**Solutions/Software Analyst/Scientist/Engineer Questionnaire**

Please limit your answers to one paragraph and do not exceed 15 sentences. The purpose is to quickly gauge your interest, goals, and experience level. Allow us to build a list of interesting questions for an in-person interview. Gauge your commitment into the depths of genetics. If you applied to every listing you could, this job is not for you; and we will not think anything of a non-response. If you’re interested in committing yourself to the bisection of genetics and computer science, then we’re hoping to hear from you.

* Why Fulgent Genetics, what makes you want to work for us?

When I interned at MCH Hospital, my eyes were opened at the state of mobile health solutions for the underserved (we were serving an aborigine tribe on a mobile medical trip). I talked to Dr. Wong during the trip, my supervisor, about how one could be effective on what seemed like an impossible task. He told me that when it came to lightweight cheap medical solutions, pre-emptive care was supreme because if one allowed a patient to succumb to a large problem, mobile solutions with our current technology would be useless and such a patient would require actual physical care at a hospital, which was not feasible for these tribes which were very far from the hospital. The irony of the situation was that these under-paid doctors who were using very primitive, old-fashioned tools were doing far more than the high paid doctors with state-of-the-art equipment in the city, at least when it came to the benefit of those who couldn't constantly access the hospital. Pre-emptive care became the name of the game; it became the theme of my solution for making the medical care system much better.

It was because of this that I learned about the importance of making pre-emptive care better if one wishes to help the underserved, including those too poor to take constant service from medical centers, which is a very pressing issue in our current time. I believe pre-emptive care deserves just as, if not more respect compared to conventional, ad-hoc medical practices to better serve the public. But we all know the foundation of all biological organisms – it's their DNA. A greater understanding of how to know the properties of an organism's DNA is paramount, and if the efficiency, precision, and cost of this process can be optimized, pre-emptive care will also be optimized, because one can see the foundation of potential illnesses, and staunch them at their source before they come to fruition.

* Genetics is a slow burn industry. Do you see yourself working within the genetics industry, not necessarily Fulgent, for more than a few years? Such that a software career within genetics is interesting to you.

I do see myself working in software applied to genetics. It may seem slow right now, but I believe that extrapolating it's growth based on current trends does not tell the truth of what it may be in the future. The issue I see with looking at it by this lens is that while genetics in itself does not seem interesting to many investors right away, the effects of improving genetic screening give repercussions to other high value industries. I'm willing to take the risk that this industry will continue to be slow-burn for quite a while. But I'm also willing to bet that the industry will expand rapidly in the future, in light of the recent issues of public health and the perceived poor quality and cost of healthcare by the public. These are all things that can be remedied by a greater understanding of DNA, which comes with better genetic screening methods and thus will improve the quality of healthcare and thus the quality of life.

Lastly, I feel that while genetics is a slow-burn industry right now, it's also a stable one (which is probably more important). There have been many large industries that have been killed off by one invention (like how scribes were obsolete due to printing presses). That being said, genetics is something that I feel cannot be eliminated because it is literally the foundation of all organisms, and I see it's importance lasting forever so long as we are alive.

* If you could work in any industry except genetics, which industry would it be? Why and what interests you about that specific industry?

I would be down to work in the computer vision industry. I think it's really cool about how computer vision offers an intersection of machine-learning, graph theory, and overall creativity with human-computer interaction to make an optimal algorithm for a device to figure out details about the world around it. I have a large passion for machine-learning and like to make new ways to extrapolate probabilistic data from incomplete data, a very common trend among computer vision problems.

* Have you worked in the medical field, genetics, and/or in healthcare? If yes, tell us about what you enjoyed, disliked, and what you would fix.

I have interned at MCH hospital (back when I was pre-med) for about a total of 6 summers. I enjoyed the large amount of human interaction, learning about the culture of the people I served (since they were aborigine their cultural traits were very different from that of those in the city), and aiding people in a way that lets them know (there were many elderly people in the hospital) that they were cared for. I would probably say the only thing I found difficulty with was the language barrier – the aborigine people didn't speak fluent Mandarin (this was in Hualien, Taiwan), so communicating was difficult when the interpreter couldn't help us. I would say that the high degree of interaction and ability to directly and physically help someone made me happy.

I also shadowed at Dr. Luke Huang's pediatrics for one summer the summer before college. I enjoyed the hands-on experience in his clinical lab tests where I got to see the slide samples where he diagnosed a patient based on saliva samples. I also enjoyed the human-interaction there with the children. I found nothing I disliked there, and feel that Dr. Huang exemplified to me at the time an example of a compassionate doctor who cared more about the well-being of his patients rather than just seeing as many patients as he could to get more income.

As discussed before, I think the biggest thing about healthcare that needs to be fixed is ability to serve the poor, as the current state of healthcare makes it hard for people with lower income to get quality healthcare for a low relative price. I believe a large part of this is a lack of investment in pre-emptive health strategies (which also would force the general public to learn more about their own health/risk factors and live healthier lives, something that is very feasible if they understand their own genetics).

* Have you worked with transactional data before? If yes, tell us about what you enjoyed. Describe the workflow and the results obtained.

Yes, in my experience working at Pivotal Ventures, Inc., I had worked with the Plaid API which allowed one to securely receive transactional data from entities including businesses and individuals. One thing I enjoyed was my exposure to how strong TLS protocols and hashed headers go a long way in terms of making something as apparently taboo to share as transaction history to be securely shared via advanced methods of encryption. It was actually this that got me into cryptography and security courses at UCSB.

The idea was that when someone signed into our app, the Plaid API for iOS would be connected to. The API actually automatically setup a login feature for the individual to sign into his or her respective bank, which would give us an access\_token which could be sent to url end-points (for a period of time, along with a client\_id) to get relevant data that would be inputted to the machine-learning model. The model would predict based on inputs and a triple-exponential smoothing model how much the student would pay in the future based on current expenses, and whether he or she was on track to pay off student loans.

* Have you previously worked in enterprise B2B software? If yes, tell us what you found most unique, what you enjoyed and disliked.

I have not had experience in this field yet.

* Do you know the difference between a disciplinary approach, interdisciplinary approach, and multi-interdisciplinary approach?

It is to my knowledge that a disciplinary approach takes a perspective on an issue via one type of perspective. An interdisciplinary approach is when a team uses members each specializing in one particular discipline to solve a problem via different perspectives. In such an implementation each member specializes in one part of the work stack, not making a significant effort to go for other parts outside his or her domain. A multi-interdisciplinary approach is where a team solves a problem with different perspectives, but each member is expected to work outside a single stack. In such an implementation people can definitely specialize in a particular part of the stack but are also expected to venture to other parts of it, and can work at least softly in other areas besides his or her specialization.

* Building software is like building tools, enabling people to do much more than with their bare hands. What are your proudest tools and by what factor did you amplify/enable in others?

I would say my proudest tool would be what I made during my time at Pivotal Ventures, Inc. - a machine-learning model to determine whether or not students were on track to pay off their loans. I feel it really amplified their ability to do so because it took into account fluctuations/noise in transaction data as well as seasonal trends to deliver a more computationally accurate value for their financial forecast. Seasonality and noise are not things that are intuitive to factor when calculating your finances by hand, so I feel that it definitely made a large difference in financial responsibility for them, which I feel is essential.

* What is your experience with SQL and/or logical data structure relationships? Describe the most complicated SQL or data structure you’ve worked with.

I've worked with SQL in my time at Pivotal Ventures because I helped configure and test an Amazon RDS SQL database (the logical data structure was relational). Database storage was done with a SQL table to store relevant user data.

* What other programming languages and tools are you fluent with that are not listed in your resume?

My resume is comprehensive on the languages I know.

* Are you a top-down designer, or a bottom-up designer? Briefly explain your approach to design?

I am a bottom-up engineer because I like to know the fundamentals of what I'm working with so I can spot mistakes or holes in logic in a much easier way in the design process. My belief in software engineering is that I need to understand all code in my project, unless a particular code is something I will never need to directly interact with and that I understand it's input and output.

My approach to design is one that heavily is based around intelligent testing, as there is no point to design if it can't be verified. I'm a huge believer that testing should be frequent across and within development stages, be concise, have reusable and modifiable test cases (to prevent unnecessary entire rewriting of test code), and have testing split into micro and macro aspects.

I believe that testing should be mainly comprised of highly frequent, short-term microtesting (to make sure basic services and newly implemented functionalities work). I feel microtesting is very valuable because a large understanding of the whole design flow is not necessary to start microtesting, which is a luxury one may not have if engaged in something he or she is doing an implementation never tried before.

Microtesting need not interfere with the development speed, but it should prevent an accumulation of problems that may show up in the coming of major project milestones. I believe such milestones should be checked with macro testing (to make sure a deliverable at the end of a development stage is usable and that new functionalities do not break old ones).

* How would you handle aggressive deadlines and relaxed deadlines? Explain your thought process.

As someone who is a leader of a startup, I more than sympathize with the difficulty of assigning the “sweet spot” amount of work for people who work with me, and because of that I'm a believer in being flexible when a boss assigns me slightly too much or slightly too little work. That being said, I understand that there are clear times when work is excessive or at a very significant deficit. In a very aggressive deadline I would likely communicate with my direct supervisor politely why I believe the deadline is too aggressive and ask for an extension or perhaps a revision of what needs to be done by the deadline. In the time of a deadline being significantly too relaxed, I'm likely to tell my supervisor that my work is finished and in my spare time help my teammates with their work, as well as dealing with house-cleaning involving my workflow to ensure that my next assignment can begin promptly and efficiently.

* OOP or Functional Programming? What's your personal favorite and why?

Generally I find the fit of one over the other quite situational - when times call for a language where the data's state is clear and does not need to be changed, I find Functional Programming gets the job done without the overhead that OOP normally would give via the structure. Object oriented programming I find very useful when I'm going to instantiate a lot of similar object-like entities as well as keeping a clear structure for transfer to another interacting entity with code objects. My personal favorite is hands-down OOP because I feel that the overhead I take in instantiating a clear object structure is more than rewarded by the overall organization and readability of what I send to the next user of my code object. In such a case where I do send my code object to another place, the person who interacts with that code needs to be able to read code very clearly, which I feel OOP is generally superior to Functional Programing in.

* Within computer science, what are your favorite topics, courses, or concepts? Explain why they are your favorite.

My favorite topics include security and networking. I really enjoy security because I have a huge belief on the right to keep information confidential for each citizen. While privacy is clearly not going to be possible in this current time because data is being collected about practically every individual, we can at least do our best to keep it secure if it's going to be taken. I also enjoy the mathematical aspect of it – my first time learning about Diffie Hellman at UCSB got me very interested in manipulations with modulo functions.

I really enjoy networking because taking networking courses has given me a stronger understanding of backends, which was a previous weakness for me and now I would argue would be my stronger suit among my full-stack skills. In some ways networking came out of a result of my passion for security, because discussion of endpoint to endpoint protection brought my interest to knowing more about how information was transmitted via networks.

* What’s your strongest example of applied mathematics, what did compute and why did it need a mathematical approach?

My strongest example of applied mathematics to industry was my time at Pivotal Ventures where I made an algorithm for Triple Exponential Smoothing. It computed a prediction of future averages of spendings based on periods of time. There was a large understanding of markov-chains and their manipulations needed as well as why double exponential smoothing failed to take into account seasonal trends in data. No algorithm at the time had been applied to finance which meant that a understanding of the mechanics of Triple Exponential Smoothing had to be understood so that one could learn how to process multiple data points in a season (usually done with one point in most implementations, which is not possible for finance) so that one could calculate the best forecast.

● What are your opinions of Silicon Beach, Silicon Valley, and startups? What are your opinions of slow and steady growth organizations?

I think Silicon Beach and Valley each have their huge contributions to the software entrepreneurship world, but I also think that in some ways those areas have polarized the perception of being successful in the software industry. To many the idea of being in the city and working in the area with the best companies is very alluding, but in my opinion I feel greatness can exist anywhere if there are passionate people willing to put in the work and who have a dream. And on that note, this is where I believe startups come to play. I believe that startups are the reason why the software entrepreneurship world keeps innovating, because it's the small company with limited budget that often due to the lack of resources will have to have more creativity to make a good product, which in itself is often innovation.

The value of a slow and steady growth organization in my eyes is contingent on more of the stability of the industry – I have no problem with slow growth so long as the company aims to improve that over time (growth is very subjective and sometimes very unpredictable) and the industry is stable. I would rather judge an organization by my overall pleasure of their mission as well as the stability of the industry.

● What are your expectations, as an individual, from Fulgent, as a company?

I mainly expect Fulgent to give me an experience that will challenge me to work outside my comfort zone whenever possible and to give me an intellectual feast via the sight of the intersection of the experimental sciences and engineering. I also expect Fulgent to have a workplace that is conducive to collaboration rather than competition and one that encourages me to learn skills from my partners that I myself need improvement in.

● What's your strongest example of programming ability you did as a personal project that is non-work and non-academic related?

One thing I really enjoy outside work and school is writing servers that speak back when messaged with particular strings (in a way it kind of reminded me of playing secret messages as a kid). I really enjoy using Postman to help me verify a server can talk to a client. I often wrote client code in python and the server code in Java. I feel my experience in this gave me a good foundation in my internships and software engineering positions because it gave me a relatively firm foundation in back-ends.

● What's your strongest example of communication ability? What did you communicate?

This has to be when at my time at Novacoast I told my superior that the CSP (Content Security Policy) was not going to be implemented in 3 weeks. I was truthfully afraid because I felt that I would come across as one to make an excuse to slack off. But truly during that time it became apparent to me that as an employee with value to the company, I am called to deliver quality work with also longevity (don't wish to burn out). I saw no point of giving a rushed product that would crash quickly upon launch. I respectfully told them that the CSP broke the whole site code, that having a complete deliverable couldn't be done by the deadline, and asked for a 3 week extension (since it literally took an overhaul of the site code).

And sure enough, my boss and his associates were actually pleased that I caught the hidden code errors that were only discernible if you actually opened the website and opened the links – they gave me an extra 1.5 months to overhaul large parts of the code, and the CSP ended up being complete (which is a very rare occurrence since it is difficult to get a comprehensive security policy, statistically at least 98% of websites lack one).

● What’s your strongest example of requirements management? How did you keep track and manage your requirements?

My strongest example would have to be my content security policy implemented on Novacoast.com, mainly because this was an example of a time where breaking requirements was not obvious to everyone I worked with, including my supervisor, but became apparent after much testing was done. The requirements for the site was literally everything had to work with the CSP without using a single in-line script that wasn't hashed or ponced. The problem was that a lot of dynamic scripts were calling their own scripts, so even if you hashed the dynamic scripts, scripts they called would be pre-injected at the server side and would appear before the CSP in the document, which would expose cross-scripting attack vulnerability. It took a lot of tinkering to figure out this (it wasn't very apparent in the beginning – it seemed illogical that the CSP was put in the beginning of the code on my machine but wasn't so in the actual loaded browser document code). I kept track and managed the requirement via frequent micro-testing across development phases and having a checkbox of specifications that were met on my agenda. I also feel it is important to understand the customer's experience, which is why besides regular testing of the code I made sure to actually go through the entire site like a regular site viewer to catch problems not apparent in development.