Lab2 System Description

- Chuuhsiang hu, Chuanpu Luo

1. Overall Program Design

As we can see below, the System includes 4 main parts: Client, FrontEndService, CatelogService and OrderService. They are all created by run.sh script (described in detail in How To Run section). **Client** class reads HTTP commands (for example, search/distributed systems) from Command List File, sends HTTP requests to FrontEndService and writes responds into Log File **FrontEndService** receives requests from Client and sends HTTP requests to CatelogService and OrderService. **CatelogService** reads and writes book data (for example, book name, book cost and etc) from and into Book Data File. **OrderService** records all BUY operation into its Log File. One thing needs to be mentioned is that, FrontEndService, CatelogService and OrderService only provides HTTP service, calls are further handled by FrontEndServer, CatelogServer and OrderServer.

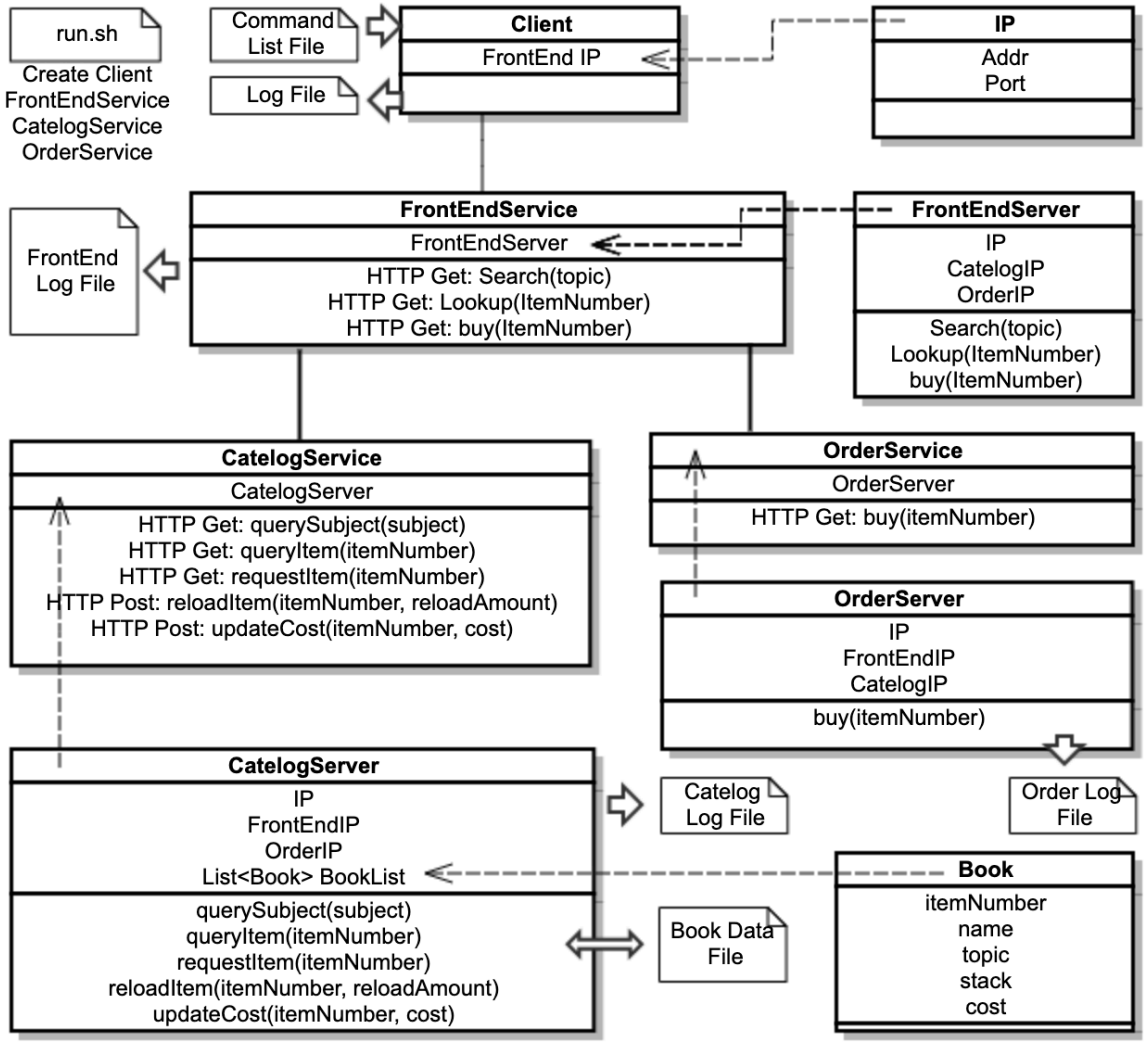

Figure 1. Program Structure of System

2. How it works
   a. Rest API Provider
      We used Java and Spark to provide HTTP service. The reason is that both of us are more familiar with Java and we are able to reuse some code from Lab1 (for example, IP.java). Spark provides a HTTP framework that is very easy to use. We define all HTTP service route in FrontEndService, CatelogService and OrderService, using both Get and Post methods. After these Service class received a HTTP request, it will call functions in FrontEndServer, CatelogServer and OrderServer to handle it.

   b. Book data storage
      We used a simple CSV file to store all book data (for example, book name, book itemNumber). In order to make sure the correctness of data under concurrency, after every data operation, we write data update into File. Next time when we need book data, we read it from File again.

3. Design Tradeoff
   When we are using a CSV file to store book data, we are thinking about if we need to write data into File after each operation. Can we keep a copy of book data in memory and write data back to File only when we terminate the program? The advantage of doing so is decreasing I/O operation and faster response. But in finally, we decide to write data back into file after every operation. Because we are afraid of program crash, if program crash and book data are in memory, we lose it. So we regard the data security over efficiency, especially when this program has tiny amount of data.

4. Possible Improvement
   In this System, only one terminal (process) is providing catalog, order service. However, if the amount of user increase dramatically, one process can be over-loaded. To solve this problem, we could create a process P which is responsible for load balance. When all processes are over-loaded, P will create a new server in the same machine or another machine. In this way, we could build a system which adjust its amount of servers based on current work load.

5. How to Run

Step 1: download project from GitHub

```
$ git clone https://github.com/ds-umass/lab-2-hung-luo.git
```

Step 2: Enter root directory, make sure file pom.xml is under current directory.

```
$ cd lab-2-hung-luo/
```

Step 3: Make sure IP address frontend/catalog/order service are correct

```
$ vim run.sh
```

```
3   # Define EDLab Multiple Computer Parameters
4   # Need to change before running for TA
5   EDLAB_FRONTEND_IP=128.119.243.164:5018
6   EDLAB_CATELOG_IP=128.119.243.175:5019
7   EDLAB ORDER IP=128.119.243.175:5039
```

Step 3: Execute run.sh to run service run.sh [local/edlab] [frontend/catalog/order/client]

```
$ bash ./run.sh edlab frontend
$ bash ./run.sh edlab catelog
$ bash ./run.sh edlab order
$ bash ./run.sh edlab client
```

Step 4: Check Output Log File

Input File	Command List File	"./tests/edlab_test_client_command_list.csv"
Output File	Book Data File	"./tests/edlab_test_book_data.csv"
	Client Log File	"./tests/edlab_test_client_log_file.csv"
	FrontEnd Log File	"./tests/edlab_test_frontend_log_file.csv"
	Catelog Log File	"./tests/edlab_test_catelog_log_file.csv"
	Order Log File	"./tests/edlab_test_order_log_file.csv"