**Assignment Report**

**Introduction:**

In assignment, we are going to implement new functionalities to the marketplace application by using concurrency and synchronization that make the application more reliable and dynamic. In this marketplace application, the multiple clients will try to access the same browsing item at a same time and the changes in the action performed by client should reflect in the marketplace. That is, when there are 5 clients in marketplace application trying to purchase the gadgets and there are only 4 gadgets that are stock. When the client 1 purchases the item then the number of gadgets in the application will be 3 and it should update in the marketplace database and it should be synchronise immediate once the client 1 purchases the gadget. The marketplace application will get benefits from the concurrency and synchronization more efficiently and it makes the application reliable for on-going transaction as well (not part of assignment). The Concurrency and Synchronization will work very powerfully in making the application reliable and the carrying the action simultaneously without the involvement of the person(administrator) for monitoring the actions. The concurrency in this application is described as multiple clients that they can access the marketplace at the same time without causing any problem. The synchronization can be implemented as when the client purchases the item from the browsing item list then the item in the list should change its state simultaneously when the client purchases the item.

**Concurrency:**

Concurrency is the ability to run several programs or several parts of a program in parallel. If a time-consuming task can be performed asynchronously or in parallel, this improve the throughput and the interactivity of the program. Concurrency enable a program to achieve high performance and throughput by utilizing the untapped capabilities of underlying operating system and machine hardware.

**Thread:**

The backbone of **java concurrency** are threads. A thread is a **lightweight** process which has its own call stack, but can access shared data of other threads in the same process. A Java application runs by default in one process. Within a Java application we can work with many threads to achieve parallel processing or concurrency.

**Process:** A process runs independently and isolated of other processes. It cannot directly access shared data in other processes. The resources of the process, e.g. memory and CPU time, are allocated to it via the operating system.

**Locks:**

Java provides locks to protect certain parts of the code to be executed by several threads at the same time. The simplest way of locking a certain method or Java class is to define the method or class with the synchronized keyword.

The synchronized keyword in Java ensures:

* that only a single thread can execute a block of code at the same time
* that each thread entering a synchronized block of code sees the effects of all previous modifications that were guarded by the same lock

Synchronization is necessary for mutually exclusive access to blocks of and for reliable communication between threads.

We can synchronize keyword for any class or method:

**Syntax:**

public **synchronized** void Demo()

{

// some thread critical stuff

// here

}

You can also use the synchronized keyword to protect blocks of code within a method. This block is guarded by a key, which can be either a string or an object. This key is called the **lock**.

**Impact of Concurrency on Java RMI:**

**Java RMI supports concurrency but there are some issues that are associated with it.**

**For instance:**

1. In a Marketplace system, if there are multiple clients who are trying to access the system to login into the system then RMI will create the remote object and the server will create the thread ‘t1’. When another client (client2) want to access the system then the server will allocate the same thread for the client2 to access instead of creating another thread. Server will run the two process with the same thread and the client 2 will get access with the thread (t1). This will create the unauthorized access to the client2 and it is unrealiable.
2. Let us think that in the system there are multiple clients and when the client 1 has a lock on the object in the server, when the other client (client2) tries to access the object then the server will not allow the client 2 to access because the client 1 has created lock on it. If client 1 is dependent on the client2 to execute some request then client2 doesn’t execute the request for client 1 because the client 2 doesn’t get access to the object in the server and this will leads to the Deadlock situation. The concurrency issue and the objects are blocked and waits for the release of the lock on the object in server.

**RMI supports threads and locks:**

Java RMI supports the concurrency and synchronization and the threads will have some concerns in the distributed system environment. If the client1 has requested the process then he lock the object in the server. If the customer 2 wants to access the object in the server then the server doesn’t perform the process for the object requested by the customer2 because the resources are in wait state that causes the resources to give priority for the thread that has highest priority and it will remain in the wait state until the highest priority thread process. This is often leads the waiting of resources for other thread to execute and this is called **Resources Starvation.**

**Domain Model: There is no change in the domain model**

**Functionality of Items:**

In this assignment, I have added the three main functionalities to the marketplace application i.e., the functionality is based upon the user’s role.

1. Browsing items (Admin, Customer)
2. Adding items (Admin)
3. Purchasing items (Customer)

**Browsing Items:**

This functionality is applicable to both roles like customer and admin. In this functionality, the customer generally browses the items in the marketplace application and views the list of the products that are in stock and views it in the appropriate view page(**CustomerView**). In this application, I have used the files to store the items. In order to retrieve the contents of the file (Browsing items) I have used the

FileReader, BufferedReader class to read the contents of file line by line. When the user enters, the login credentials the irrespective to the user’s role this functionality applicable (both customer and admin) can browse the list of items in the marketplace in their appropriate view pages. This functionality of marketplace application uses the concurrency feature of java. The multiple clients in marketplace application can browse for the items at the same time and it also uses features of synchronization, when the client purchases the item from the browsing list then the items in the browsing list should be updated simultaneously. When the client or admin checks for the item then it should show the current number of items in the list and type of items available in the browsing list. The admin views the browsing items in the **AdminView**.

**Adding Items:**

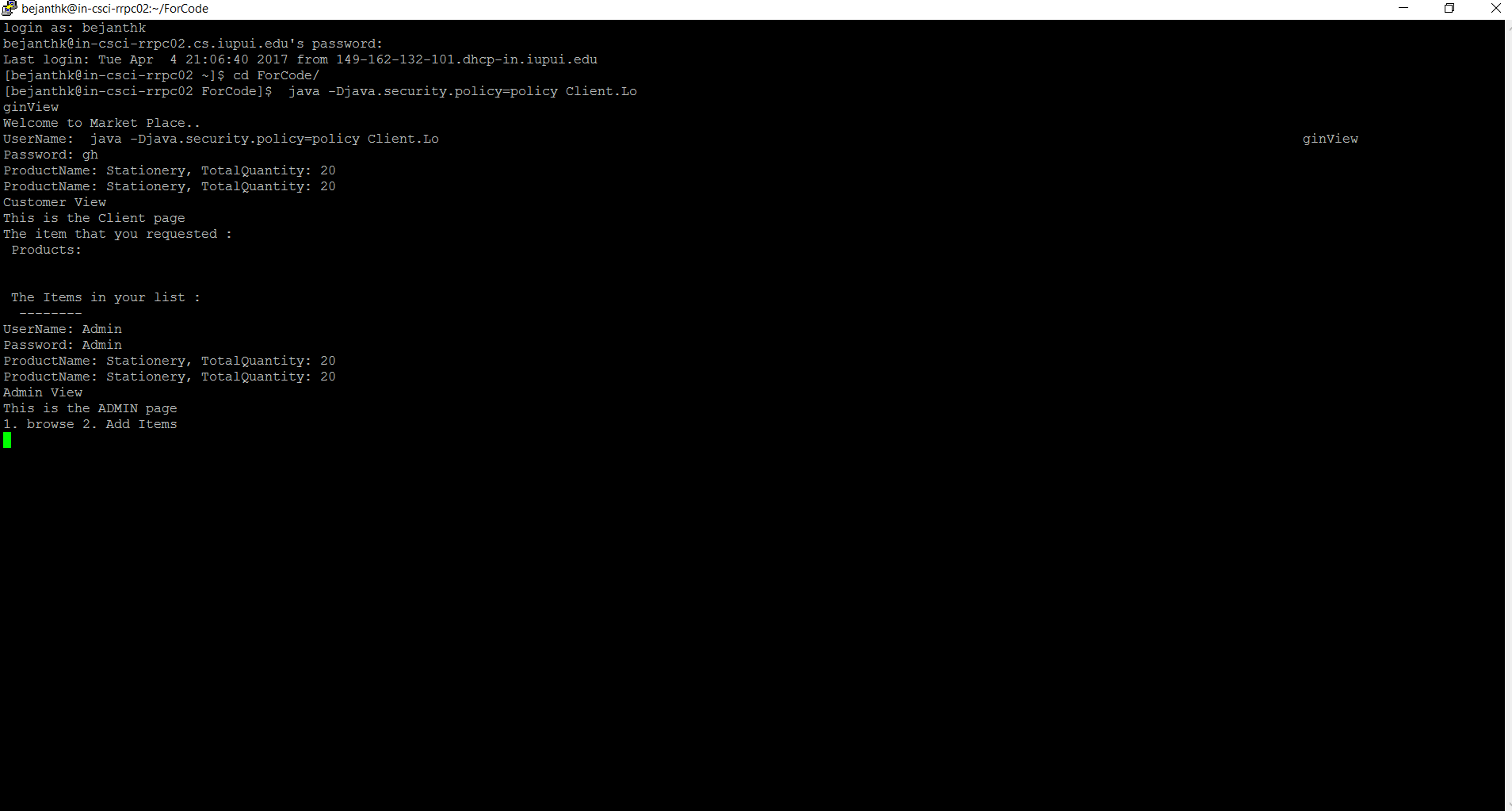
In this marketplace application, I have used the specific functionality that is based upon the role of the user. The Add Item functionality uses synchronization concept that when the admin adds the item to the marketplace application (browsing item list) and when the customer browses for the item then added item will be shown in the browsing list. In this application, the admin will have rights to add the item to the marketplace application. When the admin adds the items to the marketplace application, the item should get add to the items list immediately. And, when the customer browses for the items then added items should show in the browsing item list concurrently.

**Purchase Item:**

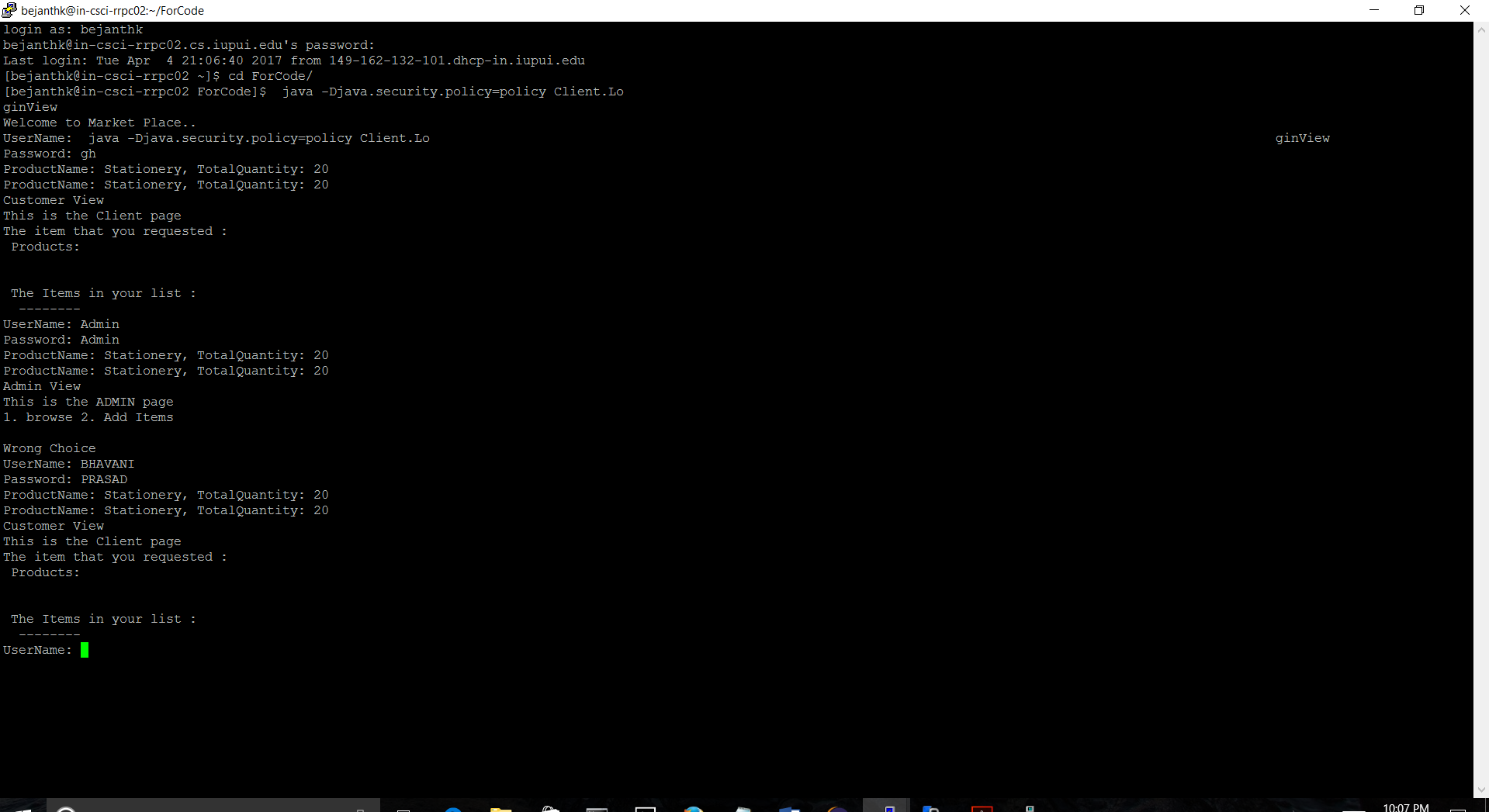
In this functionality, I have created the purchase item for the customer. The Customer will make an item to the shopping cart. And, functionality will enable the customer to purchase the item. The customer will make will selects the browsed item and carts it and then purchases the item. The Admin doesn’t have this functionality as the functionality is restricted to the admin. In order to purchase the item the admin has to have a customer account to purchase the item.

**Results:**

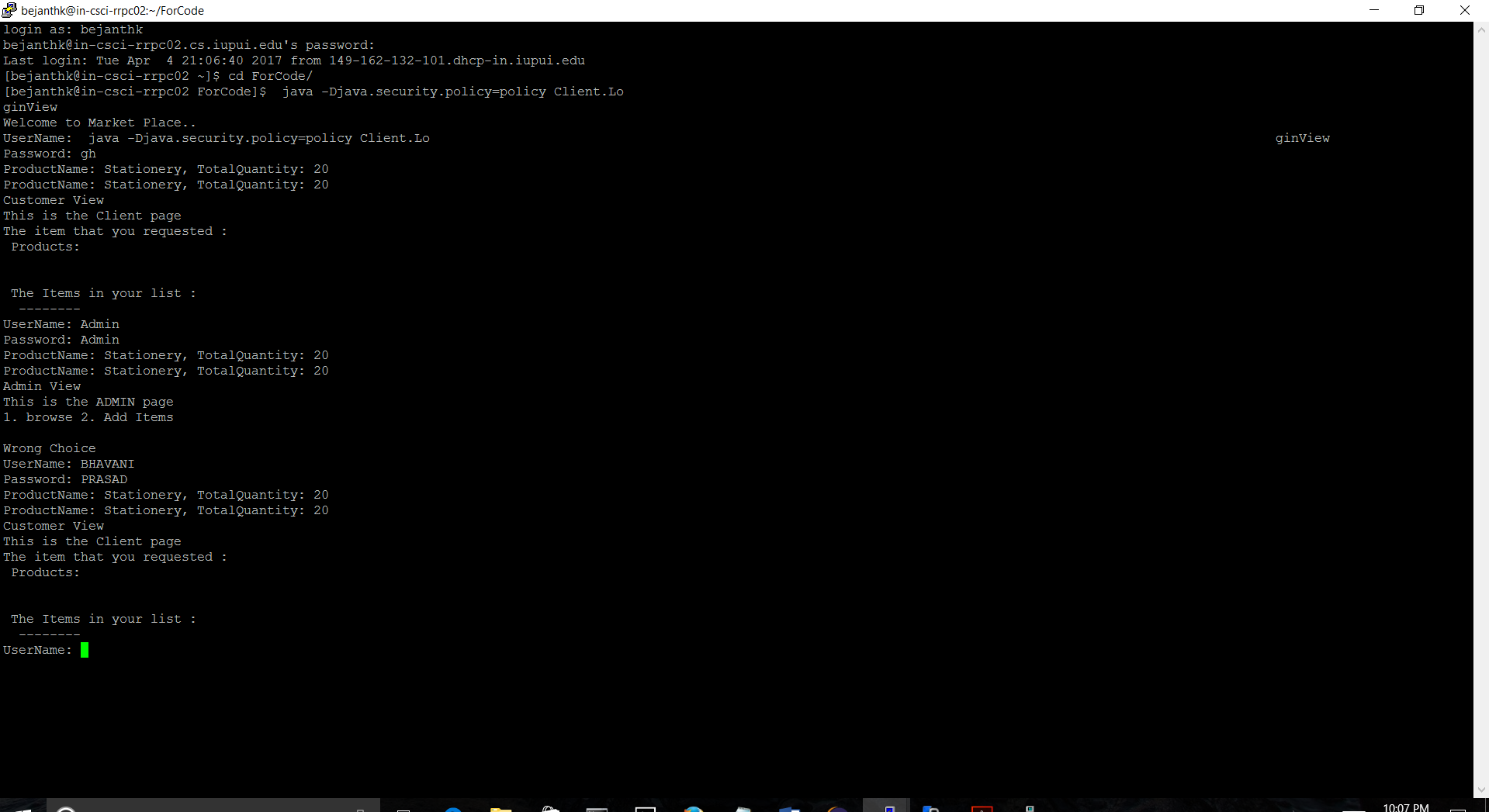
**The browse item functionality is for both customer and admin. The user view is based on the login details that they enter.**



The Admin will logins into the administrator account and adds the item to the browsing list. The browsing list will contain the new changes that made by the admin. The change in the state is shown and the it is concurrently.



The client will have the customer will have the customer functionalities like Browse, purchase item.

The admin will have the functionalities like the add the item to the list and the added items are viewd by both customer and admin.

References:

<http://www.vogella.com/tutorials/JavaConcurrency/article.html#concurrency>

<https://www.safaribooksonline.com/library/view/java-rmi/1565924525/ch11s06.html>