**AGILITY ROBOTICS**

**Besides working at Amazon**

Besides working at Amazon I was able to work as part of an industry partnership with Fujitsu with a start up they had acquired. It was a new technology for more efficient heating and cooling and we helped design the cloud platform for their systems. This seems very similar to the type of work needed with agility robotics.

At Amazon I worked as a software development engineer and was able to take ownership of our campaign on boarding process. These campaigns for security and availability are utilized by everyone at Amazon and our partners like ring and twitch our on boarding team was in charge of new campaigns and we worked in an echo system of about 100 other engineers supporting our platform. My work included the technical aspects of on boarding and also a high level of interaction with different managers and leaders at Amazon or in charge of these campaigns and different departments.I feel my professional and academic background would make me a strong contributor I also think I have a love of entrepreneurship and innovation and would love to be a part of a growing company that has the potential to make such an impact on the world. I would love to be able to speak more about my passion for what you were doing and how my experience could be used.

**FED EX**

**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 then data that is totally downloaded from a server to a client before the user can interact with it. Streaming is 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 real life 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 the can receive a 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 and they would go to the kitchen where the cooks pulled them out and put them in a 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 was 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. I also would utilize it in a microservice architecture where you would want services to be able to communicate but not become to tightly coupled.

**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 summing 2 + 2 returns 4.
* Your empty sum function returns nothing so it will fail.
* From here you can add your simple method body

I personally, enjoy Test Driven Development because writing good test coverage can often become as extensive and complex then 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 including working as a server at Red Robin during my undergraduate degree so I find that I really enjoy pair programming and how it can 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 cloud and web based applications. Stateless can be though of as something that does not remember its state whereas stateful retains this information. Growing up quicksand and amnesia where pretty popular to put in movies (though they may be a little different in real life). You can think of stateles as a person with Amnesia. You can tell them their name but if you go back they will instantly forget any information you gave them. Once they have been cured they become stateful and remember their name.

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. One of the most popular ways to retaining state is to utilize a database. Managing data in a stateful application can be very challenging especially for systems as they scale.