1. A seperate document of approximately two to three pages describing the overall program design, a description of “how it works”, and design tradeoffs considered and made. Also describe possible improvements and extensions to your program (and sketch how they might be made). You also need to describe clearly how we can run your program - if we can’t run it, we can’t verify that it works. Please submit the design document and the output in the docs directory in your repository.

The project is divided into four parts: three servers and a client.

To set up, I used a bash file to ssh into each edlab 1 2 3 for front catalog and order servers separately. In each machine it kills processes that occupy the port and finds the java jar file and runs the jar with dependency packaged by maven. The bash also rebuilds the database for test then runs test then rebuild the database for later use from client.then it calls the artifact for client to run.

I used spark in java for the three servers, they are easy and lightweight candles and multi requests well for parallelism.

The client is based on java. It scans for user input. Simply put 1, 2 or 3 as an operation or put the full name with no space. Based on each operation, it makes url with edlab machines ip and port pre determined and make HttpURLConnection. In the ends it waits for the response then prints to console.

There are three types of operation as defined from hw description search look up and buy. Search and look up are both get query call, buy is put since it acts like a put action.

The front end server takes port 34841, it takes three client api call two gets and a put. Identified by their different url //. Takes in an id or topic as an parameter then make a new connection based on the operation type and sends that to either order server or catalog.it doesn't do much computation just sort them into different address with different ip.

The order server takes in the buy request at port 34843, when a buy is called from front end, it sends a query to catalog to check if a book is available, if it is, then it sends the put request to the buy API call to purchase. If not it returns out of stock. Then return to the front end to tell the client that they bought the book.

All of the servers use a java logger from util. They log the process so you can roll back if you want to. They are all saved into respective log file on edlab machine eg. CATALOG.log in src folder. They also have a timestamp for each entry.

The catalog takes in 4 api calls. At first it records what is in the database, inventory of how many counts are there for each book.

The database I used is a CSV file. I was using a json file at first but then switched although i think json would be better. Also could use a database to improve the speed.

When a search is called, it takes the topic and finds the matching topic from the csv file. For the topic name i replaced space with a dash since it would be better to identify and include from the http call. Here we replace the topic name back to search. If the topic column matches it adds corresponsive info to a json object and then returns the object. If none is found it returns a message json object that nothing is found.

When lookup is called, it reads the csv file and find the entry with matching id and similar with search it returns all the details. The loop breaks when a matching id is found since id should be unique so it saves time.

When a query to buy is called, it first runs the look up to find the entry if it exists, since it returns all the details so we can tell if the book has any stock. If they do then return a string of true to the order server. I think the return value here could be improved instead of a string of “true” or false.

Then the order server sends the buy request as put, it takes a lock that locks the csv file aka the database. If a lock is used, sleep until a lock is obtained. I think there are better and more scalable ways to do this. Including a database like sql. Then it finds the entry while saving all of the database with the modified value of the inventory. This wastes a lot of time since i cant modify the file had to rewrite the whole thing.

There is also a makeDB file that initializes the original csv file. In bash it reset the db by copying over the inventory with the template so it can always start at 10 for each book

Clone the project in edlab and local if you want to run local client
Then go into
/lab-2-huang-shi/src
Change run.sh’s user name and location of the repo
Run ./Run.sh