Office Hours 10/24

Monday, October 24, 2022 8:09 PM

Logging LL Diagram

- Do we write psuedocode for method calls?
  - Method signatures are fine, implementation is whatever
  - If the algorithm is complex then you write pseudocode
- Logger instances are on main thread which are called whenever, logger itself is on separate thread. Should we move it to a different class entirely in its own thread to have a buffer to keep spitting to database whenever it gets anything. If it fails to send the buffer will queue up until its sent.
  - Our logging process is asynchronous
  - Asynchronous in sequence diagrams must have DASHED LINES
    - Shows it's not part of the normal sequence flow
  - Threads are harder to represent, in general you shouldn't really care
    - Just identify that it's a new thread
    - Order of operation of thread is the same
    - Fine to have loop if it's a file, but we're logging to a database
- Funneling the same service into the same pipe
  - Not an issue with a small number of users but will be an issue with 10,00+ users
- Should we show how each layer connects to logging class?
  - Sequence diagram is a use-case/scenario
  - Must show every diagram where logging is required IF THEY'RE DIFFERENT
    - If they all have the same scenario then it's one diagram
    - All logging is relatively the same unless you specify a different scenario

Database has 36,000 connections. If all logging is funneled into one connection then 35,999 other connections are not used

Bottlenecking throughput of logging

Database engine represents how it deals with connection in general

Offloads a lot of responsibility onto OS

Funnel something to a scalable

Registration LL Diagram

- Need to be more specific with type of layer classes
- What view are you on?
  - Operations are not relevant without context
- Actual input validation is done on back-end
  - Validation view on UI is only for UX, does not mean anything past that
- Database connection is expensive, want to limit access as much as possible
  - Validate as much as possible without database, then do it at the end
- Failure case is invalid email, does that mean email already exists?

  Invalid means not in right format or it already exists
- Checking all the time is bad for performance but ensures you always have valid data at all times
- OTP

- We decide how its delivered

More detailed is generally better -> less decision making when it's time to develop

- Only need pseudocode for business oriented decisions or difficult algorithm
- Method signatures are usually enough for a developer to implement

All test cases should be green, if it's the expected outcome he'll bring it up

Purple Text on Core Requirements

- Requirements that are bonus points
  - ALL PURPLE TEXT MUST BE IMPLEMENTED FOR BONUS POINTS NEXT SEMESTER