

# Block Diagram

NOTE: ALL TEAM MEMBERS WILL GET EQUAL GRADES FOR THIS ASSIGNMENT UNLESS YOU LET US KNOW OTHERWISE.

# Goal

- To build a diagram that will help us get at-a-glance view of the **different parts of the system**, how the parts are organized, and how the parts are connected!
- This diagram will help the entire team (particularly for larger teams who are not co-located) to think and learn about the design.

# Basic idea

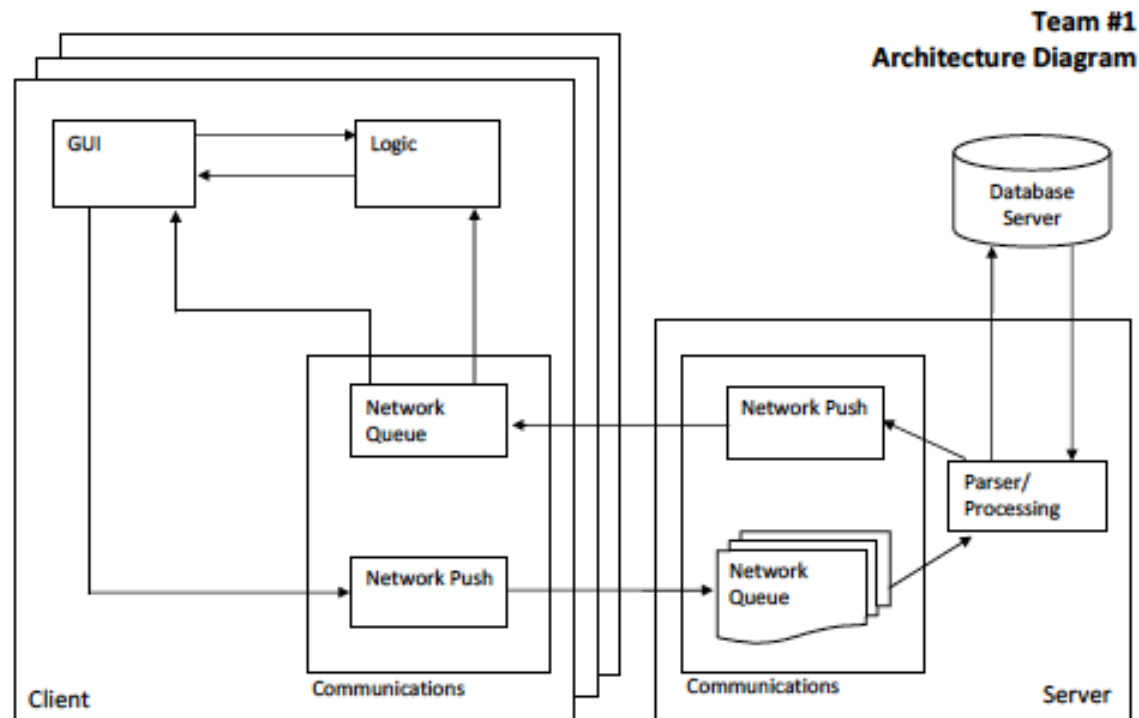
- It will consist of THREE level nested rectangles (except for DB, which will have tables/fields).
  1. Outermost rectangles will represent programs.  
Example: server, client, database, cloud-programs.  
– also, you need to show connection between programs.
  2. Second level of nesting will be packages: such as ui, communications, util, controllers, models etc.
  3. Third level of nesting will consist of individual class names.

# One page block diagram

1. put PROJECT NAME, TEAM NAME, TEAM MEMBER names at top of diagram.
2. **First** - come up with a C&C diagram showing **processes (or executables)** and their connections.
  - Show multiple instances of processes by using overlapping rectangles.
  - Use named connectors between components to indicate the type of connector (jdbc, http, tcp, udp, etc).

– See example-1 on next slide.

# Ex-1 (here focus on outermost rectangles)



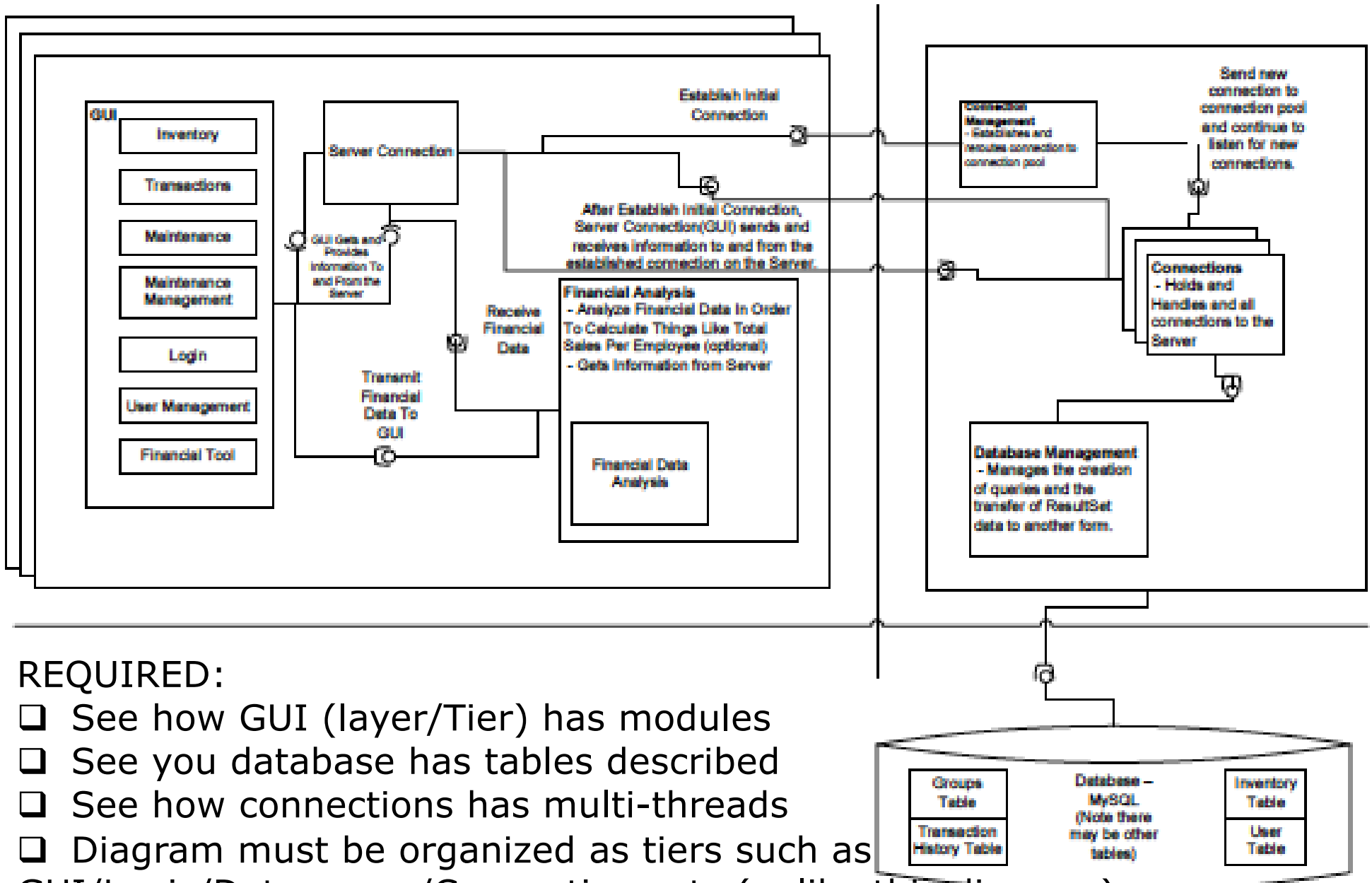
## REQUIRED:

1. See the THREE components (Client, Server, Database)
  - ❑ There needs to be such a labeled line between each component
  - ❑ The label for a line would be one of <http>/<https>/<jdbc>/<tcpip>/<soap>/etc
  - ❑ See the multiple instances of client represented by overlapping rectangles

# Modules/Tiers/Layers

- **Second** - in the C&C diagram, expand each rectangle to show the **modules** making up the process. An important criterion for a module is that you should be able to **peel the module away** from the rest of the modules and plug in a replacement module.
- Organize your modules in a layered/tiered fashion (**See Example-2 on next slide**)

# Ex-2 (here focus on GUI module)



## REQUIRED:

- ❑ See how GUI (layer/Tier) has modules
- ❑ See you database has tables described
- ❑ See how connections has multi-threads
- ❑ Diagram must be organized as tiers such as GUI/Logic/Dataaccess/Connections etc (unlike this diagram)

# REMINDER!

- Make sure to **show THREADS** that are in a process. Multiple instances of threads and multiple instances of processes are shown by using overlapping rectangles.
- The smallest entity in your drawing should be a "class"



**CHECK BLOCK  
DIAGRAM FOR  
INCOMPLETENESS**

# CHECK BLOCK DIAGRAM

- Trace through each user-goal. What modules are being called into play? What processes/threads are being called into play? What services are being needed?
  - Are all use-cases being satisfied?
  - The purpose of this tracing is to discover any missing services/subsystems. You may realize that you need additional subsystems to provide services. You may also realize that you may not have written down all the services needed from a sub- system.
- Next, trace through each non-functional requirement in your SRS. What all do you need to do to fulfill these? Again – the purpose of this exercise is to discover any missing subsystem or functionality of a subsystem. For example: you may need a “ping” ATM service to check if ATM is still up!