**Project Report**

**Introduction**

This project focuses on eCommerce and the implementation of a browser-based online store. This necessitates a large database of “items” available for consumer purchase with database columns such as item number, price, description, category and keywords. Users can browse the site and retrieve the entire item database, while also narrowing displayed results through the use of categories and keyword searches. Users have the ability to add selected items to a “cart” to mark them for purchase. When the user chooses to display the cart, it will calculate a total payment amount based on the sum of item prices.

The basic function includes six parts,

1. Register & Login
2. Search items
3. View Items: give comments
4. Add to Cart
5. Cart
6. Purchase: Google maps

The “user” class has standard read privileges for the item database, as well as the potential ability to store profile information like addresses and payment sources. Storing user payment information will require the use of hashing to avoid storing user credit card or bank account info in plain text.

We utilize several tools from the Google Maps API in concert with our own code. “The Google Distance Matrix API” is a tool that returns travel distance and time between start and end points. With this information we will determine the travel distance from our shipping facility to the shipping address provided by the user. This will provide the user with a shipping cost which will be calculated based on an algorithm that adds a small processing fee along with a per mile cost. With this information, we will also be able to provide the user with an estimated arrival date for their shipment based on the travel time returned by the Distance Matrix, an algorithm to divide this travel time by 8 hour days, while accounting for non-delivery days, will be utilized.

Our project leader is Scott Melenbrink. Here is a table that demonstrates the distribution of tasks for each teammate.

|  |  |  |  |
| --- | --- | --- | --- |
| **Num** | **Task** | **Database** | **Student** |
| 0 | ERD |  | Scott |
| 1 | Google API & distance | Warehouse, address of user | Scott |
| 2 | Search, Sort | Inventory | Xiaomin |
| 3 | Login & Register | User\_info | Cory |
| 4 | Cart | Cart | Xiaomin |
| 5 | Add to Cart/Purchase | Authorization | Xiaomin, Cory, Scott |
| 6 | Triggers |  | Scott |
| 7 | Item Page | Inventory, comment, cart | Xiaomin |
| 8 | Recent View Item | Recent\_View | Cory |
| 9 | CSS, Layout of page |  | Cory |

Scott will use his knowledge of login functionality from his Software Engineering course to help develop the login and user class system for our online store, and also create tables in the database. Scott, Cory and Xiaomin together are responsible for developing the methods to securely store the user’s payment information, including the payments table and determine how it will correspond with the user table. Scott is also responsible for the cart functionality and integrating the Google Maps API tools with our code; his database responsibility will be to make sure the interface to the Google Maps database functions appropriately. Cory Hake works on the login system, and is also responsible for developing the layout of the site using PHP, HTML and CSS, this will require working intimately on the design and implementation of the items table to display a template page for each item. Xiaomin Wang work with the items table and developing the appropriate PHP code, and she is responsible for developing PostgreSQL queries for the keyword search and category functionalities, and for the item view page, and also the comments that associate with each item, and also she is in charge of making presentation slides and the minutes of each meeting. Our group meet weekly, from 8:30-9:30am in the Engineering Library, W2001-E.

Our main dataset will be an “inventory” table that will contain all of the products available for sale to users in our online store. This table will need to have a unique item id number sequence to function as the primary key. Other fields will include product name, price, category and description. The “inventory” table will have a “comes from” relation to the “line item” table, and our “warehouse” table will have a “has many” relation to it as well.

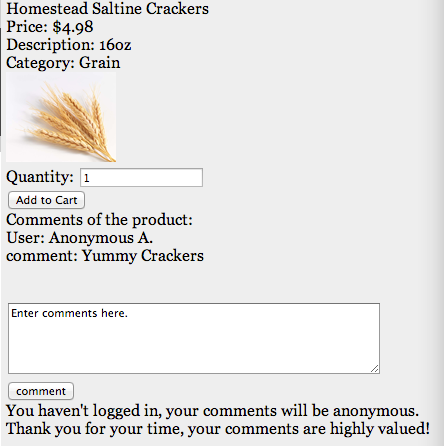
A “cart”/”Order” table will store all orders placed by users, using a unique order id number sequence as the primary key. It will then be composed of line items from a “line item” table, each of which will contain a line id as well as an item id and quantity of an individual part on the order, as well as a subtotal price for that line to be totaled in the order table. The “orders” table will also contain a foreign key to our “users” table for cross-referencing and recalling a customer’s account history. The “users” table will represent our individual users to track their order numbers, consisting of a unique user id number sequence as the primary key, as well as demographic information like phone, email and address. The address fields will be cross-referenced with data from the Google Maps API to allow for estimating delivery times. The “users” and “orders” tables will both contain foreign keys to a “payments” table where each user’s credit card information will be stored securely using hashing. The payment table will utilize a unique payment id number sequence as its primary key, as well as fields for credit card number, expiration date and CV2 code. Furthermore, we add a comment table that uses (userid, inventoryid) as primary key.

**Item page**

After you search the keyword, such as ‘milk‘, it might return several milk products, and you want to take a detailed look at each item, and compare the price. If there are reviews given by other customers about each item, which we can take reference from， it would be ideal for a user.

For this page, there are three basic functions in this item page. First, detailed description of this item, such as, price, category, weight, and so on. Second, the ‘comments’ function. You can read comments from other costumers, and also submit comment by yourself. If you have logged in, your name will be shown with your comments; otherwise, your comments will be anonymous. Third, add to Cart. If this item is in stock, you should be able to add as many as in stock items into your cart.

Here is a snapshot for this page, and I will give a detailed description of implementation about these three functions.



First, when you click the link from the list, I will use $\_GET[‘id‘] to get the inventoryid of the item, and then use te following query to display all the information of this item from table inventory(inventoryid PK, price, description, category):

SELECT \* FROM gang.inventory WHERE inventoryid = $\_GET[‘id’];

There is one photo associated with one category; there are six in all for six categories. We can also get the right image for this item based on the category info.

Second, for the comment function, we have one table comment(inventoryid, userid, text), the primary key for table comment is (inventoryid, userid), which are also foreign key from table inventory and table user\_info separately. We will use the following query to display all comments for this item, and since we want to display the user name which is from user\_info table, we will use inner join to fulfill this function.

SELECT \* FROM gang.comment INNER JOIN gang.user\_info USING (userid) WHERE inventoryid = $\_SESSION[‘inventoryid’]

For the submit comment, if you have logged in, we will use insert to insert what you wrote into the comment table by the following query:

insert into gang.comment(userid, inventoryid, text) values(userid, $\_SESSION[‘inventoryid’], ‘text’);

If you haven’t logged in, your comments will be under anonymous, here there is a trick here, I manually create a user named by ‘Anonymous Anonymous’in user\_info table, so all comments that create by user who hasn’t logged in will be under ‘Anonymous Anonymous’, in this way, we can keep track all the anonymous comments.

Third, the ‘add to cart’ function. This function should enable the user to add this item to his cart after successfully logged in, and also select the quantity he wants to buy. This function will insert this item into table cart by its inventoryid.