Shanghai New York University
Project for Databases CSCI-SHU 213
Fall 2019
Professor Ratan Dey

Objective:

The objective of this course project is to provide a realistic experience in the design process of a relational database and corresponding applications. We will focus on conceptual design, logical design, implementation, operation, maintenance of a relational database. We will also implement an associated web based application to communicate with the database (retrieve information, store information etc).

Project Overview:

The course project for this semester is online Air Ticket Reservation System. Using this system, customers can search for flights (one way or round trip), purchase flights ticket, view their future flight status or see their past flights etc. There will be three types of users of this system – Customers, Booking Agents and Airline Staff (Administrator). Booking Agents will book flights for other Customers, can get a fixed commission. They can view their monthly reports and get total commission. Airline Staff will add new airplanes, create new flights, and update flight status. In general, this will be simple air ticket reservation system.

Instructions for working with a Team: You may work alone or with one teammate.
Deadline for adding teammate: Thursday September 26, 2019. THIS IS A HARD DEADLINE. NO EXCEPTIONS.

3 Parts of the Project:

Part 1. **WORK IN A TEAM OF 2 PERSONS**: Create an ER diagram based on the Project description below. **You may work with a partner or individually for this part**.

Deadline: 10/10/2019 11:55 pm.

Part 2. **WORK IN A TEAM OF 2 PERSONS**: Create a relational database design (relational Schema, write table definitions in SQL, write some queries etc) based on ER diagram. **You may work with a partner or individually for this part**.

Deadline: 11/07/2019 11:55 pm.

Part 3. **WORK IN A TEAM OF 2 PERSONS**: Develop a web application for the system. **You may work with a partner or individually for this part**.

Deadline: 12/05/2019 11:55 pm.

The total project grade will be 25% of your course grade. Part 1 counts for about 10% of the project grade. Part 2 counts for about 10% of the project grade. Part 3 counts for about 80% of the project grade. There may also be a quiz or exam question(s) based on the project.

Teams will be required to submit a work plan, indicating who will do what, and will be required to submit progress report (due on 11/19/2019), all source codes (front end, back end), databases backup file (including table structures and inserted data), detailed report on use cases implementation, an evaluation at the end. Note that each teammate is expected to contribute roughly equally to each aspect of the project and each teammate is responsible for understanding the entire system. Normally all team members will receive the same grade, but I may deduct points from individuals who are not pulling their weight on a team.

Project Description

There are several airports (**Airport**), each consisting of a unique name and a city.

There are several airlines (**Airline**), each with a unique name. Each airline owns several airplanes. An airplane (**Airplane**) consists of the airline that owns it, a unique identification number within that airline, and the amount of seats on the airplane.

Each airline operates flights (**Flight**), which consist of the airline operating the flight, a flight number, departure airport, departure date and time, arrival airport, arrival date and time, a base price, and the identification number of the airplane for the flight. Each flight is identifiable using flight number and departure date and time together within that airline.

A ticket (**Ticket**) can be purchased for a flight by a Customer or Booking Agent (on behalf of customer), and will consist of the customer's email address, the airline name, the flight number, sold_price (may be different from base price of the flight), payment information (card type - credit/debit, card number, name on card, expiration date), purchase date and time, and a booking_agent_ID. If a Booking Agent purchases the ticket then their booking_agent_ID will be used, and if a Customer purchases the ticket then the booking_agent_ID should be null. Each ticket will have a ticket ID number which is unique in this System.

Anyone (including users not signed in) can see flights (future flights) based on the source airport, destination airport, source city, or destination city, departure date for one way (departure and return dates for round trip). Additionally, anyone can see the status (delayed/on time etc.) of the flight based on an airline and flight number combination and arrival or departure date.

There are three types of users for this system: Customer, booking agent and Airline Staff.

Customer:

Each Customer has a name, email, password, address (composite attribute consisting of building_number, street, city, state), phone_number, passport_number, passport_expiration, passport_country, and date_of_birth. Each Customer's email is unique, and they will sign into the system using their email address and password.

Customers must be logged in to purchase a flight ticket.

Customers can purchase a ticket for a flight as long as there is still room on the plane. This is based on the amount of tickets already booked for the flight and the seating capacity of the airplane assigned to the flight and customer needs to pay the associated price for that flight. Ticket price of a flight will be determined based on two factors – minimum/base price as set by the airline and additional price which will depend on demand of that flight. If 70% of the capacities is already booked/reserved for that flight, extra 20% will be added with the minimum/base price. Customer can buy tickets using either credit card or debit card. We want to store card information (card number and expiration date and name on the card but not the security code) along with purchased date, time.

Customer will be able to see their future flights or previous flights taken for the airline they logged in.

Customer will be able to rate and comment on their previous flights taken for the airline they logged in.

Booking Agent:

The role of a Booking Agent is similar to that of a Customer. A Booking Agent's purpose is to purchase a ticket on behalf of a Customer (with the same restrictions of seat availability as above), but that Booking Agent will receive a 10% commission from the ticket price.

A Booking Agent consists of a unique email, a password, and a booking_agent_ID. In order for a Booking Agent to sign into the system, they must enter all three of these items.

Once logged in, a Booking Agent will be able to see the amount of commission they received in the past 30 days, the average commission they received per ticket booked, and the total number of tickets they booked.

Airline Staff:

Each Airline Staff has a unique username, a password, a first name, a last name, a date of birth, may have more than one phone number, and the airline name that they work for. One Airline Staff works for one particular airline.

Airline Staff will be able to add new airplanes into the system for the airline they work for.

Airline Staff will set flight statuses in the system.

Each Airline Staff can create new flights only for the particular airline that they work for by inserting all necessary information and will set the ticket base price for flight. They will also be able to see all on-time, future, and previous flights for the airline that they work for, as well as a list of passengers for the flights.

In addition, Airline Staff will be able to see a list of all flights a particular Customer has taken only on that particular airline.

Airline Staff will be able to see each flight's average ratings and all the comments and ratings of that flight given by the customers.

Airline Staff will also be able to see the most frequent customer within the last year, see the amount of tickets sold each month, see the total amount of revenue earned etc.

Airline Staff can query for how many flights get delayed/on-time etc.

What You Should Do for Part 1:

Design an ER diagram for online Air Ticket Reservation System described above. Draw the ER diagram neatly. You may draw it by hand or using a design tool. Design tool preferred. Please create a PDF file and submit using NYU Classes as the solution of **Part 1 of Course Project Assignment**.

Deadline for Part 1: 10/10/2019 11:55 pm.

When you do this, think about: which information should be represented as attributes, which as entity sets or relationship sets? Are any of the entity sets weak entity sets? If so, what is the identifying strong entity set? What is the primary keys (or discriminant) of each entity set? What are the cardinality constraints on the relationship sets? Do you need to use ternary relationship sets or aggregation?

You may find it useful to read the Project Part3 descriptions but not required.

What You Should Do for Part 2

1. Following the techniques we studied, derive a relational schema diagram from the Part 1's ER diagram. Remember to underline primary keys and use arrows from the referencing schema to the referenced schema to indicate foreign key constraints.

2. Write and execute SQL CREATE TABLE statements to create the tables. Choose reasonable types for the attributes.

3. Write and execute INSERT statements to insert data representing one airline's air ticket reservation system. As for example, you can insert data in the appropriate tables as follows or you can insert data for other airline or your own make up airline:

a. One Airline name "China Eastern".

b. At least Two airports named "JFK" in NYC and "PVG" in Shanghai.

c. Insert at least two customers with appropriate names and other attributes. Insert one booking agent with appropriate name and other attributes.

d. Insert at least two airplanes.

e. Insert At least One airline Staff working for China Eastern.

f. Insert several flights with on-time, and delayed statuses.

g. Insert some tickets for corresponding flights and insert some purchase records (customers bought some tickets directly and customer bought some tickets through a booking agent).

4. Write SQL queries for executing following queries and show the results in your file (SQL query and corresponding answers):

a. Show all the future flights in the system.

b. Show all of the delayed flights in the system.

c. Show the customer names who bought the tickets.

d. Show the customer names who used booking agent to buy the tickets.

e. Show all of the airplanes owned by the airline (such as "Emirates")

You may find it useful to read the Project Part3 descriptions but not required.

Submit a PDF file for Relational Schema diagram and one .SQL file for 2 (create table statements), one .SQL file for 3 (inserting data in the database), one .SQL file for 4 (SQL queries and corresponding results) via NYU Classes as the solution of Part 2 of Course Project Assignment.

Deadline for Part 2: 11/07/2019 11:55 pm.

Project PART 3: Web based Application Development

In Part 3, you'll implement Air Ticket Reservation System as a web based application. **You must use the table definitions that we created for part 2 (derived from the E-R diagram)** unless you need to make some small additions/modifications to support your additional features. If you do modify the table definitions, you will be responsible for translating the test data/test scenarios (in case we provide) so that it matches your table definitions.

REQUIRED Application Use Cases (aka features):

1. **View Public Info:** All users, whether logged in or not, can
    a. Search for future flights based on source city/airport name, destination city/airport name, departure date for one way (departure and return dates for round trip).
    b. Will be able to see the flights status based on airline name, flight number, arrival/departure date.

2. **Register**: 3 types of user registrations (Customer, Booking Agent and Airline Staff) option via forms.

3. **Login: 3 types of user login (Customer, Booking Agent and Airline Staff).** Users enters their username (**email address will be used as username**), x, and password, y, via forms on login page. This data is sent as POST parameters to the login-authentication component, which checks whether there is a tuple in the corresponding user's table with username=x and the password = md5(y).
    a. If so, login is successful. A session is initiated with the member's username stored as a session variable. Optionally, you can store other session variables. Control is redirected to a component that displays the user's home page.
    b. If not, login is unsuccessful. A message is displayed indicating this to the user.

Note: In real applications, members' passwords are stored as md5/other hashes, not as plain text. This keeps the passwords more secure, in case someone is able to break into the system and see the passwords. You can perform the hash using MySQL's md5 function or a library provided with your host language.) Once a user has logged in, reservation system should **display** his**/**her **home page**. Also, after other actions or sequences of related actions, are executed, control will return to component that displays the home page. The home page should display
    c. Error message if the previous action was not successful,
    d. Some mechanism for the user to choose the use case he/she wants to execute. You may choose to provide links to other URLS that will present the interfaces for other use cases, or you may include those interfaces directly on the home page.
    e. Any other information you'd like to include. For example, you might want to show customer's future flights information on the customer's home page, or you may prefer to just show them when he/she does some of the following use cases.

Customer use cases:

After logging in successfully a user(customer) may do any of the following use cases:

4. **View My flights:** Provide various ways for the user to see flights information which he/she purchased. The default should be showing for the future flights. **Optionally** you may include a way for the user to specify a range of dates, specify destination and/or source airport name or city name etc.

5. Search for flights: Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or return).

6. **Purchase tickets:** Customer chooses a flight and purchase ticket for this flight, providing all the needed data, via forms. You may find it easier to implement this along with a use case to search for flights.

6. **Give Ratings and Comment on previous flights:** Customer will be able to rate and comment on their previous flights (for which he/she purchased tickets and already took that flight) for the airline they logged in.

7**.Track My Spending**: Default view will be total amount of money spent in the past year and a bar chart showing month wise money spent for last 6 months. He/she will also have option to specify a range of dates to view total amount of money spent within that range and a bar chart showing month wise money spent within that range.

8.**Logout:** The session is destroyed and a "goodbye" page or the login page is displayed.

Booking agent use cases:

After logging in successfully a booking agent may do any of the following use cases:

4. **View My flights:** Provide various ways for the booking agents to see flights information for which he/she purchased on behalf of customers. The default should be showing for the future flights. **Optionally** you may include a way for the user to specify a range of dates, specify destination and/or source airport name and/or city name etc to show all the flights for which he/she purchased tickets.

5. Search for flights: Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or arrival).

6. **Purchase tickets**: Booking agent chooses a flight and purchases tickets for other customers giving customer information and payment information, providing all the needed data, via forms. You may find it easier to implement this along with a use case to search for flights.

7. **View my commission**: Default view will be total amount of commission received in the past 30 days and the average commission he/she received per ticket booked in the past 30 days and total number of tickets sold by him in the past 30 days. He/she will also have option to specify a range of dates to view total amount of commission received and total numbers of tickets sold.

8. **View Top Customers**: Top 5 customers based on number of tickets bought from the booking agent in the past 6 months and top 5 customers based on amount of commission received in the last year. Show a bar chart showing each of these 5 customers in x-axis and number of tickets bought in y-axis. Show

another bar chart showing each of these 5 customers in x-axis and amount commission received in y-axis.

9. **Logout:** The session is destroyed and a "goodbye" page or the login page is displayed.

Airline Staff use cases:

After logging in successfully an airline staff may do any of the following use cases:

4. **View flights:** Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days. He/she will be able to see all the current/future/past flights operated by the airline he/she works for based range of dates, source/destination airports/city etc. He/she will be able to see all the customers of a particular flight.

5**. Create new flights:** He or she creates a new flight, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days.

6**. Change Status of flights:** He or she changes a flight status (from on-time to delayed or vice versa) via forms.

7**. Add airplane in the system:** He or she adds a new airplane, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. In the confirmation page, she/he will be able to see all the airplanes owned by the airline he/she works for.

8**. Add new airport in the system:** He or she adds a new airport, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action.

9**. View flight ratings:** Airline Staff will be able to see each flight's average ratings and all the comments and ratings of that flight given by the customers.

10. View all the booking agents: Top 5 booking agents based on number of tickets sales for the past month and past year. Top 5 booking agents based on the amount of commission received for the last year.

11**. View frequent customers:** Airline Staff will also be able to see the most frequent customer within the last year. In addition, Airline Staff will be able to see a list of all flights a particular Customer has taken only on that particular airline.

12**. View reports:** Total amounts of ticket sold based on range of dates/last year/last month etc. Month wise tickets sold in a bar chart.

13. **Comparison of Revenue earned:** Draw a pie chart for showing total amount of revenue earned from direct sales (when customer bought tickets without using a booking agent) and total amount of revenue earned from indirect sales (when customer bought tickets using booking agents) in the last month and last year.

14. **View Top destinations:** Find the top 3 most popular destinations for last 3 months and last year (based on tickets already sold).

15. **Logout:** The session is destroyed and a “goodbye” page or the login page is displayed.

Additional Requirements:
You should implement Air ticket reservation system as a web-based application. If you want to use a DBMS other than MySQL, SQLserver, Oracle, Mongodb or to use a programming language other than Python/Flask, Java/JDBC/Servlets, PHP, C#, node.js, or javascripts please check with me first. You will need to bring the host computer to the demo/test session at the end of the semester or make the application available remotely over the web.

Enforcing complex constraints: Your air ticket reservation system implementation should prevent users from doing actions they are not allowed to do. For example, system should prevent users who are not authorized to do so from adding flight information. This should be done by querying the database to check whether the user is an airline staff or not before allowing him to create the flight. You may also use the interface to help the user to avoid violating the constraints. However, you should not rely solely on client-side interactions to enforce the constraint, since a user could bypass the client-side interface and send malicious http requests.

When a user logs in, a session should be initiated; relevant session variables should be stored. When the member logs out, the session should be terminated. Each component executed after the login component should authenticate the session and retrieve the user’s pid from a stored session variable. (If you’re using Python/Flask, you can follow the model in the Flask examples presented to do this.)

You must use *prepared statements* if your programming language supports them. (This is the style used in Flask; if you’re using PHP, use the MySQLi interface; if you’re using Java/JDBC, use the PreparedStmt class.) If your programming language does not support prepared statements, Free form inputs (i.e., text entered through text boxes) that is incorporated into SQL statements should be validated or cleaned to prevent SQL injection.

You should take measures to prevent cross-site scripting vulnerabilities, such as passing any text that comes from users through htmlspecialchars or some such function, before incorporating it into the html that air ticket reservation system produces.

The user interface should be usable, but it does not need to be fancy. For each type of users, you need to implement different home pages where you only show relevant use cases for that type of users and you should not show/combine all the use cases in one page.

For testing/debugging you will probably find it useful to execute your SQL queries directly through PHPmyAdmin (if you use MySQL) or using your database provided client program, before incorporating them in your application code.

What You Should Do for Part 3

1. Use the tables you defined/created in Part 2. (If you defined extra tables or attributes, for additional features, merge them into Part 2.)

2. Before you start coding, think about what each component will do. If there are commonalities among many of the use cases, think about how you will modularize your code.

3. Implement the web based application for Air Ticket Reservation System. For each component
    a. Using some sample data, write the queries executed by the component, and test them.
    b. Write the application code for the component.
4. Test the component with additional values, including values that are not valid input.

5. Suggestion: Implement and test the components one at a time. When a component is ready, add links to it to your home page (or enhance the home page with the interface of the new component.) You will get partial credit if some of your features work, even if others have not been implemented.

PROGRESS REPORT for Part3: A progress report for part 3 will be due on November 19, 2019. For team projects, you must demonstrate that each team member has written some of the code. You will hand this in electronically via NYU Classes as the solution of Progress Report of Part 3 assignment. And I may also schedule brief meetings to discuss. Details TBD. **THIS IS A MANDATORY PART OF THE PROJECT AND WILL AFFECT YOUR GRADE.**

Complete Final Project Hand in instructions: You will hand in:

- Your source code. (Details about whether to zip it, etc, will be provided.)
- A list of the files in your application and what's in each file. (E.g. "homepage.phpscript to generate home page".)
- A separate file that lists all of the use cases and the queries executed by them (with brief explanation). This should be well organized and readable. It should be detailed enough to give readers a good idea of how your application works, without making them dig through all the code.
- For team projects: A summary of who did what.
- Shortly before the project is due, I'll ask you to sign up for a time slot to demonstrate your project to me. This demo will mainly consist of running application on a bunch of test cases. We may also ask you to explain some of your code. No formal presentation will be expected.
- More detailed instructions on what to hand in, how to hand it in, and test data to load into your tables will be provided later.

Deadline for Part 3: 12/05/2019 11:55 pm.