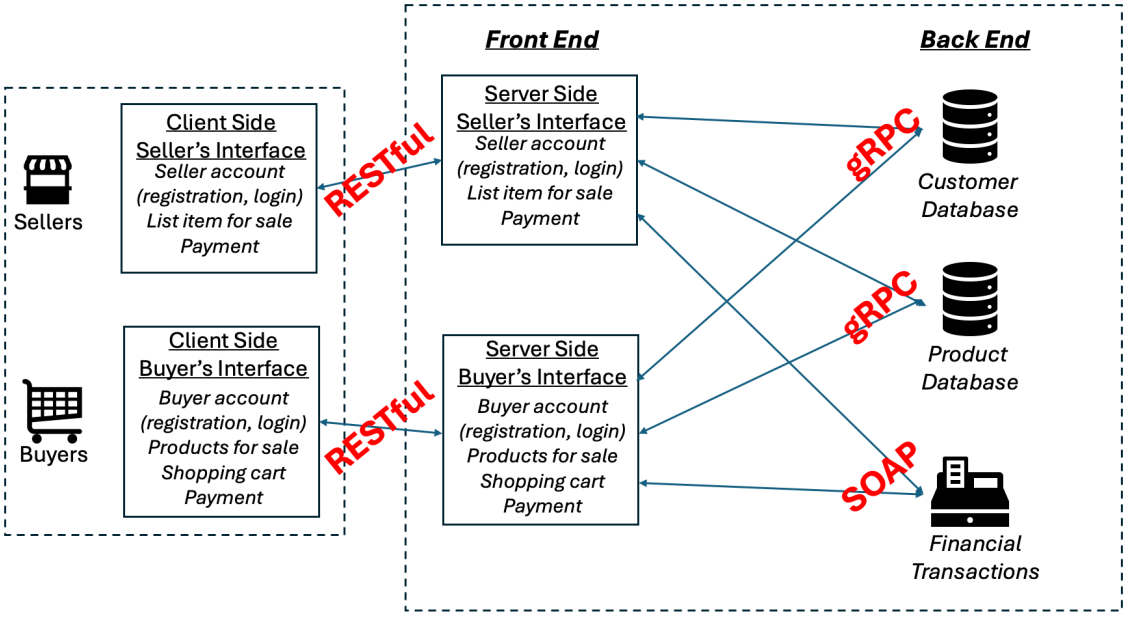
CSCI/ECEN 5673: Distributed Systems
Spring 2026

Programming Assignment Two

Due 11:59PM, Friday, February 20, 2026

Goal: The goal of this programming assignment is to extend the online marketplace system you developed in Programming Assignment One as follows:

- Move all server components (both frontend and backend) to the cloud and run them as separate instances.
- Implement the remaining API function (MakePurchase) that was not completed during PA1.
- Make the client-server interactions RESTful (Client Side Seller interface–Server Side Seller interface and Client Side Buyer interface–Server Side Buyer interface).
- Implement all communication between front-end and customer/products databases using gRPC.
- Implement a simple prototype of Financial Transactions (to be used as a third-party service) using SOAP/WSDL
- Note: Keep the front-end stateless in your design



Attributes

Attributes of an item put up for sale: Same as PA1

Seller attributes: Same as PA1

Buyer attributes: Same as PA1

API

Seller's interface: Same as PA1

Buyer's interface: Same as PA1 - however you must complete the MakePurchase API

- **MakePurchase**: Perform a purchase given credit card information (Name, Card Number, Expiration Date, Security Code) provided by the buyer.

You should also handle any errors (e.g., invalid card) in a reasonable way, such as returning a descriptive error message to the client.

Requirements of programming assignment one

For this assignment,

- Make the client-server interactions RESTful. This includes Client Side Seller interface – Server Side Seller interface interactions and Client Side Buyer interface – Server Side Buyer interface interactions.
- Implement all communication between the front-end (Server Side Seller interface and Server Side Buyer interface) and backend (customer database and products database) using gRPC.
- Implement a very simple prototype of the Financial Transactions component. It receives a request (user name, credit card number, expiration date, security code) and returns Yes (90% probability) or No (10% probability) of a completed transaction. Use a SOAP API to implement this service.
- If you have not done so already, deploy each server component (Seller Server, Buyer Server, Customer Database, and Product Database) as a process running in a separate VM in the cloud (i.e., GCP, CloudLab, VDI platform). You may run the client and financial transaction processes on any VM / locally.
- As before, make sure that your frontend components are stateless. Keep any state that you need in the backend databases (customer and product databases).

Evaluation

The evaluation for PA2 is very similar to PA1. 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 on a unique VM. 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. Finally, provide a comparison between performance from your implementation in PA1 and PA2 and explain any differences you may observe.

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.