all services.

Deliverables: Each group must submit the following:

- **project report:** explain how the solutions were implemented, all components (classes/methods) used and relationships between them. Explain you assumptions and simplifications. Present your algorithms for the recommender systems you implemented. List the used queries, explain how they were assembled to get the report for the management module (rough algorithm) and attach a snapshot of the reports and the user interface of the reservation module. Attach the entity-relationship model of your data in the database to your report. Enumerate specific features addend to your implementation (JavaScript enhancements etc.)
- **DO:** use a cover page for the project report with your name(s) and unixID(s), lecture and lab sections.
- **DON'T:** write your studentID(s) in the report or any attached document.

Demonstration: Each group will have a maximum of 20 minutes to demo the project. The TA needs to have access to your database content before the demo. Make sure the project runs well on the machines in the lab, and displays correctly your project web pages using the Netscape browser available in the lab machines. *You will not be allowed to make any modifications to your project at demo time.*

Marking: The marks for this project are divided as follows (there are no bonus marks):

- Demonstration (interface usability, interactivity etc. functionality, etc.): 75 marks
- Project Report: 25 marks

NOTE: As you develop the project with another student, the marks given for the project will be the same for all four students, even if one of the students fails to deliver his/her part. Hence, choose you partners wisely. All partners must be present at demo time, otherwise zero marks will be given to the demo component of the project to the absent partner.

Submission: Submission will be done (only) using the 'try' submission system. Create a tar file including the project report, all source code (no compiled class files) and the Makefile. *You will not be allowed to add any files at demo time,* therefore test your tar file before submitting it and make sure it contains all required files for your project to run. Additional instructions for the submission process will be added later on.

Frequent Asked Questions:

1- What technology can I use for implementing my project?
   You can use Java servlets, JSP, Perl script CGI, C/C++ CGI or PHP.
2- Can I implement my project at home and run it from my server at home?
   Yes you can. However, at demo time, you need to execute the demonstration from the lab. You also need to provide you source code with the submission.
3- Can I use another DBMS?
   Yes you can use any relational DBMS: ORACLE, DB2, Postgress, msql, SQL-server, etc. Microsoft Access is not considered a RDBMS and cannot be used for this project.
4- Will the look-and-feel of the implementation count or is it only the functionality of the project that counts?
   The user interface will count. While it won't count for very much in comparison to other aspects of the project, the look and feel is what the user sees and it is what will distinguish between implementations. In web-based applications, the interface is paramount.