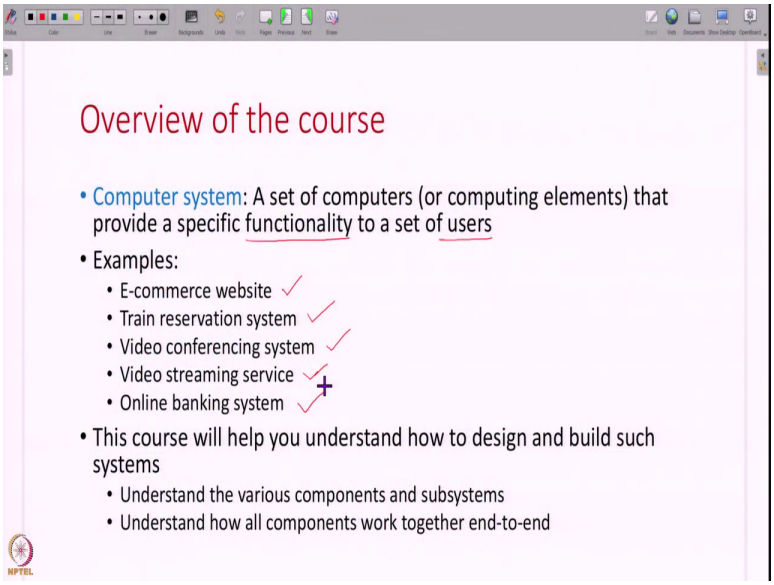
Design and Engineering of Computer Systems
Professor Mythili Vutukuru
Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture – 01
Introduction to Computer Systems

Hello everyone. Welcome to the course titled Design and Engineering of Computer Systems. So, in the next 8 weeks, we will take a journey together into the world of computer systems which I personally find very fascinating and I hope you will do so too in the next few weeks. So, let us get started.

(Refer Slide Time: 00:36)



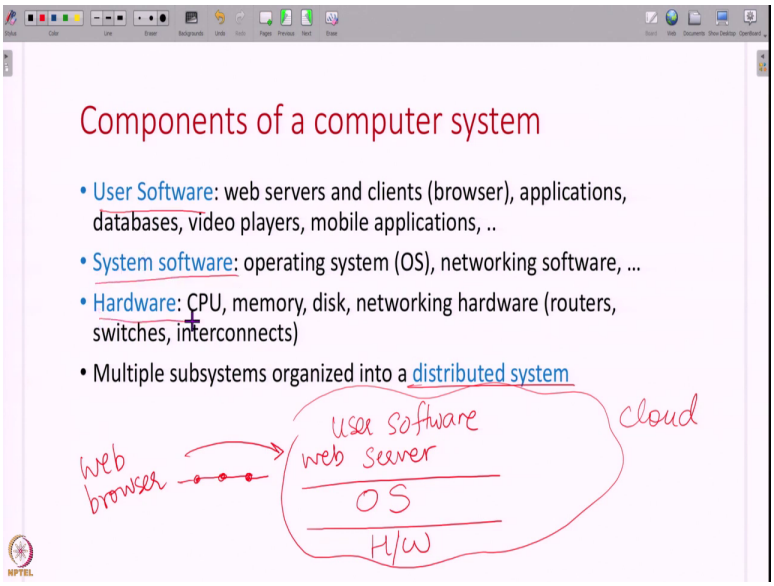
So, what is a computer system? A computer system is nothing but a set of computers or other computer hardware together that provide specific functionality to some set of users. This is the basic definition of a computer system. Of course, you have many systems in real life. For example, a building is a system. Many other real-life systems do exist. But when the systems are composed of computers we call that a computer system. And you all would have interacted with a computer system at some point of time in your life, and these are widely present all around us.

For example, if you purchase something online, you have used any e-commerce website or if you book tickets online, if you have used a video conferencing system to access content or to talk to anybody over a video conference if you have streamed any movies online, you have used an

online banking service for online transactions in your bank account, or even taking this course. You are taking this course on an online computer system that is streaming all of these videos to you, which is letting you take exams online. All of these are examples of real-life computer systems. And the goal of this course is to help you understand how these computer systems work.

So, these computer systems are made up of various smaller components or subsystems. So, we will study each of these subsystems or components. And we will also understand how the entire system is put together end-to-end. That is the goal of this course.

(Refer Slide Time: 02:20)

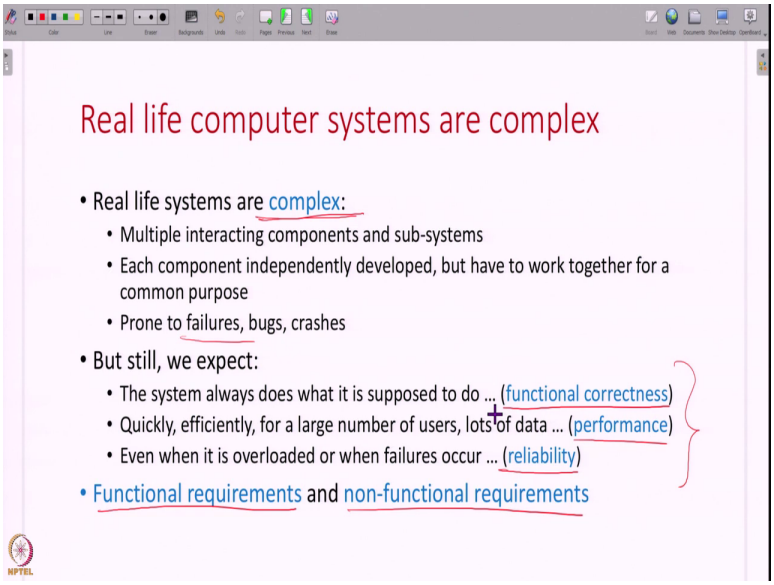


So, let us begin with understanding what are the components of a computer system. The first thing is, of course, there is some user software. This is what you interact within a computer system. For example, you have your web browser using which you are opening a website. Then there is a web server that is actually serving some content to you, that is streaming this video to you, you have video players, you have mobile applications, you have games. All of this is user software, that software programs that you as a user interact with. For example, an example of user software is a web server. And another example is a web browser on which you are watching this video. And the browser and the server talk to each other.

Then underlying this user software you have system software, like the operating system. Your browser is running on Linux or Windows or some operating system like this and there are other system softwares to manage networks and so on. And all of this user and system software that is running on the underlying hardware. You have your CPU, memory, the disk, the various routers that form the network. For example, your browser is talking to a server to fetch this video to you and this traffic is going over a series of routers over the internet. All of these are the various components in the computer system.

And a computer system is not just one or two computers. It is actually a large distributed system. There is a large set of computers that are distributed running on a Cloud or a Data Center or something like this. So, real life systems are quite complex and they have all of these components like user software, system software, hardware and so on.

(Refer Slide Time: 04:22)



So, what makes real life computer systems interesting to study? Why do we need a course on it? That is because real life systems are complex. As we have seen there are multiple different components that are interacting like software, hardware and everything. And all of these components are built independently and they are somehow put together.

So, somebody builds the server, somebody builds some database, somebody builds the system that is storing these video lectures. All of these are built by different people and they are assembled together for a common purpose. So, therefore real life computer systems are quite complex. And there are a lot of other issues that can happen like failures, bugs, crashes the power may go off, the hard disk may crash. All of these happen. And in spite of all of these complexities of real life we still expect these computers systems to do certain things.

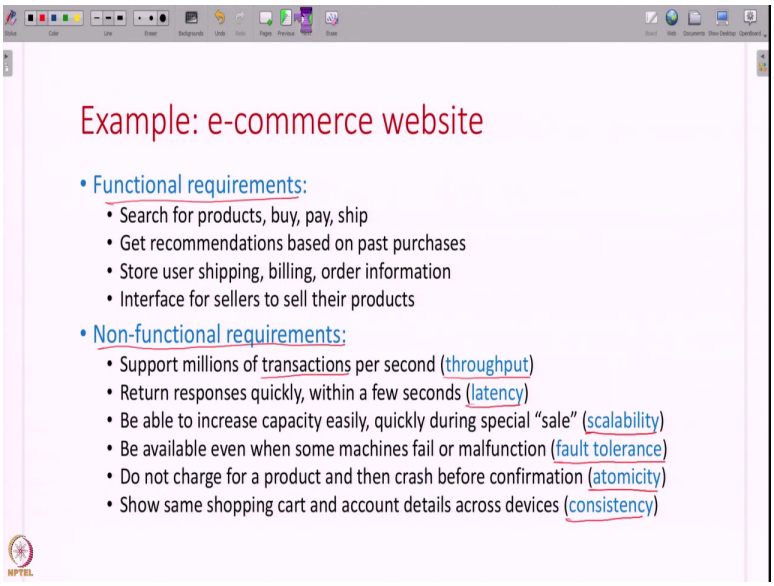
We expect the system to have what is called functional correctness. That is when you book a ticket on a website you expect that your ticket actually gets booked and you do not want the website to say oh no we lost power, therefore, we lost your reservation. Or there was some bug in our code therefore we did not book your ticket. We do not want to hear that. So, we expect certain functional correctness that a system does whatever it is supposed to do.

We also expect performance. When you book your ticket you expect the whole transaction to finish in a few minutes. You do not want to sit in front of a computer for one hour to book your train ticket. That is not what we want. We want the whole thing to be quick, efficient in spite of there being a large number of users or lots of data and so on.

And finally, we also expect reliability from computer systems. When a failure happens, power goes off you do not want your bank to come back and say, oh no we lost track of how much money you have because some hard disk crashed in our bank. You do not want to hear that. So, we have all of these expectations about computer systems.

Some of these are functional expectations dealing with functional correctness and the others are non-functional expectations like performance and reliability. So, how do you build computer systems? How do you build these complex computer systems to satisfy all of these requirements? That is what we are trying to teach you in this course.

(Refer Slide Time: 06:52)



So, let us just take an example of some real life systems to understand what are these functional and non-functional requirements. Let us take a concrete example of say, an e-commerce website. There is a website on which you are going to buy some products online. So, what are some of the functional requirements you expect from it?

You should be able to search for products, buy the products, pay online, have been shipped. And another kind of thing is you might want to get recommendations based on your past purchases. Or if you have bought this item in the past you might be interested in this item. You might want to see such recommendations. You might want the web page to store all your credit card information, for example for future access. The sellers might want to have an interface to upload their products. All of these are the functional requirements the functions that an e-commerce website should do.

In addition to these functions, we also have some other non-functional requirements or expectations. For example, you want your e-commerce system to have a high throughput that is it should be able to do a lot of a high number of transactions per second. You could have millions of users trying to buy something from an e-commerce website and it should be able to serve all of them, do these millions of transactions every second, and do all of them quickly. When I click on buy within a few seconds I want a response saying ok fine, your transaction is done. You want very low latency. You want high throughput. You want low latency.

And you want another interesting property that we expect from systems is scalability. What is scalability? Now, your system is small there is no load. It is handling that load. But suddenly suppose there is some festive offers or some special sale. Suddenly the load on your system increases. Your system should also be able to increase its resources and somehow live up to that challenge. Suddenly the load has increased the system should also increase its capacity to serve that increased load. That is called scalability. And that is also a very nice property to have in computer systems.

And other non-functional requirements are something like fault tolerance, that is even when some machine fails, the power goes off, something goes wrong the system should still be able to continue to work normally and not somehow give up. Another interesting property is what is called atomicity. So, what is atomicity? Atomicity means when you are doing multiple things you want all of them to happen or you want none of them to happen. But you do not want something half and a half to happen.

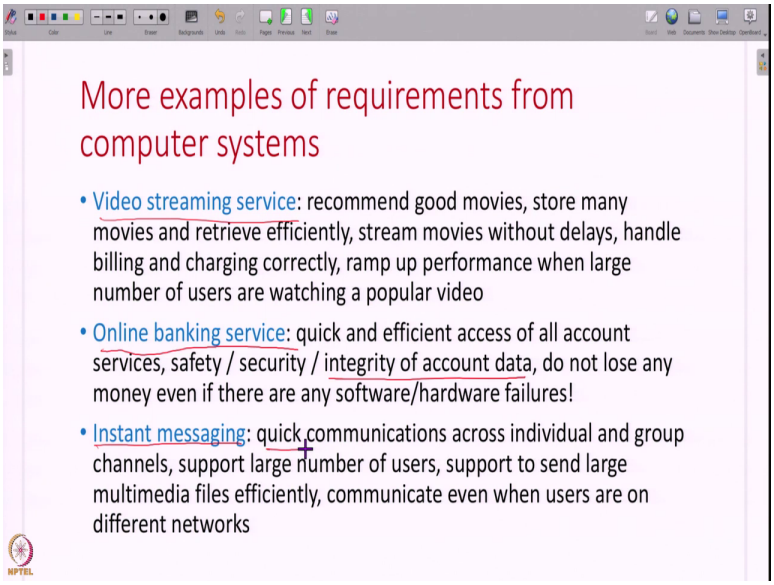
I will give you an example. So, suppose you are trying to buy a product online on an e-commerce website. You do not want this situation where the website has taken your payment, charged your credit card and somehow some crash happened and then it forgot that it has to ship the product to you. You do not want that. If you have charged my credit card you please ship the product to me. Otherwise, if some failure has happened then fine, you do not even charge my credit card and you do not ship. I am ok with that.

But you cannot do half of an operation, charging my credit card and then ignore the other half which is shipping me the product. So, for most of, for a lot of computer systems, there is some notion of this atomicity, that all of these things should happen together or do not do to anything at all. This is also so very interesting property that we may not have thought of it explicitly but we subconsciously we expect such things from the computer systems.

And another property is consistency that is, no matter where you access a system from, it should show you consistent information. For example, if I am accessing my shopping cart from my

phone or laptop I should see the same set of products. So, that is consistency. So, these are all examples of various expectations we have from an e-commerce website.

(Refer Slide Time: 10:55)



So, similarly, we can list down various other computer systems in real life and what are all the expectations we have from them. So, I have some examples here and I am sure you can also think of your own examples. For example, if there is a video streaming service. You want to be able to watch movies efficiently, quickly without a lot of lag or buffering. You might want to get recommendations for movies.

And you want to be able to you have some billing and charging infrastructure. And you want your some kind of scalability. For example, if there is a popular video being streamed with a lot of people watching. You do not want the video to get stuck too many times. All of these are expectations from a video streaming service.

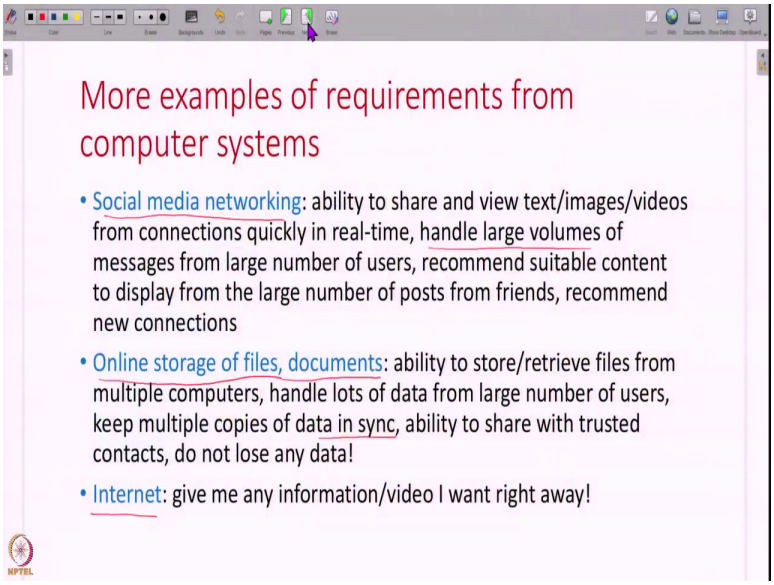
So, what happens if you consider an online banking service? Again this also has similar expectations with respect to good performance, throughput, efficiency and all of that. But for a banking service things like the integrity of your data, all of these become more important. You do not want your bank to say, oh no something happened; I lost all your account information. It is ok if some video streaming service says sorry we lost your favorite movie. It is no longer

stored. Something happened. We lost it. But you cannot have your bank lose your data. That is very bad.

So, different systems, even though the basic set of requirements are more or less the same, different systems emphasis on different things. For a video streaming service, performance, latency, quickly starting to stream is more important whereas for a bank, the integrity of your account data is more important.

Similarly, if you consider an instant messaging service quick communication. As soon as I send a message all my friends should get it. That is more important. And not so much, may be the integrity of your data or security or some, one message here or there getting lost, that it may not be as important as quick communication, supporting a large number of users, the ability to send videos and photos. All of these are more important when we are talking about instant messaging.

(Refer Slide Time: 13:01)



So, similarly, if you, let's see more examples like if there is a social media networking system. A large number of people millions of users are posting billions of messages all the time and you want to be able to see all of these messages in real time quickly. You do not want to get a social media message after all your friends have seen it two days later. You want it quickly delivered to you.

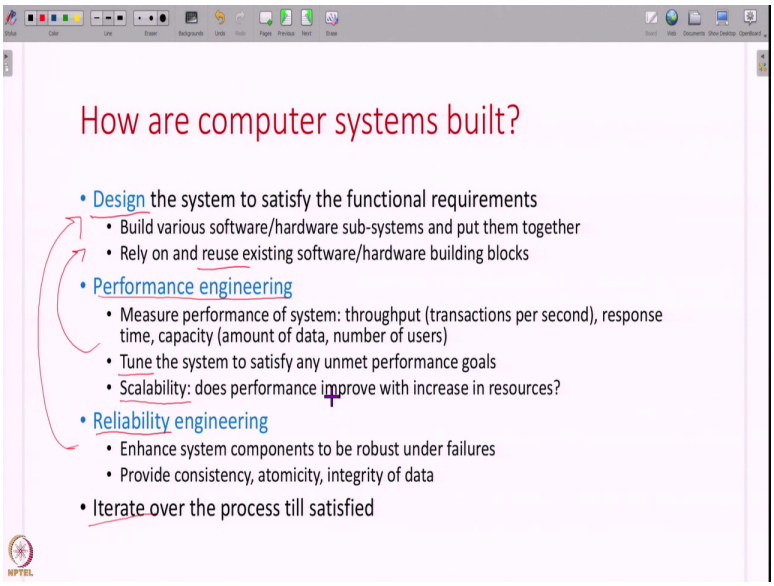
And these social media systems also handle large volumes of messages from a large number of users, and you want the system to be able to keep up with these millions of messages and billions of people on the planet using the social media website. And of course, other things like recommending suitable content or recommending new friends or new connections, all of these are other requirements. But primarily when you think of a social media system scale like handling large scale is what becomes very important.

Then similarly moving on, if you have an online storage, a Cloud storage of files or documents, for this once again storing your data safely. If you have a very important assignment that you are doing online you do not want the assignment to just be lost the day before the deadline. That is a disaster. So, the security of your data once again becomes very important. And also the ability to have multiple users collaborate. And if you and your friends are working together online on

some file you want to be able to see each other's changes quickly, to be able to sync up with each other's changes, such requirements become more important in such systems.

And of course the internet is a classic example of a large scale system that all of us use. You are using the internet right now to watch these NPTEL videos. And for the internet of course all of all of expect to download any amount of data anytime all the time at high speed. We all have these expectations about the internet. So, the summary is that we have a lot of expectations or requirements from computer systems. And for different systems we have different slightly a different set of requirements. Something is more important for some systems than the other.

(Refer Slide Time: 15:17)



So, fine, we have all of these functional and non-functional requirements. So, then how do we go about satisfying them? How are real life computer systems built to satisfy all of these requirements? So, any system starts with a design phase. You will first put together, identify what are the functional requirements, what is the system supposed to do and put together various hardware, software, subsystems, components, put them together which will satisfy these requirements. And most of the time you will try to reuse existing software, hardware.

For example, you will not write an operating system from scratch every time. You will just use what are the various subsystems available to you. You will reuse them. And you will put it together. You will design and put together a small system first. And then you might think about

performance engineering. You might think about, ok let me measure the performance of my system, how much throughput I am getting, how many transactions per second, what is the response time of my system, what is the capacity of my system. You will measure all of these things. You will define certain metrics and measure them.

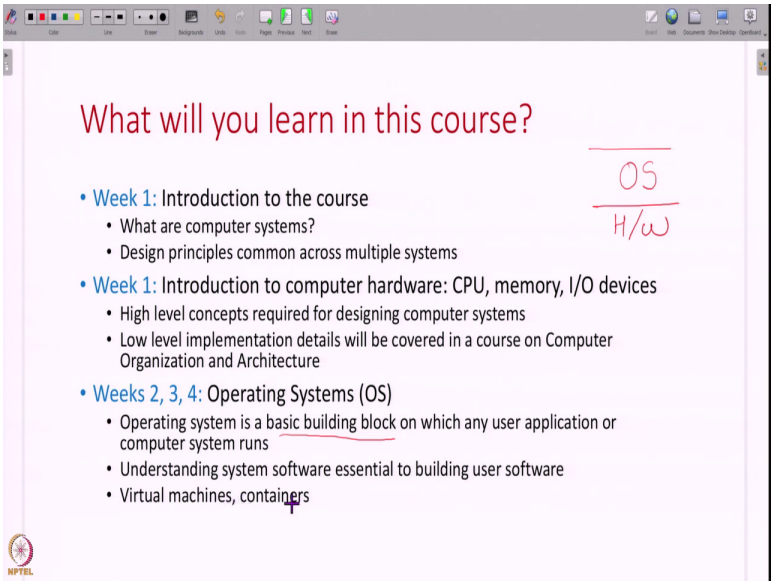
And if any of these metrics are not up to your expectations then you will tune the system. You will tweak the system somehow to improve its performance. You will also check for scalability. So, tomorrow if my load my demand on the system increases and I want to give more resources add more resources to the system will its performance increase? You will test for that as well.

And you will also start thinking about reliability. What if failures happen? Is my system robust to failures? Can I guarantee atomicity, consistency, integrity of the data all of these? And if any of these are not met you will go back to the design phase, tweak the design. Performance is not met, reliability is not met, you will tweak the design and then come back, measure it again then go back.

So, overall all it is an iterative process. Nobody writes, just builds an entire large-scale system by writing code in one sitting. That never happens in real life. It is an iterative process. We keep going over all of these things.

So, this course is meant it to give you an idea about all of these different steps. We will see how do you go about designing the real life computer system, what are the various subsystems involved, how do you put them together, what is meant by performance, how do you measure performance, how do you tune performance, how do you guarantee scalability, how do you guarantee reliability, what are the various concepts involved in this iterative process? This course will try to cover them in some detail.

(Refer Slide Time: 17:53)

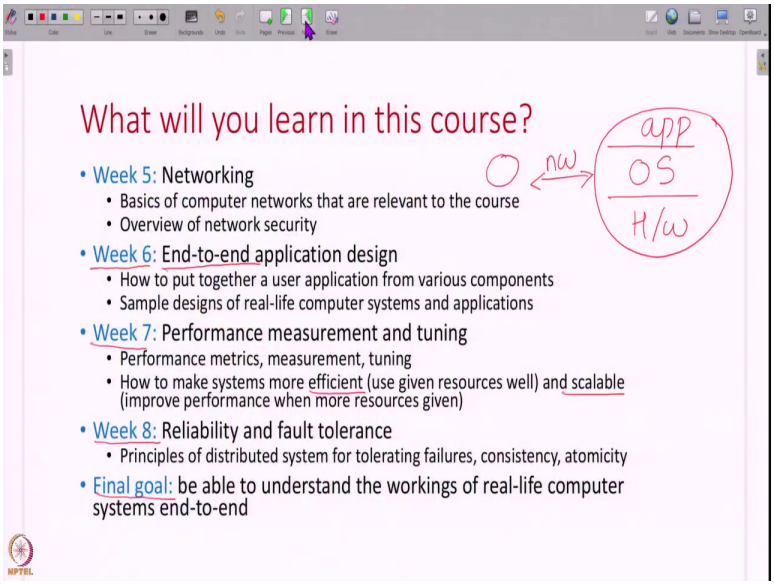


So, this is the week-by-week outline of this course. In the first week, we will simply do an introduction to the course where for example this lecture I have introduced you to what are computer systems. And in the next lecture, we will learn some common design principles that are common across all the other lectures in the course.

We will just start with a brief overview of these common principles and then we will begin by studying systems from the bottom up. We will first have a very brief overview of computer hardware. What is CPU, what is memory, what are IO devices. And of course, full detail in computer hardware if you want to know you should do a course on Computer Organization and Architecture. This course will not cover hardware in a lot of detail but just the high level concepts required for you to understand the designing of computer systems. So, we will just briefly cover computer hardware.

And then we will spend a few weeks on the operating system. This is a very important piece of the puzzle because this is a building block for any other user application or computer system that you want to build it will run on the operating system. Therefore understanding the operating system is very important. So, we will study how the operating system works. And we will also study related concepts like virtual machines and containers.

(Refer Slide Time: 19:15)



And after that we have briefly seen hardware then we have seen the operating system for a few weeks and then we are going to study networking. If you have another system like this talking to this system they talk over a network. So, we are going to get some basic concepts of networking and network security. Of course you are welcome to do a full course on computer networks also to get many more details. But this course we will cover it in one week at a high level. You will get enough concepts to understand and appreciate the design of large-scale real systems.

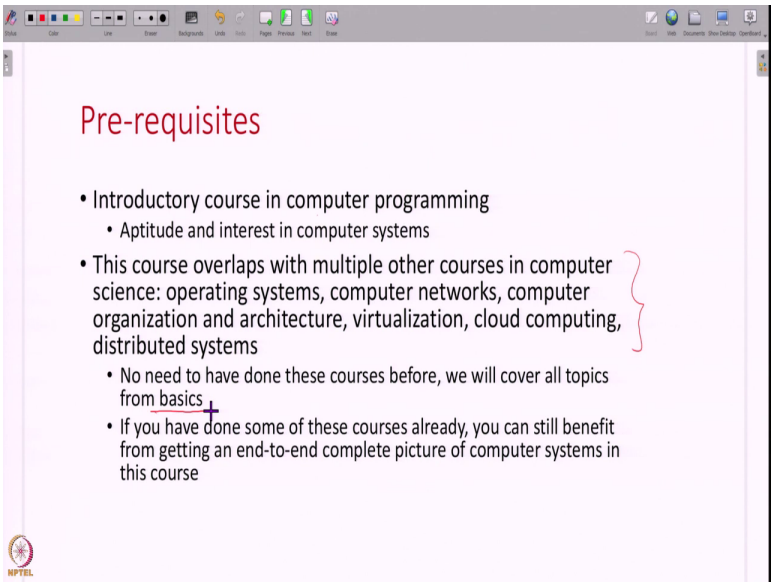
Then week 6 onwards is when the actual fun begins, where we actually start talking about applications design. Now, that all the building block, the lower layers are clear we will now move up. We will start to see end-to-end application design. We will take some examples of real life computer systems. And we will start putting together the computer system end-to-end. And of course by this by week 6 all the lower building blocks would have been clear so that we can start putting together end-to-end the application. And we will take some sample designs and start understanding about it.

And in week 7 we are going to look at performance measurement and tuning. What does it mean to have a good performance? What are the metrics? How do you measure them? How do you tune performance? How do you make systems more efficient given some resources get the best out of those resources? And how do you make systems more scalable? That is if I give you more

resources will you still continue to do well? These are two orthogonal questions and we will study both of them.

And then in week 8 is when we will move on to reliability, fault tolerance. In spite of failures how do you guarantee correct performance of the system? And by the end of these 8 weeks the final goal of this course is you will be able to understand the workings of a real life computer system, like say an e-commerce website or or a Railway ticket reservation system, whatever it is. You will be able to look at the system and understand what are all the various components in it and understand the design end-to-end. That is the goal of this course.

(Refer Slide Time: 21:38)



So, what are the prerequisites? Ideally, you should have taken a previous course in Computer Science. This should not be your first course in computer science. At the very least the basic introductory computer programming course should have been done by you. Otherwise, it is very hard to appreciate some of the things that we are going to cover in this course. And also a general interest in computer systems and programming is going to be very useful to do this course.

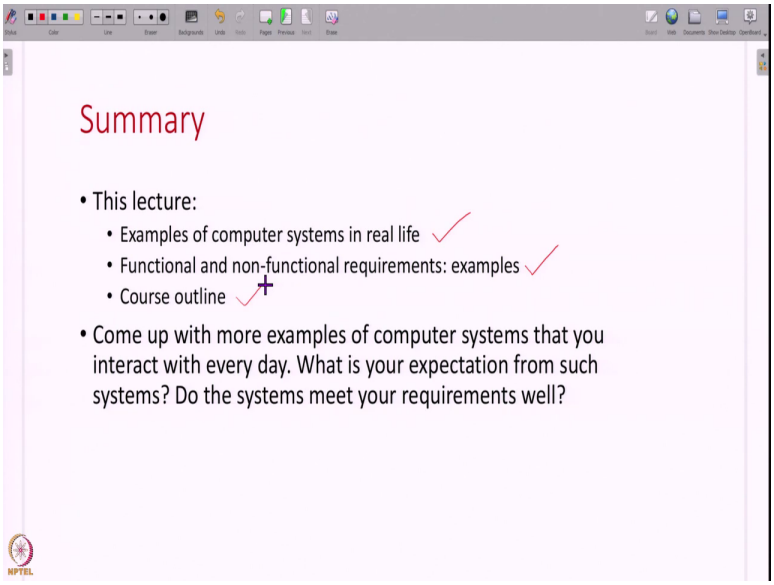
And as you may have seen in the outline this course overlaps with many other courses in computer science like operating systems, networking, architecture virtualization, distributed

systems, Cloud Computing. So, this course will have elements from all of these courses to some extent.

Of course you need not have done any of these courses before. We are not assuming knowledge of any of these courses. And whatever information about these courses about the operating system or a network is required we will cover all of that In this course. We will start from the basics. We will not assume any prior knowledge of any of these courses.

But of course, if you have done some subset of these courses already then you will still benefit from taking this course in getting an end-to-end complete picture of the computer system. if you are done just for example just operating system you can still understand there are other parts of the course that will tie up all the things together for you. So, this is the prerequisites of who should be taking this course.

(Refer Slide Time: 23:09)



So, to summarize that is the end, we reached the end of the first lecture. In this lecture I have given you a small flavor, an introduction of what are computer systems in real life, what do we expect from these computers systems and what is the process in which real life computer systems are built. And I have also briefly given you an outline of this course as to what you are going to be doing in the next eight weeks.

So, before we close today's lecture I would like to leave you with a thought of try to look around and in your daily life, see what are some of the examples of computer systems that you find, what are the computer systems that you interact with in your day-to-day life, and what are the functional and non-functional requirements of such systems? What do you expect from the systems, and do existing systems meet these requirements that you have? This is a good exercise to just get you comfortable thinking about computer systems. So, that is all I have for today. Thank you very much. And we will continue this course in the next lecture. Thanks.