CS795/895:.Net Security
Summer 2015
Course Project
Due on August 1, 2015
Secure Distributed System to Support a Retail-chain
Specification
(Teams of 1 or 2 members)

As you all know, a retail-chain has stores distributed all over a region, a state, a country, or even around the world. While each store in the chain has certain independence, in general, the chains within a region or a country, are bound to some rules. A central office may fix the prices, the inventory may be monitored and new supplies ordered from a central place, and the sales may also be monitored by a central place.

In fact, the retail-stores may be operating in a hierarchic fashion. For example, a local office may coordinate all stores of a chain in Hampton Roads. All local offices within Southeastern Virginia may have another central control say in Richmond. All in Virginia may have one central office. May be all Southern States have a central controller. This may go on.

In essence, there is a hierarchical management structure. This hierarchy could also be used for better utilization of resources. For example, if there is a heavy demand for raincoats and umbrellas in Virginia where it has been raining, and there is no demand and plenty of inventories in Texas due to drought, then it is probably better to transfer these goods from Texas to Virginia rather than ordering additional supplies from the manufacturer. Such structure could also be used to make decisions such as the location of new stores.

In this particular project, which should be completed in less than two months (June 15-August 5), you should design the interfaces used at all levels (by customers, office personnel, and system administrators) and implement the infrastructure needed to support this hierarchy. The details of individual interfaces are left to each team. This is a team project with each team having up to 2 members.

Like any other secure distributed system, the main objectives of this system are: security, response time and availability. For security, we need authentication and authorization functions. You may use a database to store the valid user information. In terms of response time, when a user has a query, the user should be provided an answer as quickly as possible. If there is a choice of providing slow but accurate information versus providing fast but approximate answer, then the user should be given an option to choose. However, it is also possible to have a default choice.

Your project must incorporate the following features of .Net

Topic	Concept
Basic Concepts and Controls	Regular Expressions
	Validation Controls
	XML control
	Data-Bound Controls: Repeaters, DataList, DataGrid
	Calendar Controls
	User and Custom Controls
ADO.NET (Data Access)	DataReaders
	DataSets
	DataAdapters
ASP.NET Security	Windows Authentication
	Forms Authentication
	Passport Authentication?
Web Services	
Use of Crystal Reports	
.NET Remoting	

What is being monitored by the system? The system basically keeps track of two types of information: merchandise and personnel. For each item carried by the retail-store, the system should keep track of the sale price, current stock, and sales for that day, last week, and last month. Obviously, each item will be identified by a unique identification and a description.

In terms of personnel, the system keeps track of the employees at each of the locations including the administrative offices such as the zonal, regional, state, and national offices. For each employee, it will keep track of the employee ID, SSN, name, sex (male or female), job title, employment begin date, and current annual salary.

The types of queries/updates you will be implementing to show the utility of the system are given below. Along with each query/update, the arguments to be provided to execute the query are also provided. **(These are only examples for illustration. You may expand them or modify them.)**

U1. Change price: Item ID, Price, date effective (sale price being changed)
U2. Add an item: Item ID, description, price. (A new item is to be added)
U3. Add to Inventory: Item ID, quantity (Items arrived from manufacturer)
U4. Sales: Item ID, quantity (A customer is purchasing)
U5. Remove item: Item ID (Discontinue the item)
U6. Inter store-transfers: Item ID, quantity, from-store-ID, to-store-ID (Transfer stock from one store to another)
U7. Add a new employee: Employee ID, SSN, name, sex (male or female) , job title, employment begin date, and current salary.

U8. Remove employee: Employee ID (Employee resigned or fired)

Q1. Query item: Item ID (Customer service wants to know information about the item)
Q2. Display number of employees
Q3. Total monthly payroll (Sum of monthly salaries of all employees)
Q4. Employee information: Employee ID (Given employee ID show all information pertaining to that employee)
Q5: Total sales information

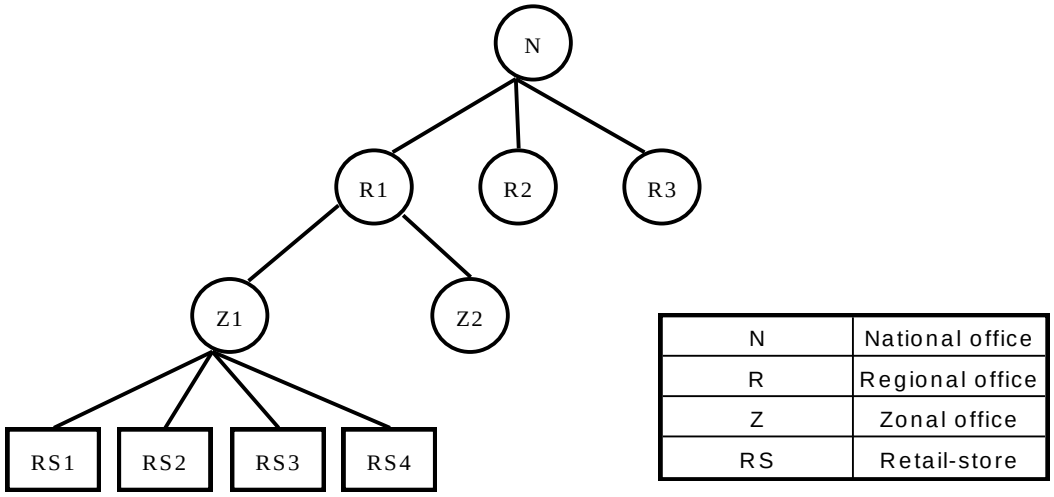

Figure 1. Example Hierarchic Structure

Scope of the Queries/Updates

At the retail-stores, all updates and queries (except U6) are allowed. At the higher level administrative office, however, only U1, U2, U5, Q1, Q2, Q3, Q4, and Q5 are possible. We now look at the semantics of each of these operations in detail.

U1: When executed at a retail-store, price only at that store is changed. If executed at higher levels, all retail-stores below that node will implement that operation. For example, if R1 executes this command, then all retail stores under Z1 and Z2 implement the change in price.

U2: All stores under the hierarchy implement the update.

U3: This may be executed only at a retail store.

U4: This may be executed only at a retail store.

U5: All stores under the node which execute this command remove the item from their inventory.

U6: This may be executed at any level. For simplicity, we assume that the from-store and the to-store will always be under the node that is executing it. So a retail-store cannot execute this command. Only higher level nodes can execute it.

U7: Only a retail-store can execute. For simplicity, we assume that the system does not maintain employee information of the higher level nodes. It is only interested in the retail-store employees.

U8: Can be executed only at a retail-store.

Q1: This may be executed at any level. Since this information includes the inventory of an item at a retail-store, all retail-stores under a node that executes this command should return corresponding values. For example, if Q1(Umbrellas) is executed at R1, then the results should display information of this item at all retail stores in this region. When executed at a retail-store, it should only display that store's information.

Q2: Similar to Q1.

Q3: Similar to Q1

Q4: May be executed at any node. Suppose an employee E1 is working at RS2, then Q4(E1) executed at RS2, Z1, R1, or N should give correct answer. In other words, a node can only enquire about employees only under its control and not everyone in the organization. (This simplifies your implementation)
Q5: Similar to Q4

Security issues

You must have a login/password system to authenticate all users. In addition, based on the user's role, you must present them with respective interfaces. You should use the role-based system offered by .Net. In addition, you may need to enhance it with your own. This will certainly need ADO.Net to store the login and passwords in an encrypted manner for authentication. The authorization information also should be stored in the database.

Availability Issues

It is possible that some servers may crash. For example, if the server at R1 crashes, it should still be possible for N to reach Z1 and Z2 directly. At the retail-store level, I suggest you use replication (two-copies are adequate) to increase reliability. So when one fails, the other can execute the queries.

Performance issues

As much as possible, try to provide good response time. For example, if Z1 were to issue a query and both servers at RS3 were down, your program should not wait forever to display the results since the other retail-stores have sent the replies. You can simply timeout and just display the results you have got with a note that RS3 is down. The switching of copies at a retail-store should be fast and transparent to the user.

Dynamic System

It should be possible to add new retail-store to a zone, add new zones to a region, and add new regions. Similarly, it should be possible to reassign a zone to a region. In such a case all the current retail-stores under that zone remain intact and simply shift with the zone. A retail-store can also be shifted from one zone to the other.

Since these are not user-level commands but more of the system maintenance, they were not listed above. In other words, you would provide two kinds of interfaces: administrative interface and a user interface.

Design choices

What is provided in this document is a broad guideline of the desired system. Each team can make its own design decisions. In fact, I expect as many varieties of systems as there are teams in the class. Spend sufficient time in coming up with the design, how it is structured, what features of the language you can use to implement them, etc much before the actual coding.

Project Grading

Each project will be accompanied by a document that describes the architecture, the features, the design choices, and implementation details.

A demo of the project should be given before August 1, 2015. During the demo, we will start from state zero, and start building the system such as creating a national office, regional offices, retail-stores etc. Alternately, I may provide you with a basic structure that e start with prior to testing so we don't spend too much adding the inventory etc. A server crash will be tested by killing a process and then see how the system will function.

Each team will be asked to evaluate members of its own team. This helps in assigning individual project grades to students.

Start early. Design carefully. Implement and test patiently. Complete successfully.