**w03-01-UserInterfaceDesign-2024**

0:01  
Great.

0:02  
So today's lecture is on user interface design.

0:07  
So as we said last time, we're walking down 1 iteration of our life cycle.

0:11  
This could be waterfall, it could be agile or something else.

0:14  
We started last week with requirements definition, and this week we're looking at the first aspect of design of software, which is normally the user interface design.

0:26  
In future weeks, we're going to continue to walk down this until we get to the end, and then we'll start to talk about other things such as agile project management.

0:36  
OK, so the user interface is somehow connected to the user stories or user requirements.

0:45  
So when you're thinking about the software, you start off with either user requirements or user stories, or maybe you used user requirements and then stories.

0:53  
But then once you've done that, you've captured what is needed, you then start to think about what it'll look like.

1:00  
What will that interface look like?

1:02  
Now, it's a good idea to produce a user interface design early on.

1:07  
Don't wait until you think you've understood all the user requirements, because you can use the user interface design as a discussion tool with the people you're building the software.

1:18  
So you can have a bit of a go.

1:19  
You've got your user stories or user requirements, and then you produce your initial version of the user interface design.

1:26  
You show it to them, you talk about what you think this user interface should be and why, and hopefully they will tell you where your assumptions are not right.

1:37  
So the benefit of it is that you can even through discussion, describe how the software is going to function, and that will be more obvious.

1:47  
Often when you're working with somebody who you're building software for, they say, oh, I didn't mean that because they actually see it rather than the user's story or user requirement.

1:58  
So we can clear up misunderstandings between the developers and the users.

2:02  
What are we thinking or they're thinking?

2:04  
The other good thing about a user interface design is that we learn things that we didn't know when we started.

2:13  
So we discussed with the client and the client says, oh, but I need a button there.

2:18  
And you think, oh, that wasn't in the user requirements list.

2:21  
Oh, it must be a new one.

2:23  
So it's common that you produce one of these designs, you show it to whoever it is, and then you think carefully about are there any features in here that have been requested which are not in our list of user stories or user requirements.

2:37  
Now, if there's a feature or features that appear, the client wants, add them to the user interface design, but then remember to add them back to either the list of the user stories or user requirements.

2:49  
And then you can tell the client, you say, oh, thank you very much.

2:52  
The design has been updated.

2:53  
Here's the new version.

2:55  
And here's the new version of the user stories with those extra ones that we discussed in our workshop.

3:01  
The other benefit of user interface design is it allows us to understand what the data will look like.

3:09  
Normally we talk about a data model as being a set of classes or data structures that are underpinning your application.

3:18  
So they could be data that are stored in a database, they could be data that are loaded into memory, or they could be data that are just displayed.

3:26  
Now once you start looking at a user interface design, you look at the read and write functionality of that user interface design and there will be some data.

3:37  
You may be displaying some values or maybe allowing some input of values.

3:42  
So once you've got the user interface design, you start thinking about the data model.

3:48  
And in some cases you might want to ask the client straight away about the data model to see if you've understood it.

3:55  
So this picture came from the first lecture.

3:58  
This is a little application with two buttons.

4:02  
So the idea here is it could be a wrist mounted application or something like that.

4:07  
It's got 2 buttons, that's our user interface, and there's some data inflow and data outflow, and then there's a little display that tells the user that something's working at some time or another.

4:21  
Now when you produce a user interface design, you need to think about how it will be used.

4:27  
So here I've written down, read some data, handle the inputs well here, OK, pressing the buttons, processing the values, outputs and write data.

4:36  
So this is a brainstorming few bullet points.

4:40  
When you're discussing a user interface design, you might want to tell somebody how it's going to be used.

4:46  
So you for example, have an initial screen here, maybe this user interface is just blank and it would say booting screen.

4:54  
And then you're going to have to hold down a button and it takes you to another screen.

4:59  
Often you present several pictures of user interfaces to show whoever it is you're talking to how it's going to work.

5:08  
Right Now, before we go into user interface design in the graphical sense, I'm going to mention a few types of different user interface.

5:17  
User interfaces aren't all graphical.

5:20  
There are many other types which are not graphical.

5:24  
The most basic is a library or application programming interface.

5:29  
The idea here is your software needs to be able to plug into something else.

5:34  
Something else is going to call your software, send it a function call.

5:38  
And so you need to describe how to call your software.

5:42  
Imagine your software as a box of functionality and on the outside you're going to allow some inputs and outputs which are in fact an interface calls.

5:53  
So they're going to send a message, call one of your functions.

5:58  
Now if you're not connected directly to another piece of software through compilation or something else, then you may have a command line interface.

6:07  
Now you may say, well nobody uses the command line anymore, do they?

6:10  
They it's all voice activated or graphical user interface.

6:14  
Well, actually the command line interface is still very important because if you are repeating and I'll repeating an action several times, having it in a command line interface is very useful because you can then write a programme or a set of scripts which are effectively programmes to call the command line interface.

6:35  
So you can automate a process if it has a command line interface.

6:40  
Now for some applications, the the customer might only want the command line interface.

6:46  
Perhaps it's for a System Administrator or somebody else who who doesn't want to have to click, click, click around an application.

6:53  
They just want to run something and have an answer.

6:57  
Now when we build one of these, we still have to think about how we build it.

7:01  
We still have to think, do the command, do the commands make sense and do the answers make sense.

7:10  
Another type of user interface is a Restful web service or a web service.

7:15  
The Restful part is really a discipline about how it deals with state.

7:22  
So if it isn't Restful, then potentially between the requests the server is storing some state information so that if you ask it the same thing twice it gives you a different answer.

7:33  
Restful is essentially a set of disciplines about how the web server should react, which is potentially a bit more robust than storing the state in a funny way in the web service.

7:48  
Anyway, for a web service you have several Http://methods, so these are the common for you've got POST, GET, BOOT, and DELETE.

8:00  
There are others which are not so common.

8:02  
So the idea here is POST means that you're going to create something in the web service, so you're sending it some data.

8:09  
GET is you asking the web service for something back.

8:13  
PUT means that you are wanting to update some data in the web service.

8:17  
So you send it something and you tell it what to update and delete is obvious, you're just asking it to delete something.

8:25  
Now other than a web service, which is actually a defined interface.

8:29  
In that you can look up how web services should behave.

8:33  
You can have network services that are not web services.

8:36  
For example, if you've used any database like for example MySQL or Oracle, they have a server which is particular to them, so you have to connect with a client to that particular server.

8:51  
There are other types of services as well.

8:53  
You'll find there's an NTP server which it gives you the time of.

8:58  
If you're connecting to it and you've got a computer you want to know the time, you can send in a message or you can make your own server.

9:04  
You can basically decide what the protocol should be.

9:08  
Now in computer speaker protocol is a set of rules about questions that you can ask the server and answers the server will give you.

9:17  
And if you so wish, you can make up your own protocol from scratch.

9:23  
Graphical user interfaces are what you'd normally think of when you think of a user interface.

9:28  
It's not always what you need, but we will obviously discuss them.

9:32  
Now, they are different depending on what you're dealing with.

9:36  
You could have a large screen, such as a laptop or a desktop computer, or you could have a small screen, such as a watch or a mobile phone.

9:45  
Now, how you lay out the data or the information on that screen is going to be different, whether it's a big screen or a small screen.

9:53  
You need to think about, does my application need to run on both?

9:56  
What libraries perhaps at some point should I be using to allow that to happen?

10:01  
You need to think about is it going to be readable?

10:04  
I know I walked around the lab and some people had this very high resolution on the laptop screen, which is fine for their eyes, but it might not be OK for other people.

10:14  
And that's something you always have to bear in mind with these interfaces.

10:18  
You're designing it for everyone, not just you.

10:23  
OK, so user interfaces, as I've already said, we can use them to capture user requirements.

10:30  
So you do the best you can with your user requirements to start with and then you generate a user interface design and you use that in discussion and potentially you collect more user requirements.

10:42  
Now the type of user interface is a non functional requirement.

10:47  
Somebody might say we must have a command line.

10:49  
Yeah, OK, fair enough.

10:51  
It's a design constraint.

10:53  
So you're going to end up building several features to fulfil one particular user interface design.

11:01  
And then the features of this user interface are fulfilled by either completing a user story or a functional user requirement.

11:09  
Now, as I've already said in the lab a few times, user stories never map to non functional designs.

11:16  
User stories are always talking about a feature, something that you are going to build.

11:21  
Whereas non functional requirements are needed whether you're using agile or anything else because they describe the design constraints.

11:30  
OK, so we need to understand what do the users want?

11:33  
What kind of hardware should it run on?

11:35  
Is it a small display?

11:37  
Is it a big display?

11:38  
Is it a mobile phone?

11:39  
Is it a custom display?

11:42  
It could be in an aircraft.

11:44  
We need to know the size of the screen, so that's the dimensions of the screen across the top, down the side.

11:50  
We also need to know the number of colours.

11:53  
Now for some low power displays, they are monochrome because that is going to be a lot less in terms of power.

12:01  
The other benefit of monochrome displays if you know anything about Kindles and Amazon is that you can have so-called I think it's electric ink or something.

12:10  
Basically it's a screen display which is not actually lit.

12:15  
It's functions much more like an old star LCD calculator.

12:19  
Now the benefit of that is you can use it in bright sunlight.

12:22  
So that is a a useful feature which might mean that you are only dealing with monochrome.

12:29  
And then you need to ask how will it be used?

12:32  
Is the person who's going to be using this wearing goggles which will prevent them from looking at these interface?

12:39  
Is it going to be used in some particular lighting conditions in the dark and the light?

12:44  
And yeah, environmental, environmental conditions, it could be used underwater.

12:49  
So what happens with water is if you hold a device with glass on top at a certain angle, it becomes a mirror.

12:56  
You'll notice that if you hold your watch underwater in the bath, yeah, you have to beware of how it's going to be used.

13:04  
Now, there are certain design principles, and I will come to these sort of more qualitatively towards the end of the lecture, but basically it needs to be logical to people in general, to your users.

13:20  
It's no good producing a beautiful interface.

13:22  
It's very complicated, got lots of lovely features, but it's just nothing like anything anybody's seen before.

13:29  
Now it might be novel, might be an interesting idea, but the risk is it will be too hard for people to learn.

13:37  
So what we normally do is we set up a user interface that has certain features or ideas that are similar to other user interfaces.

13:46  
You could still have something that's different, but it's similar.

13:49  
And then the benefit of that is that if you use something else, you start guessing where features are and it becomes much easier to use.

13:57  
One such example is Control C, Control V That's long been the copy paste on Windows and it's been picked up by a lot of different applications in Linux.

14:09  
On the command line, I think it's still Control Shift C, which is a bit annoying.

14:15  
It's because Control C means something else.

14:16  
But some of the editors in Linux are now Control C based, control V, So it's a good idea to think, do I need this to be different?

14:26  
No.

14:27  
OK, right.

14:28  
Let's just keep the standard because that will be easier for the users.

14:31  
Something else you may have noticed is that you have dialogue boxes which ask you a question, should I do this or that?

14:37  
And often we have an OK or cancel button.

14:41  
You could say yeah, or I'm not really sure button.

14:46  
And then people will like double take.

14:48  
What does that mean?

14:50  
You're better off just using standard naming for standard things like if you see I could be save rather than what could it be keep for later, right?

15:01  
What does that mean?

15:02  
So yeah, try not to be cool with your naming of buttons.

15:06  
Just go with something, somebody else will realise what it is.

15:09  
The menu systems again should have somewhat similar layout.

15:14  
If you think about applications on your Mac or on Windows or Linux, the menu naming at the top between applications is similar, right?

15:23  
You might have a file menu or an about menu.

15:26  
So if you hide the feature under some other top level menu, that's going to make it harder for the user.

15:34  
Now the last thing is you need to think about how the user interface is going to be used and what is going to be used often.

15:41  
The thing that you're doing often should be easy.

15:44  
It's no good having to take 3 or 4 clicks to do the thing you're going to do often, and two for the thing that you're going to do hardly ever, right?

15:53  
You need to reduce the amount of mouse action the user has to do to enact whatever they're doing regularly.

16:03  
OK, so word on application programming line interfaces.

16:07  
Application programming interfaces are such that you don't want to break them.

16:12  
Now, the example I gave you before was the Linux kernel.

16:15  
So Linus and friends are very careful that they don't break the Linux kernel, meaning if they've got to add new things, they add them, but they don't break backward compatibility.

16:25  
Now, the beauty of that is it runs on old computers and it keeps running, whereas Microsoft Windows probably just doesn't.

16:33  
Yeah.

16:34  
So you need to think carefully about your application interface, and you might want to leave yourself room for expansion.

16:43  
So you may want to pass in some kind of generic data object which is big enough for your future use.

16:49  
You may want to think carefully about what could be used, then define the application user interface, thinking into the future rather than what you just need now.

17:00  
You want to have some kind of logical naming.

17:03  
So for example, a function that says save data should save data, shouldn't remove data or do something odd.

17:10  
The function naming and the variable naming that you're going to pass into that application programming interface should be logical.

17:19  
The error messages should be helpful.

17:21  
I remember a funny interface that I used a long time ago that had a two layers on it.

17:29  
What it did was inside there was a process that was managing something that was running, but on the outside it checked the final state.

17:38  
So on the outside it said job successful and he said oh great.

17:43  
And then on the inside it looked and it said application failed, which to me is caused me to roll around laughing because you think that's a good design it nonsense.

17:55  
So the the error messages should be useful.

17:59  
You know, it shouldn't be something like unaccepted pointer reference, such and such line.

18:06  
You should tell the person what they've done if you can.

18:10  
Obviously, if sometimes you can't, you just say unexpected input data in this field or something like that.

18:20  
Now you need good documentation.

18:22  
Now that could be online, updated as your APIs updated.

18:28  
It could be manual pages.

18:29  
Now, if you're not used to Mac or Linux, you won't have seen a manual page before.

18:33  
But on Mac and Linux, you've got manual pages, which are great.

18:36  
They're accessible through the command line.

18:38  
But if you've got a command line feature, you can just say man, man, and then the command and you can Scroll down and find all the information.

18:47  
I've got a picture of one in a minute.

18:51  
You typically distribute the documentation with the software, so it could be compiled with the software and issued and it's automatically generated, often meaning that you put some comments in your code and then those are compiled into the documentation.

19:08  
Here is a manual page.

19:09  
This is from Linux.

19:10  
This is about the stracomp C function.

19:14  
It's been around for a very long time.

19:16  
It's for comparing 2 strings.

19:19  
And if you look at a lot of, look at a lot of C programmes, they use this one a lot.

19:23  
What's good about this?

19:25  
It tells you name, the function name.

19:28  
So this is a function.

19:29  
It tells you how to include it by including this header file.

19:33  
It gives you the format of the function call.

19:36  
And then it tells you about the function and it tells you about the return values of the function, what you can put into the function.

19:43  
It's great in terms of documentation.

19:45  
It's exactly what a developer needs.

19:47  
No chat, just simply what the function is, how it works.

19:54  
Here's another one.

19:55  
This is for a Restful web interface.

19:57  
This is using the Swagger UI.

19:59  
So with a Restful interface you can document it using Open AI.

20:03  
It's a standard.

20:04  
A Swagger UI will display that to you in a nice way.

20:08  
So here you can see it's used different colours for the different operations.

20:11  
So you've got a POST, PUT, GET and a DELETE and then it has here the the suffix.

20:20  
So imagine you had a pet shop and it could be www.mylovelypetshop.com/pet.

20:28  
And so pet here is you're posting a pet to that web interface.

20:33  
So this is how we typically document Restful web interfaces.

20:38  
And often what happens is that you tell the person what the suffixes are, and then down at the bottom here, you tell them about the data which are in Jason that can be passed the web interface or that might be returned from it.

20:51  
So you have a description of the user interface and the data that are sent to it or from it.

20:58  
Now, when you're designing graphical user interfaces, you don't want to go from nothing at all to a very complicated, beautifully designed, designed user interface in one go, right?

21:11  
You want rough sketches.

21:13  
Why?

21:14  
Because you can talk through a rough sketch on a whiteboard or a flip chart with a client very quickly, and you can quickly understand if you've roughly got it right before you start to design something in a more professional manner.

21:28  
So when I've done this personally, I have gone to workshops with people, customers, and we have had flip charts and we've drawn things.

21:37  
We've taken those drawings away.

21:38  
We've then turned them into wireframes, which I'm going to show you and talk about in a minute.

21:43  
And then we have come back with a slightly nicer picture and talked about them.

21:47  
We might communicate it over e-mail once we've got a better picture of what's going on.

21:52  
And we've used that to then improve our design before we've implemented it.

21:58  
So we often have what's called stakeholder workshops where we invite a few people, could be of maybe 5 or 6 of them, or you may have some kind of focus group with a select bunch of users.

22:11  
So we start off with no design and then we go to maybe a rough sketch and then wireframes.

22:19  
So what are wireframes?

22:20  
Essentially you can think of it as the stick man of portrait drawing.

22:25  
You just want to draw a simple outline of the elements, right?

22:29  
Where is the button, right?

22:31  
Maybe it's text on it.

22:33  
You put the boxes in the right places.

22:35  
You may not have any pictures on there.

22:37  
You may just have a little bit of text saying product image goes here or something like that.

22:43  
So you refine your design through workshops.

22:45  
We evaluate our design against other designs and design principles.

22:50  
We do that ourselves.

22:51  
The client won't necessarily do that for us.

22:54  
Occasionally clients do have intelligence and they know what they want.

22:58  
So for example, I was working on a device, it was wrist mounted and the client did know what they wanted because they had an existing one.

23:07  
They needed a new one so they could tell me about some of the concerns about that device.

23:14  
OK, So what can you do with a wireframe?

23:17  
You can indicate the functionality.

23:20  
You can, once you have created those wireframes, use a programme which will link them together.

23:26  
So there's no computer code here, but you can press on a button and it will take you to another screen of the wireframe interface.

23:35  
This takes a little bit more work, but it's very useful.

23:38  
You can distribute that by e-mail to a multitude of stakeholders and say try this out, is that what you wanted?

23:47  
Now it does take more time to implement this, so there's a cost benefit balance.

23:53  
If you can implement a simple set of wireframes and then build the application with minimal functionality, do that.

24:02  
That's cheaper.

24:04  
And if you have done it in a reasonably quick manner, you could then change your user interface design a little bit.

24:12  
You often don't want to go to a so-called medium or high fidelity design.

24:17  
So medium fidelity design is where we have taken our wireframes, but we've now made it much, much more like the final application.

24:25  
So we've put more work into animating it.

24:27  
We haven't written any code yet, but we've put more work into animating it.

24:32  
Now I can tell you from a commercial point of view that is very rarely needed.

24:37  
So don't do it unless you're sure you need it.

24:41  
You do need wireframes.

24:43  
You may need wireframes with links attached to them, but that's probably all you need.

24:49  
And then you could have a first version of the user interface with no functionality behind it.

24:54  
You just show the client what it'll look like in the final software framework.

24:58  
You haven't actually put in the code that answers the request when they press the button, but you can show it to them.

25:05  
If you do that, you could potentially get around not having a high fidelity or medium fidelity design.

25:14  
All right, what do we do?

25:16  
We've got with the wireframes.

25:17  
We have an initial version of the graphic use interface.

25:20  
We then encode that in a programme without the active code behind it.

25:25  
We can have the text box, the buttons, the menu, the images that can be placed on the screen, but they're not active.

25:32  
We could include some basic things like OK, click this.

25:35  
That happens Now when you're building software.

25:39  
It can be that the user interface design is being designed while you're building the software.

25:45  
Imagine you've built the first version based on a user interface design that you agreed with the client, and they want another version with some new features.

25:54  
So if this happens, you need to separate user interface design from writing the implementation, and you need time to write the code that goes behind a new piece of user interface design.

26:09  
So what you do is you agree with the client what it is they want before you guess how long it's going to take you to implement the new thing.

26:19  
You use the user interface design as part of your agreement with whoever you're building the software for.

26:25  
You essentially have to say, is that what you want?

26:28  
And at some point they need to tell you yes.

26:30  
And then you say, OK, we will build against that interface.

26:35  
We will build something that looks like that.

26:38  
Anyway, here are a few user interface design frameworks you can use.

26:43  
Figma is very popular.

26:44  
It's a web service.

26:45  
You can use it for free.

26:47  
I'm referencing this in the lab.

26:49  
It will allow you to design fancy looking user interfaces in mobile format or desktop format.

26:58  
This is another one.

26:59  
Mock Flow does the same kind of thing.

27:01  
It gives you a mobile and desktop layout.

27:05  
These applications normally have a tool palette to make your life easier.

27:09  
So if you want a text box you just click it and draw it and drop it in and you can have animation.

27:15  
The benefit of these web services is that you can provide your client with a link to the web service so they can go ahead and play with the user interface without having you having to send them a copy of some animated PDF.

27:28  
So here's a bunch of other ones.

27:30  
Adobe XD Balsamic I've used.

27:34  
I've used this for commercial projects.

27:36  
It's free for 30 days, but then after that you have to pay for it.

27:39  
It's not very expensive, but it's a good tool.

27:41  
Pro 2 IO, Marvel Sketch, Chevrolet pencil, Microsoft Visio.

27:46  
I've seen used.

27:48  
The pro version is expensive, so it's a balance.

27:52  
What are you doing?

27:53  
Is the tool palette going to help you?

27:55  
And typically speaking, a development team will prefer one of these tools or another based on certain set of reasons.

28:04  
Now, when you're designing a user interface, a graphical user interface, you need to think about how it's going to be used.

28:11  
So when we're thinking about these, these are described as workflows.

28:16  
So the first workflow is a sign up.

28:19  
So somebody might need to sign up to use your interface.

28:22  
Oops.

28:24  
This only happens once.

28:27  
Another workflow that could happen is that you appear at the landing page.

28:31  
So this is the first page the user sees.

28:34  
And then the landing page could allow customization.

28:37  
It could allow somebody to drag around some of the display elements on that page.

28:43  
Customizable dashboards are a common thing with web interfaces.

28:47  
You may have some sort of view, edit, or save features that you need.

28:52  
So somebody might need to list things in a table, Scroll down, expand the features, see inside.

28:58  
These are common things that we need from a user interface.

29:03  
Essentially what we're trying to do is we're trying to think about how the user is going to use the graphical user interface and therefore create something that is going to be easy for them to use.

29:14  
We also need to allow them to verify what they're doing is what they wanted to do.

29:19  
We don't want to have a button that if they press it, horrible things happen and there's no Are you sure that would be a bad idea?

29:27  
Now you can go one step beyond talking to somebody.

29:31  
You can actually use tools that will analyse how people use your user interface.

29:38  
So this one, this is called Hot Jar and the idea here is that you give it to user interface that you've built and it then will allow users to use your interface and it will record where do they touch on the screen, where are they pausing their mouse often or their input often.

29:57  
So here you can see there's a lot of input at the top here and at the bottom.

30:03  
What does that tell you?

30:04  
Perhaps the whatever it is they're doing down here needs to be at the top.

30:10  
You don't want to have somebody using a mouse and scrolling up and down, up and down, up and down the screen.

30:17  
Mice are really bad for your wrist.

30:19  
So if at all possible, keep the length of scroll shorter.

30:24  
So if you have to click at the top then the bottom and the top, that's a bad idea.

30:29  
Often you may have a save button.

30:31  
This could be save at the bottom of the the screen, which is OK because you're basically asking the user to check everything they've done before they click that button.

30:41  
So you need to think about how people are going to use it in terms of time spent with a mouse.

30:47  
Now the way the human brain works is it sees edges of shapes rather than the actual shape.

30:55  
We have some perception based on the colour as well.

31:00  
We don't want to have colours that are very similar.

31:03  
Now we have to remember that we are not all the same.

31:06  
Even if our eyes detect all colours of light.

31:10  
Some people see red slightly differently.

31:13  
Some people obviously have red, green, colour blindness, and that's an issue that you need to be aware of.

31:19  
When I published papers on scientific results, I've always printed out my graphs in black and white as well as in colour.

31:28  
If it doesn't hold the same information as in I can't tell what's going on with black and white colour, that's no good as a publication.

31:36  
For one, yes, somebody could be red, green, colour blind, but for two, somebody might not have a colour printer and I don't wish to disadvantage them.

31:45  
So yeah, we may wish to evaluate it with different colour philtres.

31:50  
We want to restrict the density of information.

31:53  
It's no good having all the information on a user interface, but it's just so much that you can't see anything.

32:01  
It's a typical can't see the wood for the trees.

32:05  
You don't want that.

32:06  
You need to have a white space around the information you want people to see.

32:13  
Avoid a so-called wall of text.

32:16  
Try and lay things out so that the eye can see the important details.

32:20  
You can have several pages of graphic use interface.

32:23  
You don't need to squeeze it all onto 1.

32:27  
Your features on the graphic use interface should be grouped so that they make sense together.

32:33  
You shouldn't have some features about one thing in one place and then one feature that's about the same thing somewhere else on the same page.

32:43  
It should be somehow grouped so that the person who's looking at it can see it's all about that data object.

32:51  
We want to think about the fonts and the sizes carefully.

32:54  
In these days, it's a really useful thing to allow somebody to scale the size of the fonts.

33:01  
Even though you may have a big screen, you could have somebody with visual impairment and they need to make the font really big.

33:07  
And if your application does funny things when the font's very big, that potentially is not a good idea.

33:15  
We will want to think about increased line spacing.

33:17  
So this is the space between the lines of text that helps people to read text.

33:24  
You may need to think about dyslexic font options.

33:27  
So again, if you have a Kindle, Amazon Kindles have a dyslexic font option.

33:33  
A Dyslexic fonts are typically heavier at the bottom.

33:36  
It prevents people from seeing a an N or AW flipped the other way up.

33:43  
Some people's eyes work like that, so you're heavier at the bottom, easier to read.

33:48  
Sometimes it's serifing as well.

33:50  
That means the little lines at the end of the font sometimes helps people read the characters.

33:57  
So we may wish to have a library in our software.

33:59  
So this is a library somebody else has written that allows us to have different font options.

34:05  
An important thing to think about.

34:08  
Simple, avoid clashing colours.

34:11  
Now on the left here I've gone for a very bad set of colours to make the point.

34:17  
Green is a very bad idea, especially light green.

34:22  
It's hard on the eyes, the eye, it receives light green light much more readily than any other colour.

34:30  
So our eyes, if you have the same intensity of green and red, we see green more brightly.

34:36  
So you want to have dark green if you're going to have green at all.

34:41  
And you want to definitely avoid clashing colours.

34:44  
So red and green, blue and yellow, don't put them next to each other.

34:48  
By all means have them, but don't put them next to each other.

34:52  
Green in particular is a very bad idea on projectors.

34:55  
You might just be able to see this, but it's a hard thing for a projector to display.

35:00  
It's much easier on my display than it is on the display behind me.

35:05  
Here's an example of colour blindness considerations.

35:08  
This is the Google logo with the red green colour blindness philtre on top.

35:13  
So here you can see Google logo as it normally is and then with a philtre on top so you can still see there is some distinction.

35:22  
But notice the L and the E here now look the same, which arguably is not good.

35:27  
You need the logo to carry the same information whether or not you have colour blindness.

35:34  
So by the way, there are references at the bottom of the slides which is where I've potentially taken things from.

35:40  
Or you can read about more ideas such as this.

35:45  
Now you need to evaluate your user interface design.

35:50  
You can't afford to deliver a bad user interface.

35:54  
Occasionally what happens is somebody will produce a user interface, which is great.

35:59  
The software's rubbish, but people like the user interface and then they're willing to put up with bad software because the user interface feels good.

36:07  
And so you can give them the nice user interface, improve the functionality behind it and they'll carry on using it.

36:13  
I have seen the other way around, which isn't good.

36:16  
Great software.

36:16  
The bad user interface, people don't use it, which is a real sad thing because potentially those developers missed out.

36:25  
We normally use user interface design to fix our price against.

36:30  
So we say, OK, we're going to build that.

36:32  
You agree, don't you?

36:33  
Yes, right.

36:34  
So we're going to deliver against it.

36:36  
As I said in last week's lecture, never, ever, ever say as a user requirement, it must be a user interface that isn't intuitively designed or something silly like that.

36:47  
It has to be a responsive user interface.

36:50  
It's vague.

36:51  
You can't say that you fulfilled this or not.

36:54  
You're much better off agreeing what the user interface design will be with the person you're building it for, and then build that.

37:02  
Now it could be that during the build phase you realise you can't build exactly that because maybe one of the software tools won't let you do it.

37:09  
And then you go back to the customer and you say, well, we'd like to change this feature slightly because it'll be a lot easier to implement in this software framework.

37:18  
Now the customer might say, no, I want the original design, in which case you just have to put the work in.

37:24  
Or they might say, OK, then, in which case you can save a bit of cost.

37:29  
Now if you are using Agile, you may allow user interface design changes.

37:35  
Well, you normally do anyway in agile development, but you have to be careful with them and you have to agree that design before you go ahead and generate any user stories from it, before you go ahead and implement it.

37:49  
Now we can be a little bit more qualitative when we're evaluating user interfaces rather than looks nice.

37:55  
Yeah, it's about right.

37:56  
We can use so-called heuristics evaluation.

37:59  
So the idea here is that we're trying to determine is this a useful user interface design.

38:07  
We could have a bunch of users who are going to evaluate it for us, discover its usability.

38:13  
We could have a heuristics expert.

38:14  
So this is a design expert really who looks at our user interface and tells us is it fulfilling best practise user interface designs.

38:23  
Occasionally I come across videos on user interface design and you know, people who are very well polished say, Oh no, it should, the menu should be here or there and they're taking these decisions or these opinions based on prior design principles.

38:39  
All right, so one formulism of heuristic evaluation was generated by these two people, Nielsen and Moloch, in 1990.

38:48  
You can read about their paper at the bottom here.

38:50  
This was a paper at a conference and so they wrote down these 9 principles which show that we should have simple and natural dialogue.

38:58  
Speak the user's language.

39:00  
Minimise user memory load.

39:02  
Meaning we don't have to ask the person to memorise lots of stuff.

39:07  
We assume that they have a memory slightly longer than a goldfish, right?

39:11  
We don't want to cause them to think too hard.

39:14  
Be consistent.

39:15  
So you can't have two actions being different to their results.

39:20  
Provide feedback.

39:21  
So tell the user what you're doing.

39:23  
You don't really want one click and the whole thing locks up for 10 minutes and the user's wondering has it crashed?

39:28  
Is it doing something don't know you want?

39:31  
Clearly marked exits.

39:33  
Exits here are.

39:34  
Ways of exiting the application or the action provide shortcuts, so easy ways to do things.

39:41  
Maybe it's keyboard shortcuts, like it could be Alt F and then a number or something like that.

39:49  
Provide good error messages to say if something went wrong, what happened.

39:53  
And yes, provide error messaging indeed.

39:56  
Now these people, Nielsen, in fact, it was went on from 1919 and Nielsen updated them again as 10 usability heuristics.

40:05  
And these are the 10 we'd normally quote from 2005.

40:10  
And we've got them here as a list.

40:11  
Now rather than go through the list, I'm going to show you some examples of them so you can see what I'm on about.

40:18  
So let's go for the next slide.

40:20  
So here is an example of visibility of system status.

40:24  
I've taken this little image to the bottom right from Windows 10.

40:28  
So this is now a little bit out of date.

40:31  
Nevertheless, you can see that the people who've designed this are trying to show us the system status in visual form.

40:39  
They're showing in this case that these speakers are muted, that we are plugged into a cable network, that our battery is charging and that OneDrive is syncing.

40:50  
So all of these things are so-called visibility of system status.

40:55  
It could be that you have an application that's doing something and you'd have a progress bar that's showing the download, for example, in the web application, or maybe it's processing some data.

41:06  
It could be that we have some kind of disconnected status or connected status where you're connecting to a remote server.

41:12  
It could be that we have some error icons like the exclamation mark saying we have a problem.

41:22  
Here's another example.

41:24  
This is System and Real World Comparison, which is one of the others of Nielsen's.

41:30  
In Windows.

41:31  
We have this picture which is a folders.

41:33  
Now these folders look a little bit like card folders that you might then put in a filing system.

41:40  
Now, of course on a computer, it's not card, but you look at them and you think, Oh yeah, I saw those in my office.

41:47  
Oh yeah, I put a piece of paper in there.

41:49  
Great.

41:49  
I'm filing something in a piece of in a folder and then I'm able to put it away.

41:55  
So here files appear as documents.

41:58  
And even more than that, the colouring or the type of this picture image here tells you about what kind of document it is.

42:05  
So these things here are Python documents.

42:08  
This I'm not sure, it looks like it might be a text document.

42:11  
Well, of course it is, because it's not TXT at the bottom.

42:14  
So empty folders appear as empty folders.

42:19  
Now in Linux or Mac, these things are normally called directories, but you still have the concept and graphical interfaces on both of those operating systems as some sort of visually empty or full thing.

42:32  
For example, if you look on a Mac, you've got a downloads menu that then pops out at you and you can see all the different documents in a line.

42:40  
Now, on a computer, they're not stored like that.

42:43  
They're in fact bits of data or different memory addresses.

42:46  
But whoever's come up with the design has thought about the connection between the system, meaning the implementation, and the real world, such as something you already know about.

42:58  
Here's another one, user control.

43:00  
So we want to provide navigation.

43:04  
We want to allow so-called emergency exits.

43:07  
So if you're going to do something, in this case it was me about to delete a virtual machine, you want to allow somebody to not do it.

43:16  
So for example, formatting a hard disc that's going to wipe the computer clean, you are definitely going to want to have a cancel button highlighted.

43:26  
So what have they done here?

43:28  
You can see they've got delete all files, remove only.

43:31  
That means remove it only from the list, but don't delete the files.

43:35  
And in this case, they have cancel.

43:37  
And if you can see here cancel is highlighted.

43:41  
What does that mean?

43:42  
It means that if I press enter, it will actually select cancel.

43:46  
So this is a safe thing to prevent so-called accidental requests.

43:52  
Somebody clicks on delete and they go oops, I pressed enter.

43:56  
Now if we'd highlighted delete all by default and then somebody clicks on it and goes oops, I pressed enter, it's gone.

44:03  
So by default, we'd normally want to highlight the cancel button or the the nothing bad will happen button, whatever that is.

44:14  
Here's another example error prevention.

44:17  
Now this you will have noticed is something you've probably come across before.

44:22  
It's the Google sign in.

44:24  
So if you with a Google sign in, try to put in an address that is not an e-mail address, it will tell you this is not an e-mail address.

44:32  
It will tell you enter a valid e-mail or phone number.

44:36  
So they're what they're doing here is they're doing input field validation.

44:40  
They're checking to see if what you provide is sensible.

44:44  
Now if you didn't do a good job here, you could put in 11365 or something, press next, and then the user interface could carry on doing whatever it's doing.

44:55  
And in the last stage, try to use 1135 whatever and crash because you can't use that.

45:03  
It's not a valid e-mail address.

45:06  
So what we normally do is we check to see has the user provided the things that we actually need?

45:14  
We can't allow somebody to sign up for an Amazon account without their name.

45:20  
Have they provided the required input types?

45:23  
If we want a number, they can't provide us with hello, that's not a valid number.

45:29  
It could be the number range as well.

45:31  
So for example, we might not allow a negative number.

45:34  
So how many bananas would you like in your shopping basket?

45:37  
I'd like -10 please.

45:39  
That doesn't make any sense.

45:40  
So we can tell the customer, no, it must be a valid positive number.

45:46  
We might want to restrict the length of the text, meaning the text can't be very long.

45:51  
Why?

45:52  
Because that data is going to go into a database and therefore it might need to be short.

45:57  
We might need to require a length of text, for example a password.

46:02  
Typically it might need to be 8 characters or longer.

46:05  
It may need to have upper and lower case characters, special characters, and lastly, we want to avoid so-called SQL injection.

46:14  
Now, this is a very nasty thing which maybe my colleague who's talking about databases will mention again.

46:19  
Maybe he will, maybe he won't.

46:21  
What will happen in some cases is a user interface has a text box and into the text box you can have text which is valid or not.

46:31  
Now, if I registered myself, I could have my first name as drop and my second name as tables.

46:39  
All right now that's totally valid.

46:41  
But if somebody that executes drop tables inside SQL, that's an SQL command and So what will happen is the tables of the database will disappear.

46:52  
So you need to think about so-called SQL injection where you want to check is the are the input fields that you're using here going to be passed directly into database queries?

47:05  
If they are that could be very bad.

47:07  
So if you want to be malicious, go and change your name to drop tables, make sure there's a; At the end of tables and see how much fun you can have with user interfaces.

47:19  
Here's another one, recognition rather than recall.

47:22  
This is the Visual Studio Code.

47:25  
And the idea here again is we're trying to reduce the load on people's brains.

47:30  
They don't have to remember that much.

47:32  
What we're doing here is we've created a Python, a text string, and then we've got a text string and we've put a dot and then Python has a set of functions that can come after this dot because it's a text string and our user interface, in this case Visual Studio Code, knows those and it's giving them to us as options.

47:52  
So here we've got a drop down menu including the available options, and we also have auto complete, meaning we start typing and it guesses and we can just press tab or enter helping the user.

48:06  
Here's another one.

48:07  
This is actually Firefox and I was running a web server locally and I stopped it running and I tried again to refresh the Firefox page.

48:15  
And so Firefox says here unable to connect, and it gives me a perfectly reasonable error message.

48:21  
It tells me that it can't connect to this local server, local host 5000.

48:26  
Why?

48:26  
Because my Flask web service has been shut down.

48:30  
It tells me some other things.

48:31  
Basically, it's not completely sure what went wrong, but it gives me some hints.

48:37  
In fact, what went wrong is I stopped the web server if there wasn't any firewall problem or anything like that.

48:42  
So sometimes an error message will suggest things and it's down to you to work out which one of the suggestions could be true.

48:53  
OK, in conclusion, the user interface design enables requirements capture.

49:00  
So don't spend lots and lots of time designing a user interface before the requirements or whatever you want to use the requirements some as best as you can, quick version of the user interface design.

49:17  
Take that back to the customer, talk about it, iterate, update the requirements.

49:21  
It's a very much one, then the other one, then the other, then you have an agreement, then you build it.

49:26  
So we use the user interface design to discuss with the users.

49:29  
We think about the data model.

49:31  
Now with data model, if we look back at the previous picture, a few back here in this Google One, we have an e-mail address.

49:40  
So the e-mail address is gonna have to be stored in data.

49:44  
So this is what I mean about the user interface design starts to tell us what we need to store or what we need to have in memory to give back.

49:56  
Now our success is very much connected to a good user interface design.

50:00  
Remember the examples I made well mentioned earlier.

50:03  
If you have a very nice user interface design and the functionality isn't great, people will still use your software.

50:09  
The other way around doesn't work.

50:11  
If the user interface design is bad, the software is good.

50:14  
Typically speaking, they'll ignore your software and people are a little bit, well, how should I say?

50:20  
They go with fashion, right?

50:22  
So if they like your design and they get used to it, they're willing to stay with you as long as you really don't upset them.

50:29  
So this is somewhat of an inertia once you've gained a user base.

50:34  
So you still need to think about that.

50:35  
Microsoft made some big errors where they made Windows 8.

50:40  
So Windows XP was well liked, Windows 7 was OK, Windows 8 people hated it.

50:47  
The idea with Microsoft was they could have Windows 8 which would be similar on desktop and mobile and that would then help them because they had Windows Mobile phones.

50:58  
But then people really didn't like the Windows 8 interface and they had to put back some elements of Windows 7 interface, which was a bad idea.

51:06  
So even though you have loyal customers, you cannot set them enough that they give up in your software and go use something else.

51:13  
OK, that is the end of what I was going to say.

51:16  
So it's open to you for questions and comments.

51:19  
It must be Monday morning.

51:28  
Everybody's very quiet.

51:33  
All right, well, we are going to have a go at this in the lab and we'll be looking at the each other's user face design and then we're going to vote on the best 1.

51:46  
So that'll be fun.

51:48  
Now best be careful.

51:49  
Best is not necessarily the most beautiful looking user face design.

51:54  
So if you look back at these ones back here, go back to the fig this for example, mock flow or this Figma one, they look nice.

52:08  
But it could be here that there's very little under the burger menu.

52:14  
This is called a burger menu, or there's very little under these menus, and if there is not enough detail in it, that's actually not very useful, even though it looks nice.

52:24  
So when you're producing a user interface design, make sure that you're describing the features.

52:30  
It could just be, as we said, the wireframes rather than focusing on the look and the feel.

52:35  
All right, that's somewhat secondary.

52:38  
OK.

52:40  
All right.

52:40  
Well, if you've got no questions or comments, we can stop there.

52:44  
I'm going to stick around in case anybody wants to ask me anything Other than that.

52:48  
I hope to see you in the lab where we have a play and see what we can do.