[03:52] RESEARCHER:

Hi, how are you?

[03:53] PARTICIPANT 5:

Very good. Thank you so much.

[04:00] RESEARCHER:

Just give me a second.

[04:03] PARTICIPANT 5:

Sure.

[04:20] RESEARCHER:

PARTICIPANT 5, okay. Okay, we'll start with the interview PARTICIPANT 5. Do you have any questions for me before we start?

[04:42]

No, not at this time.

[04:48] RESEARCHER:

Okay, fantastic. Can we start with some introduction? Can you introduce yourself and basically tell us a little bit about your experience?

[05:04]

Sure. My name is [Deleted to preserve the participant anonymity] in Chicago, Illinois, USA. I have experience in technology in software development for more than five years of software development and I've been in a managerial position about four years. I've worked in the capacity of software development and product management and a data scientist with various companies, governmental and private equity.

[05:41] RESEARCHER:

Before we dive onto the core aspect of the interview, let’s define quality. How do you define software quality in the context of agile software development?

[05:51] PARTICIPANT 5:

It can get philosophical when trying to define quality. Software quality for me it the degree a software, component, or process conforms to specific requirements or expectations. It is somehow subjective. In many cases, we know quality software when we see it. But the main attribute is a software free of errors. It also enables users to perform tasks quickly and effortlessly.

[05:41] RESEARCHER:

Fantastic. Let's start with an Agile question. What do you think of Agile?

[05:51] PARTICIPANT 5:

Agile is a framework, it's a methodology that helps software development teams have a higher quality in the smallest amount of time as well, addressing more features or issues in the smallest amount of time. There's different types of Agile. There's Scrum or Agile itself or Kanban but what all of them have in common, they try to come up with a framework and a checklist to increase the quality while also increasing the speed of addressing issues, building some features and pushing forward a project.

[06:53] RESEARCHER:

Just to confirm, you using Scrum, right?

[07:01] PARTICIPANT 5:

Yes, my experience is mainly Scrum.

[07:03] RESEARCHER:

You mention that it creates high quality software. How? what makes you say that?

[07:05] PARTICIPANT 5:

When you are working in a Scrum or agile environment, your developer has skin in the game. They need to reduce the amount of bugs that they might have, the amount of miscellaneous errors or higher structural and architectural errors. These are what I believe try to increase quality in the Agile environment. For that reason, but it’s not definitely guaranteed higher quality. The culture and experience and the people that work in that environment, work in that framework and needs to increase the quality and the speed. In the ideal world, Scrum would help to increase quality.

[07:17] RESEARCHER:

I think that’s interesting! What about agile or Scrum that make developers have vested interest on the success of the product or as you put it “skin in the game”?

[07:21] PARTICIPANT 5:

One of the qualities of Scrum is that it enhances the collaborative aspect of the software development. The team collaborate more and knowledge flow better among the team’s members. This knowledge can about users’ requirements or simply how to resolve a bug or write better code. Developers invest on quality when they feel it is safe to do so. For example, if you blame them for not meeting the deadlines and you don’t give them enough time to work on the quality to start with, then they will compromise quality to meet the schedule and avoid the blame. I’ve seen it happening; developers become disengaged and uninterested; then they produce crappy code.

[08:07] RESEARCHER:

What you say is, the maturity of the team which you referred to as the experience has a role in producing high quality, not Agile by itself?

[08:21] PARTICIPANT 5:

Yeah, Agile gives you the sense of what is the best way to do something in a faster manner but definitely the quality work that you put into your software, the people who you work, the professional culture of those have a very important role into the quality of your software and the success. So, the metrics of success of your software it's not time or low number of bugs or the best architecture it's also measured how experienced your developers are and how true you stay in the line for the Agile methodology. Maturity of the team is definitely something really important. How long you been working in that Agile environment is really important. Definitely if you just start with Agile, the team is going to have some sort of problems, just to understand it and to understanding what is going on in Agile, the timelines, the quality of work, the stories from inception of an ideal feature, all the way to release and deployment, it's going to be challenging it you want to implement Agile methodologies in a short amount of time. Definitely the experience of each roles and each individual as well as the Scrum master, and product manager is really important in implementing Agile.

[10:24] RESEARCHER:

So, you mentioned the people that contribute or influence high quality. can you elaborate further on that?

[10:39] PARTICIPANT 5:

People's contribution is that when an error happens, when a bug happens in your software, there are measures in Scrum that look at the metrics of time to detect the issue or time to lead on a fix, time to lead to change, that type of stuff. Metrics do not enforce people to communicate; they do not foster the culture behind holding everyone accountable for what they produce and what they release.

[11:33] RESEARCHER:

You mentioned accountability. Is this something that makes people accountable will produce high quality?

[11:47] PARTICIPANT 5:

The framework touches on accountability and having software developers responsible for end to end, from creation to deployment but then I would say accountability is more a cultural point than a metric in Scrum.

[12:05] RESEARCHER:

Can you be more specific? How accountability, for example, contribute to better quality?

[12:07] PARTICIPANT 5:

People write code! People test software! When they feel accountable, they take ownership of the work they do and its quality. I experienced as a team lead, when a developer is accountable, he will always take responsibility for the outcome of his work. In the business of developing software this means quality code, meeting the business needs and less defects.

[12:19] RESEARCHER:

Okay fantastic. We missed few details! We have been fast so far. Can you describe your Scrum environment? For this question you can choose any one of your experiences and describe the Scrum environment.

[12:35] PARTICIPANT 5:

Yeah so you are mainly falling into the definition of the Scrum framework which is to trying to help our team seamlessly understand what is the production backlog which is coming from features in alignment with some clients, ideas, needs, or user stories or bulletin that we create. Adding features to the software, adding features the web design and the architecture etc. Those are going to be translated to the sprint planning which we are going to say what is the priority in terms of this fix, this initiative and this feature and then communicate to the team. We have a team environment of about ten people or less, because it has some fluctuation based on what the project is or who the client. But overall, it's about that number of people working in different capacities. Software developer, data analysis, engineers, data scientists, product manager, product owner, support engineer, quality assurance and quality control engineer, and then overall the supervisor of the team or the software engineering manager.

[14:31] PARTICIPANT 5:

Then when we have our sprint planning, which is looking at the next quarter or couple of weeks, then we have the task or stories design and the tasks are delegated to different people, we try to communicate the delivery timeline and expectation, the quality expectation and then we go into the two weeks of work. In that two weeks of work, the software developer designs the software. They have the code, they try come up with the testing for that piece of code, constantly communicate with the quality control, quality assurance. You have meetings as needed to address the bottlenecks and then if there's some issue, we try to escalate the issue and have everybody on the same page, talking about the same issue and looking at the same initiative. During that two weeks, we try to have an ending point that we call either shipping, or a software demo for the next phase. If that two weeks is not enough, the sprint goes for another two weeks depending on how big or small your task order

[16:21] PARTICIPANT 5:

But then every couple of days, one or two days, smaller groups of people sit down and communicate what they have and what they need to have hand in hand with a QA, QC as well as the product manager and product owner. When these two or four weeks of sprint is over, we have a retrospective review of what we were able to address and what we were not able to address, what were the bottlenecks, what other bigger initiatives came out of these two weeks of sprint. And goes on and on. This is aligned with our timeline for clients which we release software for, which we work on their backend architecture and the cloud migration, as well as a more ordinary situation like adding a feature to the app or adding a feature to the website or an initiative that consultants handle for their clients.

[17:40] RESEARCHER:

You talked about something which is new to me which is defining quality expectations. Can you elaborate and explain that?

[17:51] PARTICIPANT 5:

Quality expectation is something that your more experienced member of team can handle, can put in place. Quality expectation is what clients need to see when you send a demo to your clients and it doesn't work and it has some malfunctioning feature, definitely the quality expectation was not addressed correctly. So, I give an MVP, a minimal viable product, to my clients and my expectation would be that every button in that app, in that software needs to have functionality. Do not put features in place if you did not have time or do not know what is behind it. This is one delivery expectation, quality expectation. These are not a one-night thing that can be developed coming out of Agile, but this is coming from an experienced developer who also happens to manage the client and knows what the clients exactly look at and know what the clients want. Agile would give you a chance templatized this framework and bring it into the software with the help of Agile methodology. Atlassian Group was a big one that help in software methodology for Agile and then you can define them and document them. Agile itself is just a framework, it’s the placeholder for that.

[20:03] RESEARCHER:

This setup you've been explaining to me, do you think this setup is a good implementation of Scrum? and why?

[20:14] PARTICIPANT 5:

We believe it’s a good setup because it's working for us and it's been working for us. And looking at the metrics, we had experienced a higher performance in our software development. It's not a clean-cut for every consultant or every environment every company or enterprise. But then big companies and enterprises have been using Agile and they've been in the game for so long and they kind of own their Agile methodology and what they mean by it that show that Agile over all should work, depending on other key factors of this. Other key factors, are like I said, the experience of your development group, the experience of your supervisor, software engineering manager, etc. You can't find a shortcut for those. Agile is just a framework in place to help you go faster with the higher quality, just in place for help. It's not guaranteed.

[21:45] RESEARCHER:

Well said. In this setup again, what do you do to assure software quality in this Scrum setup?

[22:03] PARTICIPANT 5:

We measure softer quality by multiple objectives and key results. If we are just talking about the quality of the software, there are measures there. There are KPIs already set up for that, we go through those and analyze those. If you like to know more about the KPI setup for the software quality, there is a very elaborated report by Google called Dora. D-O-R-A. Dora defines four to ten metrics, big metrics that discuss the quality of software and the speed of delivery, time to address key changes, time to fix changes, time to deployment, all that stuff. I don't have the list of those KPI in front of me right now.

[23:16] RESEARCHER:

No, it's okay.

[23:18] PARTICIPANT 5:

But then beyond that is the feedback of all the team members on that piece of software or deliverable. That would be how the developer feels about and how the developer documented that two weeks of sprint or that period of Agile that they were in. Also, we get the feedback from that quality manager, QA engineer, QA tester, we also get the feeling about what DevOps feels about it. Was it easy to deploy everywhere? Was it implemented easy within the template that you define? We also get feedbacks, technical feedbacks from people who are not putting the code together, like product manager and the product owner and the software engineer team. If a major release, major deliverable, everybody review it. And then by the end of that time, by the end of that sprint, you have about four to ten feedbacks on that software. They are all documented with templates in place, like with Atlassian, Jira, or Bamboo, or whatever off the shelf software or Agile software that you use. There is like two hundred different software that the company can use depending on their clients and their relationship, but its documented somewhere.

[25:11] PARTICIPANT 5:

It's as simple as an Excel spreadsheet on that release. You always document it and then get the technical people and non-technical people feedback on that. So, this would be two piece stuff, quality assurance as a whole, there is feedback from all the stakeholders involved, your team and the client as the metrics.

[25:43] RESEARCHER:

How about the QA? Is the QA involved? At what stage they are involved?

[25:54] PARTICIPANT 5:

Let's define the QA. QA is usually an experienced software development engineer who had been into a game of developing software and building new features, pushing along the algorithm, etc. Now he's the person at stake responsible for checking and assuring that the code works as it was defined and the software was built as expected, these all needs the definition of ready, as defined by definition of ready. And it was done by the time that you indicated. So, these individuals or groups are involved from the beginning idea. From the beginning of the build all the way, they are involved in the sprint and every individual communication, to the deployment and operationalizing the software. But let's say it's a piece of machine learning. It's data science, a small project, and clients come and say, we want this sort of software. So, you at the beginning of the game, people hire experienced and level of understanding of what it is, come and define this stuff, definition of ready to ship, definition of to deploy.

[27:40] PARTICIPANT 5:

Templates, testing, definition of quality. All of these needs to come along with the idea. The idea alone doesn't do much for the Agile methodology or the software development team. They need to know when, how and what to deliver. And then within that period, a QA person is a person who catch the errors and issues. First, try to work with the software development team to address those. And as soon as it works into the plan, it works into the sprint as soon as possible. This person has the communication skill, his skin into the game and then working from the beginning of an idea all the way to deployment. Because ultimately the quality needs to be at the certain level that you satisfy your clients or your customer. If you're building a machine learning model that doesn't deliver what you expect to deliver or it doesn't do that collaborative analysis that you want, all individuals involved have been maybe working under capacity and sub optimal.

[29:20] RESEARCHER:

I'm just pouring some tea. I do have another question for you, which is a follow up question. Can you hear me? I do have follow up questions. You talked about everybody is involved from the beginning and especially the QA are involved from the beginning. This is in contrast to other traditional methods which the QA is involved at the very end. Does this methodological behavior, for example, change the behavior of people?

[30:11] PARTICIPANT 5:

What methodology or what framework, they say that QA should be involved after the developer push and comment. Is that that what you're referring to that once the software is done?

[30:32] RESEARCHER:

Yes.

[30:35] PARTICIPANT 5:

Yeah. So, it's probably somewhere out there on the Internet that's saying the QA needs to come in place when the software says I'm done. But then what if, the software team has communicated totally different issues and different features and addressed some other issues that QA needs to catch early. So, QA is not, to my understanding, to my experience, QA is not data dog or cat task or those type of place that they put their software in. They said, cat task comes in place. And then every template that you push into GitHub, every template that you push into the cloud, they look at the task. QA is a person. You need to look at it as a member of a team working with the software development team from the beginning to the end. QA is actually there for early handholding and fixing the issue early, rather than the software development team works on it for two weeks. Push it out. And then the QA says that it doesn't work as expected or the quality is not checking the checklist. Push back and then you lost two weeks in this framework.

[32:22] PARTICIPANT 5:

You need to set up the QA dynamic with the team. Usually if you have a team of like sixteen software developers, you probably like to have four to five QA engineer team. So, something like that. That capacity to work closely with your sub teams of software developer of twos. Software developers and engineers are working on one project, you need to assign one QA to the team, so they capture something early and then that increase the success of your sprint, your timeline. It's just a saying that fix the issue as soon as you see something about it. Do something about it as soon as you soon and then communicate early. Put out the communication really, really early in their first single line of code that every software development starts to write. If there is an issue, it should be addressed the moment you see the issue. And then also it helps the team to develop the test case. Every single piece of code that you write, you want it to be tested for multiple test frameworks. That is also one thing that the QA engineer can help in.

[34:08] RESEARCHER:

So, this is empowerment, empowering everybody to be involved and transparency. Transparency in the process, transparency and in the process, transparency in dealing with each other.

[34:27] PARTICIPANT 5:

Yeah, definitely, I would put it in as a word like empowerment. It's really well said. Everything that your team, it's a machine that’s trying to handle multiple thing in the software environment, feature definition, feature development, backend, or frontend, it doesn't really matter. But then you have a team, all individuals empowered, all individuals involved and has a sense of belonging to that software, a sense of belonging to that task that they are doing. It's not cold cut teams, everyone needs to be all involved. And then the more you elaborate on this culture, I would say, the culture of transparency and empowerment, the better they get. The more they feel in the position of responsibility and power, you see the shift from one deployment per week to multiple deployments per day. You see that a team coming into place and checking all the checklist with little to no miscellaneous items and little to no errors as to speak. So, if you invest that into the people and your team, you see the return. I would say the culture is a big part of it. And I would also say that the Agile framework has something to do with it as well.

[36:20] RESEARCHER:

How? The last statement, Agile has something to do with it. Do you know how?

[36:31] PARTICIPANT 5:

Well, Agile has something to do with it because it is the framework telling you that your delivery is by the end of this month. And these people are working on it. And then you have the undivided attention of these people on this project. If it was a Waterfall situation that OK, go and have a look at that piece of algorithm and software. And then after two months you develop it, come to us, and tell us what it is and then we say no, that wasn't what we wanted you to work on or develop. Go back, work on that another two months and come back. So Agile is collapsing and kind of compressing all those communications into the shorter amount of time. It has a software piece, it has this framework, mindset, and culture, as well as it empowers what you want your software development team and product management team to be.

[37:05] RESEARCHER:

You have been talking about empowering the team. How does empowerment helps quality?

[37:11] PARTICIPANT 5:

It changes behavior which helps better quality. When developers are empowered, they make their own decisions on the quality of their work. They do not need to check with their manager to write better code or ensure an optimal design. They have more control over their work and they become committed to meeting the expectation on quality. The key here is you empower everybody to make decisions on the quality of their work. I experienced when deployed properly this should result in heightened productivity and a better quality of the software.

[37:55] RESEARCHER:

Fantastic. Well said. Now, can you share with me a positive story about Scrum and producing software quality?

[38:07] PARTICIPANT 5:

Yes, sure. We're working on the risk management and fraud detection. We were working on using a third party API to bring it into our software. Then we collect the software from the e-commerce users and then push it into the API. Their API gives us something back, and then we use that as an input into our machine learning model. It can be a location, location verification, it can be a credit card verification. So, the development team is working on it and very small errors occur that the location that they're sending into this API is in a wrong format and is in a malfunctioning format. The API return everything positive, everything thumbs up for us, like all the locations are verified and all the locations are good. We know in the real case that it's not the case because a lot of orders are fraudulent. Lot of orders are involved in fraud and money laundering as we speak.

[39:43] PARTICIPANT 5:

So that positive feedback we got early on from the QA and team who were looking at this software that was developed, pushed to the template, pushed it to GitHub and was deployed. It detected that error early on and by the time it was fixed, that spam was quick. And then we saved a big number of money in the live software that it was working on a large system. This would not have surfaced if the team of quality assurance and other peoples in the game wouldn't detect that. Because if you say that, OK, let the software development work on it and then push it to the GitHub, and you deploy it and then come back and look at the quality of the case. In that span, you already lost a significant amount of money on one single error that could have been caught earlier in the process. Because we had the QA in place and the checklist in place, the person went through the checklist for a location or whatever it was, with that check. Immediately, it escalated back to the software development. It was around the clock project, but it immediately captured what change triggered that malfunctioning - that error in the location format and what we were sending today, and time to resolve it was reduced significantly.

[41:38] PARTICIPANT 5:

So just to put it in perspective, if it were an ordinary software development and Waterfall methodology, time to resolve this issue from surfacing to resolving the issue would have been maybe six to ten days and they were losing money in that six to ten days. That was reduced to less than a day from surfacing the issue until that resolution.

[42:14] RESEARCHER:

Fantastic. That's a great story. Thank you. Now, unfortunately, it doesn't go rosy all the time. There must be a negative story. Would you like to share with us some negative story?

[42:30] PARTICIPANT 5:

I have a particular negative story, but I won't share much into it where it impacts the team overall feedback and overall performance and efficiency. Whereas even in an Agile process, in framework, teams like to sit and point fingers when an issue surfaces, when an issue come to the attention of the clients or the user or whatever feedback we get, teams like to point fingers at each other and then that clouds the whole environment that you're working in. If its significant, money is lost, or a significant client is upset with the delivery that you have then teams start to point fingers at each other. This happened in our team before. And then management is also upset about the delivery that something was going wrong in the process and something not rosy, like you put it, happened. So then teams start pointing fingers at each other.

[43:58] PARTICIPANT 5:

What is the best way to address those types of situations? Bring the culture together that something happened, let's learn from it so it doesn't hurt us again instead of pointing fingers at one another. It's not Agile fault or any other framework, even in Agile we could detect it, or the QA couldn't detect it, or the software development team couldn't resolve the issue in a timely manner. But then at that time, you have to support your team and then building that culture, that OK document the problem, communicate to all the teams, or other software development teams so it never happens again in the future.

[44:59] RESEARCHER:

That's a good example, too. Thank you very much. I come to my last question. It's a little bit provocative, but the purpose is not to upset you, but the purpose is to get you to talk and to give us your opinion. What do you think of this statement: Agile produces bad software?

[45:28] PARTICIPANT 5:

No. No framework, no software, no machine learning algorithm, produces bad software or an algorithm or a piece of code. It's teams that are working on it and the environment that you're working within. If your culture is slide it under the carpet so no one else would know about that issue or that problem, then it’s not agile fault, it is the fault of the people and the team and the culture. But if you have a team that is about communicating well, speaking well about each other, and addressing those proactively, not reactively, that's when you will be able to create the best software. There are many frameworks. From the start of technology and computer and software development, and there will be.

[46:37] PARTICIPANT 5:

Like within Agile there is a Scrum, Agile itself is Scrum, Kanban, whatever, two hundred different software off the shelf that provides something. They never replace your team and they never replace your management style or your culture within that team, within that structure that you defined. So Agile did not produce bad software. Bad software teams produce bad software. If I want to change that statement, it's not the framework or the software, it's the team. Teams are the combination of the people, technology, and the culture.

[47:36] RESEARCHER

Well said, thank you very much. I enjoyed it very much. Thank you for your input. It was very insightful. Was to the point, most the time, was very to the point. I liked it very much. Thank you. Thanks for your time. Do you have any questions for me? Because you may have questions for me. Fueler.

[48:01] PARTICIPANT 5:

Yeah, not really. No question about Agile. Would like to know more what is the big picture of your research and what you're doing and then how you're going to use this type of questionnaire and interview with different people into your research.

[48:21] RESEARCHER:

Yeah. So, let's start with the last bits because it's easy. The way how we use these interviews. I did mention that, that your participation is anonymous. I'm not sure if you mentioned the name of any company or any employer. It would be anonymous. It wouldn't be used publicly. However, your statement may be used publicly in a research paper. So, what do we do with these conversations, we analyze them using a systematic method called coding. We look for knowledge and the experience statement in this verbatim, in these conversations. And those statements are knowledge because you talk of experience. You don't talk rubbish. You don't tell us stories. Even if you told us stories, you've told us stories from your experience. So, we analyze it and we extract knowledge. We extract; there is a lot of knowledge in this conversation and that's knowledge become known. How it becomes known, we publish it.

[49:39] RESEARCHER:

We publish it in a form of an academic paper, a journal paper, or a conference paper. And it becomes known knowledge to people. This is what's happening in practice. This is what people told us. We analyze it and it becomes knowledge. What is the big picture? I'll start with the motivation. The motivation comes from looking at Agile in its abstract form. If we look at the manifesto, the manifesto is very abstract. It doesn't make clear reference to quality. When we go to the Scrum guidelines, it's abstract as well. It doesn't make a clear reference to quality. So, this is a gap. A gap in the knowledge, documented knowledge. So, we try to bridge that gap. If Agile in its abstract form does not make any mention or reference to quality, it doesn't mean that Agile teams don't make the process and the quality assurance practices embedded in the process in Agile. Hence the last question, which was a little bit provocative, Agile produces poor software because anybody that reads the manifesto would think, where is quality. So, we talk to people like yourself with experience to tell us what happens in real life and we bridge that gap. We analyze the conversations, and we make conclusions. And with those conclusions, it becomes knowledge and disseminate it in forms of publications and for practitioners, like yourself, who wants to know about Agile and read it and learn this is what most of the teams do. This is how quality is achieved in Agile. I hope that made sense.

[52:02] PARTICIPANT 5:

Definitely. That's very exciting that you try to bridge a gap between research and real world practice, its important. I can definitely testify that whatever you find in the Agile methodology websites or research are abstract and they are trying to be big picture as much as they can. At the end what they say is do what is the best for your team, for your environment, for your software. That can be a strength and a weakness of Agile. I would personally like to see more by the practitioners and industry. If you are a leading sector in technology, if you are a consultant, a user, by different sectors or by different type industry. Is it business or finance, is it marketing, is it education, or is it transportation, is it logistics, but they still have a way to go in creating that documentation etc.? A leading company like Google, Amazon, Facebook, and those big names, all develop this documentation and publicly provide how their software engineer teams work. Google has a lot of these grants to the universities and research associates like you to go ahead and develop work and QA and QC, Agile, Scrum, whatever. One that I mentioned was Dora, was looking at the elite QA/QC and KPIs for teams. Working around the clock, how to increase the quality assurance and quality control. But this is really, really exciting.

[54:19] PARTICIPANT 5:

Another question I had that I ask you, were you able to record this interview?

[54:29] RESEARCHER:

Yes. It has been recorded and we still recording.

[54:36] PARTICIPANT 5:

Would that be possible to share a link to the video as well?

[54:39] RESEARCHER:

Yes, I will do that right after this interview finishes. No problem.

[54:47] PARTICIPANT 5:

Is it ok if I publish it in the LinkedIn also acknowledging the fact that you put together this interview?

[54:58] RESEARCHER:

You are free to do. We own the data together. But if you want to publish it, I have no problem with that.

[55:12] PARTICIPANT 5:

I think it is a really helpful discussion, talk, interview and other people will enjoy it. I never had an interview so specifically about Agile.

[55:31] RESEARCHER:

You can say I was interviewed by this researcher, you type me for example. This is the conversation, this is my opinion about Agile, he interviewed me about Agile and quality, and you can do that and publish it if you like.

[55:50] PARTICIPANT 5:

So, I would really appreciate it if you can send me a link to the video.

[55:54] RESEARCHER:

I will no problem. I will do that. If you don't have any other questions, we can conclude?

[56:01] PARTICIPANT 5:

No. I would like to thank you for the time and the research you do. It's really interesting and I would like to see the end result if it's in the term of final product, publication, paper or dissertation. I would really share that.

[56:25] RESEARCHER:

Okay, fantastic. I'm just going to put a note against your name here. It's [Deleted to preserve the participant anonymity]

[56:36] PARTICIPANT 5:

Yes.

[56:36] RESEARCHER:

I do have your email, which is the Gmail one and I'm just putting a note that yes, you're interested to receive the research outcome. It will take a few months, but you will receive it, definitely.

[56:53] PARTICIPANT 5:

Beautiful. Thank you so much for your time.

[56:56] RESEARCHER:

You're welcome. Have a good day.

[56:59] PARTICIPANT 5:

Yeah, you too. Take care.