Assignment 1 Answers:

**Answer No 1:**

a).

All-or-nothing semantics means that if a transaction fails at any point, no part of the transaction data is saved. For example, in TestRateInvalidISBN(), we test that an illegal rating and a valid rating together makes no change to state.

b).

For the rateBooks we write the following tests: that a single valid rating is processed correctly, that multiple ratings accumulate on a book, that books cannot be rated if ISBN is invalid, that books cannot be rated if a rating is invalid and that trying to rate a book not in the store causes an error.

For the getTopRatedBooks we write the following tests: that books can not be got if K is not valid and that a valid K can be processed correctly.  
For the getBooksIndemand we test that books in demand can be retrieved.

**Answer No 2:**

a).

The architecture is modular in the sense that there is a clear seperation between client code, server code and the communication layer between them. Each of these modules can be upgraded or completely replaced without the rest of the program needing changes.

b).

The architecture isolates the clients from the backend code by way of an inter- vening communication layer. Thus, clients and service can only speak to each other through RPCs.  
For all intents and purposes have seperate state and environment. This is provided by the strong modularity. For example, the service could fail and the clients would still run and preserve their own state.

c).

The same kind of isolation is not enforced. Once the JVM breaks down, both the server and the clients will crash, as can be done in the tests.

**Answer No 3:**

a).

Yes, there is naming system, it binds the names with the address or resource. By specifying the name we can get the resource, the service address and the provider information. Therefore, clients can interact with a service through names.

b).

We use IP address to allows the clients to discover and communicate with servers.

4.

At-most-once semantics is implemented in the architecture. Using at-most-once the RPCs will either return a result or some error. In our implementation, when the RPC succeeds it will return the result, otherwise it will throw an exception. The implementation of the backend logic ensures that no state changes happens when a call fails.

**Answer No 5:**

a).

Yes, it is safe to use web proxy servers. We can encrypt the communication between the proxy server and the clients and service to protect against man-in-the-middle attacks.

b).

The proxies should be deployed between the software proxies that handle message sending and the server that receives and handles the HTTP requests. Notice on the diagram that these are exactly the messages sent using HTTP. These messages are the boundary between the clients and the service.

As a conservative measure, the proxy servers should probably only be used for non- critical requests. For example, buying books from a web proxy when the main service is down would not be ideal.

**Answer No 6:**

a).

Yes there is scalability bottlenecks in this architecture with respect to the number of clients. A large number of clients could overload the single backend server. We will have to rethink our implementation of the backend logic, if the number of requests grow to a large number.

b).

Technically, there could be a performance bottleneck on the server. If the requests take long to process and they arrive at a rapid rate the service might simply run out of computing power. However, as it stands right now, the computation is exceedingly simple and we would be better off focusing on the communication links.

7.

a).

Yes, the use of proxies between clients and the server would change the ways clients might experience failures.

b).

First of all, there might be difference in the time where clients are notified about the crash, because of e.g different processing times/communication times.

If the web proxies not only did forwarding and communication handling but also caching, then errors might be effectively masked. In case of RPCs that merely retrieve data, the proxies could simply serve their cached state to the clients.

c).

In case of RPCs with consequences, such as rating books, the proxies could store requests in a queue to be processed when the service comes back online. This is of course only a good idea for some systems, specifically where the service can be quickly restarted. The clients should obviously somehow be informed about this.

Other than that, using caching in web proxies would mean that the clients lose the guarantee that the data they are retrieving is the actual service state. They might be served old data and the requests they send might be delayed.