1. **How would you design the platform to be scalable**

We always need to make sure we have 2 options available. If one turns down, we need to have another of anything available.

API first. In addition to pushing work to the clients, view your application as a service with an API first

1. **How would you design the platform to be fault-tolerant?**
2. While running 2 identical systems would see the same software bug being replicated, its often a case that one system will get itself into a state where it goes wrong (for ex: a thread locking issue, running out of memory), moving processing onto a secondary server would not have the same environment and would continue running correctly (or at least until the 1st was fixed/rebooted).
3. We can use design patterns to make platform fault-tolerant
4. **How would you deploy updates to the application without impacting users?**

Use load balancing to prevent errors and keep the site up during updates. For this to work you need at least two servers behind a load balancer that sends traffic to whichever server is up

1. **Each of your webservers maintains a session for each user who logs in. What are your options for managing traffic to these servers?**

We could do: Load balancing, Failover, Conditional access, routing, redirecting