DATABASE SYSTEMS TERM PROJECT
CS6360.501 – SPRING 2007

I. OVERVIEW

Our company, *DBDudes*, proposes creating a new airline reservation system called *The ARS* (Airline Reservation System). The purpose of our system is to create convenient and easy-to-use software for passengers, trying to buy airline tickets at affordable prices.

We will have a large database supporting dozens of major cities around the world as well as hundreds of flights by various airline companies. Above all, we hope to provide a comfortable user experience along with the best pricing available. In addition, we plan to utilize modern internet technologies such as JDBC in our project.

II. PROJECT DESCRIPTION

The ARS only services registered users. Registration requires providing a unique e-mail address along with complete name, telephone number, physical address and personal identification number (PIN). E-mail addresses are stored either lowercase or uppercase so as to prevent multiple registrations with the same e-mail account.

Users are categorized into 3 mileage clubs: Silver, Gold and Platinum. New users automatically become Silver club members. Status of a user depends on the total number of miles flown with the system. Upgrade to Gold club requires collecting 20K miles. Platinum users are those with more than 40K miles. In an effort to increase satisfaction of frequently flying customers, the system keeps track of the following information: Magazine choices of Platinum customers, seating preferences of Platinum and Gold customers (i.e. window, aisle) and on-board meal preferences of all customers.

The system provides only two services: airline ticket reservation and sales. Tickets are not necessarily personal; a user can buy or book for multiple passengers. The process starts with selecting one of the services. Then, the user chooses the departure city and, if the selection corresponds to multiple airports, the departure airport. (Notice that airports are not entities of by themselves. Some may share the same name and can be identified uniquely together with the city they're located in.) Once the origin is set, the user specifies the date and quantity. In the next step, the system fetches a list of available flights and prices for the user to pick one.

Round-trip flights are very advantageous with *ARS*. Users receive 40% discount if they buy a return ticket. Omitting the city/airport selection step, the same process is repeated for the return flight.

Final step for ticket sales involves asking for the billing address. Payment by personal check is the only method, therefore there is not further dialogue concerning sales. If the selected service was reservation, then the user is informed of the reservation number. Later on, when the reservation is to be proceeded, the user simply enters this number to continue with billing details as described above. The system should respond with an error message if the given reservation is not made by the service-requesting user.

For the time being, we assume that tickets can not be returned. Therefore as soon as a ticket is sold, corresponding miles are added to the customer's mileage account. Reservations, on the other hand, might be cancelled by the user, expired by the system or proceeded to sale. Reservation requests from a user should not overlap: you can not be on two planes at the same time or you can not book a flight between departure and return dates of a round-trip.

Please note that this document does not claim to be complete. Many design issues need careful analysis. Some of these include: how user history is stored, what attributes should entities have (flights, reservations, cities, airlines), how flight capacities are handled, how users are upgraded to better mileage clubs. Yet, we believe that given this description, filling in the blanks should be easy.

III. PROJECT STEPS

DBDudes management board believes that their team is knowledgeable, attentive and experienced. Therefore, without any second thoughts about how hard it would have been to fix a design error noticed towards the end of the development process, the board decided to decompose the project into the following steps:

STEP 1. (30%) Draw the ER/EER diagram, and then convert it to a relational schema. Indicate primary keys, foreign keys and any other constraints you may have. Follow the notation used throughout the textbook. Clearly specify any assumptions you make and your rationale.
Submit all your work via WebCT in **JPEG, BMP, DOC or PDF** format by **03/01/2007**.

STEP 2. (45%) Create database tables according to the relational schema in Step 1 and populate them with a reasonable number of tuples. Implement any query/view/trigger that the end-user may need. Also include a series of 'drop table' queries that will drop all tables created.

Submit the SQL commands via WebCT as a **TXT** file by **03/22/2007**. Any line that is not part of an SQL command should be commented out. In other words, you need to submit an SQL script. Please notice that although SQL is a standard, query syntax differs by DBMS vendors. Your script should run on the campus Oracle server without any error.

STEP 3. (25%) Design and implement a program with graphical user interface for end-users. This program should connect to the database through the internet and provide the functionalities discussed in the project description.
You may use any programming language you want. But we will give support for Java only. Basics of JDBC (Java Database Connectivity) will be discussed on a simple application in class.
Please notice that all inquiries regarding connection problems (i.e. "server is down") should be directed to cs-tech@utdallas.edu. TA is not database administrator.
Submit your source code, executable and program snapshots as a **ZIP** file via WebCT by **04/12/2007**. A demonstration schedule will be created for you to present your work.

IV. SUBMISSION GUIDELINES

Please read the information below carefully. Compliance with these guidelines is a vital part of the grading process.

- ❑ Please pay attention to submit your files in proper format, specified for each step. Only one submission per group is sufficient. Include names and NetIDs of all members in every file.

- ❑ If you need to update/extend your previous work (i.e. due to TAs feedback), submit your updated documents with the next step. A brief note describing the changes would certainly help.

- ❑ Do not submit at the last minute. Your clock most possibly is not synchronous with WebCT system clock. Unless there is a good reason (i.e. WebCT system crash), only online submissions will be accepted. In order to encourage submitting partial work, multiple submissions are allowed. You can always retrieve your files, make necessary changes and re-submit until the due date.

- ❑ Source codes will be tested using software plagiarism tools. Plagiarized work is very easy to detect, and all necessary measures will be taken to identify and penalize such behavior.

GOOD LUCK!