Intermediate Focus Group Interview

Me

[00:00:00] So how did you find using the tool to teach?

Amos

[00:00:04] I think it was good, other than the few bugs. As far as the design went, I think I was happy with it for teaching. It would have been nice to have a button that says steps to the end, because there were some cases where I would have liked to just see if it evaluated correctly, but instead you had to click through a lot of times. So it would be nice to have the option to either see if something evaluates or take a look at the steps. I think a nice workflow would be implement something, run it to the end, see if it works, and then if it doesn't, use it as a sort of debugging thing to see where it might go wrong. I think that would be nice. I think overall I liked it. It was good. It was nice to have the steps clear so that it wasn't just like, there was sort of a visual aid to explaining how evaluation was happening because like I have in my head the way that I think about evaluation happening but it's much nicer to just have a thing like look this is what's happening so I think that was good I think these steps the formatting I genuinely liked there was nothing that I thought was like oh this didn't work or this I would have liked this I think so I'm just quite happy with it what do you think about the language as a teaching tool I think the language itself I think it's quite good. I think there are some things where, was there anything? I think it's a good tool because it has the, I think it has all the things you care about in Haskell without having the things that you care about. There are features of Haskell it doesn't have, but Haskell is a very advanced language. For teaching functional programming, I think it's got the right range of features. I wonder if, I think we talked about this before, but I wonder if, um, uh, what's the, what are they called? Type classes could be, like, that would be, like, maybe the next, if you were to take this, like, a step further, make it more advanced, I think the next step is type classes. Yeah. But then I guess you also would then want to get into monads, and that's a whole kettle of fish. That's, that's a next year project. Yeah. I think, I think it's definitely has, like, it's full up to the point it gets to, like, I think it has all the features it should have. I like the stuff we put in the prelude because we have like sort of basic functions. We've got good examples and we also have either maybe a list.

Me

[00:02:25] Yes. I just want to do this real quick. I'm going to copy the prelude into the main bit just so they can see it. Fair enough. So they can just kind of stare at it. And then see if you guys, what? Your V key is weird.

Amos

[00:02:43] It is, it's a bit funny.

Me

[00:02:45] You have to zoom in a little more. Oh, that's too far. Yeah, so this is the cool. Is there anything else you wanted to add about the tool?

Amos

[00:02:57] I think nothing in particular. Mostly there are practical things that I would say could be tweaked. Just having one local is a bit restricting. It would be nice to be able to define more complete programs and save them so that you can jump between them a bit. There were times where I was like, it would have been nice to jump between stuff, but I was stuck with doing it in one place, and we lost stuff at times, which was like, isn't great. It happens a lot. So getting that figured out would be a good, I think, sort of fix that would be worth quite a lot. Unfortunately, yes.

Me

[00:03:32] It would require doing more JavaScript work.

Amos

[00:03:34] Alas.

Me

[00:03:35] Alas.

Amos

[00:03:36] But yeah, I think I like it overall. I think it's a good approximation of Haskell, and it's like, Haskell-like in a lot of ways would be useful, but I think some of the differences are nice. I quite like match, because I think it gets a bit more directly at what pattern matching is doing, like that it's a specific operation and not just like

Me

[00:03:54] - I don't have implicitly matching, that's not a thing. Yeah. Haskell has case, which is the same as match. Case isn't quite the same, because case, you can do Boolean conditions as well. Yeah. I think that's just pattern matching. Oh no, that's guards. It's different.

Amos

[00:04:08] I would say-Guards are different. Guards are different to case, yeah. There's a-I think-For me, the fact that you only have this, you don't have implicit matching, is, I think, useful because then it's very clear when you're doing matching. For example, a thing I think you don't necessarily get from just using matching and guards in Haskell is that when you're doing guards, for example, you are still matching a pattern of true-false. You just evaluate down to it. I think match here does that a little bit. I think it's a little bit clearer, I think, than Haskell. It's really useful in Haskell. But if you haven't used a functional language before, I think it's, to me, I think it's nice to have match be sort of this explicit thing where you are doing matching. And then I think it's a bit clearer than what pattern matching actually is and what its requirements are, I think. So how did you find the editor to use? I think the editor was, other than the notable bug with whatever was going on with that indent, I think it was nice to use.

Me

[00:05:18] It's the same under the hood as VS code.

Amos

[00:05:22] Yeah, I think it's nice it zooms like it scales and zoomed like when we zoom in the browser it scales well, which I like. Line numbers are very useful.

Me

[00:05:33] What do you think about the way it does line breaks? For example. Do you think it would be better if there was a horizontal scroller?

Amos

[00:05:41] No, I think that is the ... It's a great question. I think I prefer that. For a demonstration, I think I prefer that because ... If it was my own code, I like horizontal scroll because I don't get confused about what's on the other line. But I would much prefer to just have all of this in there rather than have to go back and forth. I like that I can point to the entire thing. It's quite easy. It's like this. It indicates that it's a line break. I think that's fine. Okay, good. Okay.

Me

[00:06:18] Cool. I can come back to you. Cool. Yeah, I'll just go kind of along the line, mainly because the microphone is I'm not sure how good it is, so it's probably good to just start with one at a time. But feel free to jump in if you guys have anything to say, but I'll get to you guys. Yes. So, how did you understand the tool, do you think?

PARTICIPANT 1

[00:06:44] Yeah, it took me a little bit to figure out that the box at the bottom is what's changed, and then the box at the top is the entire expression after it's changed. Sure. But that's helpful to have there, I just think it's. You know, worth explaining at the start. Yeah, I liked; this is what I do in my head basically when I'm looking at Haskell, so that's, it was good. Another thing I think would be helpful to have is not having the ability to not step through a function call that's inside another, you know, if you're evaluating repeat take. I know, that's really hard.

Me

[00:07:20] Yeah. That would just require a bunch more UI stuff. That's on my list. I very much appreciate that you're bringing it up as well because it validates it being on my list. But it won't happen, I'm afraid. That's fair.

Amos

[00:07:33] Bigger and different sized jumps. I want to just do this.

Me

[00:07:36] Yeah, but that would require having different kind of boxes pointing, saying, okay, do that bit, you do that bit. It would just be complicated UI-wise. In the back end it would make no difference.

PARTICIPANT 1
[00:07:51] I was just imagining the same as free choice except with multiple buttons at the top. Except one of the buttons is just evaluate entire function when there's a function application.

Me
[00:08:04] So if you've got a function application.

PARTICIPANT 1
[00:08:06] So for example, when you have.

Me
[00:08:08] So I guess I could have another option saying apply and fully evaluate.

PARTICIPANT 1
[00:08:12] Yeah.

Me
[00:08:13] Okay.

PARTICIPANT 1
[00:08:14] So instead of take 5, repeat 5, and then expand the repeat, expand the repeat, expand the repeat. You could just say repeat and then expand the entire repeat.

Me
[00:08:23] Do you think that would be useful in, oh we didn't really look at free choice mode.

Amos
[00:08:26] I think it would be useful in lazy for sure because like there were definitely times where it would have been nice to like be able to just be like hey we see how this subfunk we've already seen how like repeat and take work we'd like to just see how this works while still being able to use repeat and take in like a useful way. So just being able to say, do the repeat thing and then let's see what happens would be good. Sure.

Amos
[00:08:51] Interesting.

Me
[00:08:52] That's a really good idea. Yeah, thank you. So how do you find the tool to look at?

PARTICIPANT 1
[00:09:01] I think it looks good.

Me
[00:09:04] As a demonstration tool, do you think this is kind of laid out well enough for you to

be able to follow along.

PARTICIPANT 1
[00:09:15] Yeah.

Me
[00:09:17] What do you think about the editor from the looks of it? I don't mean to use, I mean to watch.

PARTICIPANT 1
[00:09:24] I mean it looks like a code editor. I would like syntax highlighting. I think code is always so much easier to read with some colours. Yeah, so I don't have Massive opinions on the editor.

Me
[00:09:37] Yeah, so would you agree with Amos that it looks familiar?

PARTICIPANT 1
[00:09:39] Yes, that's true. Yeah, I might recommend, I'm not sure if switching the flow direction of the right-hand side.

Me
[00:09:51] Yeah, but then you'd have to go from the buttons at the top to the, or I could put the buttons at the bottom first.

PARTICIPANT 1
[00:09:59] Also, I'm not sure how that would work on a webpage, but.

Me
[00:10:03] It could, but you'd probably have to scroll all the way to the bottom to see things happen. It would be frustrating. Yeah. Maybe. So what effect? Why would you do that?

PARTICIPANT 1
[00:10:15] I'm not sure. I think I'm just used to things appearing at the bottom.

Me
[00:10:20] So would you say the button could be at the bottom and then things would appear top to bottom?

PARTICIPANT 1
[00:10:25] Yeah, almost like that. Because I suppose I'm used to writing or writing code and then having the changes appear at the bottom.

Me

[00:10:32] Sure. Okay, top-down, I guess, could make sense, yeah. Yeah, okay.

PARTICIPANT 1

[00:10:39] Also, maybe a label in the changes box that says what it is, just so you don't have to explain it?

Me

[00:10:49] Yeah, the problem is it appears all over the web page. I'd much rather have it explained in a separate little help section. That's true, yeah. Which is what I'm going to do.

PARTICIPANT 1

[00:10:57] Is there any meaning behind the star?

Me

[00:11:00] Yes, so a star means 0 or many. So it could actually be, so some of these steps are actually not reductions. Because if you just substitute a label, for instance, say if I have x equals 5, turning, going from the expression x plus 1 to the expression 5 plus 1 is not a reduction. They are identical. Again, you're not doing any computation. That's not a reduction. The only reductions are applications.

PARTICIPANT 1

[00:11:35] I get you mathematically, but I think visually the star confused me a bit. Whereas, if we're going from x plus 5 to 5 plus 5, I think an arrow on its own would still make sense as a student.

Me

[00:11:53] I think it's something you should get used to. I think it's useful for correctness. That's true. I think it's good to be correct because then the teachers will like the tool more and then they'll more like to use it. And some of them actually are more than one step as well.

Amos

[00:12:05] Some of them are zero, and some of them are like two or three. Yeah, some of them are two or three steps.

Me

[00:12:10] So let's say if you're applying 'if' to all three arguments at once, that's four steps. That's the substitution of 'if' for the lambda expression. If you go into free choice mode. Also, it encourages people to ask questions like this. That's true. I guess, because it is the mathematics leading. See, you can just replace this with the lambda, which is what it actually is under the hood.

Me

[00:12:37] And then, so that is a zero step. But then, if you see this, this one here, where it applies both of them, is two steps. This one here, where it applies one of them, is one step. This

one is. zero steps, because it's a substitution. So there are different numbers of steps, even though they kind of feel equivalent, I guess. Ah, yes, it makes more sense in free choice mode, where you have Because these are actually just lambda functions with syntax sugar. All I'm doing is I'm labelling a lambda function, but I just put the x before the equals. It's treated exactly the same. That's pretty standard.

Amos

[00:13:21] That that is just syntax sugar for a lambda expression I think I would add it to the list of things to explain in the list in the list yeah yeah this is what this is by the way because it's it's obviously something you would have seen before yeah I've never seen an arrow with a star after it no that's in time maybe in um PLC next year there might be an hour reduction i guess not no no that's computation not reduction but they do uh you'll do Because there are two uses for this tool.

Me

[00:13:51] A demonstration tool like this, where Amos is here to explain those kind of things to you. Or, as an individual learning tool, where I guess I'd need a help menu.

Amos

[00:14:04] There would probably be some workbook designed around it as well. Like, if it was for a-Yeah, you could have a workbook around this kind of thing. But it does that explanation, I would say.

Me

[00:14:14] Yeah, okay. Is there anything else you wanted to-No, that all looks good. Oh, I didn't ask about the language. Oh, yeah.

PARTICIPANT 1

[00:14:26] Language is fine. It's Haskell, but with a couple of limitations. I think that's perfectly all right.

PARTICIPANT 1

[00:14:36] I kind of, I both love and hate the list, the lack of syntactic sugar for list, because it makes literal lists.

Me

[00:14:44] I'm going to have it in the settings menu where you can toggle it.

PARTICIPANT 1

[00:14:47] Well, it makes literal lists really hard to read, but it makes the types so much clearer. Having to interact with that rather than just going, that's kind of an array. I'll have it in a set explaining where you can toggle it on and off. It really explicitly forces you to think in the Haskell way.

Me

[00:15:07] Cool. Okay, that's really cool. Thank you. I'm going to move down the list now. Hello. Hi. Yes, so. How did you find this lecture in the verticals?

PARTICIPANT 2

[00:15:20] I really, yeah, no, it was very helpful as in, like, I really like what the tool does, like it shows like what you can see because that's basically what I do when I'm like debugging code but um it takes like a long time to do it like by hand and to go through everything and sometimes it's like wrong if you do it yourself so this is like a good way to automate it um I really like the highlighting in like what changed because especially when like you do fold R on take five or something and then there's like two matches, so to highlight that this like you created a whole new match thing, I think was nice, okay, nice, yeah. I don't really have much to say, like we covered a lot, yeah, okay, yeah.

Me

[00:16:03] So the language itself, oh yeah, what.

PARTICIPANT 2

[00:16:06] Do you think um it took a bit to reconcile like pattern matching in Haskell and like the match here but I guess like this is That's because we have the background in Haskell.

Me

[00:16:16] This is meant to be teaching any functional language. This is called simple functional language, SFL. It's meant to be deliberately generic.

PARTICIPANT 2

[00:16:26] So it's more versatile, I guess. So it's not Haskell syntax. Besides that, I don't really have too much to add, I guess.

Me

[00:16:40] Do you think it's visually appealing?

PARTICIPANT 2

[00:16:44] Yeah, maybe the red is a bit hard to see, but that might just be the screen.

Me

[00:16:50] Yeah, the screen or the light, but I need to add light mode.

PARTICIPANT 2

[00:16:53] Okay.

Me

[00:16:53] Or to make it brighter.

PARTICIPANT 2
[00:16:55] Yeah.

Me
[00:16:56] Yeah, definitely. Okay.

PARTICIPANT 1
[00:16:58] So you say it's not Haskell syntax, but devaluing negative one.

Amos
[00:17:03] What do you mean? Negative one is about, you can do that. Yeah. You can do negative one.

PARTICIPANT 1
[00:17:10] No, you don't have to do that.

Me
[00:17:19] Very good. I mean, minus one is just a Lexer construct. It will just be minus one. Yeah. Yeah, so the language itself, you find it difficult to reconcile Haskell.

PARTICIPANT 2
[00:17:35] I mean, at first, but after, it's It makes sense.

Me
[00:17:40] Oh, then I asked about visually appealing.

Me
[00:17:46] You said you wanted to be brighter. Are there any other changes you can think of visually?

Me
[00:17:57] Literally anything. It doesn't matter. Or nothing. It's fine.

PARTICIPANT 2
[00:18:02] Not really much because I like that it's separate. Text editor here. This is. shows what it does. Yeah, there's not really much there, I think, for me.

Me
[00:18:16] Thank you very much. I mean, let me know if you think of anything else. I'll just give you another line.

PARTICIPANT 1
[00:18:20] This might be an impossible request, but the syntax highlighting would be very good in the editor. Will it be possible to have syntax highlighting? Yes, it'll work.

Me

[00:18:29] Thank you for validating me. I'm working on syntax highlighting. And I'm probably going to have line numbers on this, so when it splits into multiple lines. That would be good, yeah. Line numbers and syntax highlighting, but it'll be the same color as the background if I put it like this, and it'll be not editable. But yes, thank you for validating me. That's very useful for my evaluation. You're good. Cool, I will move down the line then.

PARTICIPANT 2

[00:18:58] Okay.

Me

[00:19:02] Hello. So how did you find this session, lecture session?

PARTICIPANT 3

[00:19:07] Ah, I liked it. I honestly really liked the match being, like, write match. Yeah. So the explicitness was good. I really liked that. It made it obvious to me in a way that it wasn't necessarily before. This is much easier to learn about pattern matching with than Haskell. Yeah. I, like, as a minor visual thing, possibly, you know how, like, you have the blue bit of the bit you've changed on the right-hand side? Part of me, like, and you've done the replacement of the list of squares to this thing that's then blue. I don't know, for whatever reason, part of me feels like maybe there should be some blue in that box to show that that's the change. Yeah, the blue on this side as well. Yeah. Because it's this one. Yeah. Maybe.

Me

[00:19:46] That could be done. I could make it like a.

PARTICIPANT 3

[00:19:49] I also would agree with PARTICIPANT 1. I don't know why, particularly in my head. I want it. I would prefer the other way up.

Me

[00:19:55] I didn't even consider that.

PARTICIPANT 3

[00:19:57] I don't know why it felt, it just felt a bit weird to me, and then I was like, fine, but.

Me

[00:20:01] Yeah, it's I guess this way you've got to look at the numbers, whereas the other way it would be intuitive.

PARTICIPANT 3

[00:20:06] Um.

Me

[00:20:07] It just feels more like writing stuff.

PARTICIPANT 3

[00:20:09] Yeah, I'm not sure, I just kind of felt, yeah. That was very messy. Oh, so could I just ask you a question? You know when, like, we talk about, like, you know the main, right? And, like, lazy. If you, say, had a different thing, not called main. on laziness of matching. Like, is this only doing this, like, uh, sorry, I'm not wording it very well. It only evaluates main. It only evaluates main. So if you had another thing just written, like, bob equals. If you didn't have main, it would be a pass error. And it would never.

Me

[00:20:38] No, it would just be an error saying you need main.

PARTICIPANT 3

[00:20:41] Okay, but if you wrote, like, above that, like, uh, like, if you have some other variable name, like, let's say type list A that's, like, repeat, take five, repeat five. Yeah.

Me

[00:20:52] That wouldn't get evaluated until it was like used in main let's say it wouldn't like go through no okay cool um these are literally just labels and they're actually unnecessary for writing a program you could just copy paste this in here for the definition main is the program itself they are only used for recursion but that could be done anyway with fix um which yeah there are ways around that Fixed point combinators Like the Y combinator is a fixed point combinator You might have heard of it So no it wouldn't This is the program These are just supplementary I guess I could make that clearer why they're different things Yeah that makes sense That would probably be something in like a little help menu Okay Cool. Because I wanted to show you, I'll do one more; I'm just going to get this ready first. Do you have any more questions, comments, concerns?

PARTICIPANT 3

[00:21:59] Uh, no, I mean the only thing was that like I don't really know what I would like is I really, really liked it kind of going through I felt like everything at the start and then when we got to some of the more complicated examples, yes. But I didn't I also don't know what so like like Yeah, but I wasn't really sure if I would then hate that at that point, because I like to go for everything, so I don't actually know.

Amos

[00:22:23] I think having a middle ground, like a setting, that lets you do the similarly trained evaluation, but skipping steps that you don't need, that you've already figured out. Because ideally you should, if you build up your understanding.

PARTICIPANT 3

[00:22:52] I really liked the, like, well, because we don't have the constructor we had in Haskell

for, like, lists. Abundantly much clearer to me what was going on.

Me

[00:23:04] So you preferred it without fancy list syntax?

PARTICIPANT 3

[00:23:07] The only bit I didn't prefer is, like, when we were then doing functions with lists, it might be nice if you just had a couple of places that, like, such that it didn't, like, because we were doing, like, take five, repeat five, but you didn't want to write, like, list five, list, like, cons that, cons that, blah, blah, blah. Like, you could just have a couple, like, default things that are just, like, that you have written above somewhere. So you could have a default list of 1, 2, 3, 4, 5. So you could just write that value in.

Me

[00:23:34] Yeah, yeah, yeah. I guess I could have that as, I've got range. The range does that, I guess. If you use range, I think range solves that. But I could have some, yeah.

PARTICIPANT 3

[00:23:51] Because then you get the way I prefer it with this list, this cons and the nil. But also you don't have to write it out for five hours as the teacher.

Me

[00:24:01] Yeah, okay. Okay, fantastic. Thank you. Let me know, feel free to interject if you've got other comments. I'm just going to move on. Hello. Yes, thank you very much for coming. How did you find this session?

PARTICIPANT 4

[00:24:14] Actually, I find it quite interesting, and I actually really love the functionality of this app. That's what I appreciate. I do have one suggestion. I think you can create an auto-complete function for a code editor. Sorry, say that again? You can create an auto-complete.

Me

[00:24:38] Thank you. Yeah, that's just a setting I need to turn on. I could do that. Auto-complete. What do you guys think about auto-complete? Yeah. Yeah?

PARTICIPANT 3

[00:24:45] Like a drop-down, like a VS Code? I mean, I would have thought it'd be nice, but if I was watching it as a lecture, I wouldn't really mind. I mean, it's like a lecture, so I don't know. For me, it wouldn't make any difference to me. It's not meant for you to write big programs in.

Amos

[00:24:59] I can see it being handy if, like, I don't know, if you want to quickly, if you've written your function and you just want to write out an example, you can write an example out of it quicker. Specifically, I'd be fine when it came to the function to be defined, it was a little bit like

because of the misconstruction. And because of, I guess, not having auto-read, it was like that feels a bit more friction when using the function to define, so I can see that being one way to help with that. Yeah.

Me

[00:25:27] Okay, very cool. I mean, yeah, so, yeah, I mean, feel free to repeat what other people have said as well, if there are things you liked or didn't like, because that's all very useful to me, even if it's all a good set.

PARTICIPANT 4

[00:25:39] Do autosave for the code editor. So you can just save it to the local storage. Yeah, that's what I'm doing already, but it's broken for some reason. That's how it should work. Also, I find sometimes when you're writing big programs, it's quite difficult to track the conversion between two steps. So I think it would be good to have some kind of description when you make a transition between two of the steps.

Me

[00:26:10] So, do you mean other than this, I've got this, which says what it's doing, do you mean like after it's already done, say here, I did this?

PARTICIPANT 4

[00:26:20] Yeah, I think, yeah, because it's only showing once. Yeah, so it's only showing and it's losing the information.

Me

[00:26:27] Yeah, okay, that's very, I guess I just need to turn it to past tense, wouldn't I?

PARTICIPANT 4

[00:26:33] Okay, interesting. Is it possible that I think currently we evaluate each step one by one, right? Yeah. So it's possible that the user can just jump to a specific point of the program.

Me

[00:26:49] How would you specify the point? That's the problem. That's the problem with functional animation. You can't put a break point in. I kind of can put a break point in. I can have a special label called break that when the label break is evaluated, it stops. But.

PARTICIPANT 4

[00:27:08] That's the only kind of thing I can do in terms of keeping going until this point that's not really not really a great point but because now we're in order to jump into the end of the program, we have to evaluate everything right yeah so yeah I guess I was mentioning that earlier.

Amos

[00:27:34] Evaluate only this function. Yeah, they'll need to evaluate the conditional every time.

Me

[00:27:46] And I don't want to do that on the boolean. Because that would cause crashes if it's infinite. So I'm not sure about that. But I could do definitely break points. But you can put breakpoints in if statements, which is bad.

PARTICIPANT 4

[00:28:04] Very interesting. Thank you. And also have some UI changes. Yeah, absolutely. Any would be welcome. So I'll prefer the buttons to be on the header instead of at the center of the page. Okay. I think that will leave more correspondence in place for people.

Me

[00:28:19] I was going to show you, this is the designed UI. This is what I'm working towards. I will have an evaluate to completion button. So even now, with all those buttons, would you still prefer to have them at the top?

PARTICIPANT 4

[00:28:42] Maybe. I think so, because it will give more space for the code edit. Okay.

Me

[00:28:48] I guess maybe they could be toggleable, or I could have a hamburger menu, or something like that. Or getting them out of the way of the code editor. Yeah, that could be valuable. I'll have to think about that and do some experiments.

PARTICIPANT 3

[00:29:07] You could have it just like when you've got your cursor in the code editor.

Me

[00:29:12] Yeah, I'll just put a hamburger in there or something like that. Or when your cursor goes here and expands.

PARTICIPANT 3

[00:29:20] Cool.

PARTICIPANT 4

[00:29:22] I mean, that's really great. Thanks. I think you can also add a toggle so that the user can choose whether to have the code editor to be horizontally scrollable or just wrap around. Yeah, I guess I could have input settings, output settings, I guess. Great. I think that's it. I think it's a really good one.

Me

[00:29:43] Yeah, well, feel free to, well, thank you. I much appreciate it. Its available online at functional.kiransturt.co.uk bear in mind, do not rely on the tool because it may be broke because I'm very much in rapid development right now. This isn't even beta. This is like, but. It's not

stable yet. I'll have a stable and unstable version at some point. So don't rely on this if you ever want to explain something to somebody. But yeah, so feel free to use it. It's always there.

PARTICIPANT 2
[00:30:43] Cool.

Me
[00:30:45] I think that's pretty much everything. I'm happy with that. Does anyone have anything they want to add?

PARTICIPANT 2
[00:30:52] I don't know how useful it would be for a use case, but could there be a thing that's like, you know how when you have a browser and you can, if you want to make the code editor bigger?

Me
[00:31:03] Oh, what, like, yeah, you could drag this. Yeah, I know, I'm just terrible at CSS, but that's, yeah, you could drag this separator in the middle. Yeah, sure. You could and I should do that.

Amos
[00:31:18] Yeah, because then you can focus on editing and evaluating your own music.

Me
[00:31:23] But this is all kind of functionality-based. That's why I've got you guys here, to work on the user experience.

PARTICIPANT 2
[00:31:33] About the autocomplete you mentioned earlier, I guess it would be nice, for Liz especially, you have a thing that like, when you make a list it automatically has like 'cons', 'cons', 'nil' and you just replace like the things you want inside.

Me
[00:31:50] Yeah, then you'd have different length lists. I'm not sure about that. I'd have to do some fancy custom autocomplete rather than just like autocompleting labels. I'm not sure I've got time for that. But that's an interesting thing that could be explored. So, this project is likely to be, well, I'm hoping that one of the second years will take us on with the project. Somebody will do this.

Amos
[00:32:14] Or you can do something similar.

Me
[00:32:19] The functional programming project is going to be really fun. Does anyone else have

anything else? Oh yeah, I'll just show this again. Anyway, it looks pretty much the same as what I've got now. Do you think this looks nicer, or do you prefer the first one? I prefer this one. You prefer this? What do you guys think?

PARTICIPANT 3

[00:32:55] I am for the other one. I guess it depends if I want to be looking at the beam. This one has a how it works button and thing. It's a saving mode, which you probably do want. But I did quite like just the big buttons and not that many of them.

Me

[00:33:10] It's a shame to get rid of the big buttons. I like the big buttons. Really? I prefer more buttons because it justifies them being that tall. Yeah, because they're just ridiculously big.

PARTICIPANT 2

[00:33:18] Wait, for this one, where does the free choice and Lizzie go? Here. Oh, okay.

Me

[00:33:26] You can also add strict and other evaluation types. But that should be fairly trivial. But then we've got this evaluation step by step. It might be worth it.

PARTICIPANT 1

[00:33:37] Having an indicator that email strategy is a dropdown and not a button.

Me

[00:33:41] Yeah, I'll have like a little down arrow. Yeah, anyway, I think I'll call that a day because this is the allocated time. I'm going to stop recording now. Yes, we'll be out in a sec. Yeah, no worries. Cool. Yeah, thank you guys so much for coming. I'm just going to be in the cafe. I'm going to stop the recording. I'm just going to be in the cafe if anyone wants to chat about anything. Feel free to ask about implementation stuff if you're interested. But yeah, that's not really what I wanted to go on today. But if you guys are interested, I'll be in the cafe. I've also got a recording. Yes, can you please send it to me on Teams? Yeah. So message me. My name's Kiran Sturt. You can just find me on Teams. Message me if you have any questions or comments. Or if you have any concerns about your data privacy, I do have to mention that you can, until I've actually written the anonymous transcript, you can revoke your data.