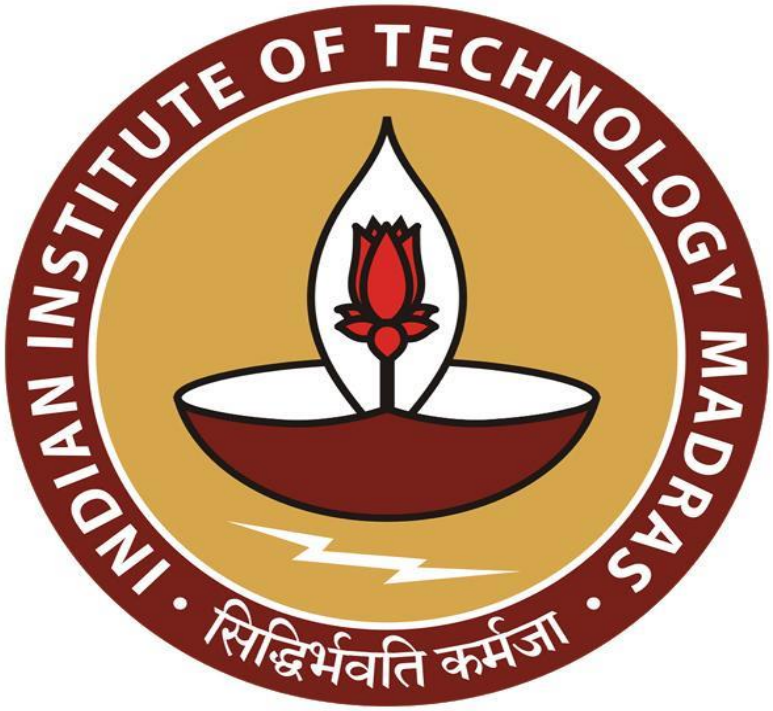


IIT Madras

ONLINE DEGREE

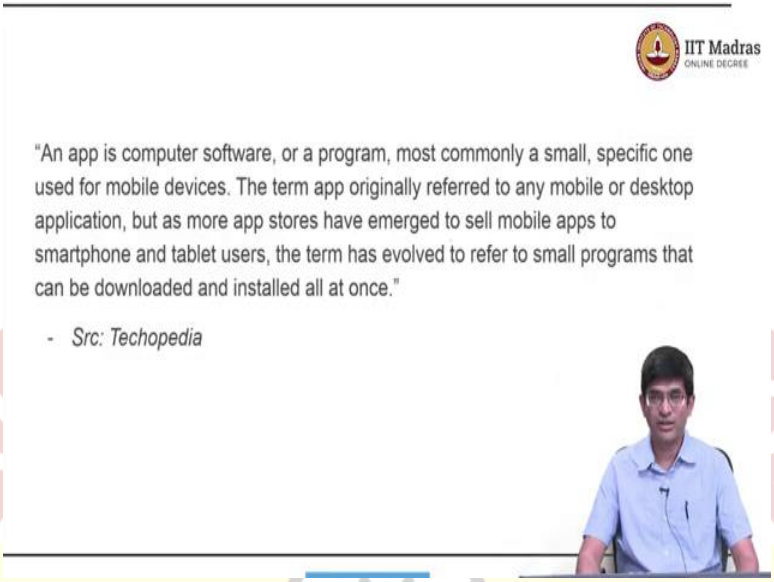
Modern Application Development - I
Professor. Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
What is an App?

Hello everyone, and welcome to this course on Modern Application Development.

(Refer Slide Time: 0:16)

As part of this course we are going to see various aspects of what it means to develop an application in modern technologies and as part of that the first thing we would like to know is to understand what exactly we mean by an app. So, the question that we would like to answer is what is an app, anyway?

(Refer Slide Time: 0:36)



So, one good way to start is just by looking up a definition and the simple way to do it would be to look for either Wikipedia or Google at this point because after all those are our main sources of information. So, what I did was looked up on Google and what does Google do it points me to a source on Techopedia which has a definition of an app which I thought was workable and it sort of conveys all the useful information that we need at this point.

So, according to this definition an app is computer software or a program, in particular most commonly a small specific one used for mobile devices. Now, as we will see later we are not going to restrict ourselves to that but this is good I mean it sort of tells us what we are looking for.

The term app originally referred to any mobile or desktop application and at some point especially after the rise of the Apple iPhone, app stores emerged and now more and more app stores have emerged that sell mobile apps to smartphone and tablet users. And since then the term has sort of evolved, in general to small programs that can be downloaded and installed all at once.

Now, is that really the context in which we are going to be using it? Largely yes, we are going to be looking at how do you solve a specific problem, let us say you want to create an app with a fairly specific goal in mind, what would be the steps that you need to follow, how would you go about doing it?

(Refer Slide Time: 2:03)

These icons should be familiar to pretty much everyone these days, you have a number of different things out here. So, for example, right now this smiling icon over here is the Amazon app. This off course is twitter which has been in the news for a number of reasons good and bad of late. Microsoft word is probably familiar to most of you simply because we use it to edit documents. Firefox and Chrome are both extremely famous as the dominant browsers, off course Firefox has been declining a little bit of late but Chrome, Firefox, Microsoft Edge, Safari from Apple, those are all the main browsers, web browsers that we are all familiar with.

Now, the browsers play a very important part in general in a lot of what we will be doing moving forward and we will spend some time on understanding what they are capable of later, throughout the course we will be looking at various aspects of them. But apart from that there are also mail clients that you could just use to read email, something like Instagram, social media similar to Twitter.

And also something like Visual Studio Code which is just used as a text editor, something which you use in order to edit text files for what purpose? Either plain text or maybe some kind of formatted text or maybe as we will be seeing later markup, markup languages like HTML or programs, programs in python, programs in C, programs in pretty much any programming language.

What makes a text editor special and different from a word processor is that it is sort of oriented more towards handling text and then you are trying to let say write a program it provides a lot of peripheral support to you in the form of for example, you could have syntax highlighting, you could have automatic indenting and various other things that make it easier to write code.

Visual Studio Code is one example, some of you are probably using Sublime Text others may be familiar with VI or E-macs. There are a number of others, Notepad is the one that comes by default with Windows, G-edit comes by default with most Linux systems. In general the common thing out there is they are all just good at very simple text editing, you can't do much of sort of making parts of the text bold or italics or other highlighting but you can just write text. And as we will see later, when we are talking about markup languages that becomes important, you should be able to manipulate text so that you can then control what it displays.

And finally off in the right corner over here I have an icon for a terminal right? And what do we mean by a terminal or a command line prompt? That again is something which all of you who are doing this course will have to develop some familiarity with as we move forward through the course. The reason for that is that apart from writing the code itself some familiarity with what it takes to actually run programs on a system is also important in order to get our entire application running.

(Refer Slide Time: 5:31)

Now, the applications that we looked at previously just covered a wide spread of different types of applications I mean you could think of them functionally, there are web browsers, there are

Code editors, there are word processors, there are social media applications. Another way of sort of dividing up applications is to look at the platform on which they run. And here we could start by thinking about desktop apps.

Now, what is a desktop app? Usually what we mean by a desktop app is something which is typically a standalone program what I mean by standalone is? You could potentially use it without even requiring a network connection, you just have your laptop or tablet, well, tablets are increasingly bound to the network these days, but a desktop or a laptop very often is meant for running maybe even without a network connection.

Nowadays we are used to network connections but for a long time there used to be a setup where you just had your computer, you had your files locally on the machine and you needed to work with it. So, many desktop apps in that sense are standalone, editors, word processors, Microsoft word, visual studio code all of those are things that you could just run on your machine without requiring any network connection and everything, the source where the files are from, what you are doing, the keyboard that is attached everything is just in one machine.

Now, web browsers are also standalone desktop applications except that their entire utility is in the sense that you know you need to be connected to the network otherwise you are not going to be able to browse very much, you can just browse around your own local machine but that is probably not really what you want.

Similarly, a male client, Outlook or Apple mail or Thunderbird all of those are examples of email clients, what they allow you to do is to read email, compose email, sort it into different folders, delete email. And what happens in such a situation is the actual reading, composing, deleting, all of that is happening on your local machine, but finally in order to send or retrieve mail you need to have a network connection.

So, many of these desktop apps or most of them need to be able to work offline which means that they would need some form of local data storage, so even an email client would typically download your email may be using pop or some other i-map or some mechanism like that, download it onto your local machine and then allow you to work on it. And then finally just connect to your outgoing mail server in order to send the email.

So, in most cases they may require a network connection for some activities but maybe not for all. So, a word processor for example should be able to go along perfectly fine with no network at all. Now, the way that these applications are developed is using what are called software development kits or SDKs. These are usually custom frameworks that are in many cases OS specific, operating system specific.

So, for example if you had an SDK for developing apps or applications for let us say Microsoft windows, it is unlikely that the same applications would work without changes on Linux or on a Mac or for that matter on tablets or on phones. And this was ok for the longest time because the primary computational interface that people had was a PC and in most cases that would be a windows PC.

So, you would find that there would be many windows specific software development kits, some kinds of frameworks including things that allowed you to create windows and graphical user interfaces and various things that would have API or application programming interface functions that allowed you to link to the underlying operating system. So, for example if you wanted to create a file or connect to the network all of those would be through API calls.

So, desktop apps even though I call them apps usually the term app is not really used in that context, these are full-blown applications. So, the whole idea of an app is that it is a smaller version of an application. Nowadays, however the boundary is blurring, I mean we very often have situations where you might actually find that the application that you are running is sufficiently focused and specific that you want to make it into a standalone app by itself, something that could then be distributed to other users or could even be run on multiple operating systems, we will get to all of that later what are mechanisms that you can use for that.

(Refer Slide Time: 10:18)

Now, nowadays one of the biggest sort of targets for apps and in fact that is where the term originally came from is the mobile platform, phones and to some extent tablets, phones are still dominant, I mean the number of phones is significantly larger than the number of tablets in existence today, so whenever people are talking about developing a mobile app it is more targeted towards a phone than a tablet but the IPad, various Samsung tablets or even less expensive tablets have led enough people to say that the tablet is also important simply because it has a larger screen and you could sort of do a little bit more on it than you could do with the phone.

Having said that there are constraints, you are limited in screen space, the interaction mechanisms that you have, you do not always have a keyboard and mouse. So, many things that we are used to in applications like for example the fact that I can type something into a box or I can hover with a mouse over a link or over an icon to get information, suddenly have gone away.

On the other hand, new interfaces have arrived, touch has opened up a whole new set of ways of interacting with the computer, there are also audio interfaces you could speak and there are speech to text interfaces that would interpret what you are trying to say. There is a camera which can be used not only to take pictures but also to sort of look at gestures and recognize what you are trying to do and respond accordingly.

Some of the constraints that you face when you are dealing with these kind of phones and tablets is that very often they have much less memory and by that the term that is usually used is the RAM; the random access memory than a desktop or even a laptop would have. And similarly the processing power is typically going to be significantly less than you would find in a desktop or a laptop.

And one of the main reasons for that is simply the power consumption, these handheld devices the mobiles and tablets need to run on battery and they need to run for long durations, people are no longer willing to accept a phone that will just run for one and a half hours or they probably would never accept something of that sort, but even a tablet it is not okay for something that just runs for maybe one and a half hours or so, you need to have something which can last at least a full day, which means that you cannot have a power hungry processor. And there are limitations that automatically sets limitations on how fast your processor can run, how much work it can do, how many computations it can perform and so on.

Now, what are the kinds of frameworks that we are looking at in this context? Some of them are of course operating system specific, even now the Apple ecosystem basically means that if you develop an application for the iPhone or the iPad you would probably be using the swift programming language with Apple's SDKs, the software development kits.

The good thing of course is if you develop for an iPhone then almost the same thing should work on an iPad, a very similar thing would work on MAC OS, so they have integrated tightly within their ecosystem and they have a variety of devices. On the other hand, in sheer numbers the number of android phones out there vastly outnumbers Apple iPhone. So, what do you do? Can we have something which is cross platform, something that would allow you to develop both on apple iPhone as well as on android phones? And yes a number of frameworks have come up that try to address that problem.

Now, one other thing that we need to keep in mind when we are talking about these mobile apps is that given the nature of the device, the fact that a phone by its very nature is expected to have some kind of a network connection at the very least so that you can make calls but also for various other purposes, means that many of these apps are strongly network oriented. Twitter, Instagram, the Amazon shopping app, all of them are pretty much just connecting you with

servers at the other end that are giving you information. And operating them in standalone mode is more often than not even particularly meaningful.

(Refer Slide Time: 14:32)

Now, we come to the web apps. Now, the web is effectively providing a platform on which we are going to build our applications and the whole idea is that these browsers that we saw briefly before Chrome, Firefox, they have been ported already to run on multiple devices. So, Chrome runs on Linux, it runs on Windows, it runs on the MAC. Similarly, so does Firefox. And the interfaces that it has on all of those and not only that even on mobile devices is almost the same.

This helps to create a common base that works across operating systems. It is heavily network dependent which means that I mean after all everything about the web means that you need to be connected to a server. And the reason why it has become a platform for developing apps is the rise of dynamic websites ever since the so-called web 2.0 in the early 2000s which basically means that not only can you create pages that change dynamically each time a person loads it and depending on information about the person you could behave differently, but it also allows you to program with languages like JavaScript that can actually manipulate what is done on the browser itself. So, a lot of the interactivity, lot of the processing could also be pushed onto the user's end “the browser”. Now, these web apps, this platform that we are talking about is going to be the main focus of this course. What are the reasons for that? Primarily because our main motivation over here is to understand what is involved in developing an app.

So, we are not really concerned about Windows or MAC or even android, rather we would like to see what are the fundamental principles, are there anything that you need to know irrespective of the final platform on which you want to develop your application. How do we distill that out and then use it with certain justifiable assumptions hopefully in order to develop applications that are sort of neutral. The web gives us a nice platform to do that and we are therefore going to use it for much of this course.