CSCI/ECEN 5673: Distributed Systems
Spring 2026

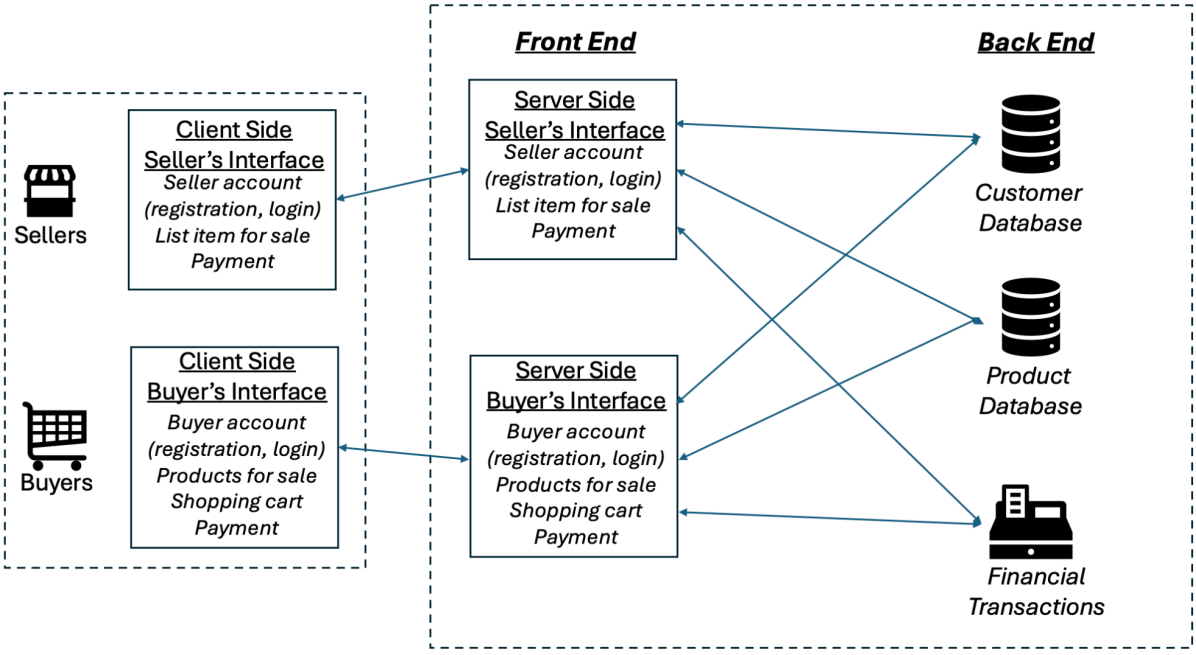
Programming Assignment One

Due 11:59PM, Friday, January 30, 2026

Goal: The goal of this programming assignment is to review client-server programming using TCP/IP (using socket interfaces) and to develop a system that you will enhance over the next 2-3 programming assignments.

You may work in teams of size two students.

An online marketplace is an e-commerce site that brings sellers and buyers together in one place. It allows sellers to put items for sale and interested buyers to purchase those items. We outlined the design of this system in class, comprised of ***seven*** logical components (See L1_Intro slides on Canvas): Client-side seller's interface, Client-side buyer's interface, Server-side seller's interface, Server-side buyer's interface, Customer database, Product database and Financial transactions. Over the first few programming assignments, you will implement and enhance this online marketplace.



Attributes

The following describes attributes of objects that will be handled by the online marketplace.

Attributes of an item put up for sale

- Item name: a char string of up to 32 characters, assigned by the seller (item names may not be unique).
- Item category: an integer, assigned by the seller.
- Keywords: up to five keywords, assigned by the seller. Each keyword is a string of up to 8 characters.
- Condition: New or Used, assigned by the seller.
- Sale price: A floating point number, assigned/updated by the seller.
- Item quantity: Integer number of units of this item available for sale. Initially assigned by the seller, and maintained by the server as items are sold or the seller updates units.
- Item ID: <item category, integer>: unique ID associated with the item, *assigned by the server.*
- Item feedback: <integer number of thumbs up, integer number of thumbs down> associated with the item, *assigned and maintained by the server*. New items should start with <0, 0> feedback.

Seller attributes

- Seller name: a char string of up to 32 characters, provided by the seller during account creation (Seller names may not be unique).
- Seller ID: an integer, a unique ID provided by the server during account creation.
- Seller feedback: <integer number of thumbs up, integer number of thumbs down> associated with the seller, maintained by the server. New sellers should start with <0, 0> feedback.
- Number of items sold: an integer maintained by the server. New sellers should start with 0 items sold.

Buyer attributes

- Buyer name: a char string of up to 32 characters, provided by the buyer during account creation (Buyer names may not be unique).
- Buyer ID: an integer, a unique id provided by the server during account creation.
- Number of items purchased: an integer maintained by the server. New buyers should start with 0 items purchased.

APIs

The following describes the API for the Seller's and Buyer's interfaces.

Seller's interface

- CreateAccount: *Sets up username and password for a new seller. The server should return the registered seller ID associated with this seller.*
- Login: *Seller provides username and password, begins an active session*
    - *Note that a seller must be logged in in order to perform the following actions. The following actions are associated with the Seller that is actively logged into the active session.*
- Logout: *Ends active seller session.*
- GetSellerRating: *Returns the feedback for the seller of this session.*
- RegisterItemForSale: *Given item attributes and the quantity of available items, register items for sale. Server should return the assigned item ID.*
- ChangeItemPrice: *Update item ID with new sale price.*
- UpdateUnitsForSale: *Given Item ID, remove a quantity of items for sale.*
- DisplayItemsForSale: *Display items currently on sale put up by the Seller of this session.*

Buyer's interface

- CreateAccount: *Sets up username and password for a new buyer. The server should return the registered buyer ID associated with this buyer.*
- Login: *Buyer provides username and password, begins an active session.*
    - *As with sellers, buyers must first be logged in in order to interact with the server. After logging in, all following actions are associated with the buyer of the active session.*
- Logout: *Ends the active buyer session.*
- SearchItemsForSale: *Given an item category and up to five keywords, return available items (and their attributes) for sale.*
- GetItem: Given an item ID, return attributes of the item.
- AddItemToCart: *Given item ID and quantity, add items to the shopping cart (if available).*
- RemoveItemFromCart: *Given item ID and quantity, remove items from shopping cart (if available).*
- SaveCart: *Save the shopping cart to persist across a buyer's different active sessions. Otherwise, the shopping cart is cleared when the buyer logs out.*
- ClearCart: Clears the buyer's shopping cart.
- DisplayCart: Shows the item IDs and quantities in the buyer's active shopping cart.
- **MakePurchase**: Perform a purchase.
    - Note: this API does not need to be implemented in this assignment.

- ProvideFeedback: Given an item ID, provide a thumbs up or thumbs down for the item.
- GetSellerRating: Given a seller ID, return the feedback for the seller.
- GetBuyerPurchases: Get a history of item IDs purchased by the buyer of the active session.

You may design the APIs of the three backend components as you see fit.

You should also handle any errors (e.g., a Buyer attempts to add an unavailable item, a Seller sells an item without logging in, etc.) in a reasonable way, such as returning a descriptive error message to the client.

Requirements of programming assignment one

For this assignment,

- Implement the following six components: Client Side Buyers interface, Client Side Sellers interface, Server Side Buyers interface, Server Side Sellers interface, Product Database and Customer Database.
    - Your implementation must allow for each of these components to run on different servers (different IP addresses/ports). In other words, each component should be able to execute as a separate process (e.g., a Buyer Client, a Seller Server, etc.) on an individual server.
- Implement all APIs except “MakePurchase”.
    - (NOTE: You will possibly extend/modify these APIs in future assignments, so make sure that your implementation can be easily extended in future)
- Design your own (reasonable) semantics for the search function in terms of “best” keyword match, etc. Clearly state your semantics.
- Use TCP for interprocess communication. You will use other methods of communication in later assignments.
    - YOU ARE REQUIRED TO USE SOCKET-BASED TCP/IP API FOR IMPLEMENTING ALL INTERPROCESS COMMUNICATION. DO NOT USE REST, RPC OR ANY OTHER MIDDLEWARE.
- Assume that each buyer or seller maintains a separate TCP connection. However, a single buyer or a seller may connect to the server simultaneously from multiple hosts.
    - Note that a TCP connection is distinct from a session. Sessions are identified using a session ID returned at login and passed with each request.
- Session Timeout**:** Logout a buyer/seller automatically if there is no activity from the user for at least five minutes.

- **Stateless frontend**: Frontend servers must not store any persistent per-user or cross-request state in memory, including login/session state, shopping carts, and item metadata. Any state that must persist across requests, reconnects, or frontend restarts must be stored in the backend databases (customer and product databases). TCP reconnects should not affect session state. In other words, the frontend must be resilient to restarts/reconnects without affecting the semantics/experience of the client.
- Registration and login: Implement a very simple mechanism, e.g. store/transport login name and password in clear text. You will address security issues in a later assignment. Allow a buyer or server to login and interact with the server from multiple client machines simultaneously.
- CLI interface: Implement a CLI interface for the Buyer and Seller clients, allowing buyers/sellers to interact with the server.
- Evaluation setup: Implement a setup that allows you to create multiple Buyer and Seller sessions concurrently and to issue API calls from each session (i.e., multiple independent connections to the server).

Evaluation

We define the following metrics that you should be able to measure:

- *Average response time*: Response time of a client-side API function is the duration of time between the time when the client invokes the function and time when a response is received from the server. To measure average response time, measure the response time for ten different runs and then take the average.
- *Average server throughput*: Throughput is the number of client operations completed at the server per second. To measure average throughput, measure the throughput for ten different runs and then take the average. Each run should consist of each client invoking 1000 API functions.

For this assignment, run one instance of each of the four server components using different processes and ports. You may either colocate processes on the same machine or distribute them across different machines. However, your design must not assume colocation: all communication must occur over TCP, and components should be deployable on separate machines without code changes. Report the average response time and average throughput for the following scenarios:

- Scenario 1: Run one seller and one buyer.
- Scenario 2: Run ten concurrent sellers and ten concurrent buyers.
- Scenario 3: Run 100 concurrent sellers and 100 concurrent buyers.

Provide a description of how you set up your experiments. Provide explanations/insights for the performance you observe for each of the three scenarios. Provide an explanation for the differences in the performances you observe among the three scenarios.

Submission

Submit a single *.zip* file that contains all source code files, deployment files, a README file and a performance report file to Gradescope. In the README file, provide a brief description of your system design along with any assumptions (8-10 lines) and the current state of your system (what works and what not). In the performance report file, report all performances measured along with your explanation.

Infrastructure

You are free to use any (distributed) infrastructure to build both your programming assignments and final projects. We provide the following platforms which you can use:

- Google Cloud Platform: We will provide $50 of Google Cloud credit to each student (via a separate announcement). You may spin up individual VMs through Google Compute Engine https://cloud.google.com/products/compute using these credits.
    - Note that the amount of credit is limited, so be sure to conserve your budget (choose small VMs, shut them down when not needed) for further assignments/your final project if you go this route.
- CSCI VDI Platform: We've set up a cluster through the department's VDI platform, providing each student with up to 5 VMs to use for the class. Please access it (and find tutorials) via https://vdi.cs.colorado.edu/.
    - The username/
- CloudLab: We have also setup a project on CloudLab for you all to use. Go to https://www.cloudlab.us/signup.php?pid=csci5673-sp26 to register for the project. You can find documentation on how to use CloudLab here: https://docs.cloudlab.us/. Note that CloudLab is a shared resource, so please be conscious of the amount of resources you use on this platform (like with GCP).

You are also free to use any programming language that you want. Note however that future assignments will require the use of Raft (https://raft.github.io/), gRPC (https://grpc.io/), and SOAP/WSDL libraries. Be sure that your choice is compatible with future libraries that you will be using.