[00:05] PARTICIPANT 7:
Hello?

[00:06] RESEARCHER:
Hello, PARTICIPANT 7. How are you?

[00:09] PARTICIPANT 7:
I'm fine. How are you, Researcher?

[00:11] RESEARCHER:
I'm doing very well, thank you. [Deleted to preserve the participant anonymity]

[00:15] PARTICIPANT 7:
You too. You too. How is it there?

[00:18] RESEARCHER:
It's a long day, it's been twenty hours here.

[00:22] PARTICIPANT 7:
[Deleted to preserve the participant anonymit]

[00:35] RESEARCHER:
Yes, it's twenty hours. But we managing OK, I guess.

[00:41] PARTICIPANT 7
Yes, [Deleted to preserve the participant anonymity].

[00:42] RESEARCHER:
Thank you very much. Do you have any questions for me before we start the interview?

[00:49] PARTICIPANT 7:
Not actually. I got the idea. [Deleted to preserve the participant anonymity].. I know exactly...

[01:01] RESEARCHER:

So, you know what I'm doing, right?

[01:02] PARTICIPANT 7:

Yes, yes.

[01:04] RESEARCHER:

OK, fantastic. Let's start then. Can you please introduce yourself and talk a little bit about your experience?

[01:12] PARTICIPANT 7:

Ok. My name's PARTICIPANT 7. I graduated as a computer engineer in Turkey. After that I had one year experience as a software developer for a company and then I pursued my Master and PhD in the United States. I was at the [Deleted to preserve the participant anonymity].working with [Deleted to preserve the participant anonymity].on the requirements engineering topic. After I actually finished my PhD, I stayed there for a few years and worked as a requirements engineer. So actually, I'm pretty much familiar, a different software engineering processes. During my stay at the university, I was a teaching assistant and a research assistant. I facilitated more than a hundred of requirements negotiations. And some of those projects actually developed using Agile during that time.

[02:39] PARTICIPANT 7:

We were more focused on actually on lean development processes and Agile was pretty new, I can say. It was around 2005. But I get the taste of Agile during that time. From the industrial perspective, while I was working there as a requirement to engineer it was a health care company. So, they are more stricter on getting the requirements then developing. And they had different teams for different tasks. So, I was actually in the requirements engineering team. Mostly we interacted with stakeholders to get the requirements and write the specifications. We were also involved in testing of the system during that time. I stayed there for almost for two years then came back to my country and start working as a scholar, as an assistant professor there. During that time, I think it was around five to six years, I told lectures on software engineering. Now, instead of applying, using, or practicing what I have learned in the software engineering domain, I tried to teach them.

[04:27] PARTICIPANT 7:

I had teams of students, assigned projects to them. And actually, try them apply Agile techniques, agile development techniques like they were developing their projects. It was to write projects. We used to do actually making, let's say, finals. I tried to actually put more effort on the project so they can actually test taste how things happen in the industry instead of just learning to tell the theoretical part. I have actually seen good results, but mostly people, actually students, I can say, are more interested in doing development in waterfall. It's linear. It's simple.

They actually handle the task, and continue to the next phase and the other ones. But when they understand the philosophy behind Agile, they like it. After that, actually, for the last four years, I worked for a software company, a software technology company. They had two programs, like an embedded part in the software department part. In the embedded part actually, they were running, they were actually implementing or manufacturing some devices, these Bluetooth devices.

[06:20] PARTICIPANT 7:

I think, you know, the BLE devices, Bluetooth Low Energy devices. They were hot actually around that time. And in the software part, we were actually developing some user scenarios, services using those IoT devices. I have taken quite different roles in that company. I started as a project leader and then our team expanded. We had a CEO who actually act as a product owner. Then after I'd learned actually what he's thinking or what kinds of services he wants, I took over the product, the product ownership the role. And while our team is actually, a few people, I actually have taken the quality assurance role also within the team. So, in those cases, I actually experience many of those roles in the development cycle. I have done little in the development part, in a few years, actually, because I worked there for almost four years, our team expanded to twenty people. So, my roles, at the end, I can say become actually the product owner or the product manager for those products in the software side and as scrum master or a project manager for the embedded part. So, in the embedded part, we actually used Kanban most of the time because it suits better. In the software projects for the [inaudible] applications and mobile applications, we practiced scrum. And I think further I can expand based on your questions there what we do have done in those.

[08:42] RESEARCHER:

Okay. Fantastic. Thank you very much.

[08:45] PARTICIPANT 7:

You're welcome.

[08:46] RESEARCHER:

Let's agree on the definition of quality, because we will be talking about quality in this interview. How do you define quality in the context of agile software development?

[08:53] PARTICIPANT 7:

Quality take place in many aspects of what we do in software development. From my experience, the focus has always been product quality, which is basically delivering a software that works and without defects. But with the rise of agile, I saw a shift toward getting the process right, code quality and good design. Agile believes in responding to change that’s why we aim for a design that is flexible and aims for reusability. Agile calls for continuous improvement. As a team, we always reflect on the way we work and our process. We learn, we identify improvement and implement them when possible. It is a journey!

[09:16] RESEARCHER:

Let's move to the next question. What do you think of Agile? What is your opinion of it?

[09:20] PARTICIPANT 7:

Well, I can say that I'm in the software domain for twenty-five 25 years after I graduated from [Deleted to preserve the participant anonymity] a lot of projects and worked on the requirements part. And I can say that there is a relationship between the complexity of the projects from the requirements point of view and also the process that you are following. So, by experiencing actually the waterfall, the spiral, the iterative and evolutionary processes, actually the Agile process comes in life, from those experienced people, in order to lighten the process. Because the most important part actually for a software development project, is like it's the people and their interactions, as far as I see.

[10:07] PARTICIPANT 7:

Because it doesn't matter how good your process is, you cannot follow it if you don't have the people take on those roles and they accept it, it's very difficult. So, you have to make it simpler for those developers because at the end, broken software and a product is most important then actually what process you have followed. So, in that case, the principles that Agile provided those software developers makes it easier for the developers to just focus on producing better software and in a shorter time. That's actually another gain in that perspective, because in the early days actually, the projects were bigger, actually bigger companies were trying to have developers or development software companies, develop projects for themselves.

[11:21] PARTICIPANT 7:

But for the last ten or fifteen years, I can say, because of the Internet, now people are developing smaller applications, products, and they can actually publish it with the mobile phones instead of having cumbersome, bigger software like the financial software or healthcare, or you can say like Space or Military. Now, people are more interested in smaller software in the telecommunication and that makes it easier to adapt Agile software development processes. Because it's not easy to develop the traditional ways. You have to be faster in a few months, actually you have to deliver. You have to publish it because if you cannot, then it's going to be late. So Agile gives that flexibility to the people to develop faster.

[12:42] PARTICIPANT 7:

And another one is like it's easy to change the requirements in the Agile domain. Because you can decide on what the requirements are. If you spend too much time on detailing those requirements later, you see like the customer changed their mind. So, it's not a good practice because then you cannot actually pass the requirements specification phase and stop development. But with Agile, this is now easier. You only decide on the most important features for the viable product. And then what you can do is change your features based on the customer feedback and the development cycles are going to be shorter. So that's very fantastic, actually. If you can apply it. From the development team's point of view, I think transparency is very important because when you are applying the traditional ones, there are different teams for different phases. There's a requirements team, maybe product management team, you have

developers, there will be one or many software architects. Then there will be testing team, deployment teams. But because there are different teams, what they do is like instead of working together, they work actually separately.

[14:37] PARTICIPANT 7:

So, one team finishes the job, test the work, the other one continues, and they do the iteration way. However, in Agile, they work together. So, you have developers, project manager, product owner is part of the team, QA is part of the team, the DevOps is part of the team. So, there is a transparency. So, what happens is like they actually know what the customer wants, what is the output at the end and working in short cycles, they can actually deliver it. So that's actually another aspect, transparency side, as I see. And one more thing that I'm also interested in from the scholar point of view, is while you are developing something throughout the cycles, within the Scrum point of view, between those sprints, you actually evolve. You learn also. You learn from your mistakes and you do a better job in the next sprint. So that's actually one very important aspects of Agile development processes. Because in the software engineering parts, I think you are familiar also with the CMMI, the maturity models, they request that it's not only developing, you're actually collect, measure your performances, this way you actually improve your processes. Now with Agile actually, it becomes faster. Because in a time box, let's say sprints, you can achieve those kinds of things. So that's actually what I like. The time box of Agile development processes.

[16:51] RESEARCHER:

Ok, I have some follow up questions. You touched onto some very interesting values of Agile which is transparency, working together, which is collaboration, learning from mistakes, and continuous improvement. How do these values contribute to achieving quality in Agile or Scrum? Because your experience is more Scrum, right?

[17:15] PARTICIPANT 7:

Yes, mainly Scrum. Ok, let me start with collaboration. Now if the development team worked as one team, not like individuals, because what I have experienced actually doing my roles in the industry, is like sometimes people actually argue. They actually blame the other one because of those tasks. But with agile development process because you need to have regular meetings and within those meetings you discuss what kind of problems you have. And this actually allows the other team to contribute to help you. So, you can collaborate. And it's not just one person owning the task or the feature, the team owns it. You cannot have someone to blame. It's either you blame the whole team or there's something wrong with the features.

[18:31] PARTICIPANT 7:

So that's important. Another thing is with some practices, I apply actually some of those. I require my teams to apply some of those like the peer programming. So, this way, people learn from each other. So, when I had a senior programmer, I assigned a junior one to that programmer and they learned from each other. And this is actually part of like the continuous improvement and learning. Developers learn from each other's how to write better code. So, this improves the quality of the software by applying those techniques there and learning from each other's. Another one is let's say we apply these...actually we are using GitHub. And with the

practice of GitHub, when we assigned a task to a developer, we request him after he's done programming and write the unit test for the work he has done. He sends a Pull Request. And another team member, and most of the time it's the project leader or a more senior one, reviews the code and provide feedback; developers learn from it. It's not just the unit test that's running. He reviews the code. And if it's correct, then he approves it. And this allows that feature to be included in the software. And the person who is responsible for the quality, the quality assurance engineer or tester actually tests and provides feedback.

[20:44] PARTICIPANT 7:

So, in these cases, actually teams work together. So, it's not just one person who's doing all the job. It's different people collaborating with each other, it allows better quality. Because from one person's perspective, for example, when he reads something, he understands something else. So, this way, we actually find if there are some ambiguities in those requirements or features or the work assigned or there are some bugs there, we can capture them easily. So, this are really helpful tools. Let's say in a short basis, when you apply those, I see the benefits there. I can also maybe add these ones, when we have those user stories written, I was actually most of the time responsible for those, and when I have written acceptance criteria, the development team understands them better. The testers actually understand the requirements and can write test cases easier before the development starts.

[22:25] PARTICIPANT 7:

So that way, the developer has the requirements, user stories or the task assigned and the associated test cases. They know exactly what kind of conditions they have to change while they are writing. So, their job becomes a bit easier. And the unit tests that they have to write becomes easier to write and test. Now, actually, all these lessons learned comes at the end. We did a lot of mistakes in the beginning. What happened is like when we started, we give the requirements first. We see that by giving the requirements, it's really difficult for the developer to understand. You have to split them into pieces, tasks. But when you give those tasks sometimes what happens is, they actually just focus on their tasks, not the whole system. So, we start these regular meetings. So, we ask what they are doing if they are having any problems. We actually applied scrum meetings and they learn from their colleagues. They learn what good practices they applied. And definitely actually they improved themselves. We just hired some students, new graduate students, and within a year, they become very good at doing their job. So, it becomes a very, let's say, practice or I can say from a scholar's point of view, a very good lessons learned for me. Because I was trying to do this in the university, in my software engineering course but I didn't see that much effect, the efficiency. But in the industry, I see the benefits more.

[24:50] RESEARCHER:

Okay, fantastic. Thank you very much.

[24:53] PARTICIPANT 7:

You're welcome.

[24:54] RESEARCHER:

Can you describe to me your Scrum environment? You can choose any one of your experiences, or your projects and talk about it. How does the process work?

[25:08] PARTICIPANT 7:

So, what happens is like first we start with an idea from our CEO or sponsors, we receive an idea. So, the product owner, what I did is I just elaborate having some meetings with the CEO and doing some research on the idea. I get some details before we move on to the user stories, I discuss that with the CEO, with the sponsors, the stakeholders, and try to actually scope it down to the most important features. After we have those features, I have written the user stories. So, these users stories becomes the product backlog for our project. So, what happens is like we create a project, by the way, we use Jira for our management, so it is agreed to in order to actually follow up both the project management part and the product management part.

[26:38] PARTICIPANT 7:

So, I have written down all those features to the product backlog and then we prioritize those features. Based on those prioritization we have a sprint planning meetings. So, the team is involved in those meetings and we decide on what features needs to be provided in the next sprint. And based on those user stories, so we have chosen a project manager to elaborate those features into tasks that's going to be assigned to the developers. We were using a board where we can follow the workflow. And we started with a simple, agile workflow, but late in the process when we became around ten people in a team. Then we decided to add more steps within the workflow. We divided the development and the quality parts, in the workflow, instead of just saying like, To Do, In Progress, In Review, or Done, we actually elaborate this one. Because what we have seen is of the process, sometimes we had a feature being implemented, but it's not ready to deploy.

[28:26] PARTICIPANT 7:

We just keep it in the loop. So, we developed a very complex workflow for our cases, for our company. The tasks are divided, and we assign work, let's say, how difficult this work is, work hours for those and then we assign these to the developers. So, the developers then take each task based on what task they're assigned. Because sometimes what happens is within the, let's say, sprints, we have assigned, we had, let's say, five user stories. And the project manager actually comes up with twenty-five tasks to be assigned because we have four developers assigned for this project. The project manager assigns related corresponding tasks to developers. Then the developers actually choose from which tasks they want to start. And each day, most of the time, we get to the scrum meetings in the morning. They discuss about what they have done, what are they going to do next, and if there is anything like any hindrance or drawbacks for their job.

[32:24] PARTICIPANT 7:

Once they actually finish their job, they just moved their task into the code review. And the project manager, or senior one actually review the code. And if it's OK, then they pass it to the quality bucket. The quality assurance person just take steps, the user story and checks if it satisfies whatever acceptance criteria, we have given for this user story. And if it is satisfied

then it's actually pushed to the next item into workflow which is like it's okay to be published into production. So, what happens is, we had three stages in our development. One is the deployment stage. That's the server. Another one is the QA and another one is the production. So, these are three different servers we have. The development team works on the development parts when the code review is done and the feature is actually satisfied, the code is pushed to the QA environment. Then the QA person checks if the feature is implemented correctly, and if it is not, it's actually opens a bug, an issue. And sends it back. If it's correctly done, then it actually says this is fine, now this feature can be published in the production. So that's the process actually being followed.

[34:46] RESEARCHER:

Do you think this is a good implementation of Scrum, this process?

[34:52] PARTICIPANT 7:

Yes, because the thing is, like things are changing fast. So, what happens is like on the background, we try to make it simpler for the developer. The developer only is concerned about the tasks assigned to him. And when he's done, he sends a pull request. After he sends the pull requests, he's done. The next step is the responsibility of the project manager. He reviews the code. And if there is a problem, actually, he sends it back. So, what happens is we are following, let's say, instead of publishing, whatever the developer has done, we have a check mechanism. So, it's a quality assurance that's done during development. The code review is actually quality assurance that we're doing there.

[36:01] RESEARCHER:

Yes, it is.

[36:04] PARTICIPANT 7:

Although I explain it maybe in a complex way. There are different roles here. So, you are not overvaluing the individual person within the process. So that's the good part because we differentiate the roles. The project manager is responsible for code view. If it's OK, then he sends it to the QA environment. He passes it there, then the person takes it because everyone knows exactly where to look within the workflow. So that's easier. And actually, that's one transparency that's put into the process. Because when you are a developer, you only see what you need to see. The project manager sees the overall picture. He sees whatever the developer's done, what QA's done and the features. So, he's more concerned there because he's only doing that job, it's not a big deal at that time.

[37:15] PARTICIPANT 7:

For the QA, instead of pushing all the features at the same time in front of the QA, what we do is when something is done, then we test it there. So, this becomes easier for the QA also to handle the job because I also experienced the same thing. What happens is like the developer will just keep the stuff, do a lot of the implementation, they have written less unit tests, but at the end, when they actually test all the code or the features to the QA, what happens is the QA just informs that he needs a few weeks in order to test the system and he cannot give any

responses. And during that time, the developers are, I can say, like out of work because they were waiting if there are some bugs or not. And actually, it becomes a very inefficient way. And one lesson to be learned there is like instead of just waiting for the QA to send bugs, or defects back, we actually go to the next sprint. We start the next sprint, we start developing the features.

[38:50] PARTICIPANT 7:

And when actually there is that defect or bug within the system, we assign these to the new sprints that we have the continuous sprint. We were working this way and have a continuous development with QA together. So, I can say, is this actually the way proposed for the scrum development process, maybe no. But I can say, the reason why we did that is like at that time, the development team learned also how to apply to Scrum. Because one thing that I learned from my readings and from my industrial experiences, Agile is not not easy. For a junior or inexperienced developers, because at least you need to understand. You need to experience all the different phases of the software development. Or you need to have really experienced people within the team. So, they teach the other ones.

[40:26] PARTICIPANT 7:

Although I was a scrum master and tried to mentor them, guide them. Now what I see is like for people to change their behavior, for developers to change the behavior of doing, building, or doing their work, is very difficult. Because most of the time they think that I can only work on one thing, I can only code, I cannot do unit test or other, I cannot think while I'm coding. So, this is not the case there. For experienced people they think, they design before they code. So, in the beginning it become very difficult for us to implement the principles. But later they actually understand the value and it becomes easier for us. So, I don't know, I detailed a lot.

[41:31] RESEARCHER:

That's good. We like the details. Yeah, that's great. That's good. I'll move to the next question. You already touched a little bit on this, but it doesn't matter. We can repeat it and repetition is good. What do you do to assure software quality in this Scrum setup?

[41:53] PARTICIPANT 7:

First, what we did is like as I mentioned like when we had received an idea and the short description about the product, we go over and elaborate that one. But before we move on, before we start creating a projects out of these, we actually discuss it within the CEOs, sponsor, or the high level if everyone is on the same page. So actually, if it is feasible or not, and what are the most important features of these products. Because we need to make sure that this is durable. And this is the first phase in the quality, because if something is going to be really difficult and we don't have enough resources, both men and budget point of view, it's not a good idea to push. So that's the first part. After it's been buy-in by the management, what it is, we write the detailed requirements and user stories as I said, and others.

[43:25] PARTICIPANT 7:

These artefacts are also reviewed before we move on. Because from the project leader or senior, developers, they review this provide feedback. If from the implementation point of view if

something is actually feasible or not. Because sometimes you write the requirements from the product owner point of view, it seems like everything is clear, crystal clear but from the development part, it's not. They need more details. And in these cases, the acceptance criteria becomes very important, very critical. We think these processes actually, when we have dedicated QA people in our team, they also review the requirements and try to write the test cases and to document to test plans. So, we also get feedback from the quality team. And this is actually another feedback in the process to improve the quality of our work.

[44:58] PARTICIPANT 7:

And after actually everyone buys in, what we did is like we tried to split these features into trunks or sprints. After this is done, they assign the number of weeks they need to get implemented. And project manager actually divides this into tasks. You are also familiar like the product owner is mostly interested in identifying the features, the user stories. But from the development point of view, and user story needs to be divided into different tasks like backend development, it can be a database written tasks, or it can be a cloud service, there are different tasks there. So, you need to split it. And this is the task for the project manager because you cannot assign the user story directly to a developer. In our case, a junior with a few years of experience. So, when they divide those, it becomes easier, at that time. So, the project manager then checks the quality of the process. Because here what I'm doing is, I'm explaining both the quality within the process that we are following and the product because they are both important, and you cannot separate them.

[46:50] PARTICIPANT 7:

I think you are aware of the validation and verification within the quality assurance because you have to make sure that you are developing the right product and actually you are developing the product right. Because those are two different terms, that's very important here. We are actually interested in both of those. So that project manager is actually more interested if you are doing the correct job. The QA is more interested in if he implemented the correct features. I also have lessons and experiences like we provided a description and requirements of the feature, and the developer just did something else. But when you look at the feature, we see that yes, that's possible because there is a misunderstanding or there was an ambiguity. The features can be understood that we also but later we think like what's missing, and we just provide that information and that way, it becomes easier. After the developer does his task and the developer, the coding part, what they do is when they code the piece of code, they also write unit tests.

[48:37] PARTICIPANT 7:

We see the benefits of the programming. So, we tried to implement those in some cases. For the quality point of view, we integrated the continued deployment, integration in the deployment. So, this is another part for the quality because instead of doing things manually, we automated it. So, we used the Microsoft Teams and the TFS, in our cases. What they did is like when they send their pull request, the unit tests written on the background automatically executes it. And if there's a problem, the pull request is denied and actually returns to the developer. This comes before the project manager reviews the code. So that's another thing that we detail for the quality assurance part. And if everything goes fine and it's ready for the QA to test, they test it in a QA environment. Because what we did is, we don't want the production, our marketing

people tests the system for the acceptance part. What we did is like it's the previous stage, they test the features. And if the feature is ready, it's ready for production.

[50:39] PARTICIPANT 7:

But still, in that case, we didn't publish. We didn't deploy the features. We had due dates. So, what we did is like actually we just collect those features and based on those completeness of those features we deploy to the production. So, the QA is mostly responsible for doing black box testing. In some cases, white box testing. In the black box testing, what they did is like they have given the features, they have given the inputs and the expected outputs. So, they don't know exactly how things are done. And in some cases, they are given how things are done within the code in order to understand if things are done correctly. And because the testers and the quality ensures team, let's say part of development team and they attend the regular meetings, they were also aware of the details, how things are done. And this allows them to do the white box testing. And when they actually test those features, they also did integration testing.

[52:17] PARTICIPANT 7:

Because we had different pieces, different modules within the system which interacts with each other. So, what they did is like while they are testing the features, which is only related with one module, which does not affect the other one, in some cases we have some features which actually is related with multiple modules. In those cases, they've actually done integration testing. And they check if they did something that affects the other one or not or what happens if they actually combine those modules and after they have done, I also [inaudible] as a product manager. We did the system wide testing, so we did system testing.

[53:12] PARTICIPANT 7:

I also had experience in exploratory testing also. And in those cases, instead of just focusing on features we have in our hands, we tried to actually just go over the system and try to do different tests and try to see like it if it's breaking the system or not. So that was actually very beneficial for us because in some cases what we have seen is like when we did a feature test, a functionality test, it tests the test cases. However, when we do an exploratory test and click some other functions, we have seen that it affects actually the way things are displayed or the output of the function.

[54:27] RESEARCHER:

Okay, PARTICIPANT 7, I have to ask you the last question because we have five minutes. I have another interview after you. The last question, what do you think of this statement, we're trying to be a little bit provocative, Agile produces poor quality software?

[54:44] PARTICIPANT 7:

That actually depends on who is applying the Agile. As an analogy, you have a BMW car, but you are not an experienced driver. So, do you think you can get the all the benefits of driving a

BMW? Or what happens is you can just drive it ineffectively or hit somewhere. So, it's similar. Having a process doesn't mean that you produce a good product. It depends on the people. Agile highlights the process. It gives you what is the most important things you have to focus on. But at the end, it's the buy in from the developers. If they don't have apply it, then you cannot get the good quality software. Agile allows development teams to produce software in a shorter time. And that's a benefit there. Because if you apply the traditional ones, then it's going to be, you have to write the requirements, all the specifications, do the design, then assign the test.

[56:34] PARTICIPANT 7:

But it's a long duration and this is not good because we are the Internet time. You have to be very fast. There are lots of competitors out there. So not easy. Agile, if it is applied correctly, can produce good quality. But the people who are applying it, needs to understand it and needs to be aware of it and they to learn from their mistakes. Agile allows those kinds of things because you learn from your mistakes. And if you are okay that you are not doing everything perfectly, you benefit from Agile development.

[57:31] RESEARCHER:

Well said PARTICIPANT 7. Thank you very much.

[57:35] PARTICIPANT 7:

You're welcome.

[57:36] RESEARCHER:

If you don't have questions for me, do you have any questions for me?

[57:42] PARTICIPANT 7:

No, actually. What's your time schedule for publishing this paper? I'm interested actually.

[57:51] RESEARCHER:

If you are interested yes. I'm just going to check if I have your e-mail. It's going to be ready in August.

[58:03] PARTICIPANT 7:

August?

[58:04] RESEARCHER:

Yeah. Do you want to receive a copy?

[58:06] PARTICIPANT 7:

Yes. Yes, please. I'm also interested if you need a reviewer.

[58:14] RESEARCHER:

Yes sure.

[58:15] PARTICIPANT 7:

I can give you feedback.

[58:16] RESEARCHER:

Yes. I'll get in touch by email. Thank you very much.

[58:21] PARTICIPANT 7:

Yes. You're welcome. Well, I'm not currently in the university. I'm not in a scholar department, but I want to continue. This is a nice experience for me also. I was waiting actually.

[58:38] RESEARCHER:

Yeah. Fantastic. Thank you very much, PARTICIPANT 7. Bye.

[58:41] PARTICIPANT 7:

You're welcome. Bye.