Final exam 2021

Question 1:

(a) - Communication: Threads are easier to communicate than processes, as they share same memory while processes communicate through message passing

- Context switching: Threads are easier to switch as their state has no need to be preserved as they have shared memory while processes should store their state across different processes

- Security: processes are more secured because of the difference in address space that is handled by the OS while threads share same memory and such activity is done by the developer.

(b) - By extending the Thread class, the derived class itself is a thread object and it gains full control over the thread life cycle

⟹ MyThread thread = new MyThread(); [create thread]

thread.Start(); [start execution]

⟹ Better used when you don't need to inherit additional classes.

- By implementing Runnable interface, developers do not have any control over thread itself.

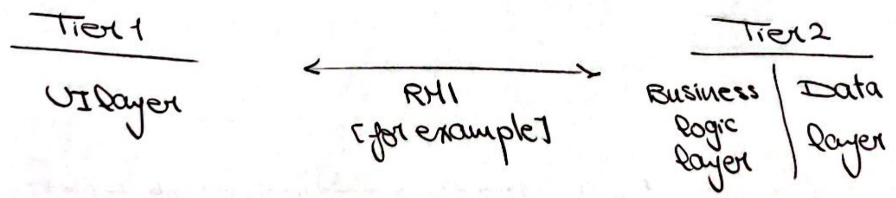
⟹ MyThread thread = new MyThread(); [create object]

Thread th1 = new Thread (thread); [creat thread object]

th1.Start [start execution]

⟹ Better used when you need to inherit additional classes.

(c)



layer? a group of related functional component [conceptual]

Tier? physical server/ device that holds one or more layers

(d) * Application servers are muchmore heavy weighted than web servers

مش عارفة * In web servers, there is no headache of changing the programming language as the functionalities are only exposed to be used. While in application server is prescriptive! Exact rules should be used while the implementation process

(e) limitations	Monolithic	micro-service
① Start time	• More time as the whole program/app is starting	* Less time as only needed service is starting
② change cycles	• Recompile all	* Re-compile only needed service
③ Reliability	• Single point of failure	* Much better, can make replicas

④ Scaling ⑤ Modular structure

Question 2

(b) * Scaling-up: to increase the application service hardware [configurations]

Example: CPU or memory

Pros: Simple

Cons: ① Single point of failure

② Not costly as scaling out.

* Scaling-out: by replicating a service and running multiple copies on multiple server nodes

Pros: No single point of failure

Cons: ① costly

② complex

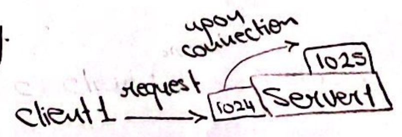
③ Database inconsistency

Question 3

(a) port is the unique number of an application that you need to access on a remote device

(b) i) As they are completely different servers (programs)?

ii) when client 1 makes a connection request to server 1 on port 1024 and if every thing goes well, the server accepts the connection and creats a new socket that is bound to a different port eg. 1025, So as any other client can request a connection. client 2 also, connects to port 1056 and server creats a new socket with port eg. 1057. Hence, it is guaranteed that each client can use the application independently.



(C) i & iii are not enterprise applications

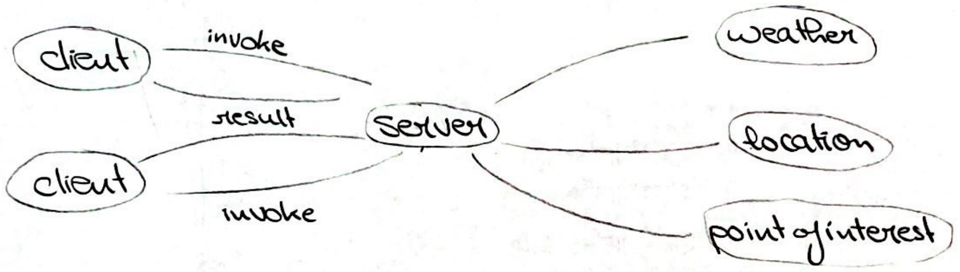
⇒ very simple programs compared to enterprise one. No need to use such heavyweigh method to execute ~~these~~ programs. These programs are targeted to smallgroups or individuals.

(ii) Enterprise application, there is the need to handle many concurrent users at the same time

(iv) Enterprise application, there is the need to handle transactional workloads and support for scalability to handle future growth.

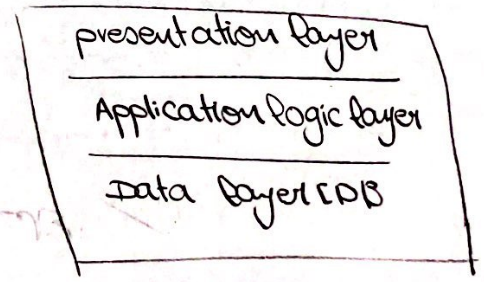
Question 4

(a) i. Client-server, anyone who needs information should request it using the available API where the server process the request and gather information from various sources and send the response back to the client



ii. No clue!

iii. Layered, as there are three layers in this system. A UI layer that displays information and an application layer that controls the system's functionality and data layer that process data.



Question 5

(a) Stateless: No state is preserved/stored for each user between calls $* \rightarrow *$

Stateful: State of users are maintained between calls $1 \rightarrow 1$

Singleton: instantiated once per application + exists for the application's lifetime

(b) i - Singleton, same game is instanciated once for any player who wants to play.

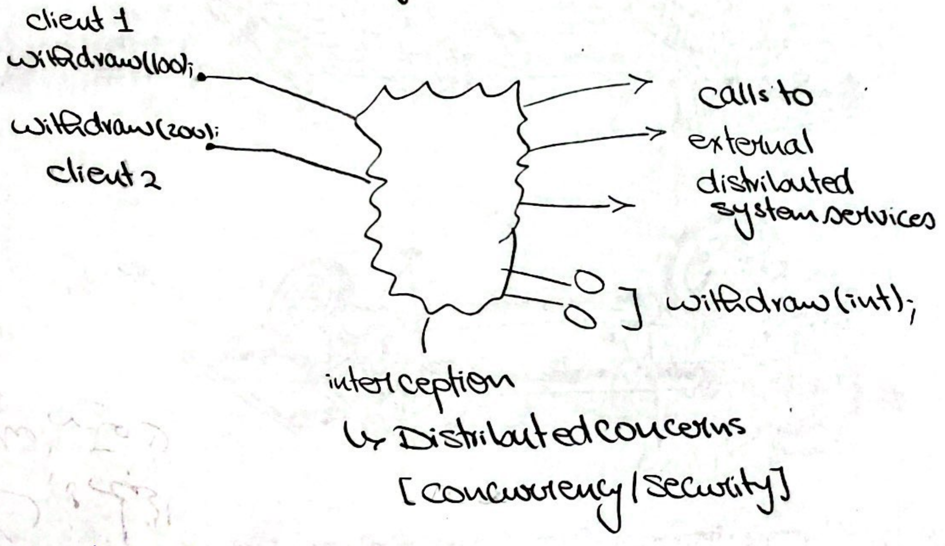
ii - Stateful, the shared state is stored for all players.

iii - Stateless, no state is preserved for interested shoppers between the different actions

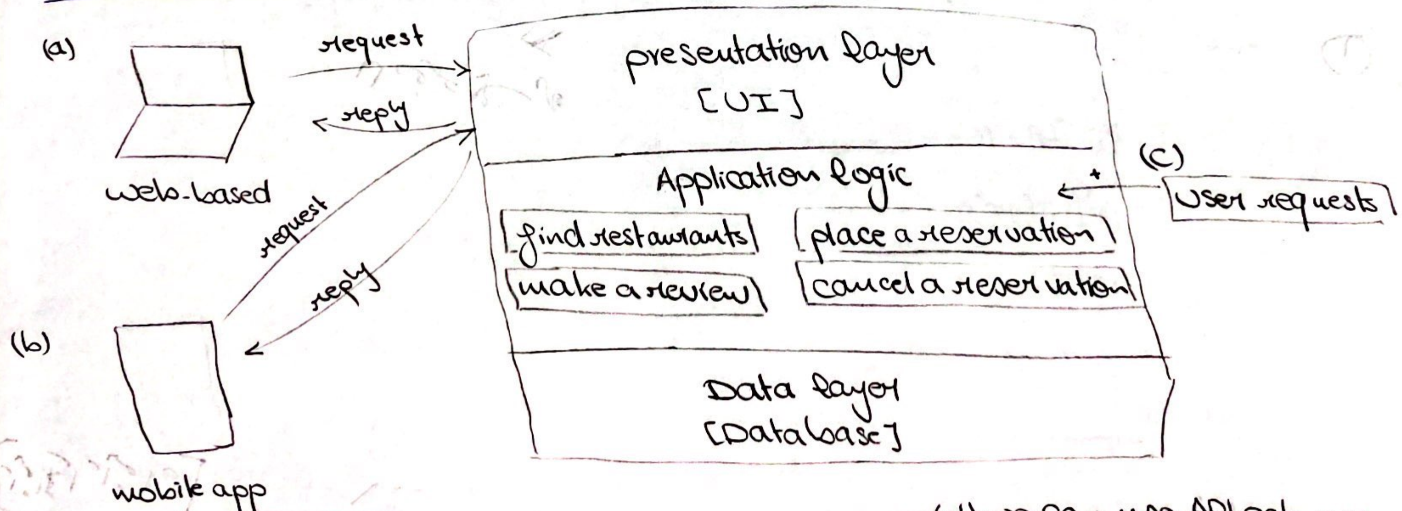
Questions

(c) Distributed concerns means the characterists that makes any system meant to be distributed as consistency during different concurrent calls. In distributed objects, for example, I should write some code to handle this inconsistency, like: Syncronized();, in addition to the application logic. While in distributed components, I only need to write the application logic with some notations and this will do the job.

4ii



Question 6



↳ the specifications of the device will be changed based on the type of device

(d) we can use API gateway?