# Group Project Managing Inventory
CSCI221 Section 1
Assigned: Monday, April 9, 2012
Due: Wednesday night, May 2, 2012

## I. CORE REQUIREMENTS

This project involves you working in groups to get practice for the follow-up class, CSCI321, which is a project-based class in which all of your grade is based on one group maintenance project and one other original group project. The objective is for you to gain experience in working effectively in groups using the Git concurrent versioning system on a C++ project that is more complex than the small individual programs you have had thus far.

For this project, the class will be broken up into teams. The teams will be listed separately on Blackboard. Below are some of the core requirements for this project:

1. For your group, create one Git repository on Bitbucket which also contains a Wiki and Issue Tracker (which are options provided when creating the site.) You are to invite me to the repository so I can monitor your progress.
2. In your Wiki, you are to provide a Software Project Management Plan (SPMP). Given that you only have a short time to work on this, I do not expect you to be accurate in your planning. However, I do want you to do the best you can in assigning work with milestones and deadlines to each member of your group.
3. You must provide a reasonable set of test cases for each of your classes.
4. All information must be stored in flat files and read in when running the code. When a change of data takes place, you may simply store it in memory until the program is stopped, at which point you will write it all to disk in each file. Make sure you have a separate directory called `data` to hold your files. (You can, however, choose to keep data updated properly in the files as changes are made if you choose. This is more difficult and more correct, but not required.)
5. You must use the Standard Template Library data structures to store your information in memory while the program runs. For the `Category` and `Merchant` classes, you must use a `map`.
6. For each new order that is placed, increment the day by one so that they are not all occurring on the same day.

## II. CORE CLASSES

While not complete, below are some of the classes and attributes you will need to provide when developing your software:

TABLE I: `Customer` class

| |
|---|
| customerID |
| user name |
| password |
| full name |
| address |
| city |
| state |
| zip code |
| amount in account |

For `Order`, you may assume that there is only a single `SKU` purchased per order, though the customer may purchase as many of that `SKU` as they choose as long as that quantity is available in inventory and there is enough money in the customer's account.

TABLE II: `Order` class

| |
|---|
| orderID |
| customerID |
| SKU |
| quantity of SKU purchased |
| price (store in cents) |
| date ordered |

TABLE III: `Inventory` class

| |
|---|
| SKU |
| item description |
| categoryID |
| price (store in cents) |
| quantity in inventory |

TABLE IV: `Merchant` class

| |
|---|
| user name |
| password |

TABLE V: `Category` class

| |
|---|
| categoryID |
| category name |

For your `Category` class, you may assume that whatever is entered in the file that stores this data will not be changed when running the program. That is, you can hard code the items in the file, read them in when the program runs, then close that file and leave the data in memory. You don't need to provide any means of changing, adding, or removing categories. It can all be done with `vi`.

## III. USER INTERFACE

For your user interface, you must only provide a text-based system. The requirements for these are:

### A. Customer UI

Ask if new or returning user; if new user, create account then return to same screen asking if new or returning user to allow person to actually log in. If returning user, log in. Once logged in, user should be able to:

1. display items for sale
2. purchase item (1 or more of the same SKU, but only as many as are in inventory and only up to amount in customer's account.)
3. display all of own orders
4. change amount in account (the only property of Customer that needs to be changeable once Customer is created.)

### B. Merchant UI

The merchant should be able to at least do the following:

1. log in (you do not need to provide a means for creating new accounts as you did for customers)
2. add and change inventory price and quantity (For SKU already entered, do not change description or category ID.)
3. list inventory
4. remove inventory
5. display orders by date or customer ID
6. when adding inventory, show categories to choose from

## IV. FINAL POINTS

Please understand that this project involves you changing how you have developed software before, even when working in groups. Effective collaboration involves using a concurrent versioning system properly, which means:

1. submit your own work; I will track each of your changesets to see who does what and will assume that whoever submitted a change is responsible for it (your grade depends upon it, but more importantly, it is not an effective collaborative strategy when working on much larger projects)
2. do not work all together, email code bits to each other, copy and transfer files to each other with flash drives, copy files between yourselves (be that on londo or any other system.) Again, if you copy files to each other and one person commits all of the files, I will assume that person wrote it.
3. update your Wiki whenever you change your SPMP or anything there regarding your project.