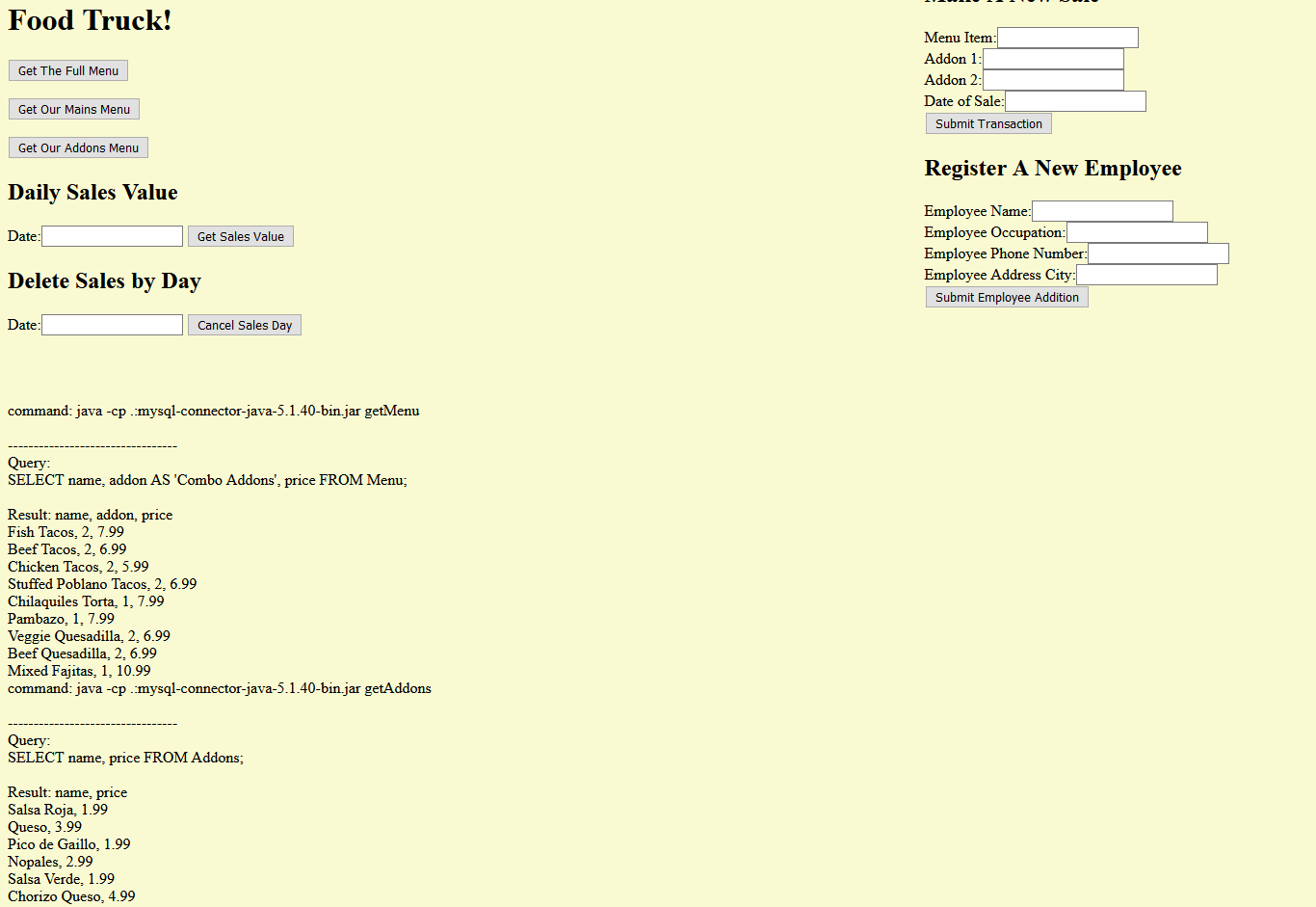
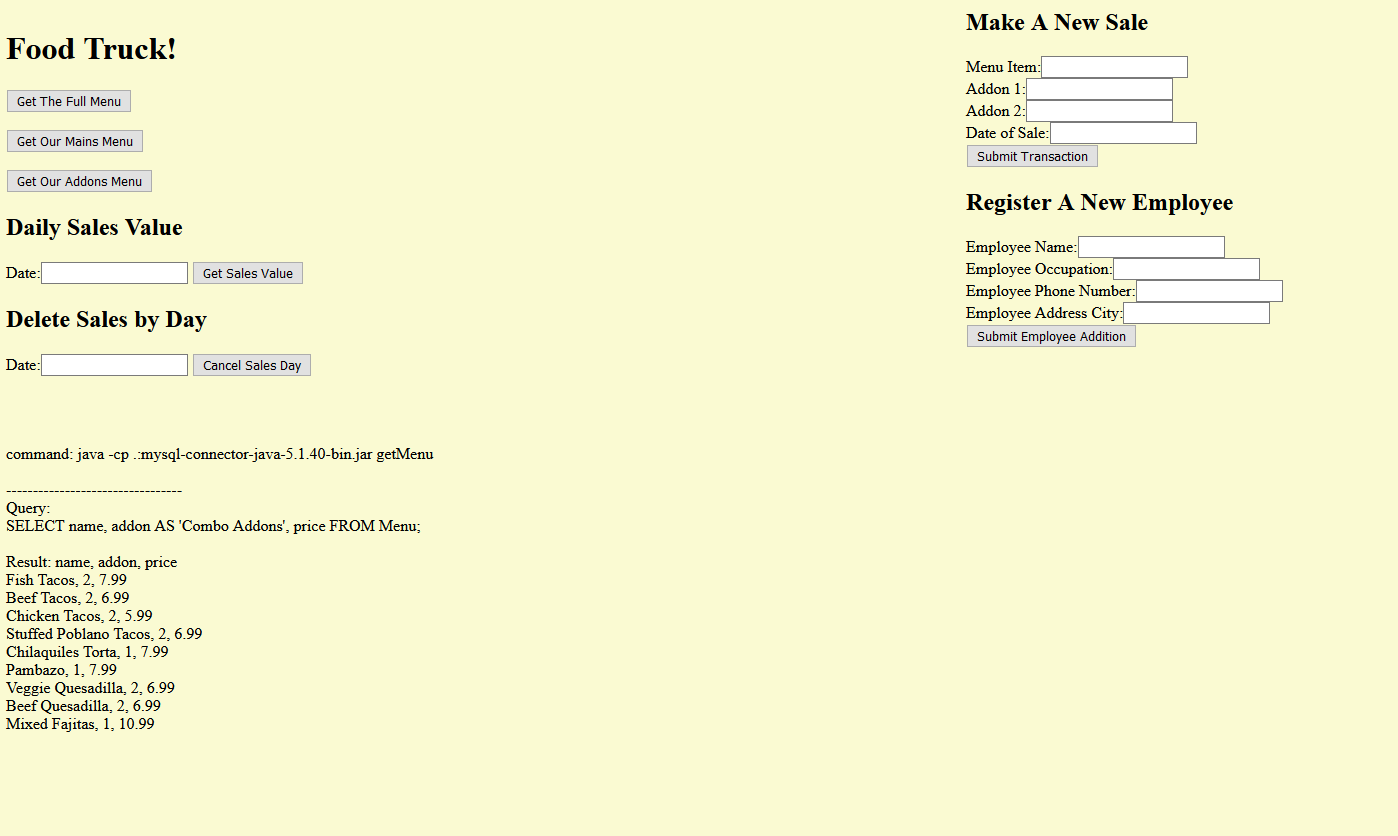
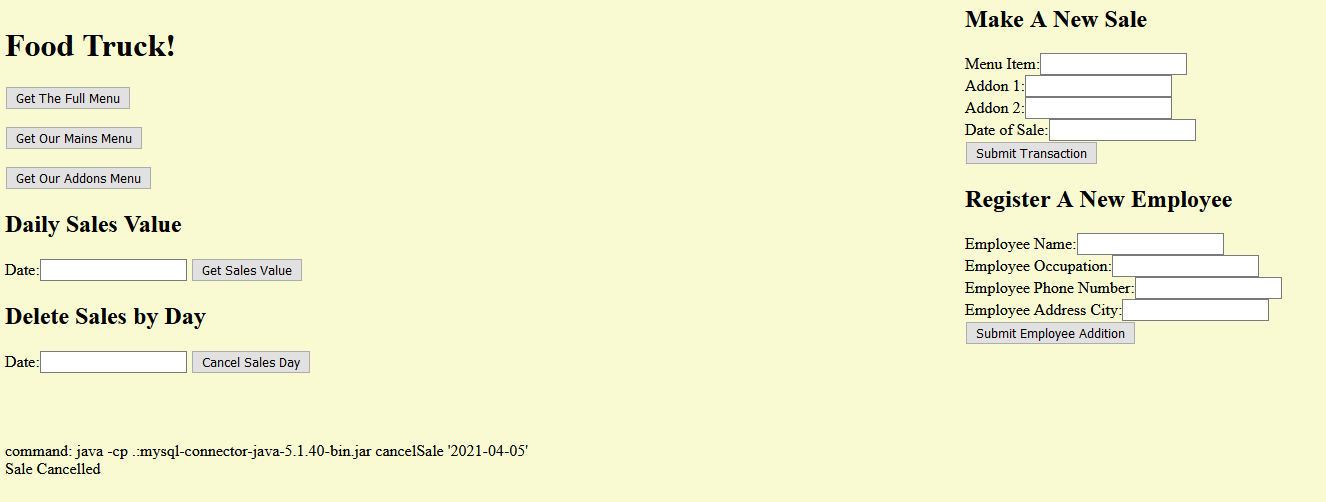
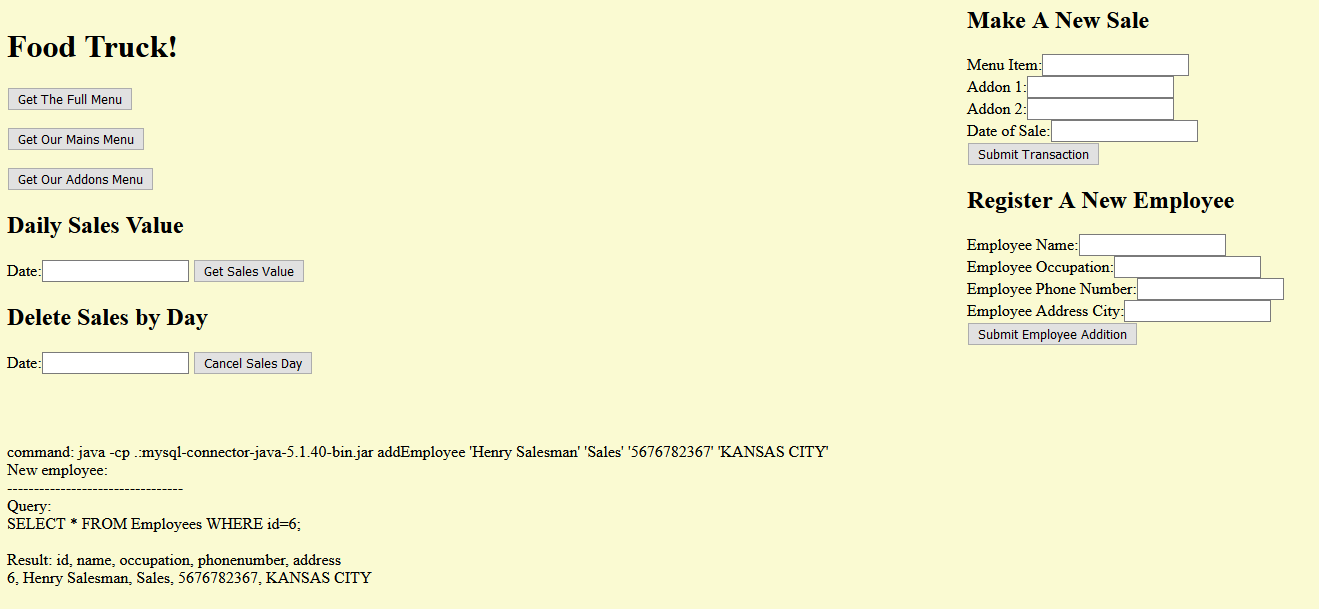
# Homework 5

My database example is a simple database for a food truck including tables for menu entrees, side dishes and extras, employees, and sales. The menu table contains attributes id, name, addon, and price. The menu, addon, and sales tables interact with the addon attribute in a specific way. Menu items will either be addon 1 or addon 2, and this specifies how many sides a menu item comes with. The addons table contains attributes id, name, and price. The sales table contains attributes transid, itemid, addon1id, addon2id, and sale date. Addon2id can be null, as some menu items only allow 1 addon. The least involved table is the employee table with attributes id, name, occupation, phonenumber, and address.

For my implementation, I used Java and PHP like the given example code. I used the apache hosting on turing out of the public\_html directory and used mysql on turing for my DBMS. The url for my page is <http://csce.uark.edu/~robell/web_DB/homepage.php>  
  
The website design is somewhat rudimentary at this stage but is functionally complete.

The functions I used for the database begin with accessing the menu and addon tables. Both functions access a different table but are functionally similar (both are full table accesses). There is also a button to call both functions in one POST request to present the full menu for creating new sale entries.  
  
*Full Menu Result*  
 *Mains Menu Result*  
 *Addons Menu Result*The Interactions with Sales make up half of the website’s functionality. Daily Sales Value takes a date in the form yyyy-mm-dd and returns the menu price of all orders made on that date by joining menu and sales to identify the price of mains and joins with addons twice to get the price for each addon in a sale. Then the sum of these columns is summed into a single one-entry column that is then printed onto the website.  
  
*Daily Sales Value Result*Delete Sales By Day takes a date similar to the previous function, but instead deletes sales made on that day. It is a functionally complete, but impractical function. To extend this and make it more useful, I would extend this with a multi-conditional delete, as many sales would occur in one day and only deleting by date without time or another identifier is impractical. A method was added to jdbc\_db delete() that deletes records like the given insert method. I would have to add a new attribute to the table that would make identifying a sale beyond id easier, as using menu item or addon combos is not very precise either.  
  
*Delete Sales by Day Result*  
Make A New Sale takes an order complete with Menu item, both addons, and date of sale and adds the entry into the DB. To generate the id, a new method getQueryRecords() was added to jdbc\_db.java that takes a parameter set number of results and a query and returns that number of results at the method call. This was used to query the DB to get the largest id in the table. This id is incremented by one and then set to the new sale’s transid.  
  
*Make A New Sale Result*Register A New Employee behaves like Make A New Sale in that it will also complete the id field with getQueryRecords() by taking the largest id from Employees and incrementing it. However, the employee table does not have any access in this website besides a result string that would not appear in a better implementation. An extension of this would include more functions to access Employees in some useful manner such as access by occupation.  
  
*Register A New Employee Result*

The Functions were all checked for error in the website and in the command line on turing. A consistent warning given for all functions is a

*javax.net.ssl.SSLException*

*MESSAGE: closing inbound before receiving peer's close\_notify*

This exception seems to have to do with disconnecting from the DB at the end of any function call and is associated with SSL and Java but has no impact on the website’s functionality in all testing. This issue is also only visible in the commandline and not on the website (screenshots were taken with this issue present in commandline calls). For a lack of impact on the project at turn in, this is currently unresolved.   
  
User input is my biggest concern with this project’s functionality, and currently opts for graceful SQL failure. If input is incorrect or does not give meaningful results the page will either return a null value, return a confirmation that is only functionally a confirmation that the java command did not fail before disconnecting from the DB, or will leave the database unchanged in the event of an incorrect insert. In the future, I would like to make more comprehensive user feedback on SQL query failure and implement more user-friendly input. However, user-friendliness was prioritized after functionality and time only allowed me to improve the layout and page presentation somewhat.