**FedEx Ground Package System**David Vasquez

**What is streaming?**

Streaming is a way of continually transmitting data from a server to a client. Streaming is often seen with video and audio files. This is different than data that is totally downloaded from a server to a client before the user can interact with it. Streaming can be more efficient than downloading an entire file first. The data is sent in data packets which are small pieces of the file being sent. Utilizing a CDN can greatly improve streaming performance.

**What is asynchronized queueing?**

Asynchronous queuing is made up of a number of important concepts. The first is that a queue is a simple data structure very similar to a line of people. It follows a First in First out principle. The next is asynchronous processing versus synchronous processing. Synchronous processing is somewhat of a blocking process in that a task is sent and other execution is halted until it is finished; essentially it blocks the execution thread. Asynchronous processing means a request can be made by a client and other requests can continue to be made without waiting for a response.

Asynchronous queuing means a queue can hold requests and pass them to the server for processing based on the capability of the server. The client essentially acts like a producer and can receive a quick response that the request has been received. This type of behavior can be seen in the real world. When I was in college I worked as a server at Red Robin. On busy nights we would put our orders in and they would go to the kitchen where the cooks pulled them out and put them in a pending orders queue. If five cooks were working they could pull the next in line ticket and work on it so that multiple orders could be getting worked on in the order they came in. I feel like this is similar to the client server relationship of asynchronous queuing.

**When would you use streaming vs asynchronized queueing?**

I would most likely use streaming when a client is requesting large data files (like video) that could be sent in parts and that the end user may not need once it has been consumed. There are a lot of uses forasynchronized queueing, but I would most likely implement it in situations where a high volume of requests are coming in that need to be processed in a specific order. An example of where this could be utilized would be in a microservice architecture. You would want your services to be loosely coupled and able to communicate without processes being blocked. This type of architecture can help scale services as they are placed under a higher load.

**Describe TDD?**

Test Driven Development is a process that generally means writing a unit test first that will fail. The next step is writing enough code to allow this unit test to pass. For instance, in the simplest sum function:

* You could have a unit test that asserts that sum(2 + 2) returns 4
* Your empty sum function actually returns nothing so it will fail
* From here you can add your simple method body return num1 + num2 to create a passing method

I personally, enjoy Test Driven Development because writing good test coverage can often become as extensive and complex as the problem you are working to solve. Writing good test code into an existing complex application can be quite challenging and TDD can help with writing better code with intelligent test coverage. I have found it can help me better understand problems and develop solutions more incrementally which is a technique I have found to be most effective when writing code.

**Have you done pair programming? if answer is yes, how long did you stay on a story while pairing?**

I have done pair programming at Amazon and Oregon State University developing a cloud-based application as part of an industry partnership with Fujitsu. At Amazon we were working to develop a new user application with a number of rollouts across multiple business quarters. I spent about six months and generally paired with two other developers and spent smaller time with tech leads. At OSU it was more informal as a number of us developers shared office space and would work together for about a year. I am a social person and find I really enjoy pair programming. I think that besides helping to write better code it can also help strengthen a team and build stronger relationships.

**Walk me through Stateless and Stateful**

    Stateless and stateful are two important concepts in software design and particularly in cloud and web based applications. Stateless can be thought of as something that does not remember its state whereas stateful retains this information. You can think of stateless as a person with Amnesia like they show in old movies where the person forgets everything all the time. You could tell this person their name but if you go back they will instantly forget any information you gave them. Humans are normally more stateful in that we remember important details. Of course, we are not computers and forget things all the time so this is not a perfect analogy.

    With more applications moving to mobile and IOT it is important to have a uniform interface for them to communicate with. A RESTful API is an example of a stateless technology. I personally find that managing state is something that makes scalable applications incredibly complex. One example of stateful data is how session data for a client server relationship is related to a single user. Oftentimes this stateful data must be processed by the server holding this session id in memory so scaling servers can become more complex. Besides server memory databases are utilized to retain persistent information. Building secure, available and scalable database infrastructure can also be much more complex than an application that simply processes data.