Slide 6.10: PHP MySQL delete
Slide 7.2: Mobile-commerce transaction processing
Home

PRINT VERSION

Programming Exercise II: A Native Mobile App for a Bookstore

(An Industry-Level, Second-to-None Comprehensive Specification)

Absolutely no copying others' works

Development Requirements

When start developing the exercise, follow the requirements below:

- Superuser (managing the bookstore through desktop web browsers):
  - Have to use the LAMP technologies consisting of Linux, Apache, MySQL, and PHP/Perl/Python.
  - The system entry page must be located at `http://people.aero.und.edu/~userid/457/2/` and all pages must be hosted by `http://people.aero.und.edu/~userid/` .
- Customers (interacting with the bookstore through emulators/smartphones):
  - Have to use Android Studio to create a native mobile app, instead of a mobile website.
  - The exercise should be located at `C:\457\Bookstore\` .
- The book and customer data should be saved to and retrieved from the server-side MySQL database at the Aerospace school.

Due Date and Submission Methods

Monday, April 10, 2017 in class and have completed the following three tasks:

- Turn in printouts of all source code (no documentation needed) including Java, XML, PHP, SQL `create` statements, and whatever languages. The code will be examined during the demonstration too.
- Send an email to the instructor at wenchen@cs.und.edu to set up an appointment to demonstrate your exercise to the instructor individually, so misunderstanding would be minimized. The mail lists the times you will be available. The instructor will prepare a set of test data to be used by all students.

- Students may save the whole app in a flash drive and demo it at a classroom or conference-room machine.

Objectives

This exercise is to design and implement a native mobile app on Android devices. Students learn how to build a mobile app connecting to the servers.

Requirements

This exercise is to build an app for a local bookstore by using Android Studio. It includes the following requirements:

- The data of a book includes
  - a unique ISBN (10 characters),
  - a unique title, and
  - a price.

- The data of a customer includes
  - a unique ID assigned by the system automatically after signing up,
  - a name,
  - the information of the all purchased books (one quantity for each book and no duplicate books), and
  - the total amount spent on the purchased books (one amount per customer).

- **[Superuser (managing the bookstore through desktop web browsers): 20% total]**
  For the one and only one superuser:

  - **(Entering books: 05%)** Enter books one by one.

  - **(Listing all books: 05%)** List all books each including (i) an ISBN, (ii) a title, and (iii) a price.

  - **(Listing all customers: 05%)** List all customers each including (i) an ID, (ii) a hyperlinked name, and (iii) a total amount.

  - **(Listing all purchased books: 05%)** After clicking the hyperlinked name, list all books, each of which includes a title and a quantity, purchased by the customer.

- **[Customers (interacting with the bookstore via emulators/smartphones): 60% total]**
  For each customer:

  - **(Signing up, in, and out: 10%)** Customer can sign up, in, and out.

  - **[Searching for books: 20% (undergraduate) and 10% (graduate)]** List a book if its title includes any of the case-insensitive keywords in a query, where the keywords are

separated by spaces. If the query is empty, list all books. The search results are a list of books, each including (i) a checkbox (for purchasing), (ii) a hyperlinked title, and (iii) a quantity to purchase (in an input form such as `EditText`).

- **[Ranking: 10% (graduate students only)]** The search results are also sorted based on the length of LCS (Longest Common Subsequence).

- **(Purchasing books: 15%)** After searching for books, purchase several books (but not all the listed books) at the same time. Add the quantities to purchase to the existing quantities in the database.

- **(Checking his/her own account: 10%)** List the customer's (i) ID, (ii) name, (iii) titles of all purchased books including a quantity for each book, and (iv) total amount.

- **(Showing the details of a book: 05%)** List the book's (i) ISBN, (ii) title, and (iii) price by clicking its hyperlinked title.

- **(Instructor's requirements: 20% total)**
  Other than the above system requirements, the instructor has the following requirements:

  - **(User-friendliness: 15%)** User-friendliness will be heavily considered when grading. In the past, some exercises were awkward, which made the grading or browsing difficult.

  - **(System reset: 05%)** The system can be reset, which is to clear all data stored in the database and files, so the instructor can test the system by using only his own test data. That is the system has to include a button such as "Clear system" at the system entry page.

Evaluations

The following features will be considered when grading:

- **Specification:**
  - The instructor (or your assumed client) has given the exercise specification as detailedly as possible. If you are confused about the specification, you should ask in advance. Study the specification very carefully. No excuses for misunderstanding or missing parts of the specification after grading.
  - The specification is not possible to cover every detail. You are free to implement the issues not mentioned in the specification, but the implementations should make sense. Implemented functions lacking of common sense may cause the instructor to grade your exercise mistakenly, and thus lower your grade.
  - The exercise must meet the specification. However, exercises with functions exceeding the specification will not receive extra credits.

- **Grading:**

- This exercise will not be graded if the submission methods are not met. Students take full responsibility if the app is not working.
- A set of test data will be used for all students. The grades are primarily based on the results of testing. Other factors such as performance, programming styles, algorithms, and data structures will be only considered minimally.
- Before submitting the exercise, test it comprehensively. Absolutely no extra points will be given after grading.
- The total weight of exercises is 36% of final grade and all three exercises have the same weight, 12% each.
- If not specified, no error checking is required; i.e., you may assume the input is always correct for that case.
- Feel free to design your own interfaces; user-friendliness will be heavily considered; each function/button will be tested extensively; and from the source code submitted, the database design and programs will be examined.
- The systems have to be active and the exercise printouts will be kept until the end of this semester. They will be used to check against others' exercises for any plagiarism. The instructor will inform you the exercise evaluations by emails after grading.

- **Desktop web browsers:**
  - Desktop web browser will be used by the superuser to manage the bookstore.
  - Firefox 51.0 or above browser will be used to grade exercises. Note that Internet Explorer, Chrome, and Firefox are not compatible. That is your exercises may work on the IE or Chrome but not Firefox.

- **Android platform:**
  - Android platform has to be used.
  - Since no advanced Android features need to be used, the following SDK (Software Development Kit) and IDE (Integrated Development Environment) will be used to grade exercises:

    - Android SDK 6.0, API 23,
    - Android Studio 1.3, and
    - Android Virtual Device (AVD) (3.7" FWVGA slider and Lollipop API level 22).

    Make sure that your app works for the SDK and AVD.
  - Android emulator (AVD) will be used by the customers to interact with the bookstore.
  - No mobile browser will be used to check your app.

- **MySQL database:**
  - MySQL database at the Aerospace School has to be used to save and retrieve book and customer data.
  - The SQL DDL commands such as "`create table`" have to be submitted, where SQL is Structured Query Language and DDL is Data Definition Language.
  - From the source code submitted, the database design and programs will be examined. Poor database design or uses will result in a lower grade.
  - **(-05%)** if the database design is NOT optimal.

- **(-05%)** if the SQL `create` commands of database implementation are NOT submitted.
- Android includes an embedded database, SQLite, which has limited storage and functionality. You can use it to save other data, except book and customer data.
- Try to perform the tasks by using SQL (Structured Query Language) as much as possible because SQL, a non-procedural language, can save you a great deal of programming effort. The same is applied to the server-side PHP programming because client-side Java programming is more difficult.
- There are many advantages of using databases. If database is not used, the problems caused by not-using-transaction must be considered. For example, if two customers are enrolled at the same time, an ID may be assigned to different customers if databases are not used.

- **Comments:**
  - Make the exercise work first. Do not include extra features, such as passwords, in the beginning. By the way, you will not receive credits for the extra features.
  - One way to build a mobile-commerce system from scratch is to design the user interfaces first and then implement the system button by button. By doing this way, it could simplify the construction. The recommended construction steps are

    i. Database design (E-R modeling or normalization),
    ii. Database implementation (SQL),
    iii. Interface building (XML), and
    iv. Button-by-button implementation (Java and PHP).

  - Each function can be accomplished by either client or server -side scripts, or both.
  - Many times, simplicity is the same as user-friendliness.
  - Security concerns are mainly ignored here. For example, a customer may be able to check others' account information if protection is not enforced. Small or medium -size businesses do not usually implement their own secure payment schemes. Instead they use a third-party payment system such as PayPal or purchase software from company like Cayan and integrate it with their websites.
  - The function of automatically sending emails/texts is important for m-commerce systems, but will not be used here since companies complained our students sending out numerous mails because of faulty programs.

Slide 6.10: PHP MySQL delete
Slide 7.2: Mobile-commerce transaction processing
Home