James Quach

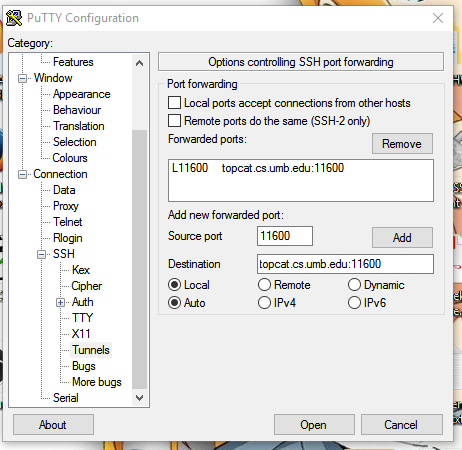
Elizabeth O’Neil

CS636

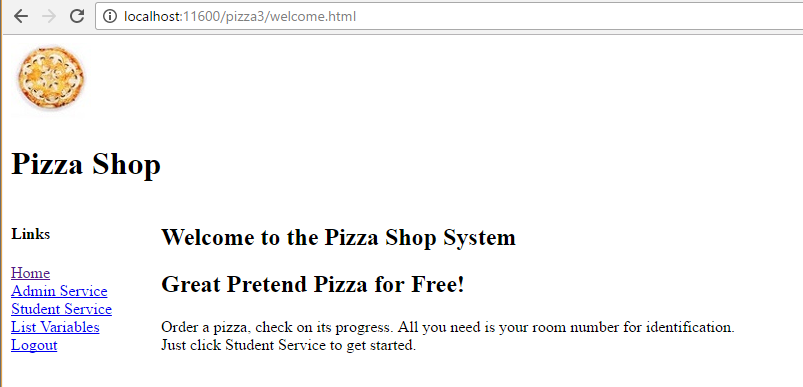
10/17/2016

Homework 3

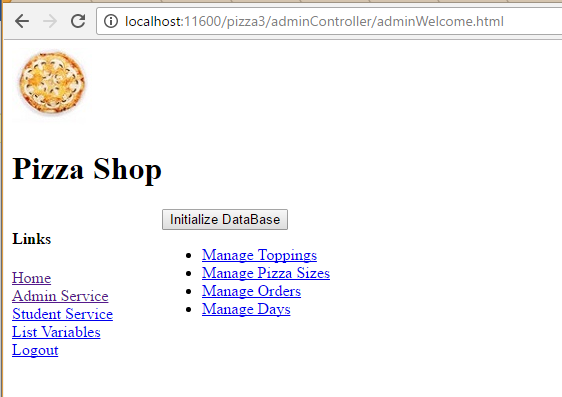
1. **Page flows**--UI analysis. Read about the [Pizza Project](http://www.cs.umb.edu/cs636/PizzaProject.html), and the two handouts on Pizza1. This is the running example for the course, and is available as a client-server app "pizza1". The last version, "pizza3", is the final web app. The deployed version is now available as at http://topcat.cs.umb.edu:11600/pizza3, but we can access this directly only from inside the cs.umb.edu network. To access it from home, add a tunnel from local port 11600 to this port 11600 on topcat.cs.umb.edu, following the pattern of the mysql tunnel. For putty, your tunnel setup should look like this after the addition of the new case:

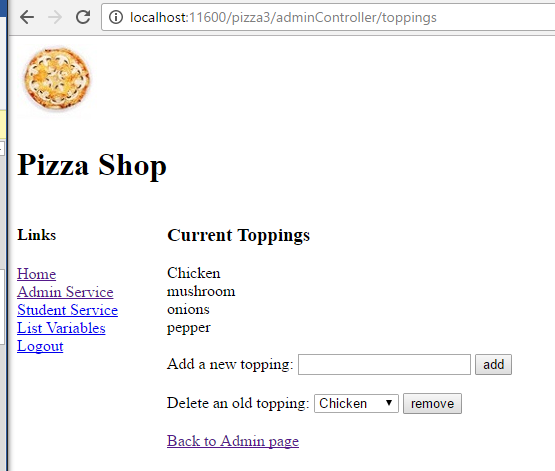


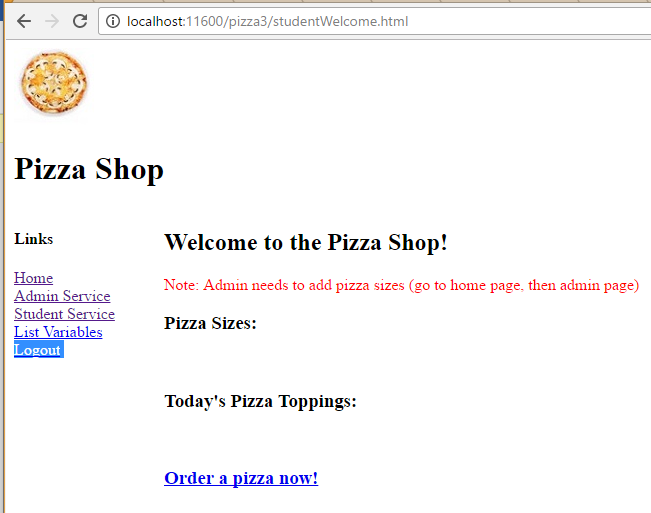
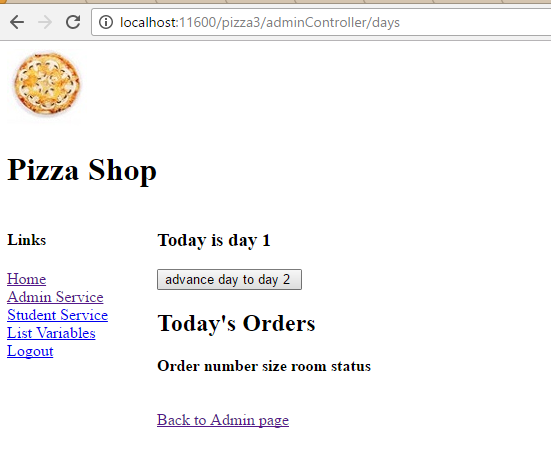
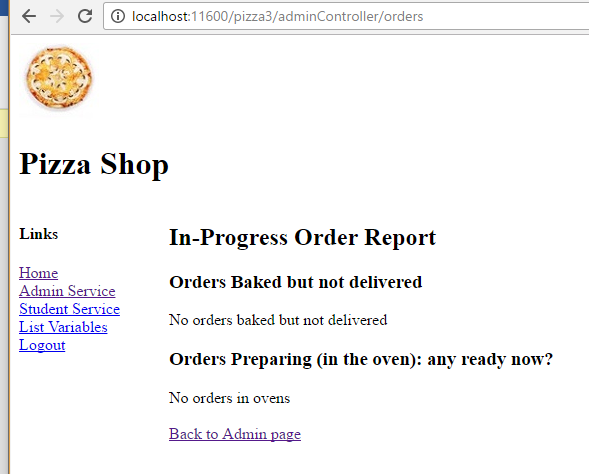
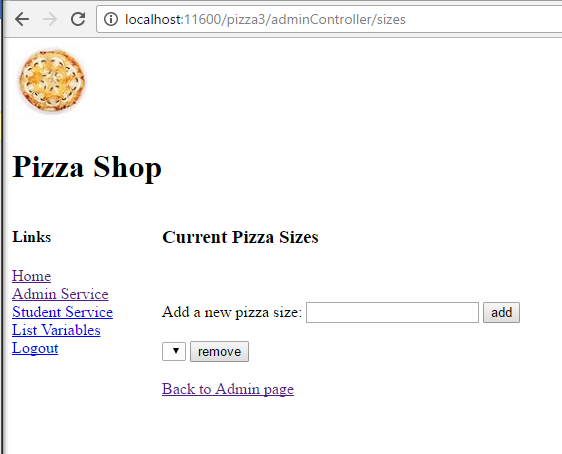
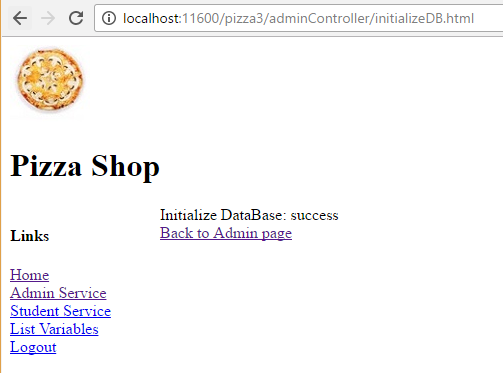
1. For each page you reach starting from the "Home”, do the following. Start from the top link, so that the result is easy to grade.
2. Record the URL displayed by your browser. Note that they all start with http://topcat.cs.umb.edu:11600/pizza3, so you can just record the part after this start.

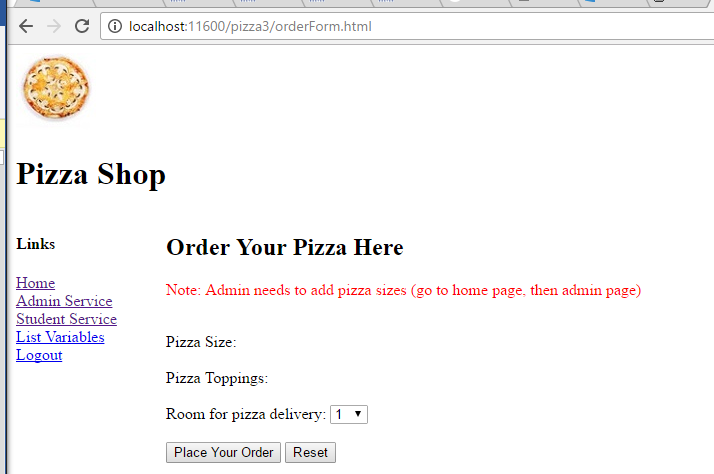


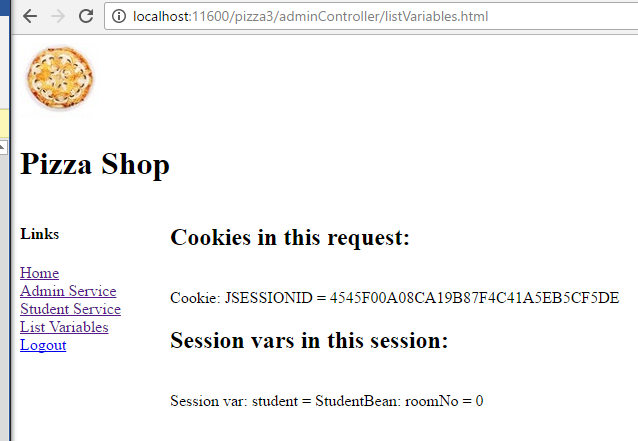
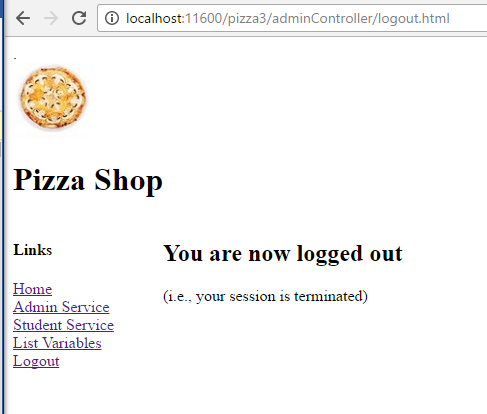
1. Record how you got the next page to show from this page: click on a link?  click on a button? Anything else? Also, don't use the links you see on the page shown above (Home, Admin Service, etc.) until you've exhausted more local links, so your exploration goes: everything accessible from "Admin Service", everything accessible from "Student Service", etc. Note: For this exercise, don't use the "back" button of  your browser. A well-designed dynamic web site provides enough links so that the user can get around comfortably without using the back button.  Handling the back button (and forward button) properly in a dynamic web site is an advanced topic. In fact the use of the back button can cause trouble in many real dynamic sites, so it's best to avoid using it on, say, your online banking site.



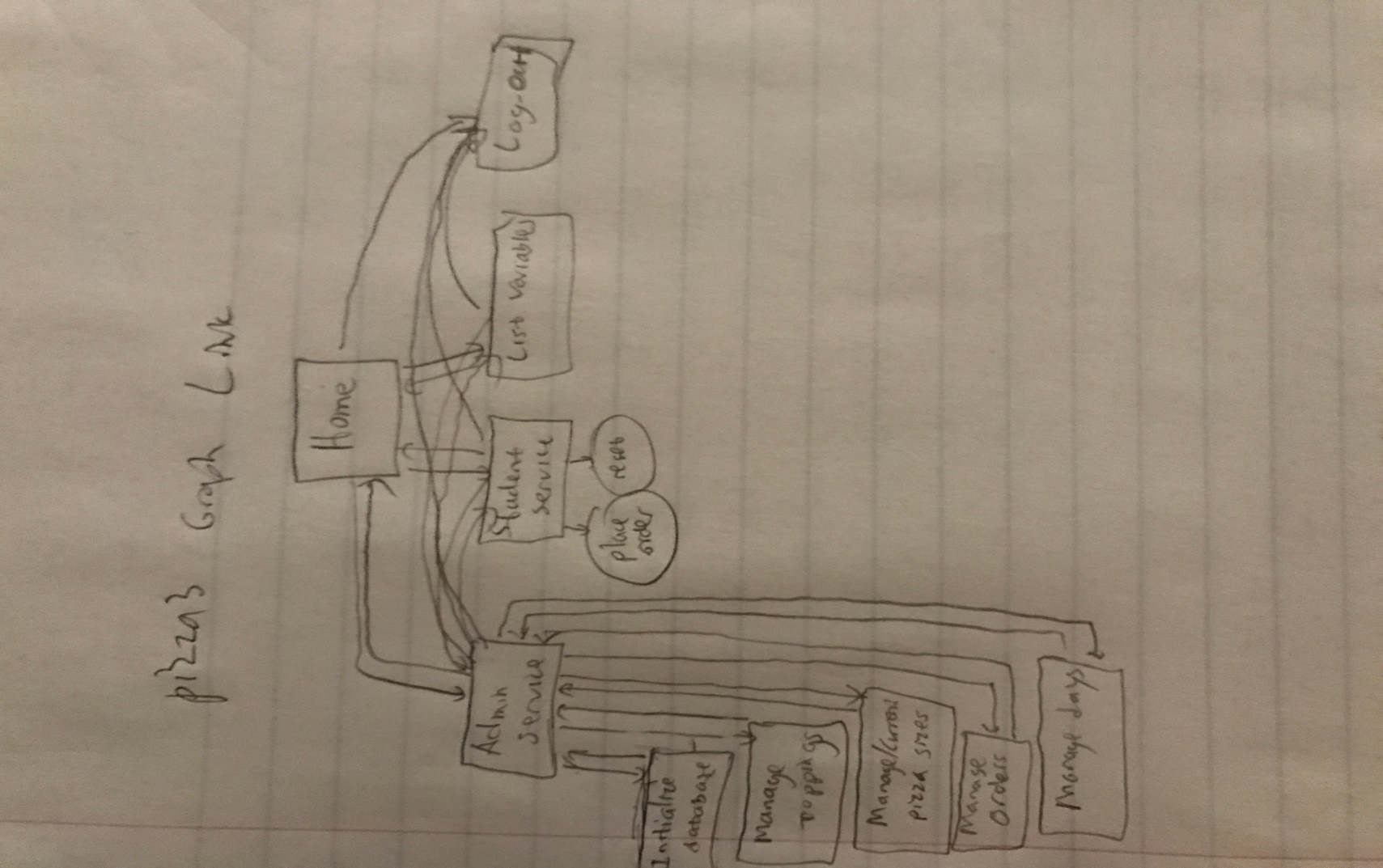






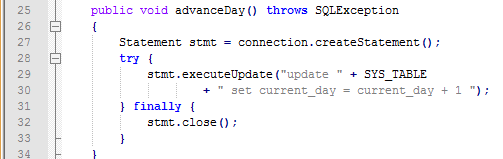


1. Draw a graph of the pages you navigated, with arrows showing use of links or buttons to get from one page to another. There are 9 pages in this pizza app: Home, Manage Toppings, Manage Sizes, Manage Orders, Manage Days, Student Welcome, Pizza Order Form, List Variables, and Logout, so there should be 9 nodes in your graph, with arrows between them as appropriate. You don't need to show links that express that "Home" is accessible from all pages, and similarly other "extra" links, just the links that show the user navigation paths.

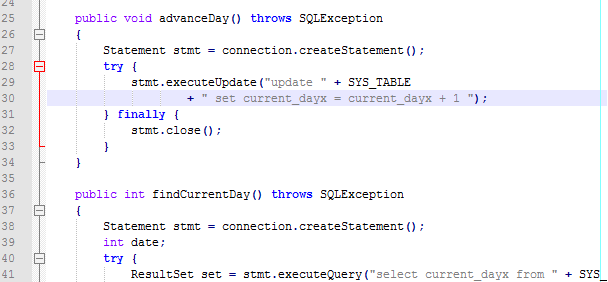


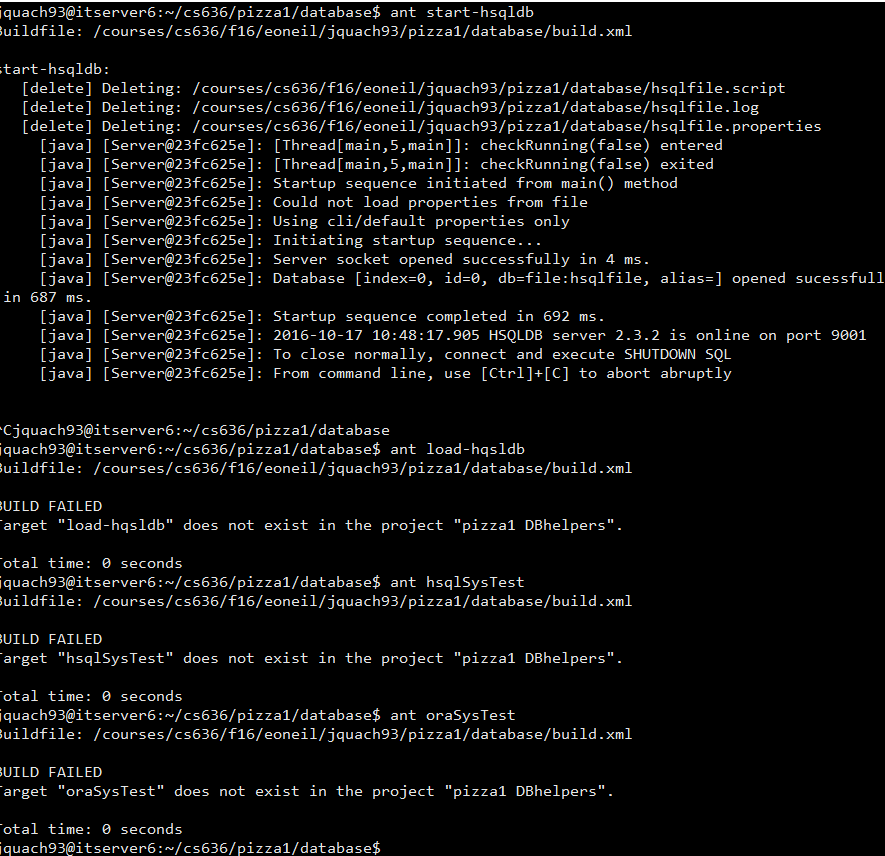
2. **Java Exceptions**. Review exceptions in your Java text. Look at the exceptions in pizza1:

a. If the SQL in AdminDAO.advanceDay throws a SQLException, where is it caught?    What happens to that exception next?  Where is the ensuing thrown exception caught?  What is printed out in this case?



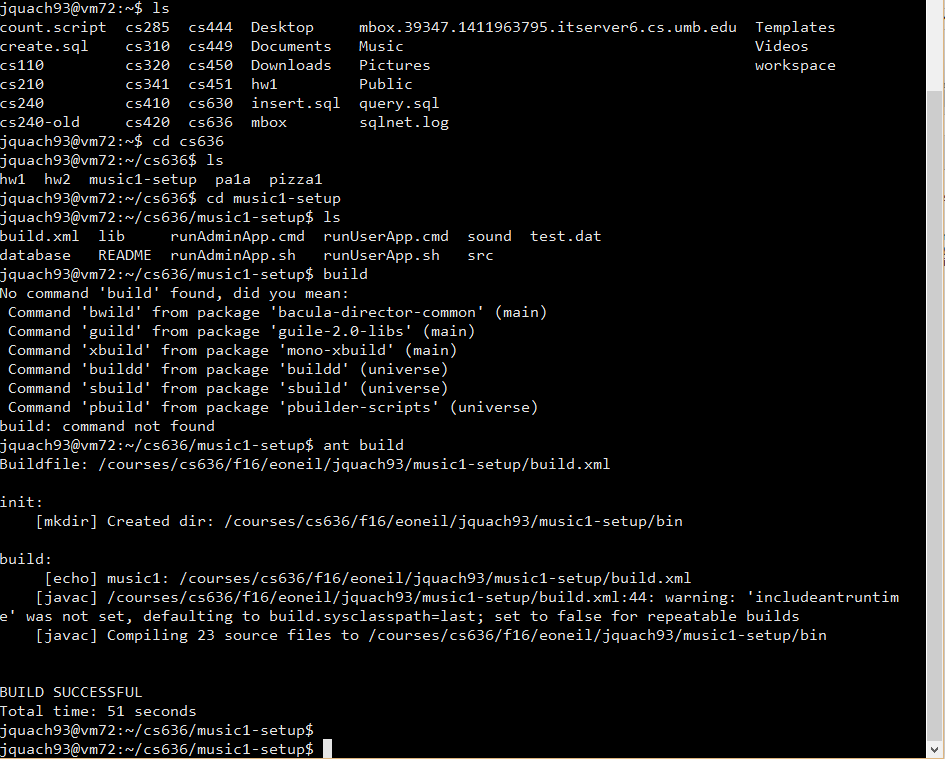
b. Make the scenario of part a happen by changing current\_day to current\_dayx in the SQL update statement.  Run SystemTest on all three databases (ant oraSysTest, etc.) and record the exception report from each of the three.  You will see that the databases report errors with different messages and sometimes different information included in the message, so sometimes it’s worthwhile to run on more than one database to try to figure out what’s wrong.



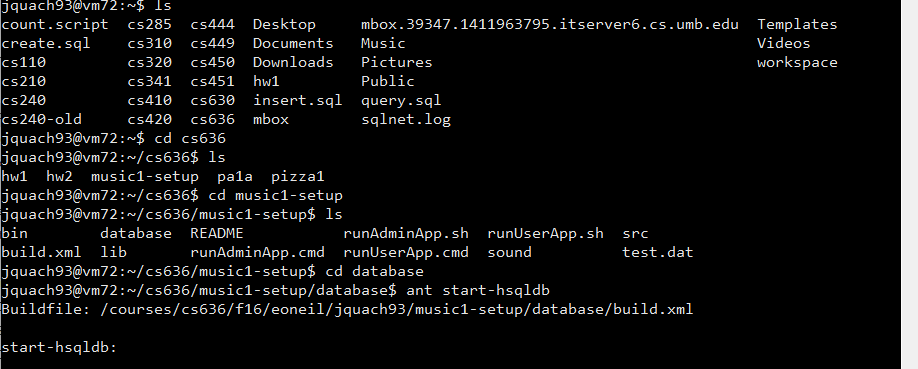


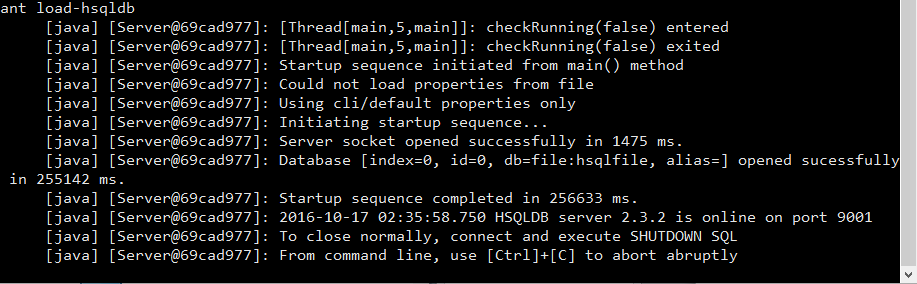
3. **Music1-setup.**

a. Install the supplied music1-setup project and copy it to music1, for further development.

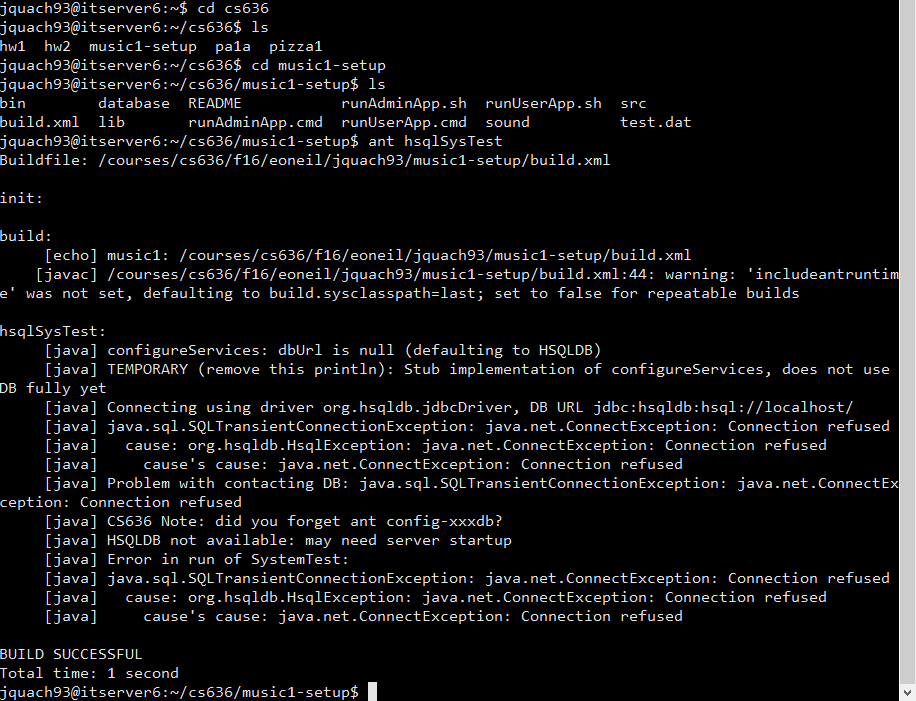


b. Use music1-setup’s database/build.xml with ant to start hsqldb and load all three databases. Report your ant commands and success or problems.

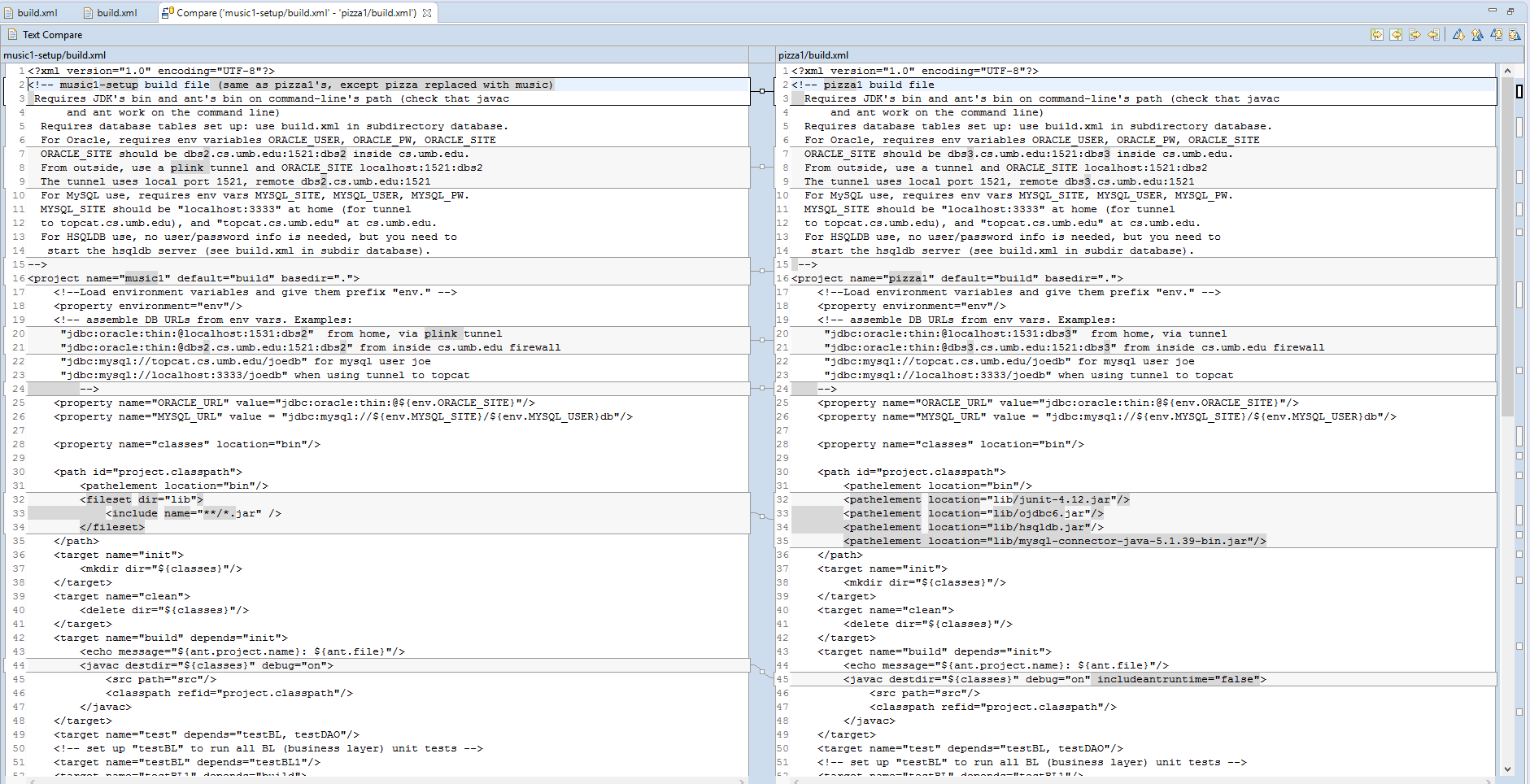
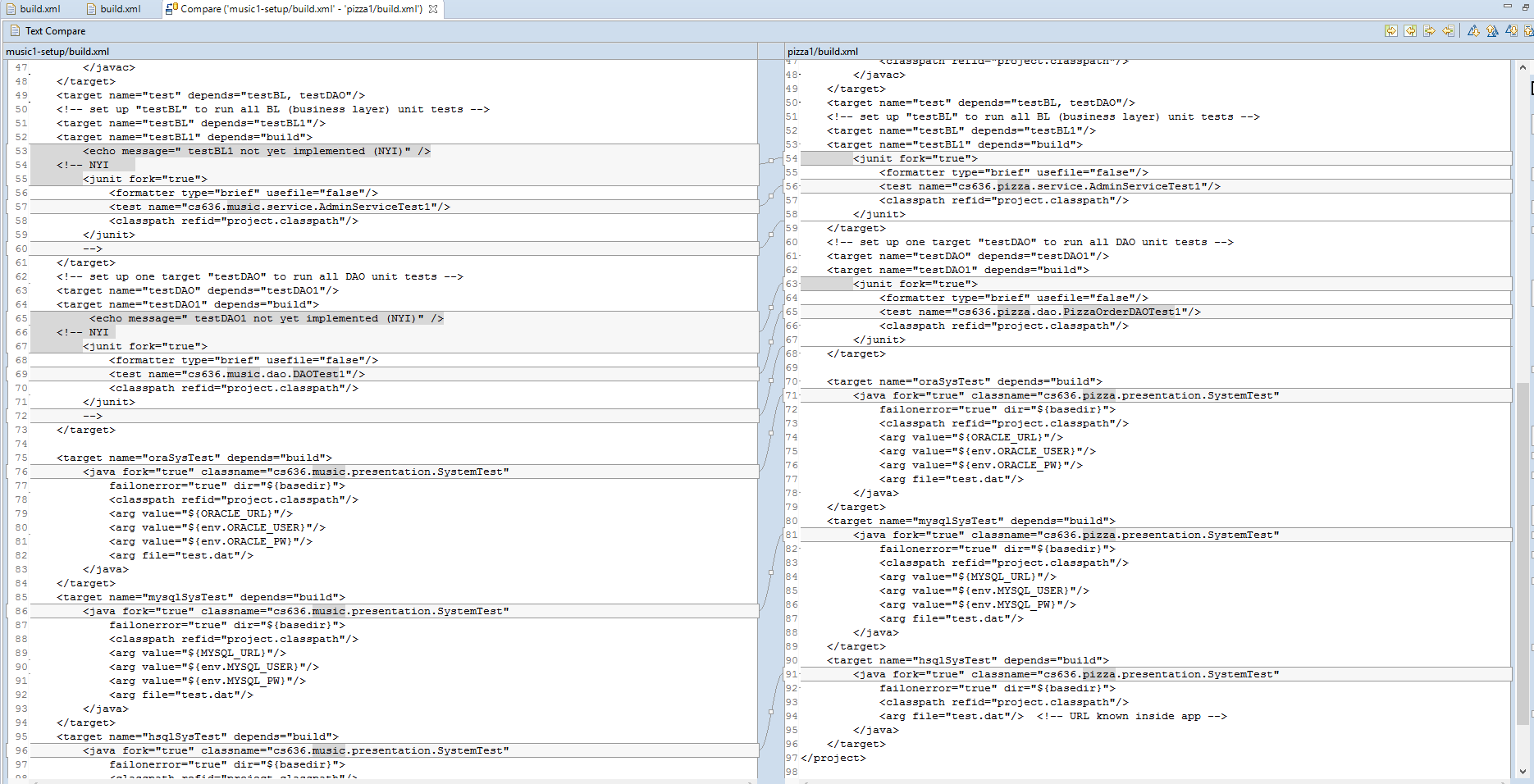




1. Use music1-setup’s build.xml to run the SystemTest on all three databases. Report your ant commands and success or problems. Note that the output from SystemTest will be skeletal, with not much content yet, but it shouldn't crash.



d. Set up an eclipse projects for music1-setup and music1, and compare music1-setup to pizza1, by selecting one in Project Explorer, then hold down control and select the other, then right-click and Compare With>Each Other.  Then display the differences for build.xml by selecting it in the resulting file display.  Describe the differences.  Note that this technique will be helpful to track your changes from music1-setup to music1.

Differences vary in path id location, javc destination directory.

Difference in junit fork in which one is part of the music admin service test vs pizza admin service

Differences also vary in .presentation.SystemTest

4. **Music DB SQL**.  Write queries as follows and show the queries and their output (after the inserts of d. are done) in your hw paper. Each part except d should be answered with one SQL query. You can use eclipse Database Development tools to try out the queries, or write a .sql file and use the same method as in "ant show-oradb", etc. Or if you want, login to dbs2 and use the sqlplus tool or topcat and use the mysql program. Note that many tables are empty after the ant-load. You can load some sample rows into table downloads with the provided load\_downloads\*.sql (ant load-mysql-dl, etc)

For queries, report the SQL and results, if any.

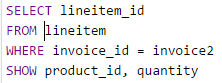
I barely luck getting sql to run, but I’ll make do.

1. Find all users (user\_id's) with email addresses ending with "gmail.com".



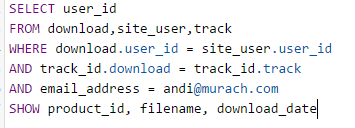


1. Find all lineitems of invoice 2 and list their product id and quantity.



1. List all the downloads (product id, filename, and date) for the user with email [andi@murach.com](mailto:andi@murach.com).







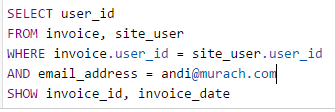


1. There are no invoices in the tables in the original load, making it hard to test queries involving invoices.  Fix this problem by inserting an invoice 2 (dated 9/2/10, for user 2) with lineitems for products pf01 and jr01.  Show the three insert statements in your hw paper.



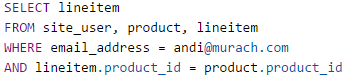


1. List all invoice ids and invoice dates for the user with email [andi@murach.com](mailto:andi@murach.com).



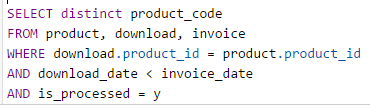


1. Find all product ids purchased by the user with email andi@murach.com (i.e., find the lineitems)



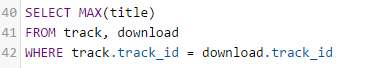


1. Find all products (product codes) where a download preceded a purchase. Avoid duplicates in the result.



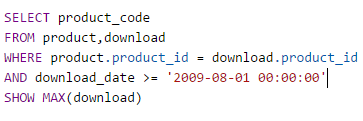


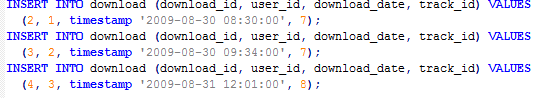
1. Find the downloaded file (.mp3 filename) that was downloaded the highest number of times.





i. Report on the number of downloads (total over all files for that CD) by CD product code, since August 1, 2009.



]



3 downloads.

5.  **A Little start on music1 (pa1a).** See the [project 1 spec](http://www.cs.umb.edu/cs636/pa1.html). Write a program Register.java (in src of your music1 project) to insert a new user, directly, no User object, no "DAO" yet.  This brute force starter program follows the idea of using all the pieces of the needed technology as soon as possible.  Elegance can wait a bit.  Show the source in your hw paper. Feel free to copy code from JdbcCheckup.java and Java files in the music1-setup project.Register.java should accept database information the way SystemTest does, but it has no input file. Don’t take any user input for the information about the new user but rather just invent values in the program. Add targets to build.xml of the same form as the build.xml targets oraSysTest and mysqlSysTest and hsqlSysTest, except call them oraTest and mysqlTest and hsqlTest, and drop the test.dat file argument. This way, the final test run (after freshly loading databases) is “ant clean”, then “ant oraTest” and “ant mysqlTest” and "ant hsqlTest", using username and password from environment variables.

