**Interviewee**

He's kind of interviews. It's likely that I wouldn't have questions at the beginning, but it's fine. I'm looking forward. And you know, thanks for the opportunity. Yeah, I think, you know, I, I would love having this conversation. And I'm sure you know, when I look back, I will find a lot of interesting things coming out of it. So I'm looking forward to it. Sure.

**Interviewer1**

So we'll start the interview with some demographic questions. Then we'll jump into the interview related question about the tools and all of that. So, the first question is, how old are you?

**Interviewee**

Okay, I am 46 years old.

**Interviewer1**

Okay. And how do identify your gender identity?

**Interviewee**

I'm male. I identify myself as male. Okay,

**Interviewer1**

and what is your highest level of formal education?

**Interviewee**

I completed my Bachelors of Engineering from Bangalore. So an engineering graduate? Yeah.

**Interviewer1**

Okay. How many years of experience do you have in the software engineering field?

**Interviewee**

I think about a little more than 20. So 21 years is

**Interviewer1**

that's a long experience.

**Interviewer2**

And as a quick kind of follow up. Your bachelor's was also in computer science.

**Interviewee**

Yeah, I was I completed in computer science and engineering. Yeah. Okay.

**Interviewer1**

How many years of experience do have in leading teams, like software engineering teams?

**Interviewee**

I think this is more than probably. Yeah, I would say a little more than 15 years. Teams of various sizes actually started up with small teams of, you know, few people up to leading the company.

**Interviewer1**

Okay, so these are all our demographic questions. Now, we will jump to the interview related question. So the first question is, what is your perception regarding airbag localization? So what do you think about that?

**Interviewer2**

And before we go into that, "", do you want to give a description of what we mean by localization?

**Interviewer1**

I think we should hear the description of our participants has done you can describe whatever we are thinking about it.

**Interviewee**

I think I've had a little bit of, maybe not when I was informed about the interview, I had a bit of discussion with Nasif as well. And he told me about globalization and just, you know, came to know about it. So basically, as I understand, it's about identifying, you know, where are a bug exists in your code? Probably not. That's what I would interpret it as. And then the question was about bug localization tools, right? So I think we haven't, I will not say we have used many automated tools for bug localization. So we've used something like sonar cube, which is not really a bug localization tool, but it can analyze the source code, and then perhaps give some insight on, you know, potential code flaws. Like you know, you have assigned a variable, or you have used a variable without really assigning something. So that sometimes gives some hints on on some areas other than that. The rest of the identification is mostly I would say, manual. And when ever there's a bug, you will probably not think in your head that what might be the cause of this bug. And you would go through into your system to find that out. So I guess, brain is the other tool, probably.

**Interviewer1**

Yeah. So in this study, we define a bug localization tool is a way that it's a tool that can identify potentially buggy files, given the bug reports or stack trace, or both. Suppose a repository contains 700 source code files, and someone logs a bug report in some system. So for example, in GitHub, and that tool will shortlist a few files potentially responsible for causing that bug. And the developer can start further investigation or fix procedures from their, from the shortlist. That is our description of bug localization tool. So given the state of art of such a tool, would you use such a technique?

**Interviewee**

I think, first of all, in terms of adaptation of such tools, right. I think it depends a lot on how effective it is, right? So I think as a developer, we always want to do things in our own way, because that's a part of the treasure as well. Right? If a tool gives me such facility, I think at any point in time, we always have to evaluate how useful it is. And I think if a tool is really useful, then it saves a lot of time, then eventually, you know, I would probably start using it. So yes, if there's a, you know, the good tool that is able to tell me that, okay, this bug has been reported, and this is the source of the of the bug, and then most likely, you know, I would, I would use it, as long as it's, I find it to be reliable.

**Interviewer1**

Okay, so our next question is, what do you think will be the benefits of that tool, like, all the benefits of that tool like time or experience or what kind of benefits?

**Interviewee**

I think, of course, you know, it will save time, especially, you know, in software engineering, or in development teams, we frequently have, you know, team members shifting and switching, right. So many times, new team members will have to come and join in a project. So for them, it can be really helpful, right, because in order to just get to an identification point, they'll have to run the code, you know, do some debugging to find out understand the code. So they will spend a lot of time just understanding the entire code base to finally get to the point where you can solve the bug. So of course, for new members, I would think it would be a pretty helpful and time saving tool.

**Interviewer1**

Yeah, and what do you think will be the harm of such a tool in use of your company or any kind of organization?

**Interviewee**

I think I can think of two. The first one is, of course, you know, as people go through the code, so they get to learn about, you know, I mean, when we were first when we first joined the industry, right? So I think the first few years, whenever you're part of a new team, the best way to engage a developer initially was to give them a couple of bugs, you know, they solve the bugs, and they walk through the code, and then they get to learn about the code base and different functionalities. So if something you know, immediately identifies where the bug can lie, then I think, you know, it takes that opportunity out a little bit from that. The other thing is, I've seen a few instances, right? Where, you know, we look at a file, and we look at a potential bug, and we think that, okay, just changing this line will fix it, okay. But it fixes that bug as well. But it has some, you know, side effect that, you know, developer was not aware, which can be caused in some other area, right. So, you know, there's a possibility of more of those incidents, so probably will, you know, fix this bug pretty quickly, but we are likely to introduce some other bugs, because we do not have sufficient knowledge of the existing code base, it's just bizarre, you know, to think the other thing that I can think of is that, you know, like I said, developers, they always enjoy this entire business of troubleshooting, writing, or writing a function. So I think, you know, if that time came, I will probably miss out on this, this opportunity, or, you know, miss out on this thing that I that was enjoyable to me learning about while I was learning about a new system.

**Interviewer2**

Yeah, that is such a unique take, right? Like, would you say that you enjoy fixing bugs more than you enjoy writing new features, for example?

**Interviewee**

No, not in that way. I think what we enjoy or what I enjoy is probably, you know, having that, that that feeling that I could conquer something, right. So be it be the, you know, new functionality that was written nicely that you know, fun factor. And then you know, you have a critical bugs that people are finding difficult to solve, but you don't look at the code and you can sort of identify. So that's also sort of, you know, you come up with a self satisfaction at the end. So I will not say, you know, it's, it's related to know exactly, whether fixing bugs and you haven't, it's more about, you know, making use of your intellect to, to make the system better in some way and fixing bugs is of course, one of those is right. Yeah.

**Interviewer1**

And the next question is, even though I think you have already answered but still is new, have you ever used any bug localization tool?

**Interviewee**

Like I said, not really bug localization tool, some code analysis tool, like sonar cube I've used in the past.

**Interviewer1**

Okay. Can you describe the tool in brief for us?

**Interviewee**

Okay, yeah, so sonar cube, basically, what it does is, it can scan through the source code, it has some in a pattern matching, and it can identify some some areas, like, like I said, like, if you assign a user variable that has not been assigned, it can identify, if you have, like, when we use strictly typed languages, then you also have a lot of compile errors that the compiler fixes, right. But if you're using scripting languages, like Python, for example, and you have a path that you, you know, haven't returned anything. So it can identify those areas and tell you about specific things that you shouldn be doing. It also has some ability to, you know, on code formatting, for example, if your code is not properly formatted, you can get it I mean, there are other tools to identify that, but you can also have reporting in sonar cube. It can also report potential bugs, in some areas. So and then the other thing that it does is, and of course, you know, you can as you go through your build lifecycle, or, you know, yeah, as you go through your releases, you can have a report of these things, you know, whether it's going up or you know, whether it's going down, and various functionality around that. And test coverage report, also, it can show, if you have like tests in your code, how much coverage of the source code that you have had, it can show. So altogether, you can use it to sort of identify some some issues, make some fixes in your code, and probably get rid of some some issues before appearing.

**Interviewer2**

So kind of like as a follow up, what kind of tools do you use for debugging? Like, once you know that there is a bug report? Not using a bug localization tool, but what would you use to find the source code? Like, how do you do that?

**Interviewee**

Okay, so I think few things. I think the initial I mean, what we typically try to do is, the main step is reproduction of the issue, right? In most cases, you have issues coming in, in production, and then you want to try and replicate the situation in your development environment where you can debug, right? So I think that's the first stage. And then, of course, you know, from your experience, or, you know, knowledge of the source code code base, you sort of try to identify some areas where you think this bug can be, right. Ideally, if you have good test coverage in your source code already, then you can start by writing a test case, with the scenario where the bug reproduces, to see if you can make the test fail or make, you know, write a failing test. And then you would, you know, that it makes it very easy to identify, and then you probably, you know, go into the source code and try to find a solution of how you do it. And then, you know, if you don't have test cases, then we probably not try to inject these values in the function or, you know, whichever they see you have that bug and then you you either put a debugger breakpoint and try to execute your code if you're in a scripting language. In JavaScript, if we call where it's mostly browser, you would put some in a console dot write line in those specific places and try to identify what's what's going on.

**Interviewer2**

Okay, so in, in those kind of, so like, in that explanation of debugging that you just gave, like, what are the aspects that you like the most in that? Right, like, which functions of the tool do you like the most and the flipside is, what functions do you think are missing or could be improved in that process?

**Interviewee**

But I think I think mostly what we enjoy is putting up a breakpoint into your source code. Right? Yeah, I think that's the most important one, or, you know, something that everyone enjoys. It's also probably one of the most, what do you call it?

**Interviewee**

spoiler, I would say I'm in it spoils developers as well, because it's very easy to just set up a breakpoint and, you know, start using that. And then, of course, you look at stack traces, or, you know, what do you call, call stack, right. So which function it came in, through, if you have exceptions, in the data, the tracing tool, and then you know, watches, or like, variable watches that you can pull to see values of different variables. So I think, you know, these are the ones that we use, frequently. I haven't really seen, I mean, it's difficult to debug in some, in some languages, perhaps, like, you know, maybe Python is a little more difficult to just do inline debugging like this. Because you have to set up some code or, you know, attach a debugger, debugger to your running processes, etc. But I think most of the, you know, languages have tools that have seen, I've seen that, okay, you can, one way or the other, you can do it.

**Interviewer2**

So essentially, the feature that you use the most would be something that you can watch the code as it is executed, like find out values that are in there, find out what gets executed, or what doesn't get executed kind of features you use the most? And is there there was a magic wand, then you could get something, what would you? What do you what do you miss? Or what would you where do you find the most difficulty in in this process?

**Interviewee**

So I think it depends a lot on how well you know the tools. Right? So if I have to think I mean, what do I miss? I think the most difficult stage is probably getting to the point of code where you need to debug, right? So yeah, I think that's the most difficult part. And you know, that's where we spend most time on. So for example, you know, let's say I'm testing an API, right? And with certain set of data, it's it's failing, then you have to prepare this data to sort of, you know, make an input into the API, then, you know, if you're not sure exactly where the issue is, you would have to debug through different stages until you finally find the place. So I think if there could be something that would sort of help me identify this data, or you know, this or somehow automate this step, I would find that really helpful.

**Interviewer2**

So wait, so you mean like, and I talk back just to see if I understood it? Yeah, that's, that's my first. So where to start? Debugging is something that would be useful.

**Interviewee**

Yeah, exactly. Exactly.

**Interviewer2**

Okay. Okay, that's great. And so, if we were to build this bug localization tool that would, what do you think the minimum set of things that you would need in this kind of tooling, right, like, is it because the right now the tooling, what it does is, it gives you, you give it a bug report, and it gives you these five files are likely these n files like 2 or 1 or 10, or something like that. Different tools give different, but it just says this set of files are where the bug is likely to be in. And you kind of were saying that you would appreciate more, if you know where to start debugging rather than Then where to fix the bug? Like that? What minimum set of features would you want?

**Interviewee**

So I think, you know, as we're doing development these days, right, so I think the bugs that take most time are not really related to single applications. Right. So the challenges that we face are, in many cases related to the infrastructure or communication between different services, right. So I think the reason why I'm saying is that, you know, if we, if we look at, you know, one of the major applications that we work in, so we have, like, you know, many different applications, and then, you know, there are there are different microservices, which communicate with each other. So, identifying and fixing issues in a single application is not as difficult as, you know, when your application has interaction with a few other services. And you know, you get issues there. So I think one of the useful things would be if that could be put into consideration as well. Right. And then I think the other see thing that I see is that right.

**Interviewee**

I don't think that's a relevant question for me at this point. Yeah. So I think Yeah. Yeah. I'm not really sure exactly, you know, what would be his minimum set of requirements? I think, yeah. I mean, if you if there's a bug report, and if it can tell us where exactly we should look for for this issue, that would be helpful. If I mean, one of the issues that we probably often see is, in front end applications, right? I mean, one of the scenarios we had is that I mean, we used to have these forms, where there would be a set of validations, it's very easy to put these validations. But developers would often miss that. Right? So I mean, we have this form, maybe in a similar kind of form, repeating for maybe, you know, three different applications. And in each three of these, we have this same set of bugs coming in that this validation, this required validation is missing, right. So I think if it could analyze source code, and maybe you know, give us a sense of these kinds of issues that we'll get from the back, you have this, this set of bugs, and you've written a piece of code, this piece of code is likely to have, you know, one of these bugs, that could probably be helpful as well.

**Interviewer2**

And, like, as a follow up, would you like to use this tool for more difficult bugs? Or would you like this for more mundane, easier bugs? Like if the tool can only work on one thing? Which would you prefer that this tool work better on? Do you care? Or do you want like, doesn't matter? It should be equally?

**Interviewee**

I mean, if the tool would identify, you know, the difficult ones, then of course, I would say that that's better. So I would say that, you know, if it's able to identify the difficult ones better if it can protect the easy ones, then even better. Okay.

**Interviewer2**

Okay, awesome. "" Do you want to ask question 10?

**Interviewer1**

Yeah. So the next question is, what kind of output Do you prefer from a bug localization tool? Like it can point out the buggy files, or it can point out the person who is better fit for fixing that bug, or it can point out the similar bags that has been filed before and has been fixed by someone, what kind of output do you expect?

**Interviewee**

What was the first option sorry,

**Interviewer1**

it will just shorlist all the source code file in the repository, and we'll just give you five files that are possibly related to the bug. And the second option is it can point out the best person in your team sho is better fit for solving that bug?

**Interviewee**

Okay, I think I would go for the first one actually. Because you know, anything that's related with, with people coming in the picture. I think we all know that software engineering is something where, you know, it's not just your skills, your team also matters a lot, your cooperation matters a lot. So I think in as long as if you have, you know, people's name coming in as part of some, some tool that always clear creates a conflict, and it's probably better avoided.

**Interviewer2**

In the first part, would you prefer that it is just the files that are given? Or would you prefer the similar bug reports? bug reports that are similar to the current bug report?

**Interviewee**

Yeah, I think both actually. Okay.

**Interviewer2**

So both would be useful. Yeah. And in the similar bug reports, would it be helpful to also find out where those bugs were fixed as well?

**Interviewee**

Yeah, I think so you can, you know, you can dig through and find out. So I think one of the other utilization for this, this tool can be is that sometimes you have no repetitions in code, right? Or you have maybe a similar service, maybe you're working on two services, but both of them are using you know, copied code from each other. Right. So, you have bug in one place, then you know, it will trigger a situation where you can say that, okay, I have some common functionality in both of these services, we should probably, you know, unite them into one library or something and try to make use of it.

**Interviewer2**

And I think that, yeah, final question.

**Interviewer1**

Yeah, we have another question. So, the other question is about the performance of bug localization tool, the tool we are using in definition, that can short list files, like potential buggy files, they can return you like top five potential buggy file or they can return just top 10 potential buggy files. So what do you think about what should be the top number? Like should they return the top five or top 10 or top 100? What should be the acceptable maximum number for a developer

**Interviewee**

I think, you know, are you talking about maximum or minimum because you know, in terms of minimum minimum, the better? Of course, the tools reliability?

**Interviewer1**

Those tools actually the performance actually increases with the number like if they return top 10, then they can perform in 100% accuracy, but in top five, their accuracy will drop that kind of way. Okay,

**Interviewer2**

okay. The reliability is indirect, correlated with the size, so, are directly correlated to the size of the returned set. So the larger the set, the more likely something is there, but maybe not the first answer, right. So, what will be our threshold for reliability we like? So for example, at some point, you would say sonar cube is useful or not useful, right? Like, what is that threshold here? Like, in terms of, should it be the first result that should be the thing or

**Interviewer2**

also, among 100? Bugs? how tolerant will you be for? Errors? Like how many times if it gives the wrong file? Would it be okay? So

**Interviewee**

it's a difficult question to answer because, you know, I can't envision so I, I would guess that, you know, if I have like, for a bug, if I have 10 files, and one of the files is likely to have the issue, I will probably be able to know by looking at it, I might be able to rule out three already, because you know, I know something about it. And then I look at seven more to see. So yeah, I think 10 is probably an acceptable number, if you can, you know, find it, find it within 10 files. Yeah, I think 10 is a good number. Of course, you know, the shorter the better, or, you know, smaller if I have more probability because the effects but I think what I would not like to do is probably not if I have such a report, I didn't want to go through all the facts. And then find out that these are not the files. I mean, none of these are there. Probably, if I have a list, then I would probably like to have at least one hit from from that list.

**Interviewer2**

So the cost of an incorrect classification would be high, like the cast, you would not like to go down around rabbit holes.

**Interviewee**

Yeah, I mean, in my opinion, that's an important criteria.

**Interviewer2**

And, like, static analysis, tools, like sonar cube also have false positives. So what would be a threshold for a false positive V here? Like, we think it's a problem from our tool, but then it actually is not? How frequently can we screw up before you say, I'm not going to use this tool?

**Interviewee**

I think like I said, right, it's when you have anything new, it will always have an adoption challenge. Right. And I think we can see that in, in many developers as well. Right. So I mean, sometimes we add up very fast. Sometimes we don't, it also depends on on individuals as well. So I think in order for developers to really, you know, start loving and using it on a frequent basis, I think it needs to be pretty accurate. So yeah, I think first few trials are very crucial, right, you need to find the target before going to test and you know, improve, to improve it initially. So I don't know. I mean, I was having a discussion with someone about the the tool from GitHub, right, code. Something that writes code automatically right? Yeah, so yeah, I was wondering, you know, how much people has actually adopted, how much, you know, useful, people are finding it, and so on. But I think, you know, I haven't adopted to it yet, right? In my day to day work, or anything, so I do feel that, you know, if whenever you have a new tool, you can probably have some some, you know, adoption issues. That's one thing. The other thing is, of course, it probably needs to be easy to use. And so on. So I think, I don't know, I mean, I always try a little later than others, but probably other people who are, were much more faster adopting.

**Interviewers2**

So, following up on, you said, easy to use, like, what? How would like what would be, what would make it easy to use?

**Interviewee**

So I'm just thinking, so. I mean, if I think of, you know, how it will do it, it will need access to source code, right? What will, you know, I need to do in order to give it this access. That's, that's part of the setup. And that needs to be easy in some way. Right. So I think that's, that's one of the ways and then of course, you know. Yeah, I think that that is the part that I would say, when I say because, you know, at the end of the day, it shows me a list of these files, right. So that's pretty intuitive. You know, you have a bug report, you also have a bit of a list of files, but in order to, you know, get it to this position, How much work do the developers or you know, whoever is setting it up? They will need to do? Yeah.

**Interviewer2**

Would you like the output to be on the issue management system or in the IDE

**Interviewee**

I think IDE would be nice. IDE would be nice, but again, I'm thinking because right now, we don't really have a lot of annotation on on our, you know, there's not a much relation between you know, your issue management systems and IDEs at the moment.

**Interviewer2**

Yeah, Okay, and the last question about performance is how long for compute time, do you think is an acceptable time? For this tool? Like, does this tool need to respond in seconds is minutes Okay? Should it be? Is it okay? If it takes a day to calculate this and put it in? Like, what is an acceptable timeframe? Do you think for?

**Interviewee**

Your I am thinking it will, of course, you know, depends on depends on, you know, overall processes, and you know, how the different processes in a company work? So, I mean, the project that we work in now, right, so, here a QA reports a bug? So, the first hurdle that QA has sometimes is that because, you know, there are different applications that make up a large product suit, right? So they're looking at bug from, you know, one application, but maybe it originated from another application application, right, or, you know, input from another application is wrong, because of that you have this bug in your target application. So this identification they do, they have to do it themselves or not, they first maybe, you know, issue the report and assign it to the team where they first found it, that team does an investigation, and then, you know, assigns it to another, to the appropriate team, that team actually investigates and, you know, try to find a solution to the issue. So I think in our case, this time, can take maybe half a day, in many cases, because there's also time differences, right? QA reported that at night, we find out in the morning, it's wrongly logged, we assigned it to another team, that's again, in United States, so they look at it. So we have this, you know, half a day gap, at least we have. So in our use case, it's okay. You know, if it takes, you know, half a day to run everything and gives us an accurate report, and especially for QA who needs to identify exactly, you know, which application and you know, a lot of cases we have bugs that are wrongly identified, if it can, at least, you know, identify that it originated in this application. That's also something that is very helpful in our case.

**Interviewer2**

And it's interesting that you mentioned this. And so I want to ask another follow up here is the do you think this kind of tool would be helpful in the hands of the QA instead of the developer where they can provide a little bit more information? Because the QA right now is reporting what is happening in the outputs, but and maybe they attach a stack trace if they have one? But. And this way, they kind of also know, is there a similar bug report? Is there or at least part of the

**Interviewee**

similarity? As long as you know, bug reports, similarities concern? That's very helpful for QA? Because, you know, right now, we have issues that are reported, like, you know, we had something similar in this situation. And that comes, you know, sometimes it comes through, you know, related issues. Sometimes it comes specifically from the QA said, you know, because the QA remembers, you know, something like this. So, of course, you know, that would be very helpful if it can identify similar bugs. So, that would be helpful for, you know, QA side, who executes, you know, maybe, you know, like I said, in some cases, our QA would find it helpful that if they can quickly have an overview of which application, they should actually lock this bug into, right. And then maybe, you know, not much use of finding the exact source code for the for the QA in general probably. Also a well-maintained codebase may not need such tool. They may have been able to localize the bug in a viable time without the help of any tool. On the other hand the tool may not work properly in a poorly maintained codebase.

**Interviewer1**

Okay. I think that's it, right. Yeah, that's the end of her interview. I'll just stop here.