Participant 4

Thu, 2/10 12:36PM • 1:01:09

SUMMARY KEYWORDS
task, team, quality, junior developer, working, code, discuss, people, senior developer, improve, team members, fantastic, senior, proposed, problem, application, communicate, reject, environment, sprint

SPEAKERS
Researcher
Participant 4

Researcher 00:18
Good morning. So Participant 4, how are you? Can you hear me? Hello, good morning now I can hear you. How are you?

Participant 4 01:16
Yeah, I'm good. Researcher, how are you?

Researcher 01:18
I'm doing very well. Thank you. I think we should start because I do have a packed agenda with questions. Yes. Thanks for accepting to do the interview. And I'm looking forward to the discussion.

Participant 4 01:36
Thank you so much. So am I.

Researcher 01:37
Okay. Can we start with a brief introduction? Can you introduce yourself your education, your experience?

Participant 4 01:45
Absolutely. So my name is Participant 4, and I graduated back in October 2005. And from January 2006 onwards, I've been working in software development. Initially, I started out with PHP, and did a bit of WordPress. But then I completed a couple of certifications. And I moved to dotnet and Angular. So I started out with Angular JS and now I'm working with version 13. And on the backend, I'm using dotnet core API mainly. Apart from that, I've also worked on Beanstack. And right now I'm exploring myrn stack as well.

Researcher 02:21
Okay, fantastic. Thank you very much. Currently, you're working on an Agile team, right?

Participant 4 02:28
Yes. So currently, I am actually managing two projects, both of their MVPs have launched successfully. One of them is [redacted]. It's a startup based out of Netherlands and the other is [redacted]. It's a startup based out of Canada. So my involvement on [redacted] is slightly more than the other project and on [redacted] the team that I'm working on is the one that I have referred to in all of my answers as well.

Researcher 02:54
Okay, fantastic. We will keep all the answer based on that team. Are you using Scrum or what type? Yes, yeah.

Participant 4 03:03
Yes, we are. Yeah, we are using Scrum and our Sprint's are 15 days based. Usually, in some situations where stakeholders want something to be delivered quickly, then we modify that sprint to seven days, usually it is seven to 15 days, we have our tasks defined in Trello. And then each of the team member is assigned to them. And we complete them and we track the progress. And sometimes we also do stand-up meetings in which all of the all of the relevant members are involved.

Researcher 03:32
Okay, how large is the team?

Participant 4 03:36
So the development team specifically consists of five to six members, the product team is also there, which consists of about five members then we have a QA team, which basically is also involved with client stakeholders and all of those discussions. So they are about three members for all of them, which are very senior. So the, the CEO of the company actually is overseeing everything. So he is also a developer himself. So he usually takes part in all of the QA activities specifically.

Researcher 04:08
Okay, fantastic. What type of software are you developing in this team?

Participant 4 04:14
Yeah, so the application that we are working on is a startup known as [redacted] and it's a task management and team performance evaluation tool. So imagine that if you're working in a cross functional environment in remote Setup, then you have many people from many different backgrounds working alongside with you. And this tool basically helps us to psychologically evaluate every individual their strengths and weaknesses and how they are moving along with the team how well are they connected and communicating and how the team is overall progressing to achieve certain tasks. So, with the help of psychologist and with the help of other core insight behaviour evaluations, we are able to track each person's behaviour and then get our needs across into it so that we can maximise our productivity.

Researcher 05:03
That's an interesting concept. I like it very much. Thank you for sharing this with me. How long have you been working together this team? Is it a new team? Or you've been working together for a while?

Participant 4 05:17
Yeah, we have been working together for a while now. It's, I would say close to 1.5 years now. But the product itself, yeah, the product itself is being worked upon for the last three to four years, probably, I would say now it is a lot more structured. Because initially, when we started out, as you can imagine, startups usually have a lot of things to do in a very short time. So things were quite haphazard in the beginning. But now, thankfully, things are very much structured, we have a lot of, we have a lot of templates, standards, tools, as you can see, when we are developing, when we are pushing code, how we are communicating, so things seem good now.

Researcher 05:51
Okay, so what the degree of control over the team is that, for example, do you feel like you are self managed, or there is a lot of control over what you do? So what level to what extent you are a self managed?

Participant 4 06:08
Okay, so in the start, I would say I was controlled mainly by the senior staff as well as my team lead himself. But now as time has passed, I'm a lot more self managed. So my tasks are usually defined in Trello. And sometimes there is an overlap, that I have to discuss with someone else. Because the task also, you know, depends on his task on my task might affect his task. So sometimes that happens, but usually, in most of the scenarios, I'm working, you know, myself on something. And then I also communicate with the QA, on making sure that the task is running fine on different environments. So or you could say, a lot more self managed,

Researcher 06:50
Okay, is it the whole team or just yourself?

Participant 4 06:54
I would say just myself, because the rest of the team are very much in communication. Again, as you can imagine, a junior developer might need more help, as compared to a senior one. So my role on the team is more of a senior developer. So if I have to assist other members, but usually, you know, they are communicating within themselves for certain tasks. So yes, communication is going on.

Researcher 07:15
Okay, fantastic. We will be talking specifically about software quality in the rest of the interview. So why, because the definition of software quality can vary, and it's debatable, so I'd like to make sure that we're talking about the same thing. So we use ISO standards definition, I'm going to read it to you and give you an opportunity to either agree or disagree or reflect on it. So the ISO definition says software quality is the degree to which the system satisfy the stated and implied needs for its various stakeholder and thus provide values. And the same time the ISO model also cover many non functional characteristics of the software, mainly performance compatibility, usability, reliability, security,

maintainability, and portability. First, do you agree or disagree with the, with this definition? And would you like to comment on it?

Participant 4 08:21
I absolutely agree with this definition. It mainly encompasses everything that I was also thinking about, I would say something that I want to add to this definition would be the amount of software testing that we do or the methods that we use for it, because I have seen in various teams that usually some teams only rely on working with the unit tests and integration tests. And they think that if the unit tests are working fine, then the software is fine. Otherwise, some other teams might also involve the actual end users within their within their testing purposes. And then they use sort of a survey like situation where they deploy an application in its beta version. And then they involve the end users. And then they use that application, they provide feedback, and then that is integrated as well, sort of like how you see games being launched. You can see a version of the game is launched as a trailer, and then people play it, and then they provide feedback. So it depends on how rigorously we test the application. But the definition itself is very, very much Correct.

Researcher 09:21
Okay, fantastic. That's bring me to the next question. So, in order to achieve or to assure this quality, you might have processes and practices, what type of software quality assurance practices and processes you have in place?

Participant 4 09:37
Yeah, so, the first thing of course, is the developer himself. So whatever he is writing the code must follow the clean code, you know techniques. So the code must be clean, it must be commented well, and it should have unit tests written alongside the code. So for example, if we are using Angular, then we must have the unit tests written in Angular. Then the next is the application deployment on various stages. So we have multiple environments, we have a staging environment, as you know, a development environment, a production environment. And all of these environments are there for testing purposes. So we basically test it, you know, the QA team actually then goes ahead and does some advanced testing, including Phyton, Blackbox testing, and also a be testing and making sure that the application does not slow down if we throttle the speeds of the network and things like that. And finally, when once we have the go ahead from QA, then the final step would be to deploy it on production. And then again, thoroughly tested before making the feature live. And all of this is done, almost automated using the Microsoft Azure CI and CD pipelines.

Researcher 10:47
Okay, fantastic. So that's quality assurance. How about software engineering best practices, for example, what code standard, etc.? What type of best practices do you follow?

Participant 4 11:01
Absolutely. So I would say I would divided into front and back end. So on the front end, the most important thing, of course, is to make sure that whatever code you're writing, it adheres to the standards defined by JavaScript if we are working mainly in JavaScript. So that basically means that like the use of interfaces, the use of proper design patterns, the use of reactive dynamic form bindings, all of these techniques that basically allow us to write a lot more structured code, and a code that anybody can put take in and modify. And as capabilities. On the backend, things are a bit more complex, because it also involves databases. So making sure that we communicate with databases in a secure way, we have proper authentication and authorization on the backend, we have proper repository pattern or other design patterns implemented. And again, there could be a situation which is ideal. And sometimes this will happen that somebody would come in and review your code, and then tell you what things you can improve on it. So code revisions or reviews should also be done periodically.

Researcher 12:10
Okay, fantastic. Let's get on to the fun part of the interview. Sorry. You provided some answers. Prior to the interview via email, thanks for that. Well, when I saw the answers, my assessment was that the place or the team is relatively safe. What do I mean by safe is it's a working environment that provides a sense of security from repercussions. So people feel that it's okay to admit mistakes. They feel that it's okay to propose initiatives and discuss problems. So there is a sense of confidence that the team will not embarrass or reject or punish someone for speaking up. So there is also a mutual respect and trust among the team member. So do you agree with this assessment, that is a relatively safe working place? Environment?

Participant 4 13:10
Yeah, I would I would agree with it. As I mentioned before, I would say that now it is a lot more safe, and a lot more structured. In the beginning. Since things are moving very fast, I would say that usually a lot of discussions, and a lot of frustrations were discussed related to deadlines or related to, you know, the quality of the work, because we didn't have enough time. But now I would say since the MVP is launched, and we are actually moving towards more of a stable product. So now things are relatively safe. Yes.

Researcher 13:44
So do I understand that in the past, there was pressure on the team to deliver and created unsafety environment work? That's what you are saying?

Participant 4 13:54
Yeah you could say, Yeah, pressure was there because of time constraints. I wouldn't say unsafe environment, I would say more of a pressured environment. And yes, since the time was short, we had a lot more tasks to achieve in a very small time. So if somebody was working on something, then he must quickly work with other people. He must also test his work. He must also deploy things. So things were very, very fast moving, I would say.

Researcher 14:26
So in this fast paced environment, people were reluctant or less inclined to, to speak up. Is that Is that what I understand?

Participant 4 14:37
Yeah, you could say that and that actually allowed the team to be filtered out. Many of the team members initially that were not performing up to the level or which weren't achieving the tasks that were that were expected. were eventually either let go, or they were associated with more senior developers and then they basically picked up everything and became part of team. So I would say it's not really that much of a situation where person cannot speak up he can, but it depends on how much time we have for the task and how things were rolling back in the day back in the past, but now as I said, it's a lot more structured and safe.

Researcher 15:19
Okay, so it was a performance was very important before, right. You had to pay for me. Yeah. So to what degree do you understand? Do you agree that is safe today? Strongly disagree, disagree? Neutral? Agree or strongly agree?

Participant 4 15:36
I would say I agree. Since there is always room for improvement. So I would keep the strongly agree part out of out of the question. Now, I would say agree. It can be better, of course, by using a lot more other techniques that we are still not employing. So I would say agree.

Researcher 15:55
Okay, fantastic. So the first questions I proposed in the email, I said, if you make mistakes in your team, it is often held against you. And as part of this discussion, you said in the past, maybe it was possible, but now it's not. Can you elaborate a little bit? When people make mistakes? What has happened? And how do they talk about it?

Participant 4 16:23
Absolutely, absolutely. So I'm going to take two examples. And the first one was related to myself. So when I was starting out, I think in the in the second month, I had a very complex task. And it was related to how the dashboard widgets were displayed and how the API was communicating to get that data. And I would say that since the tight, tight deadline was intending, so the unit tests that I wrote, but not up to the part, some of them failed. And the other ones were not complex, or you would say thorough enough. So I suppose that code and then I was immediately contacted by the QA member, I was also contacted by the senior developer, and we sat down and we discuss whatever the shortcomings were, and I fixed them, and I re-watched the code. So that basically gave me an idea of what to expect in the next task. Another example that I would, you know, discuss is a junior developer who was working on the backend, specifically, and he didn't handle exceptions well, so in certain situations, where the result was not found from the database, or something else failed, whatever the front end was receiving was not in the friendly exception error message that the we can show to the end user. So it was actually returning the entire stack trace of the error and things like that. So that we

also discussed and it was, again, since it was sort of the beginning, it was not more of a blame on you. So we had to talk with senior person team lead and resolve it.

Researcher 18:00
Okay, let's go back to the first example, your unit tests were not good enough. And they created a quality issue and so how the team dealt with it, were your blame for it. Were they okay with it? And what type of feedback did you get?

Participant 4 18:20
I wouldn't say a blame in a negative term, I would say I was, of course, approached by the QA member. And he basically sat me down and discussed, why the situation was failing and showed me exactly what he was expecting from the unit testing, you know, tests. And then I fixed it. And then I was approached by the team lead, we just chat about what are the expectations related to the unit tests, and what usually will be test other than just the unit tests, because as I mentioned, you know, it's also the staging environments, and the production environments on which things are tested. So he basically gave me an overview of what to expect when we deploy on different environments, that configuration settings are different. So I would say it was initially again, a more of a friendly chat in the beginning. But of course, if you keep repeating the same mistakes, then the tone of discussion might change to a more serious discussion. But thankfully, I didn't make too many mistakes. So it was a friendly chat.

Researcher 19:23
So I understand from your answer that the expectations of quality were clearly and friendly communicated. Do you agree with me after admitting the mistake?

Participant 4 19:36
Yeah, I would say yes, I agree with you.

Researcher 19:38
Yeah. So after that, what was your behaviour? Did you make sure that in the future, you don't do the same mistake so your unit test and have become better?

Participant 4 19:49
Absolutely. So it's my usual practice of my own that I always locked down everything that I learned in a day. So if I did some mistakes. You know, I fix it later, whatever I did I log it down in a notepad. So next time if I have to do the same task, I already know the things that I have to follow and the things that I have to avoid. So keeping that in mind, yes, I was very productive in my approach I discussed sometimes directly with QA before even completing the task and pushing it just to make sure that what I did was sort of what was expected. So things don't break too much. Of course, bugs and errors are always there. It's not a problem. But if something entirely breaks, that is an issue. So that thankfully didn't happen, because I was very productive in my approach.

Researcher 20:36
Okay, fantastic. Do you think the same thing happened with the junior developer not handling exception very well?

Participant 4 20:44
Yeah. I mean, the junior developer, actually, as you can imagine, maybe didn't have enough of the information, or the skill set yet for certain tasks. So with time he, he gradually improved. And as I mentioned, some of the junior developers are now actually senior developers. So thankfully, over time, they discussed it with myself also and other members also. And they eventually learned on the things that that should be done and should be avoided.

Researcher 21:14
Do you think this environment where it is safe, friendly to talk about errors and not skilled enough, this environment where you come forward? And you talk about your skills and your error? Does it help accelerate the learning process? Do you think this junior developer learned faster? Because he can talk about his errors?

Participant 4 21:39
Yes, absolutely, he does. Because it's also a matter of responsibility. If the junior developer is always blamed for something which is not working, then he would be demotivated and his morale would go down. But if in a situation where he can freely express his concerns, freely express what he is not getting, or what he is making mistakes with, then the senior developers can jump in and can help him out. So I think more of a collaborative and safe environment and a friendly environment actually improves productivity and people can feel safe to discuss the shortcomings also, because not everyone is strong in one thing, people are strong in multiple things. So that basically allows the team to get stronger eventually.

Researcher 22:21
Okay, that's fantastic. When the when the team learns faster and become more productive? Do you think that the quality also improves?

Participant 4 22:32
Absolutely, it does. And the team basically then becomes, you know, becomes capable enough to not only achieve the quality, but also improve the quality. So I would say it's not just about achieving something and then saying that's enough for the team. It's always about setting new goals and improving the quality further, and employing more techniques and more ways to better mitigate the risks, better identify the potential problems, and better solve them. So I would say risk analysis and mitigation also become stronger as part of improving quality.

Researcher 23:07
Okay, fantastic. So you take more risk, and you assess accordingly, and you learn and the quality of your work gets better, right?

Participant 4 23:16
Yeah, right now, I would say yes, it is. It is an environment where we can experiment a bit. Initially, when we were starting out, we usually had templates designed by the by the designer for the front end, we had a lot more involvement of senior developers on the backend, so everything was already done. But now, things are more safe and stable. So we can propose things, we can suggest new ways to doing things. We can maybe experiment a few things and see if they work, then we can discuss with team lead or team members and we can implement that.

Researcher 23:48
Okay, fantastic. Thanks for that. We will move to the next item, which is the member of your team can bring up problem and tough issues. Your answer said yes. Usually, the company hires a mix junior and senior team. And sometimes people write bad quality or buggy code. And when it's not properly tested, and has to be pushed onto the server, it's always happens that people aren't paying enough attention and not focusing while actually working. Can you elaborate a little bit on this?

Participant 4 24:26
Yes, again, I'm going to share a couple of examples. So the first one was related to a UI UX developer, that was part of the team. And what he did, he basically pushed the code and it didn't have enough testing done. And we basically ended up with something that was not scaling well on mobile devices, the UI code, so the front end, again, the sprint was almost about to complete, so we didn't have enough time to discuss it. But later we sat down and we discussed the sprint blocks and everything that was done that should not or should have been avoided. And eventually it was sorted out. Another example that I would say is, you know, there was a developer that that basically worked on a task, again, related to the widgets that are discussed about the dashboard. And his task was mostly related to the front end and making sure that the widgets are getting the data from the API correctly. And his approach was not, you know, optimal, because what he was doing was issuing one single call, and getting all of the widgets data at once from the API, which was slowing down the application. So I basically sat down with him, and we discussed a better approach. So that each of the call to get the API data was independent of the other calls. And that basically, hence increase the performance of the page load itself.

Researcher 25:50
So do you think that in this case, he learned from you, he learned a better way to do it better and avoid this type of bugs?

Participant 4 25:58
Absolutely. Yes. And I also saw him use the same technique, I think, a couple of months later in another task. So yes, he did.

Researcher 26:06
So his behaviour has changed accordingly, right?

Participant 4 26:09
Yes, yes. His code quality has also improved because he started asking more questions.

Researcher 26:16
Okay, we will move to the next example. The next item, which is people in your team sometimes reject other for being different? You answer that sometimes is yes. But I think in your understanding of rejection here is an individual rejection. But what we mean here is, like you said, and the rest of your answers, usually the team environment is kept healthy for all those working within and supervised by HR. But other times, you may face criticism of your approach, perhaps your methods are unorthodox. I like that the way is somebody has something working a different way. And he was rejected because of his way of working. Is this happening in your team?

Participant 4 27:06
I would say, if there is a task that we have to achieve, and the approach to achieving that task, from a UI perspective, usually it is defined. So we use that approach. Like for example, we have a template ready that we use that. But in cases where we don't have that, there are certain things in which we propose something. Now, I would say 98% of the time, usually my suggestions are incorporated in some way, or they are modified and then incorporate it in some way. But in certain times, my suggestions might be over, might be totally rejected from the start. And I'll share an example. So we basically had to implement a quiz interface in the application for the users to take assessments. And my initial proposed suggestion was that we use something like type form. So we have just one question in the in the screen in the middle of the screen, very large font, and then we enter the answer. And then we click and then animates and fades into the next question. So that was outright rejected. And it was not just rejected, because the team lead didn't feel like it, it was rejected on solid grounds that it wouldn't scale nicely on various devices and in various orientations of the devices. So for example, if the mobile phone is in portrait mode, then it wouldn't scale properly. So due to that reason, it was rejected. And then we opted out for more of a horizontal step of UI for the for the quiz. So I would say it depends on the task itself.

Researcher 28:36
Yeah, that's a good example. Because it's a legitimate rejection, they didn't reject, reject you because your idea was, was not sound, they reject you, because they have a strong argument, which is the scalability of the solution. But you felt it's okay. It's okay to go and talk to them about your solution, you didn't hesitate, because the work environment is relatively safe to do so. Right?

Participant 4 29:08
Absolutely. And because of that, I also just outright discussed. And I actually, you know, the reason that they gave me was very solid. And because of that, I then suggested some other techniques. And we mutually agreed on using the horizontal stepper because that was something that didn't only satisfy the problem, but it also enhanced the issue and we had a very beautiful UI eventually made that was fine. So yes, speaking up is very important, because the team is safe, then you can speak up.

Researcher 29:42
Okay, fantastic. Do you think that the rest of the team have similar feelings?

Participant 4 29:48
Now they do. Yes. From the past, like one and a half years now. Things are very stable. Before that, as I mentioned, people did speak up, but it was relatively lower input. to what we can do now.

Researcher 30:02
Okay, how does it help quality, this feel at ease of speaking up? How does it help the overall quality of the software you're producing?

Participant 4 30:14
Yeah, so it's usually the senior developers, I would say, I have a lot more experience in not just coding aspect, but also how the application is deployed, how we are following different practices, how smooth is the overall sprint goals being achieved. So, if you see there are blockages or if you see the communication is a problem or quality is a problem, or this thing could be improved, then your suggestions should be heard and I speak up. And the only way they can be heard if the team is stable and easygoing, feeling safe, and they are open to communications. So if you can go ahead and talk to your senior members or to your CEO directly, and discuss your concerns, and if those concerns are legitimate, and they are being acted upon, then this is a very satisfactory and a very motivating thing for the for the employee himself.

Researcher 31:04
Okay. Do you have an example where you brought a quality issue either has to do with the process or the quality of the work forward and it was acted upon?

Participant 4 31:17
Yes. So basically, as I discussed, initially, we didn't discuss sprint blockages at the end of the sprint. So I said, I propose that whenever the sprint ends, we must have a meeting before the next sprint starts. And we must discuss the blockages that we'll face in the previous sprint specifically related to quality, how things could have been improved, how the code quality could have been improved, how the other tests or the other bugs that failed, could have been resolved. So all of the members join that meeting. And we discuss everything I noted down, and then I shared the minutes of meeting later. And that basically helps us to get on the same page a lot more faster in the next sprint.

Researcher 31:59
So you took actions upon the meeting after the meeting, did you take actions to implement them in order to improve?

Participant 4 32:08
Yeah, so basically, my job on the meeting is taking notes, sharing those notes with all the relevant members, and then also having a quick chat with the QA team, or sometimes with the CEO himself to discuss what we what we just talked about, and how we can improve it. And then we can maybe

mitigate a solution for it. As I mentioned, the QA team itself is very senior and very capable. So they usually take up things on their own. But I also share my thoughts with them. So the improvement is we now have more time to vet for the quality of the code and test more at the end of every sprint.

Researcher 32:38
Okay, fantastic. Thanks for sharing that example. That was a very good example. So the second items sent you in the email. It says it is safe to take risk and initiative in your team. You answer that it is a yes. I would say yes. And you said it depends on the team leader. If the person is very narrow minded, conservative, and only concern about deadline, he may actively reject you from taking risks. If you are subordinate, by but if you have control lead team, I would say take calculated risk. But don't go overboard. You don't want to jeopardise sprint and deadline. Can you elaborate a little bit on this?

Participant 4 33:32
Absolutely. So initially, when I started out, I was a junior developer, I wasn't in the senior role. And I have an I had some other members on the lead positions. And there was basically a task that came along that a person basically worked on. And what he did was he didn't actually write any unit tests at all. And he pushed the code and the lead, I don't know what happened. But the lead basically merged that PR, the pull request with the actual master branch. And that basically backfired in in a huge way. Because I don't know if the lead didn't check it, or if it was overlooked in some way. But it was actually part of the production code, and it broke. And when it broke, it basically caused a lot of problems and frustrations within the team and the CEO was involved, QA members were involved. And they were trying to find out what happened. And then it later came out to the fact that things were done. I mean, the risks were taken, even though it was a very small thing. But writing unit tests even for that was important, which wasn't done. And that risk backfired, in a huge way. So it depends on who's leading the team and how responsible he is on making sure that quality is the utmost priority. Because if we just push the code and that's it, it's a huge liability for the team and then the company.

Researcher 34:58
So how did you dealt with that problem? Did you talk to the guy? What happened?

Participant 4 35:06
Yeah, since I was in a more junior position, and I was in another, you could say, module, I was working on something else. But I still knew about everything I knew of the situation. And I talked to the guy who basically push the code. And I discussed with him and I talked about what he can could have done to improve the situation. And, you know, the things that we must take care of quality. So we didn't have any templates back then, about any software quality practices. So I urge the team lead to make one and distributed among the team members so that every time they push the code, they know exactly what to check before pushing the code.

Researcher 35:46
Again, the expectations were communicated and become clear. And the junior developer, attitude and behaviour towards his work has changed, right? So did you notice improvement in the quality of his work and how?

Participant 4 36:02
Yeah, so he continued working on the other modules, but I didn't hear any complaints, again, about his work. So since he was working directly with me, I know for sure the quality his code has improved, for sure it did. And I didn't hear any issues after that incident. And things were smooth. So he eventually got, you know, promoted in his role. And that basically implies that things were good.

Researcher 36:27
Okay. That's fantastic. Let's move to the next one, which is about helping each other and the team. It is important in software development environments, that you help each other because not everybody in the same level, right? So the statement says, it is difficult to ask other members in your team to help and you say, in most cases is now I can talk to other people. The key is to ask nicely, ask relevant question. Don't just clean around the other people to annoy them. If someone is specially senior or feel intimidating, or math to them first and asked questions that have to remind them too much with friendly reminder, that sounds are normal work environment. So as long as you now how to approach people, and you approach people in a constructive way. So you can ask for help. Right?

Participant 4 37:31
Exactly. And I would share a couple of examples, if you don't mind. So I was working on insights dashboard, a task that I just talked about, that had widgets and a lot more complex communication with the API. And I had a junior developer working with me that I talked about that he was working on the front end, and his approach was not optimal. So immediately, instead of rejecting his pull request, and just outright blaming, blaming him and creating frustrations, I sat down with him and I talked to him. And I told him that this can be improved in this particular way, I basically coded in a live example in front of him for a small widget. And then I showed him how exactly this can be done. And I asked him to carry that forward and apply it on the other widgets. And that turned out to be very, you know, fruitful for us. Because not only the application performance was increased, but also we were able to push the good quality code one day before the expected deadline. So that was a very good thing. Another thing that I would say is the second example is what happened with me. So I was working on something in my initial like four or five months, and the tasks was to communicate with the backend and to handle real time notifications. So my approach was not working properly, because whenever we went on to a certain page, and we refresh that page, the real time notification triggered twice. So that problem was resolved. Because the team lead himself he approached me proactively. And he sat down with me, and he asked me exactly why I was taking a bit more time than expected. And I discuss the problem with him. And he helped me resolve that problem. So I would say, if the team member if the team members are more open and collaborative, then it is very safe to talk to them and very comfortable. But if the team members are more involved in their own work, and they don't want to share enough information with you, or maybe they are too occupied with things then it sometimes feels intimidating for especially the junior developers.

Researcher 39:37
Okay, I understand. So openness is important in this type of safe environment. Yeah. You mentioned the problem. The example that your senior team member came and helped you. How did that collaboration and how helped to improve the quality of that piece of code?

Participant 4 40:04
Absolutely. So I was expected to push the code by a certain timeframe. And I couldn't. And I shared the reason with my senior developer that I'm still working on something, and it is not working properly. So he approached me himself here, we sat down, and we discussed what I was doing. And my approach was not optimal. Because from a quality standpoint, we need to achieve a certain page, page load speed for the application, if the page is loading too slow, then the end user would not feel that the page is worth waiting. So the quality of the application actually had the unit testing had a clause in which we basically checked the performance of the page. So that clause was not being achieved. And my senior basically helped me achieve that clause. So that's why we were able to push the code successfully. And not only that, we basically cleared it from the quality assurance team itself. So that basically helped improve the page Sspeed. And I also noted it down for future reference.

Researcher 41:02
Okay, fantastic. That's a very good example. Thanks a lot. So far, we've been talking about the quality of the product, which is good. Yeah. All the examples you shared with me, it's about the end product quality. Does the same thing apply to the to the to the processes you use to assure quality? I'll give you an example. For example, you talked about code review that you do code review. So for example, the feedback people share about each other's code. Do people feel comfortable getting negative feedback about the code? Do you have example regarding the process to use to achieve quality, whether this safety things apply, in the same level?

Participant 4 41:50
Absolutely. So I have actually two examples for you. One example was something that I was involved in myself, and I learned from it a lot. And that that was pull request reviews. So as I mentioned, we have different staging environments for the code to be tested upon. And once somebody completes a task, he basically pushes that code to the repository, and then generates a pull request for the senior developer to review and then merge it with master branch. Now, what was happening is that whenever the pull requests were being generated, they were generated on the same day by many developers at once. So the senior developer couldn't have enough time to merge it with the master. So you can imagine, at 10am, in the morning, there are 15 Pull Requests to be merged on the master branch within the same day by this by the same single developer, his work would be too much, you know, you could say stressful, and sometimes things were not getting properly reviewed, and that was causing quality problems. So I alongside a couple of other members, we discussed within ourselves to better mitigate this and assure better quality. And what we proposed was that we don't submit the same pull request by everyone on the same day if we'd rather split it. So if we have 15 days sprint, then every five days, we submit a mini pull request, and it starts getting it, you know, merged with the master plan so that at the end of the sprint, we don't have a lot of work to do in a single day. And that basically tremendously improve the quality because a lot more errors were being caught part of the review. In the pull request, a lot more potential bugs would be identified. So that basically proved the performance as well as quality. Another example that I would share, related to this is how the application structure is so in the backend and front end, at the start, we didn't have enough design patterns implemented. So things were slightly more spaghetti code, as you can imagine. So things were not optimally coded. So I went ahead, I also had a couple of other members that basically discussed within ourselves, again, we

discussed with team lead, and we proposed that we must write a more standards or a more library design pattern oriented code, so that also increased the quality as well as performance.

Researcher 44:10
Okay, fantastic. That's another very good example. Thank you. How about best practices? Do you have an example where a team member or yourself have proposed a best practice our software engineering best practice, for example, a static analyzers, which they have become popular in the last 10 years? For example, I'm just using it as an example. So has any member of the team or yourself or junior or senior says, look, we could use this practice of static analyzer. It helps check the code against our own standards, and that's something new that he or she brought to the team to help the quality. Do you have an exam example where this has happened because people feel comfortable to do so?

Participant 4 45:04
Yes, yes, I'm gonna share two examples. So the so the first one was a junior developer that was very sharp. And he was very high skilled on front end optimization because he was working exclusively on Angular. So he proposed that, you know, we'll be just pushing the code, and we are not checking the page performance speeds properly. So he suggested that we use some libraries and npm plugins, in order to check the performance of the page and to upgrade the packages whenever there are new packages, you know, available. So instead of manually searching for a package, and then manually updating them, for example, jQuery or Bootstrap, he proposed a way to simultaneously check bulk packages at once, and then upgrade them with a single command. And not only that, but also deploy it on GitHub and run a CI CD pipeline against it, and then add a commit message and everything. So that was a very nice way to automate package upgrades that saved us tremendous amount of time. And it's a practice that we still use, that was very much appreciated. Another thing that I basically proposed on the back end was using a better authentication and authorization technique. So initially, what we were doing is using custom JWT implementation Javascript Web Tokens. So whenever somebody logged in, we basically created a session for him a token for him ourselves. And then we managed permissions and everything. But it was getting too complex. And I suggested we use odd zero for this, and use their way of using permissions and event management, everything. So basically, that streamlined everything and made it a lot more easier to handle, because a lot of our implementation was abstracted out of the application itself, and more handled towards the auth. Zero. And so our job was made a lot more streamlined and easier.

Researcher 46:50
So do you think this this dynamic of talking collaborating, proposing initiatives, talking about errors, helping each other how to avoid them, helps to promote quality?

Participant 4 47:07
Yeah, it again depends on the on the balance, if we are just talking too much, and we are not really working on the deadlines, then it's a problem. But if you're just working on the deadlines, and we are not talking about it, and again, it's a problem, I would say a balance is required. So it's more of a situation where if somebody is working on something, first of all, he must have some templates, or must have some know how before actually starting the task on what to do and what to expect. That basically helps a lot of things, especially the time wastage that we can avoid. And secondly, as I propose the

sprint blockage meeting, at the end of every sprint, we basically sit down and we discuss the entire sprint that was just, you know, completed, and we discuss everything that we could have done better. And we also appreciate the team members, we also have a very friendly and supportive environment where we can discuss things that were being concerned sometimes a team members themselves have concerns that they can discuss. So that is a very nice way to make sure that team is moving forward in every situation.

Researcher 48:12
So the balance you talked about, which is a very good thing you mentioned, this balance should be between safety of the environment, where they can speak up, and also put in the right processes, and the right measures, the right standards to help them to do their jobs. That's the balance you're talking about?

Participant 4 48:36
Absolutely, absolutely. And it also involves the actual tasks that we have to achieve in a certain timeframe. So if we have to achieve five tasks in five days, the first thing, of course, is to make sure that not only those are being achieved, but also the quality and the practices are being followed. And people feel safe to put effort on quality. And if one of them is out of the balance, then we have problems. But it is very important to proactively identify those problems and to productively discuss them instead of waiting around at the end of the sprint to discuss them. So it's always proactive communication and standards which are being followed along with the tasks and how open the team members are to communicating, bringing problems and helping each other that actually and eventually helps in promoting better quality. We now also feel safe because we have time to invest in developing ourselves and our technical knowledge and capabilities. We feel safe to schedule time for this.

Researcher 49:20
Okay, fantastic. We have two items to go through and we don't have enough time to go through them. The next one, it stayed no one on my team would deliberately act in a way to undermine my effort. Your answer was quite strong. You said I'm sure no one would. As this is highly unprofessional, unethical. It's something like this happen. It's unfortunate to say the least and should be properly investigated. We are all professional here. If you cannot work with me, talk to me about it or ask involve a mediators. That's really nice. I mean that the work environment is very professional. And this sense of safety is facilitated? Who do you think promotes this safety or where it comes from?

Participant 4 50:23
So, sorry, sorry to interrupt you, I'm going to share an example. If you don't mind. Yeah. So if you remember, I talked about the junior developer and the PR getting merged in the master branch without retesting. So that was a situation which I believe from, from the junior developers perspective, he could have actually blamed the senior developer that he has deliberately trying to undermine my effort. So you're deliberately trying to make me look bad in the eyes of management, because instead of merging the PR, you could have just told me that you should write unit tests, and I could have written them, and then this wouldn't happen. So it depends on how you see the situation. So in this case, for example, yes, the junior developer might say that, but the developer, but the senior developer would have an argument that my job is to just merge the PRs. And I don't care if you have written the unit test or not,

that is your job. So I must merge the PRs. So again, it depends on how the situation is being tackled. And for me, thankfully, speaking, I have always been very collaborated with an open in my situation. And I always make sure that whatever I'm working on, I totally know the requirements. And I also know the expectations of that task. And the safety that you talked about, that comes up after having multiple discussions with team members, making sure that everyone is on the same page, making sure that the team goals are clearly defined. And then also making sure that people are sometimes, you know, shuffled in their way of programming practices. So for example, if I always work with certain person, then if I have to work with someone else, I might feel not comfortable enough with them. But if the team is constantly being shuffled, and people are working with each other all the time, and tasks are being divided among the strongest, and the weakest team members alike than the weakest team members don't feel like they do not have enough skill set, they would have some sense of security and safety that of course, they can discuss with their senior developers, and they can resolve that issue, whatever they are facing. So it's, it's a collaborative and open environment that promotes safety.

Researcher 52:30
So what's the role of management here or the team leaders, they must have a role for this environment to flourish the way it is.

Participant 4 52:40
Yeah. So team leaders, of course, from a technical standpoint, their role is to make sure that whatever sprint tasks are defined, they are properly achieved. And they are also making sure that if they assign a task to a certain person, that is the best person for the job. So for example, I'm a full stack developer, another person is a full stack developer, but I'm more strong in Angular implementation, and he's a lot more stronger in the databases, then whatever tasks are there, they must be divided in a way that, you know, basically accelerates the progress of the task. So if I'm working on the Angular aspect, I would have completed a lot more faster than if I'm working on the databases. And of course, as I mentioned, the shuffling part is also responsible for making sure that I increase my skill set over time. So a person's individual progress as well as a team progress is being monitored by the team lead. He is also responsible for communicating with stakeholders and CEO and other seniors, in order to discuss what further requirements are to be implemented, and whether the implementation that was already done is properly your quality, quality assurance team has properly given a sign off on it, and whether they have been deployed on the production environment so that if there is any bug or any issue, he proactively communicates it with the team. So team leaders are more of a senior people who guide all the others and also make sure that the team stays on track and safety is practiced not just a label.

Researcher 54:10
Fantastic, thank you very much. We come up to the last item, which is working with members of my team, my unique skills and talent, our value and utilised your answer. Again, it's dependent a team I would say yes, if the team is a unit, then things are pretty good. And everyone's happy and appreciated. But if some if someone taking credit all the time, either he gets away due to supervisors allowing him the favour or he gets caught eventually wish will not be a good day for that employees. So you the answer here is very interesting because you mentioned a team is a unit you which you talk about in previous example, can you elaborate a little bit more on this?

Participant 4 55:08
Absolutely. So I'm going to start with the team as a unit apart. And what I mean by this, as I mentioned, it's not just about your personal goals that you have to achieve within a certain company, it also the team that you're working with, because if you're just working on your things, and you're not being friendly and open with the other team members, then eventually the team will not feel like you are part of the team. And that basically undermines your efforts, eventually, because somebody might jump in and say that this person is not somebody who we can work with. And we do not want him on the team and all the other frustrations that might come along with it. So by making sure that the team members are very, very cohesive with each other, and the tasks are well managed, and they can communicate with each other, that basically helps us to get motivated enough to do our job. And everyone feels like he's part of the team and is appreciated. But at the same time, unfortunately, I have been in the past, not this current team. But in the past, I have been part of some teams that didn't actually have this environment. And it was a lot more like bullying and politics and a lot more of the negative environment that you can think of, in which if I'm a junior developer, and I'm working on something, I'm working a lot more than other senior developers maybe. So I'm putting in 14 to 16 hour days, instead of eight hour days for the seniors. And at the end of the day, whatever I pushed, were labelled as the team effort and labelled as something that the senior developers basically pushed. So I felt very demotivated on that. And I sometimes discussed it with my, with my colleagues as well that this is maybe unfair, that, you know, the efforts are not being not being appreciated. But thankfully, the teams after that, that I am currently, especially the part, the part of the team, they are very open and very appreciated, appreciative of the efforts. And the unique skills, part that I talked about was just the example that I discussed previously was related to how strong a person is in a certain aspect. And if he's strong in something, then we should not only achieve, you know, utilise that to achieve a task, but you should also utilise that to help him train other people. So it's a constant form of training that's happening within the team and team leaders are usually the people that that have to take an eye out and be keen about it. So that every person is eventually growing in his own field and so on skill set and also the quality of his work.

Researcher 57:44
Fantastic. That's a really good example. How does it make you feel? And what? What's the impact in your attitude to work the quality of your work?

Participant 4 57:55
Because you say, yeah.

Researcher 57:59
Yeah, go ahead. I am looking forward to your answer.

Participant 4 58:02
Absolutely. So I would say, as I mentioned, you know, it's, although I'm a senior developer now, and I have a lot more control on things. But still, if I was a junior developer in the same team, I would be, I would be very happy because I would have a chance to work with multiple people, I would have to have the chance to speak up. And to raise my concerns. If I have a problem with someone, someone, I

would have a chance to very friendly in a very friendly way, communicate with them and communicate with the mediators, I would have a chance to work directly with stakeholders were directly with QA, work directly, even with my boss, because usually the bosses do not communicate with you, well, they, they basically have a chain of command. But in this type of situation where everybody is collaborative, it basically enhances my skill sets a lot more rapidly than if I was just working on it myself. And it also helps me learn a lot more new things. It helps me to better, you know, increase my productivity and the quality of my work because I know exactly what to do, and what to avoid. So that the next time if something happens, I also know exactly how to avoid that. So I'm constantly learning. It's a it's a very fun environment to be to be a part of. And I think that is something if you achieve that eventually you wouldn't want to leave that team.

Researcher 59:23
Okay, you mentioned a very good word you're always learning is this continuous learning contribute to the quality of your work and have

Participant 4 59:34
Absolutely it does. So, eventually the quality is tested against a certain task or as against whatever deliverables you are you are putting forth. So if the quality is increased, that eventually means that your way of doing things basically enhanced and you did them better. So whatever mistakes you were making, or whatever code you were writing that was, that could have been improved and you keep on learning it and you keep on improving that, then eventually you will start writing good code and you will eventually start creating better unit tests and start managing things in a in a more optimal fashion. And that would translate into a higher quality of the code output. And that would actually not only increase the quality throughput for the for the team overall, but it would also have you have a sense of, you know, accomplishment that I did something better. And that sense of accomplishment, accomplishment then fuels further motivation, and you keep on improving yourself.

Researcher 1:00:35
Fantastic, solid. Thank you very much. We consumed our hour. I enjoyed the interview very much. I like to thank you for the good examples. And thanks again for participating in the interview.

Participant 4 1:00:47
Absolutely. Thank you so much, Researcher. It was very insightful and I really enjoyed all of the questions, and I hope that we work again in the future.

Researcher 1:00:55
Okay, definitely. I will reach out. Thank you very much. I wish you a good day.

Participant 4 1:00:59
Thank you so much. Me. Sure. Good bye. Thank you. Bye