upGrad

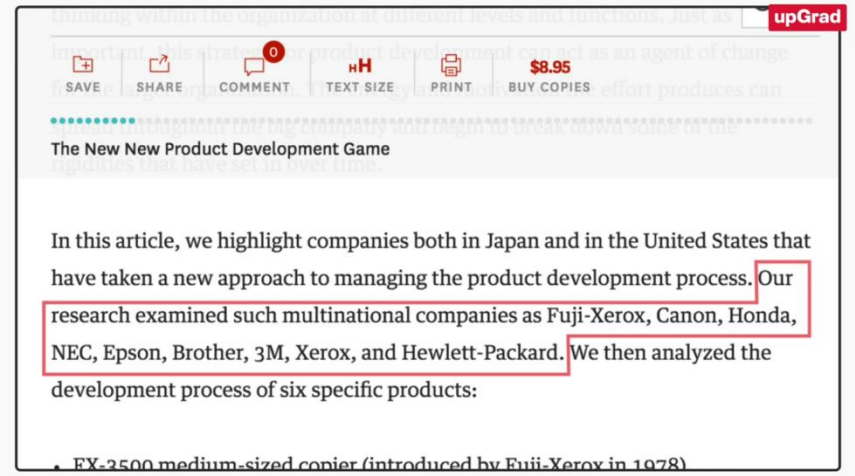
Transcription

The Scrum Framework

Hey there. So, in the last session, you were introduced to a new way of developing a product and you saw how this process grew from the waterfall method to the agile methodology. You learned that agile is an incremental iterative method of product development.

After learning about agile in the previous session, I did a bit of reading on my own, and there's one phrase that popped up everywhere, SCRUM framework. But what exactly is scrum framework? Where does the word scrum come from? And what does this framework involve? Let's see, what our subject matter expert has to say about it.

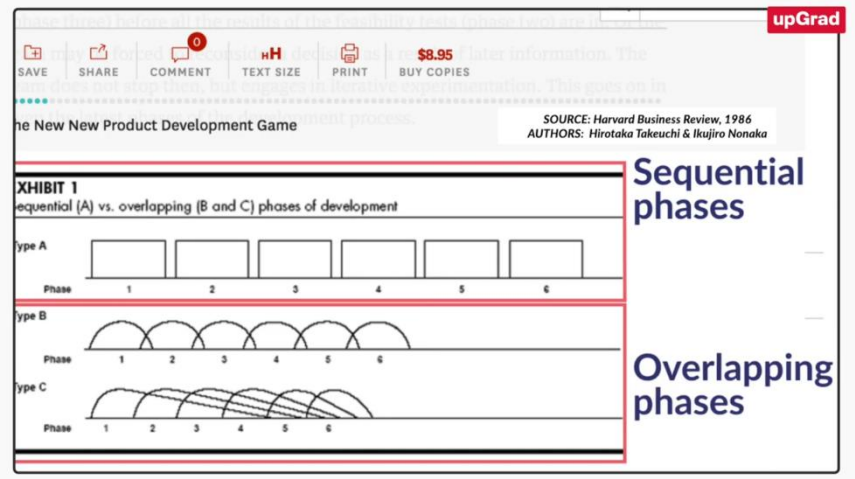
Scrum is one of the agile methodologies, which is used by majority of the agile practitioners. Whatever we talk about, scrum may not apply to other methodologies. But I think scrum is a beautiful starting point for a lot of teams because it allows them to understand agile based software development to the maximum extent possible.



It is also a very simple framework. Let me talk about one of the most influential articles that came out in Harvard business review. It came out actually in 1986, two Japanese professors, they were studying the product development, how it was happening in some of the Japanese and American companies, specifically Honda, Fuji, Canon, and few of the other companies.

And they found that these companies, unlike a lot of other companies were actually doing the new product development in a new manner. And the title of their paper was the new-new product development game.

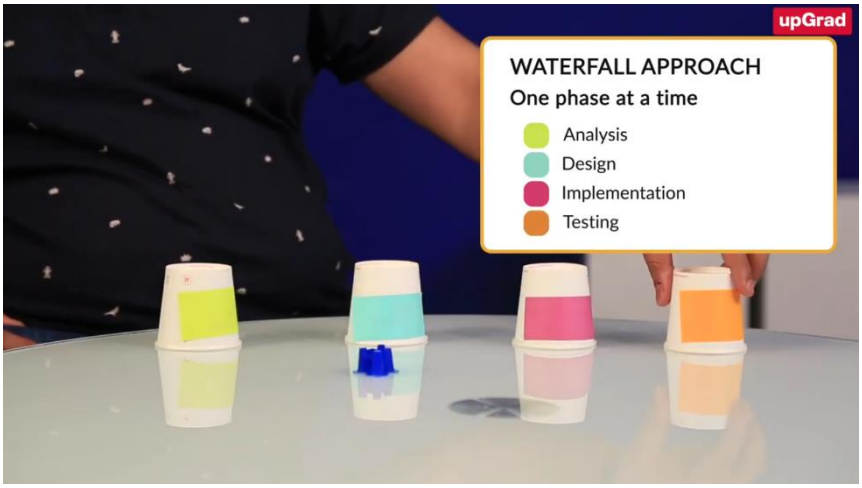
And they found out that speed and flexibility are not only essential in that, they also found out that the old sequential way, which means I do step number one, then I do step number two, step number three and so on, this kind of a sequential approach to developing new products was not enough anymore.



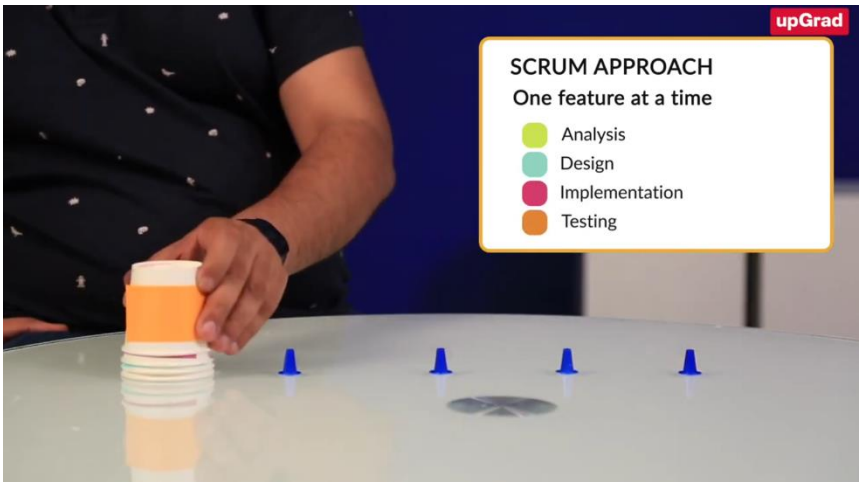
In fact, they found that these Japanese and American companies were actually using ideas about overlapping phases. They were basically now saying that we are in requirements and design or implementation or testing phase, but they were basically doing all the activities at the same time.

So, scrum is about speed and flexibility. But what does overlapping phases mean?

What new product development suggests is that doing things in a linear manner may not be very effective. We now need to do everything altogether. Let's say you have to develop a product which needs to have five features. You name those features F1, F2, F3, F4 and F5, and to implement each of the features, you need to carry out analysis, design, implementation, and testing.

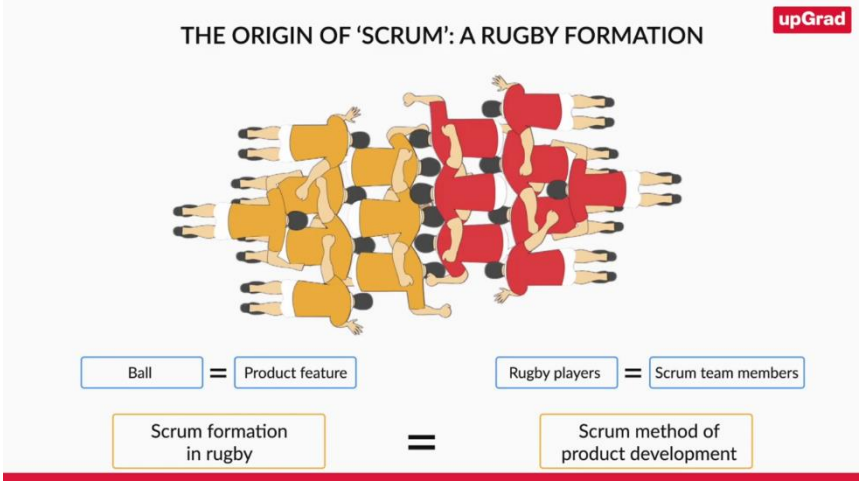


So in a traditional approach, such as waterfall, you would do analysis for all the features first, followed by design for all the features, followed by implementation of all of the features, followed by testing for all the features.



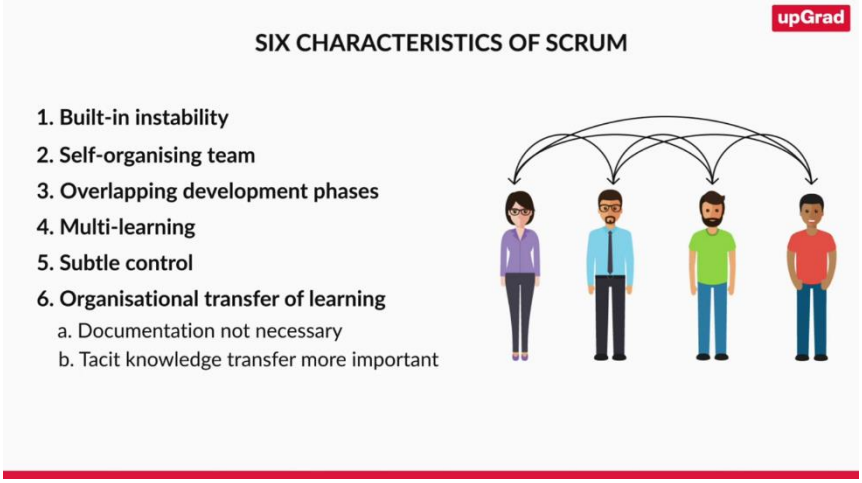
However, in this new approach, you would carry out analysis, design, implementation, and testing, all at the same time for a given feature. Once you're done with all the processes for a given feature, you would do the same for all the features thereafter.

That sounds interesting, but why do we call it scrum? Where does the words scrum come from? What does it mean?



In this article, they said they were using the concept, they said, it looks like the game of rugby. Because in the game of rugby, this is one of the foundations where the team of eight people, they actually come together in a human formation. And instead of in a football kind of a thing where people might be making short passes, in this formation of players, the entire eight people team moves together as a team and carries the ball inside between them. And in game of rugby, this is known as scrum.

So, they said that this style of product development was very much like how the game of rugby is played and specifically how the human formation of scrum happens there. These kinds of product development have six essential characteristics.



They talked about a built-in instability, which means that at a high level, we have defined the vision of the product there, but we haven't really spelled out every single nuts and bolts of that. The team will come together and the team will kind of figure it out how exactly it is to be done.

They talked about self-organizing project team. The basic idea they said was that this was a team which had everybody in the room who could together put the product into life. They were not looking for or seeking approvals or instruction from outside the room.

And the whole idea was to cut down the decision making, to cut down the wait time, to make the team more accountable for their decisions and execution. And that is why having a self-organizing team was the right way to proceed with that.

They talked about overlapping development phases. As we saw that they were not really looking at linear sequential phases, but these were the phases that were collapsed and overlapping with each other.

Further, they also talked about multi learning, which is the learning which is happening at multiple levels. One is across the division. So, I could be in sales and marketing. I could be in product development. I could be in pricing. I could be in customer support. I could be in quality. And all of them are working together.

Secondly, I might be a junior engineer. I might be a product strategy manager. I might be an execution team member, but all of them are again working together. So, the multiple levels of learning that were happening were not left isolated or fragmented. They were all happening together for the entire team.

Then they talked about something known as a subtle control. And the whole notion of subtle control was instead of explicitly checking or micromanaging, the nature of work or the progress of work, the team was putting its own subtle form of peer control or a self-control, if you will. And the idea was that human beings typically tend to become more accountable in those conditions rather than situations where somebody is necessarily checking them.

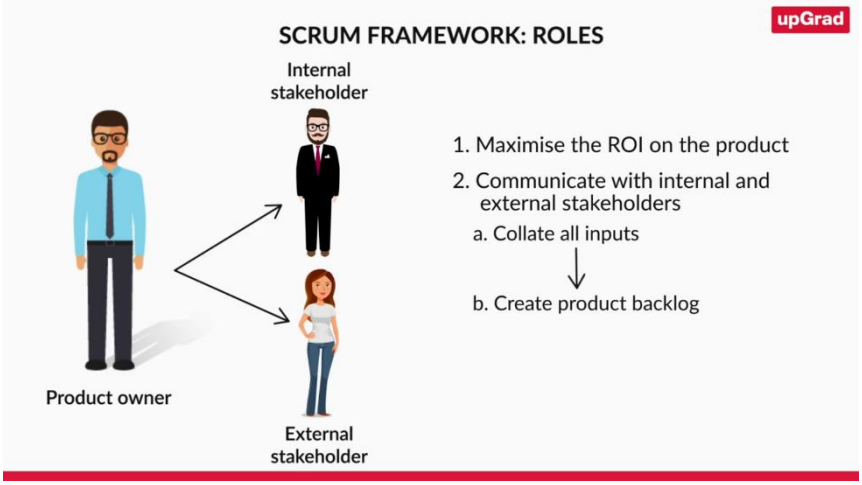
And finally, they talked about organizational transfer of learning, and they also talked about that the transfer of learning is not necessarily in the form of writing documentation, but also making sure that there are multiple ways of transmitting that tacit knowledge among each other. So, these were the six characteristics that they talked about in their article in 1986.

In this video, you learned that scrum is one of the agile methodologies that is followed by the majority of agile practitioners. In scrum, instead of undertaking one product development process at a time for all the features, you would go one feature at a time and conduct all the steps for it. Then move on to another feature.

The term scrum comes from the game of rugby in which players come together in a formation and try to get the ball. Scrum has a number of features such as built-in instability, self-organization, overlapping development phases, multi learning, subtle control, and transfer of learning. In the next video, we'll dive deeper into the scrum framework. See you then.

In the previous video, you learned about scrum and its characteristics. Now let's get into the details of the scrum framework. What are the roles different people play in this framework of product development? Let's find out.

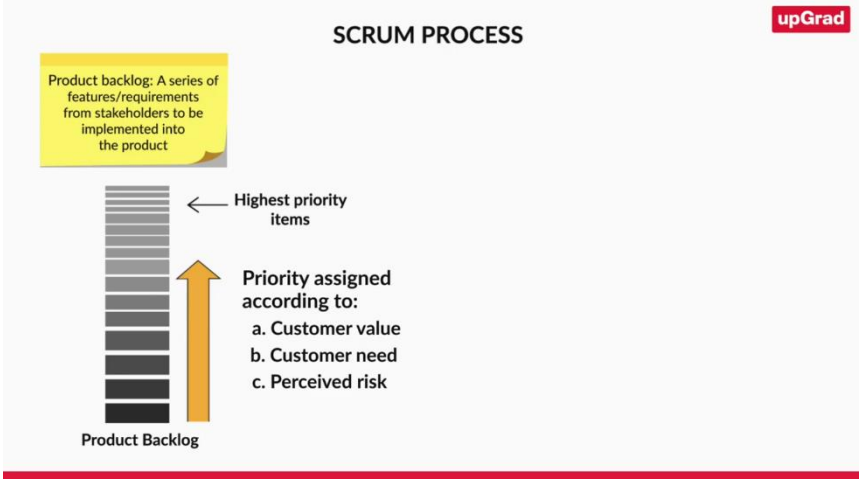
When we talk about scrum, I will explain you the whole framework and we will talk in detail about the roles and the artifacts and the events. But let me explain you the framework right now. The first role that we talk about in scrum is known is the product owner.



The product owner is all about maximizing the ROI and the product. If I have to put a hundred dollars into building a certain product or even building a feature, how do I decide that I will get the maximum out of these hundred dollars? Maybe I have four options.

Maybe I can put some money and I can get $60, I can make $120, I can make $130 or I can make $200. How do I decide what is the payoff in each of these features? How do I decide which is the right investment to be made? This is in a very short form the role of a product owner, maximize the ROI of the product.

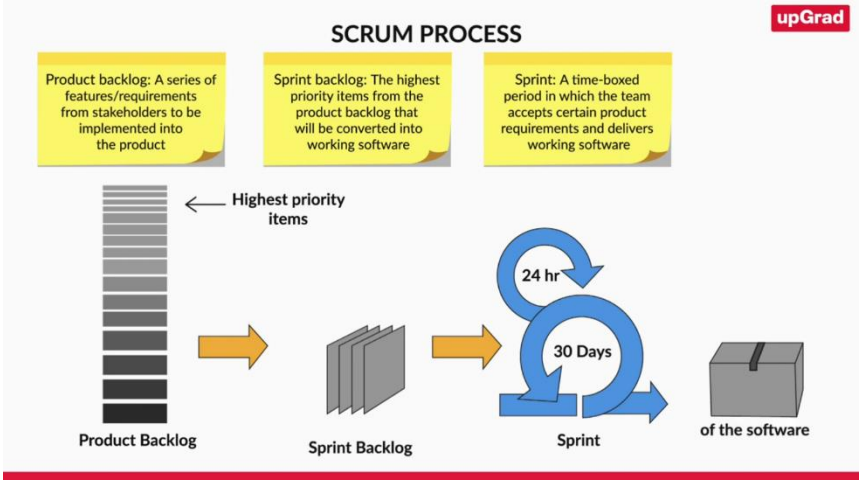
What do the product owner do? They are operating mostly in a network scenario. That means they are talking to stakeholders from inside the organization and outside the organization. And they're collating all these inputs. And these inputs are then translated into something that we call as a product backlog.



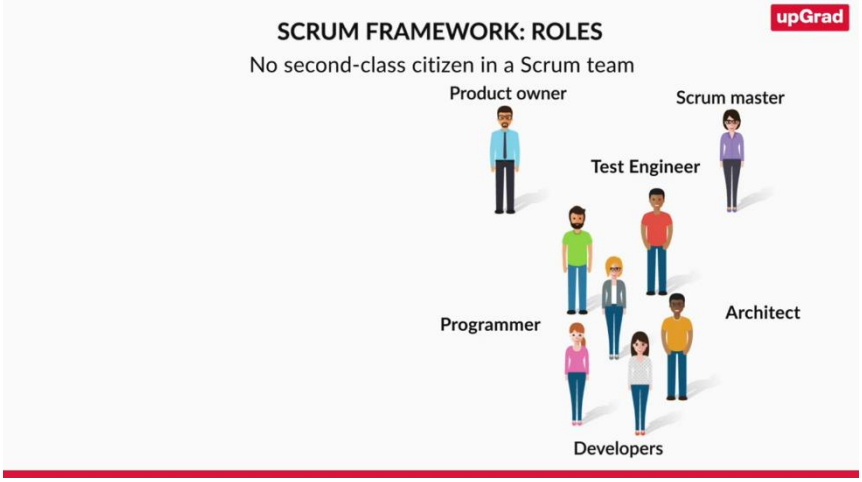
A product backlog is essentially a series of feature requests or requirements from the customers that the product owner would like to see in the product at some point in time. They are ordered in only a certain priority.

This priority could be determined by customer value, it could be determined by the need from the customer. It could be determined by the risk that is perceived, as long as there is a uniform way and as long as there is a very clear stack order in which these items can be linearly ordered against each other.

Now, as you can imagine, when you order something based on certain priority, there will always be the top most item which have the highest priority. And these highest priority items are the ones that the product owner would like to implement in the most immediate unit of delivery. That immediate unit of delivery is what we call as a sprint.

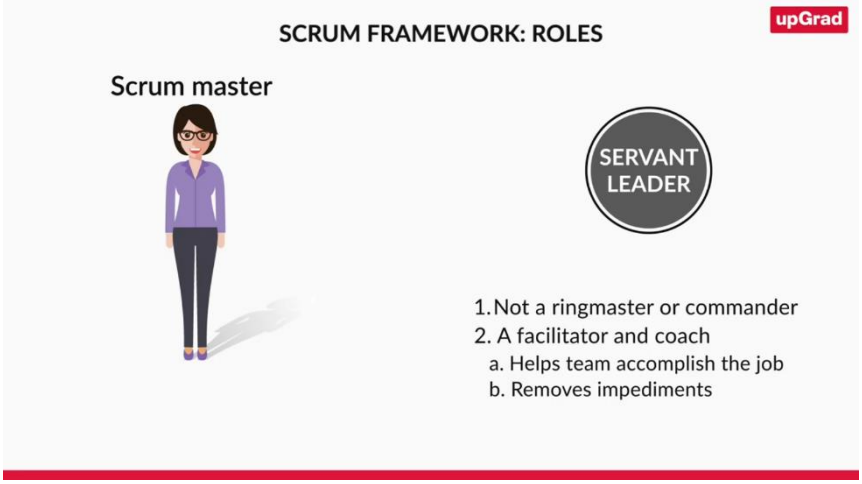


Basically, sprint is a period of time in which the team accepts certain part of the requirement and delivers working software. Now, the part of the product backlog that the team accepts is known as sprint backlog. So, what is sprint backlog? It is the highest priority items in the product backlog that the product owner would like to get translated or converted or created into working software.



We also have another role in scrum known as a scrum master. Everybody else in the team in a scrum team is known as developer. Irrespective of whether you are a programmer, whether you are a test engineer, whether you are an architect scrum calls everybody as a developer.

Why does it do so? It does because it believes there are no second-class citizens on a scrum team. Everybody has equal opportunity, equal responsibility and equal rights to be a part of the scrum team. Scrum is all about team rather than individual. And that is why, it has so much a focus on the team and the term developer.



The one role that is separate as we were talking is the scrum master. Now scrum master might look like a ringmaster, but it has nothing to do with really command and control or somebody really controlling and whipping around people.

A scrum master actually is a facilitator, is a coach for the team. In fact, the term that we use to describe that is servant leader. They are the people who really go around and help the team members accomplish their daily jobs. They are the people who will remove the impediments from the team members pathways, just so that not everybody has to go out and fight the fire, but people can focus on their work while the scrum master goes, talks to the people and remove the impediment.

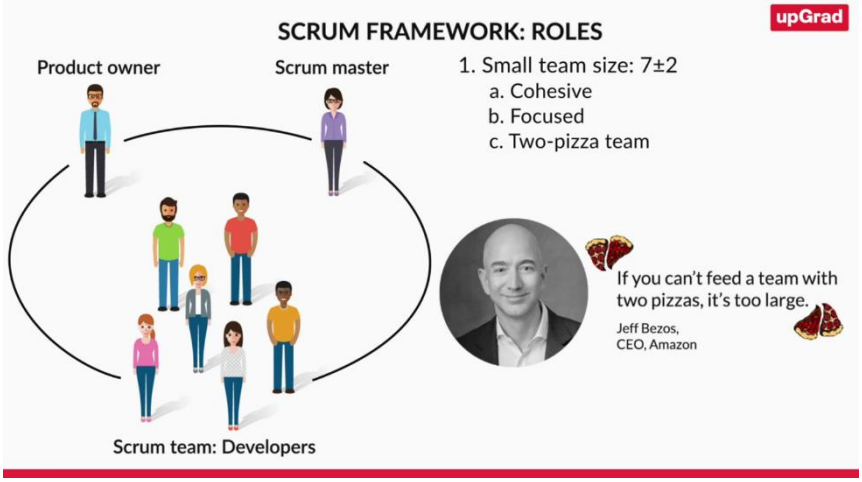
This video highlighted the key features of scrum framework and two important roles within the scrum team.

- First, the product owner is responsible for collating the input from all the stakeholders and translating them back into the product backlog, which is a log of user requirements in the form of user stories or epics.
- The highest priority items from this product backlog are transferred into the sprint backlog and are delivered through sprints.
- The second role was that of the scrum master, who is the facilitator for the scrum team and can be described as the servant leader.

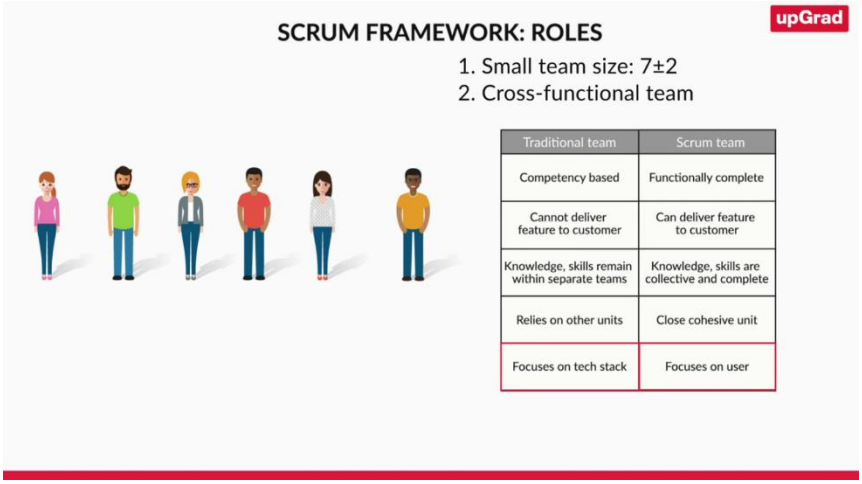
After this, we'll learn more about the rest of the scrum team in the next video.

The previous video gave us important information on two roles in the scrum team, the product owner and the scrum master. But what about the rest of the team? What are the roles and specifics related to other team members? Let's check out the answer.

So, the scrum teams are small teams. Now, what do we mean by a small team? There is no mathematical definition of a small team, but typically in scrum, we say a team size of seven plus minus two is a desirable team size.



Why do we care about a small team? We believe that the small teams are more cohesive. They are able to come together without losing a lot of focus in that process. Amazon, in fact, calls them as a two-pizza team, which is you can get two pizzas and you can feed the entire team. So, that's first characteristics of a scrum team.



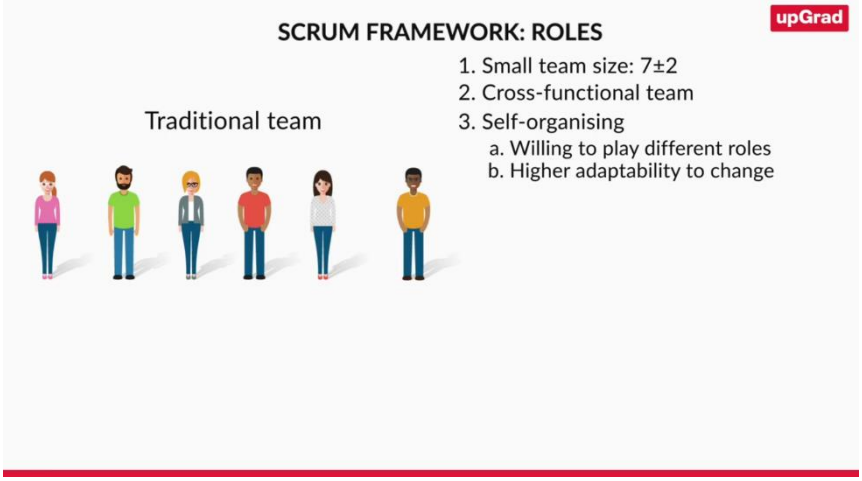
The second thing that is important is that they are cross functional team. That means unlike traditional teams, which is there is a UI team, there is a database team, there is a backend team, there's a middleware team and so on, a scrum team does not believe in really having competency-based teams. They believe in having the complete team that can deliver the entire feature to the customer.

So, I might have a team of eight team members. There are two of them, Java programmers, and other two might be PHP skilled engineers. There are two who know selenium, for example, one of the team members might know how to do user experience design, yet another might be expert in doing the production operations and pushing the cohort into production and running the data centres and so on. But together everybody has collective knowledge and skills to ship the feature out.

Why is it important? It allows the team to function as a close cohesive unit, rather than relying on some other units and saying, Oh, we have done the UI module. Now we are waiting for backend team to give us the module. This results in a loss of agility at an overall picture. While you might have delivered your UI module very fast, but unless you can

get the third-party integration or a middleware component integrated, you cannot deliver the feature to the customers.

So, the scrum team as a cross functional team is the capability to basically deliver features that are centred from a user point of view and not from a technology stack point of view. This is a big change from the traditional thinking.



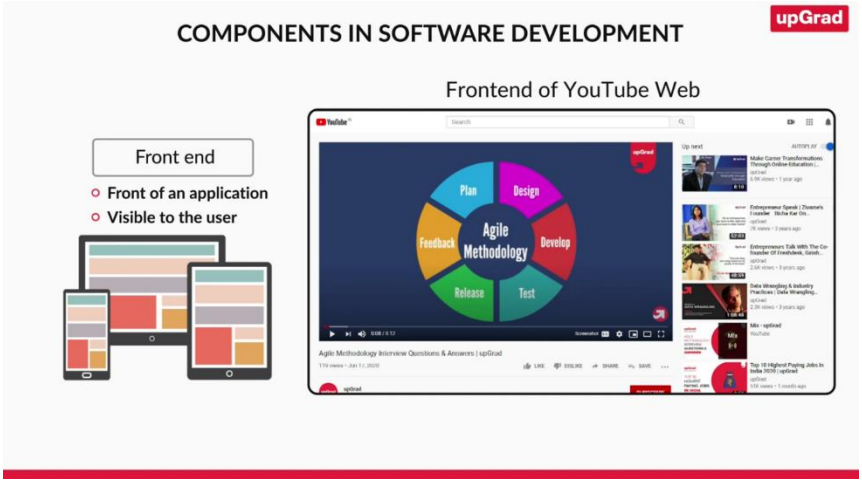
One more word about the scrum team. They also strive to be a self-organizing team. Now, what is a self-organising team. In a traditional team, people typically take a position and say, I am a programmer. I am a test engineer, I'm a database architect and so on, and they don't change their position a lot.

But it's unlike, for example, you might have a fixed position when you play the game of cricket or when you play the game of football or when you play the game of basketball, but a good team, a self-organizing team is something that is able to, where the players are willing to change their position and play a another role or another position because that's the right thing to do for the team. And that is a true self organising team.

A self-organising team has higher adaptability to the future changes because we cannot predict them. We cannot plan for them. But the best thing we can do is have the right mindset of people, have the people who have a varied set of competencies and build a self-organising team. So, that is about the composition of the team.

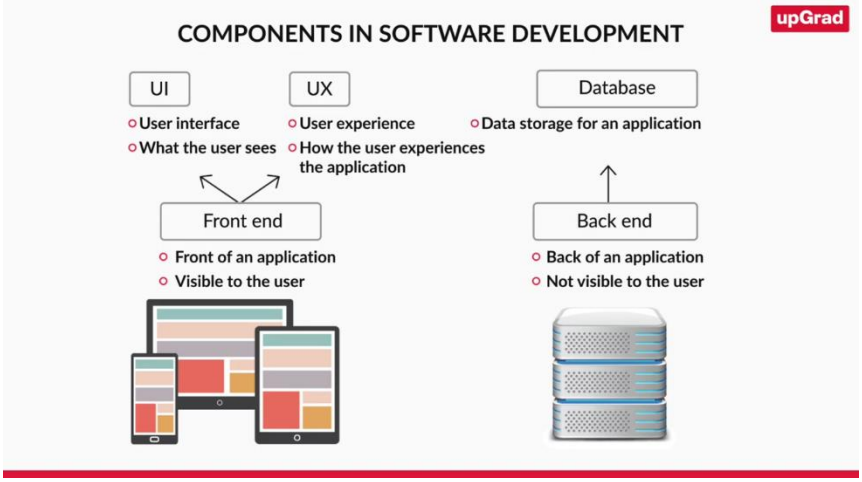
Let's take a break here. Our subject matter expert mentioned a few technical terms. Let's find out what they mean in the next video.

Okay. Picking up from the tech terms, our SME spoke about something called technology stack, what is that? What does frontend, backend, database and middleware mean anyway? Let's ask our tech guy.



What does front end mean? Front end just means, as the name suggests, is the front of an application or what the user sees. Let's go back to the YouTube example. We talked about frontend in brief. Let's understand that better.

So, when you go onto YouTube, we see an interface where we can watch videos, like and share them, add them to favourites and do a lot of things. YouTube also has mobile apps which look different, but finally allow us to do same things. So, these interfaces are known as the front end.

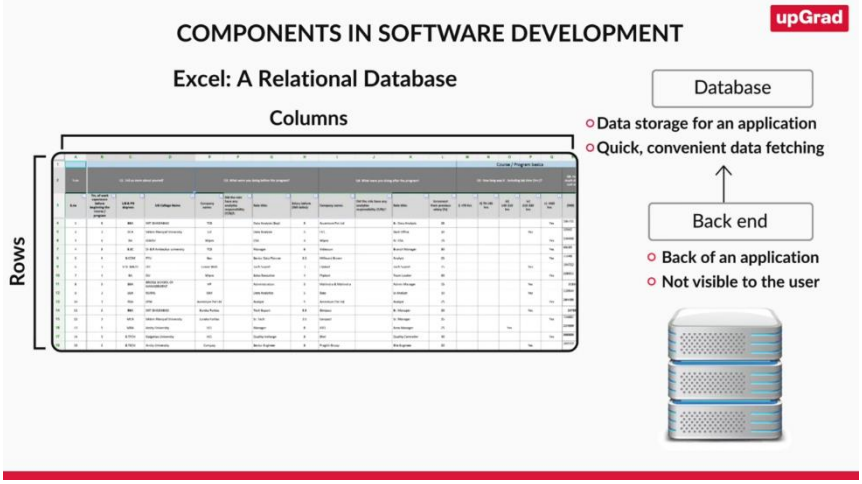


Now what is UI and UX? UI stands for user interface, and that consists of exactly what the user sees. UX goes a little deeper. UX stands for user experience and relates to the experience that the user gets when they use the application.

So, when creating an application, you can create a beautiful UI, but if the application is filled with errors and bugs, or is slow or consume a lot of data, it will have a bad user experience. Front end development covers ensuring great UI as well as great UX.

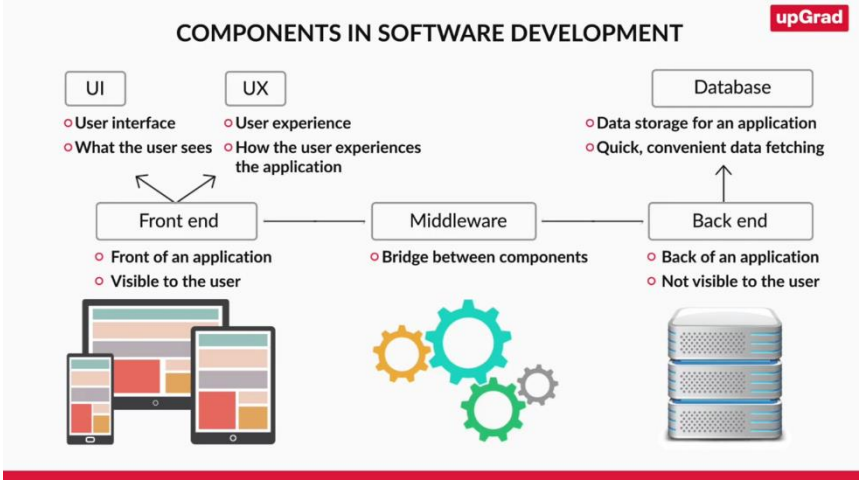
Moving on, what is backend? I'll take again take YouTube as an example. We see a great interface when we use YouTube, but we never get to see where the videos are stored or how they are stored. That is where the backend comes in.

Again, as the name suggests, backend is the back of an application, something that is not visible to the user. This backend is responsible for doing all the background tasks, storing data in the database, fetching data requested by the user and so on.



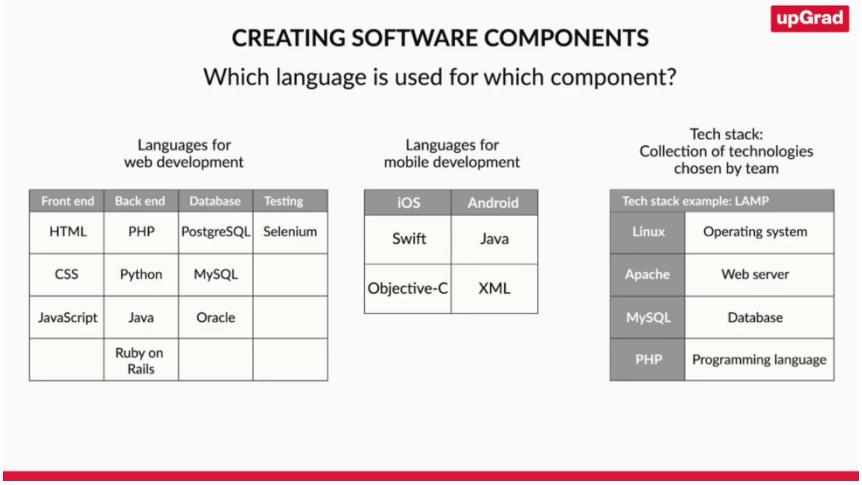
Okay. I mentioned database right now. What is that? Database is a place where all the data of an application is stored. Most of you know, what an Excel is, right? It is an implementation of a database or what we call a relational database.

To explain a relational database, it's a collection of data formatted into different tables, which are formed the rows and columns. Yes, exactly like Excel. A database makes it easy to fetch data quickly and conveniently.



And finally, middleware. So, we learned that we have a front end, a backend and a database. But how do these components interact with each other? Middleware is a bridge between these components. It is not necessarily that every application needs a middleware, frontend and backend directly can interact as well.

We talked about all the components, but how does a developer create these components? Well, that's an obvious one. They write code. But what code do they write? What language do they write? That's where these competencies come in.



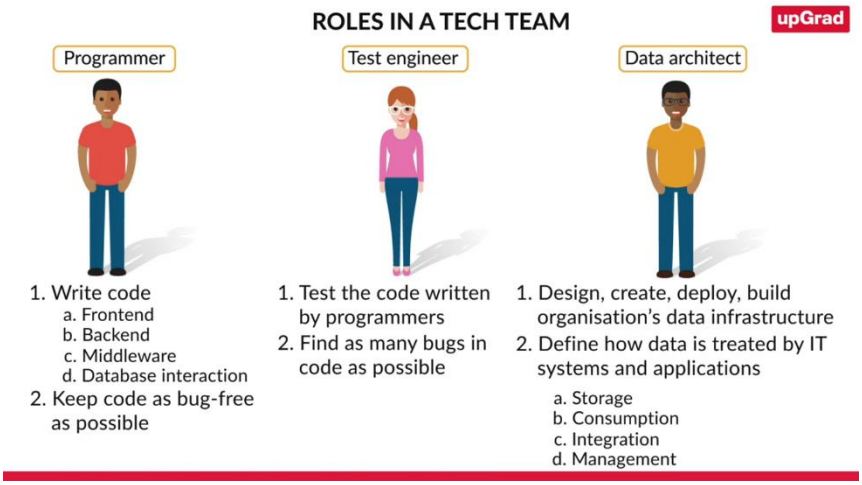
Every component has a corresponding language to write in. For example, the most common language to create front end applications are HTML, CSS, JavaScript, etc. The backend includes a web server that hosts an application built with languages like PHP, Python, Java, Ruby, and rails, etc.

Now they communicate with the database using something like PostgreSQL, My SQL or Oracle to serve up the information that the front end presents. And finally, selenium is a portable and convenient tool that eases the task of testing web application, in sort of an automated way.

On the mobile front, these languages differ but the whole process remains the same. For example, iOS uses Swift as main language or objective C. For Android, the languages are Java and XML. We'll cover all of this in upcoming modules.

As we have seen for each layer of web application, there are many choices available. When a team decides on these choices, the collection of technologies they choose is collectively called as tech stack.

One such example of a commonly used tech stack is lamp, LAMP, which stands for Linux in operating system, Apache, which is a web server, MySQL, which is a database and PHP is a programming language. Using these four components, we can now build a self-sufficient system, which can be used to build an application to share images online, watch videos, anything.



So, now we know what the components are and the commonly languages for these components. Now coming to people, programmers are people that write code, be it front end, backend, middleware, database interaction, anybody that writes code is a programmer.

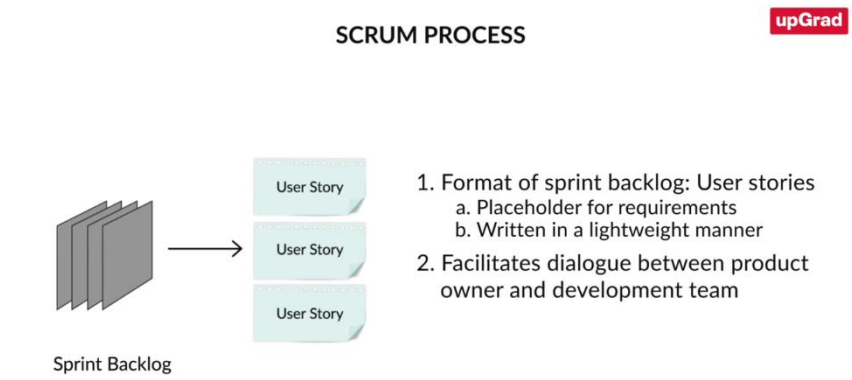
Test engineers on the other hand are responsible for testing the code that is written by the programmers. It's the task of a programmer to keep their code as bug free as possible. And it's the task of tester to find as many bugs as they can. And the eternal battle between programmers and testers continue.

A data architect is concerned with designing, creating, deploying, and building an organizations' data infrastructure. Data architects define how the data will be stored, consumed, integrated, and managed by the different data entities and IT systems, as well as any applications using our processing that data in any way.

So, in these two videos, you learned that a scrum team usually consists of five to nine developers and is characterized by its traits of cross functionality. That is, the team has the capability of delivering a feature that is ready to ship to the customers. It is also self-organizing where team players are ready to dawn multiple roles if required.

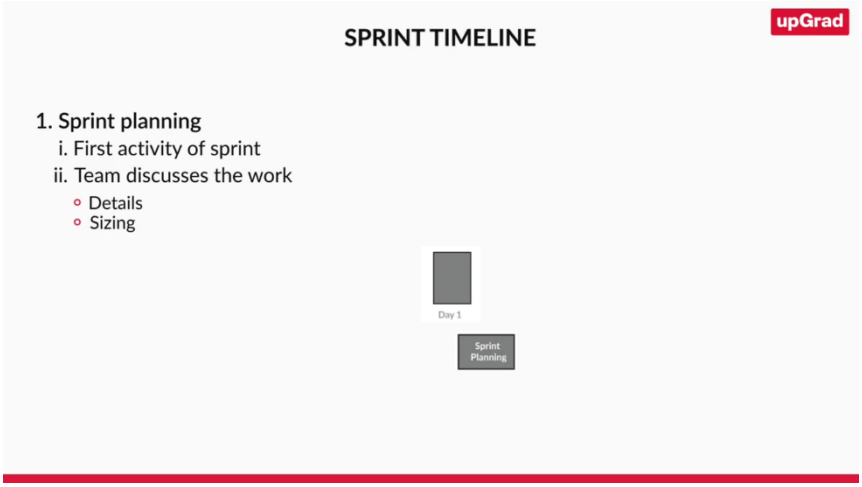
You also learned that in software development, front end is what the users see, backend is where the data is stored and retrieved. Database is an organized collection of data and middleware is the bridge between them. So, next up, a dive into sprint and how it is conducted.

So far, you've understood that scrum has two roles, product owner and scrum master, and the rest of the team are called developers. Now let's move on to sprints. So, how is a sprint conducted? What are the different steps involved in conducting sprints?



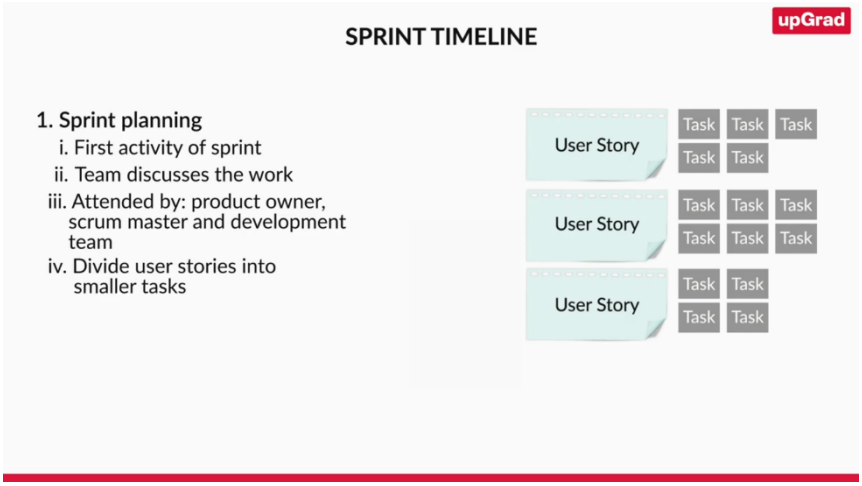
So, the product owner has identified the sprint backlog. What dis the sprint backlog? These are essentially requirements that are identified and written in a very specific lightweight format. And we call this format as user stories.

The user story is a placeholder for requirements. It's an invitation from the product owner to the development team to come and have a conversation about one of the user requirements. Now, these are specified in a very lightweight manner because the idea is not to describe everything on a piece of paper, but really have a face to face conversation between the business and the technology, between the product owner and the development team.



During the sprint planning, which is the first activity on the day of the sprint, the team comes together, they discuss their work. If they have enough details about it, they do the sizing of their work. And then eventually they do the planning of the work in terms of, Hey, we are going to have two weeks of sprint, we have these many hours available and this is the effort needed to complete the stories and the tasks. How much can we sign up for?

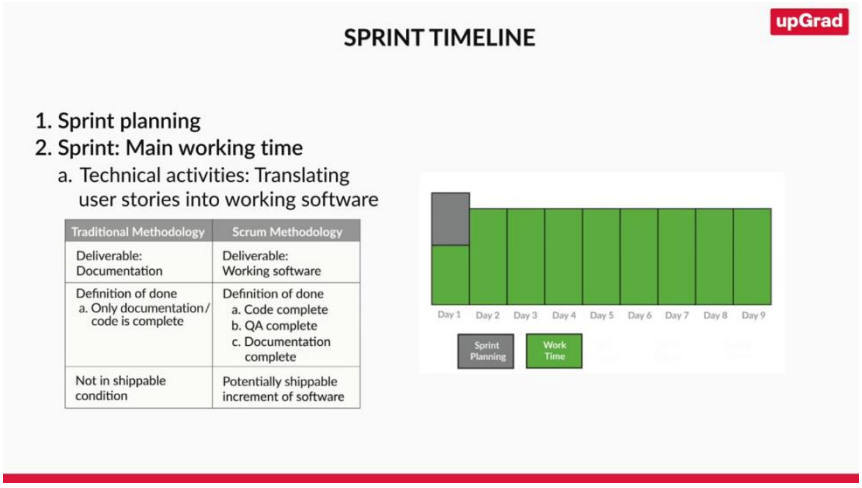
So, sprint planning happens on the first day of the sprint. Let's understand this better through an example from BookMyShow.



So, there are a couple of things that happen at BookMyShow when we do a sprint planning meeting. First of all, the attendees at our sprint planning meeting or the scrum master, the product owner and the entire team.

So, essentially what happens is the entire team takes items from the backlog and adds it to the sprint. Now, while these items are added, once they go into the sprint, they're further broken down into smaller tasks that need to be done. And that's essentially what happens simply at a sprint planning meeting.

Okay, now that sprint planning is clear, let's get back to learning how the rest of the sprint is carried out.



At the end of the planning, the team comes up with a very clear picture of how much work they can commit back to the product owner. From the next day onwards, the team really starts working on the technical activities, and these technical activities are all about translating the user stories into a working software.

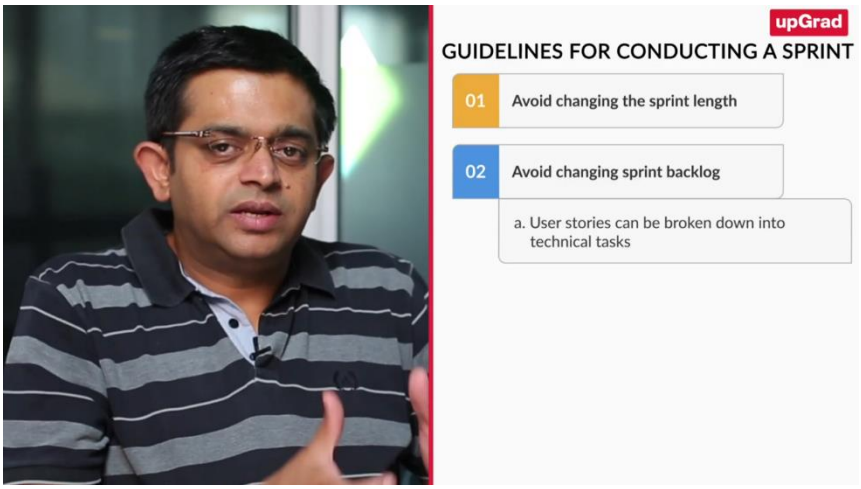
Scrum is extremely high on delivering working software at the end of the sprint. Unlike the traditional methodologies where the deliverable was a documentation, here we are talking about working software as the key deliverable at the end of it.

Now this is again a big departure from the traditional software development where we have developed documents, but we have never delivered code. And even if we have delivered code, we have said, Oh, it is code complete, or it is feature complete and so on.

On the contrary in scrum, when we say a software is done, we don't simply say done. It is actually done-done. Or in some cases, you might say it is done, done, done. What does it mean? It means as a developer, it's complete, as a QA it is complete, as a documentation it is complete. Everything has been done. Nothing is needed between now and releasing it to the customers. That is why we call it as potentially shippable increment of software.

We call it potentially because while the feature may not be complete, every ingredient into it that has been put into is complete there.

Now during the sprint time, the sprint could be on between one week to four weeks. That means the team can decide we want to do a sprint within one week or two weeks or three weeks or four weeks. There is no such rule in scrum that says, everybody must follow only one sprint length. What I've seen is a lot of teams start with four weeks and they find they settle down for two weeks.

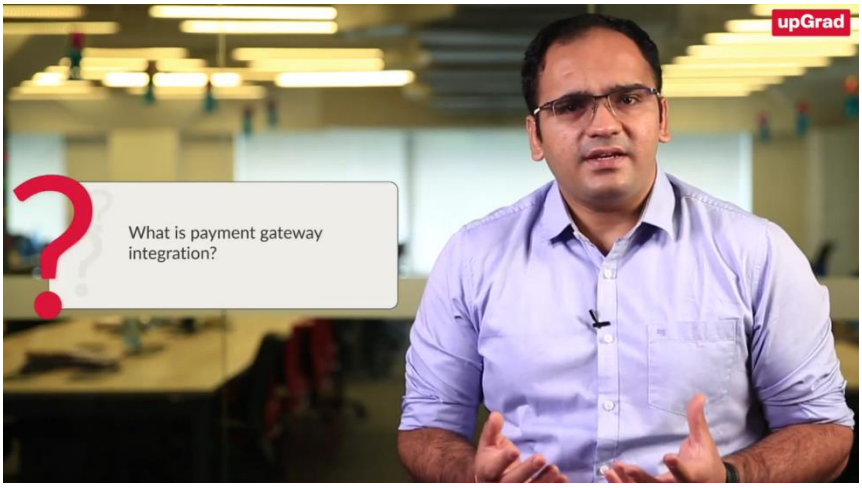


The key thing in scrum is once you decide the length of a sprint, you do not change it. Why do we don't change it? The simple reason is every time you will have to reset your mental filter of how much work you want to commit.

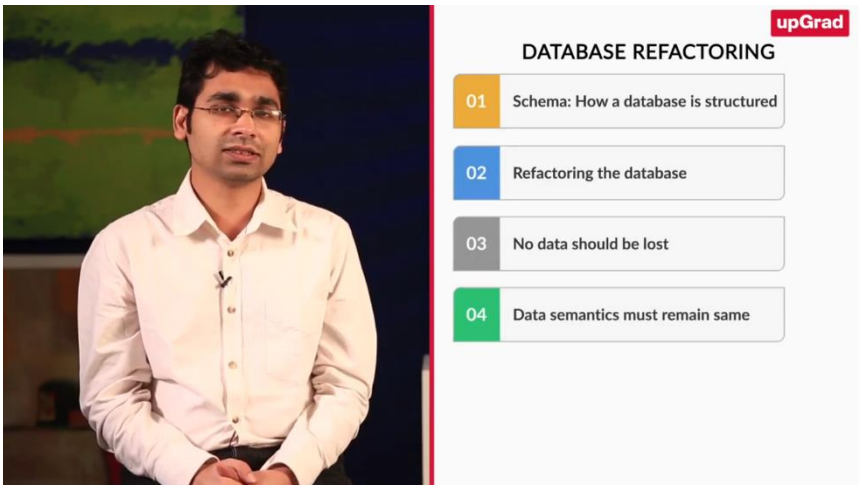
The other rule that is there in sprint is, once the sprint gets underway, nobody can change the content of the sprint backlog. Which means if we have these five stories that we need to complete, neither the product owner, nor the team can change the stories.

Now with that said, the stories themselves will be broken down into the technical tasks. For example, if I have to implement a webpage with a one click checkout, it will have a UI widget. It will have a webpage. It will have an integration with the payment gateway. It will have some kind of a login module and so on.

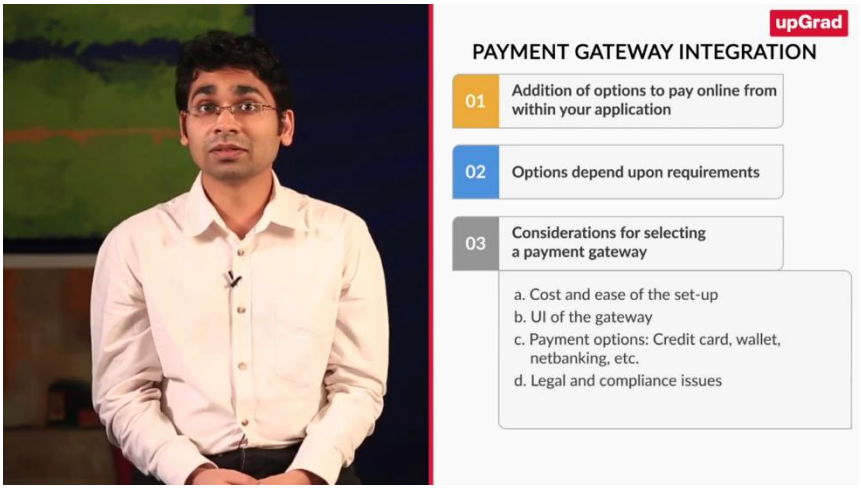
Now, as a part of implementation of that story, we might suddenly discover, oh, we have not taken care of refactoring the database because we need to add one more column. That is acceptable. But we are not changing the story per se. We are only discovering new tasks that are associated with the story. And that will happen because that is the nature of the software development.



Hold on a bit. There were a couple of terms that I'd like to understand better. What is payment gateway integration? What is database refactoring?



We know now what databases are. Every database has a structure or schema as it is called, how its table are structured, what kind of data each column stores, etc. Refactoring a database means making some changes to the schema of the database or to improve its design or effectiveness, while ensuring no data is lost and the semantics for the data is not changed.



And what is payment gateway integration? Simple, adding the option to pay online from within your application. You know how when you are on Flipkart or Zomato, you place an order and you get the option to pay online using your credit or debit cards. That is because these websites or applications have integrated the payment gateway into their application.

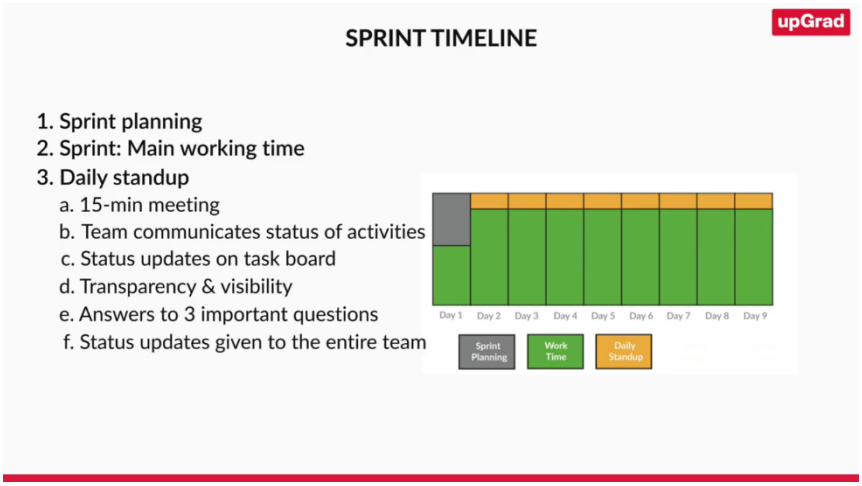
These days, adding a payment gateway is not difficult. It just has a small cost associated to it. In fact, there are several options for payment gateway integrations, and the choice depends on the requirements. Some of the examples are CC Avenue, Pay U Money, Instamojo, etc.

While trying to decide your payment gateway, keep in mind the following considerations:

- Cost and ease of setup.
- User interfaces of the gateway.
- Payment options like credit cards, wallets, net banking, etc.
- Legal and compliance issues.
- After sales support.

Okay, that's much better. Now we know what payment gateway integration and database refactoring mean. More on sprint in the next video.

Previously, we started with how a scrum team conducts a sprint. Now let's get back to the sprint and its components.

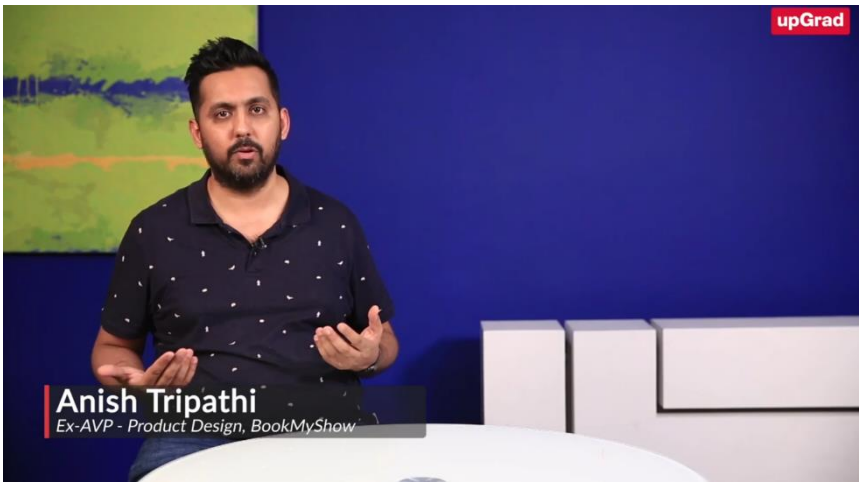


During the time the sprint is on, there is one more activity that the team does on a daily basis. They come together every day for a period of 15 minutes and do something known as a daily scrum or a daily stand-up. In that meeting, they all come together and they communicate the status on activities that they have signed up for.

One more thing to remember in scrum is nobody allocates the work. So, the question might be, who allocates the work then? The short answer is people select their own work. In scrum, we call it as open allocation. So, now if I have chosen some piece of work, I go and update the task board. In scrum, we believe in using a very high level of transparency and visibility about the work.

Everybody essentially answers three questions. What did I do yesterday? What am I going to do today? And what is blocking me? A lot of times people mistake it that this is the update they are giving to the product owner or to the Scrum Master. Both are wrong answers. The right answer is they are giving the status updates to the team.

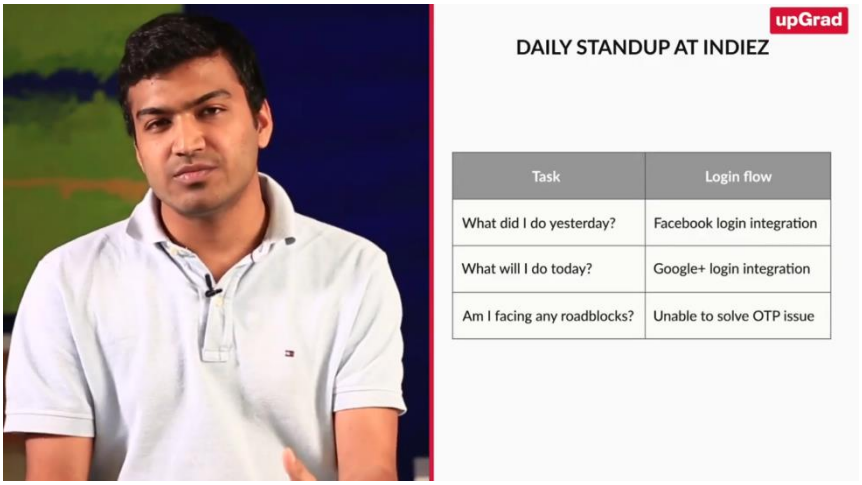
After a bit of theory on the stand-up meeting, let's take out an industry example from BookMyShow.



We regularly have daily stand-ups at BookMyShow. And if you come at any day in the morning to office, you will realize that there’s groups of about 10 or 15 people, which are part of a squad and a scrum team. And they typically stand around in a circle facing each other, which is very important.

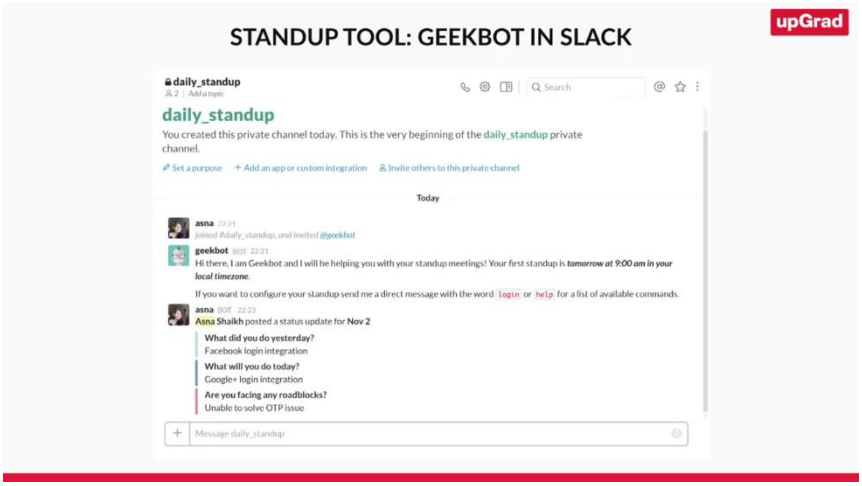
And they usually discuss about the impediments that they're facing or the trouble they're facing, or maybe there's an achievement, you know, they need to talk about, or it's a very open discussion that needs to happen every morning, just to get a sense of where we are and it's like a check. So, I think daily stand-ups extremely important, and they're very healthy for the progress of the product.

Let's check out another example of how you can conduct stand up meetings, this time for Indiez.



Everyone works remotely in Indiez. So, stand-ups do a phenomenal job for us to know what exactly that person is working on, what you did yesterday and does easy facing any blockers. Everyone has to update everyone on, A, this is what I did yesterday. This is what I'm going to do today. And I'm facing this kind of blocker.

So, for example, let's say, if I am working on log-in as of now, so I did what I did yesterday, I integrate Facebook login into a plot. What I'm working on is I'm going to add Google plus log in to our plot. Any blockers that I'm not able to solve the OTP issue. This is how we conduct any stand-ups.

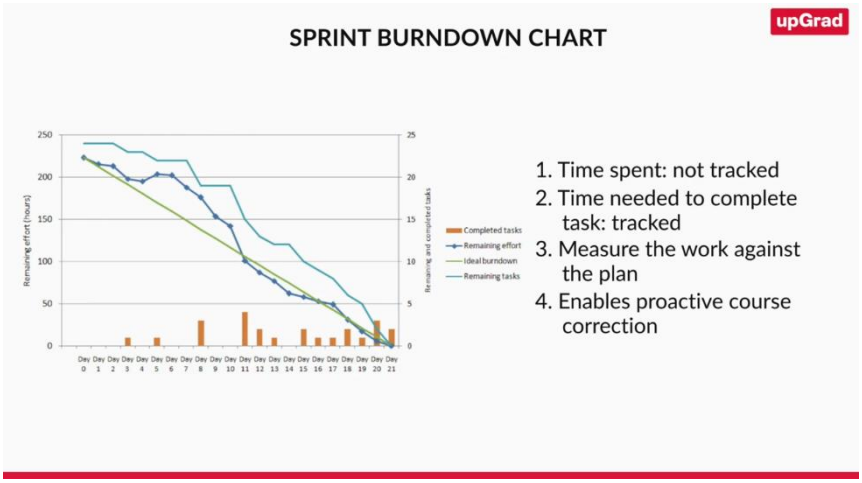


Now we use amazing tools again for daily stand-ups. So, I came across one phenomenal chat bot for Slack, which is called geek bot. What it does, it automatically asks everyone questions and you can choose the question according to yourself.

So, it helps you ask questions like, Hey, @Nitesh, what did you do yesterday? Hey @Nitesh, what are you going to do today? And hey @Nitesh, are you facing any blockers? And at the end of stand-up, what it does is, it summarizes everything in one go and push a summary on that particular channel.

So, as we just saw through examples of BookMyShow and Indiez, stand up meetings can be very useful to keep the team informed on the tasks and issues, if any. Now let's get back to the other aspects of sprint.

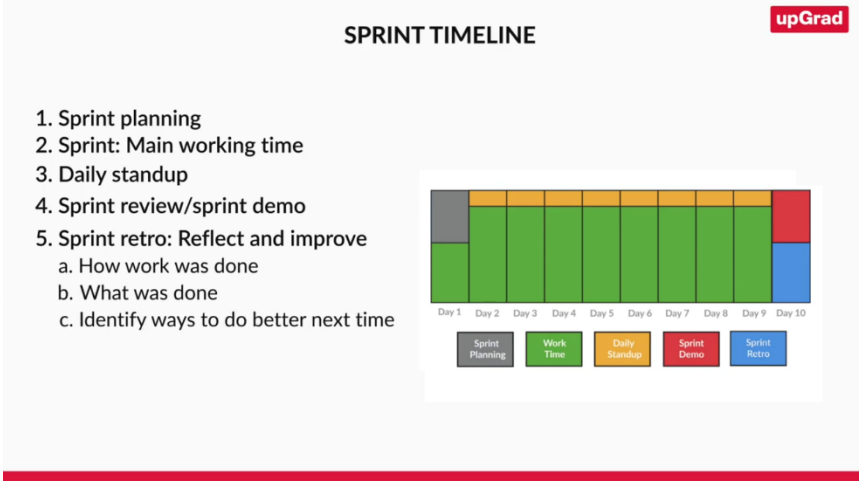
Unlike a traditional project, we don't track the amount of time that we have already spent because we believe that is a sunk cost. It is of no value to anyone. What is more important is to track how much more time is needed to complete the task.



Now that when you track it on a daily basis, it allows you to track the progress of the sprint in what is known as the sprint burndown chart.

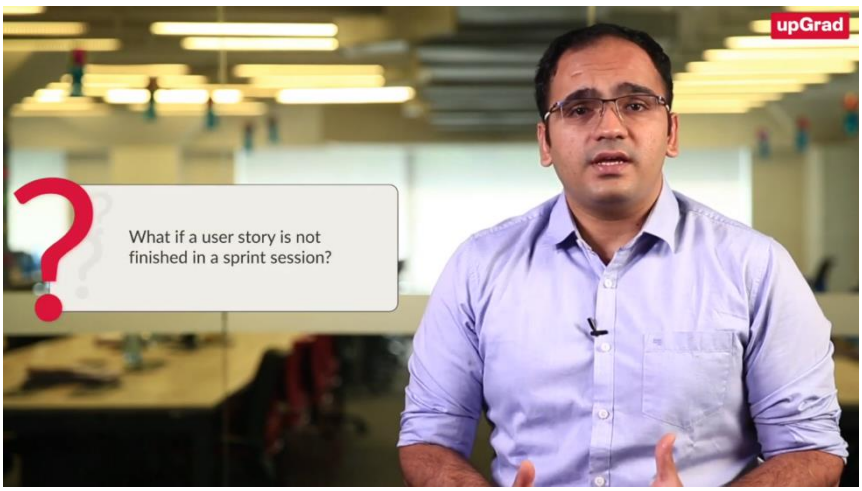
Again, we'll go into the details of that in due course of time, but it's important for right now to know that because the sprint itself is a very short duration sprint, often two weeks to four weeks, unless we track it on a daily basis, we might find that suddenly the project plan is totally out of control.

The idea is by doing a daily tracking, we are actually measuring the work against the plan on a daily basis. And if there is a need for a course correction, we are able to take proactively an action to align that with the project goals, sooner rather than later.



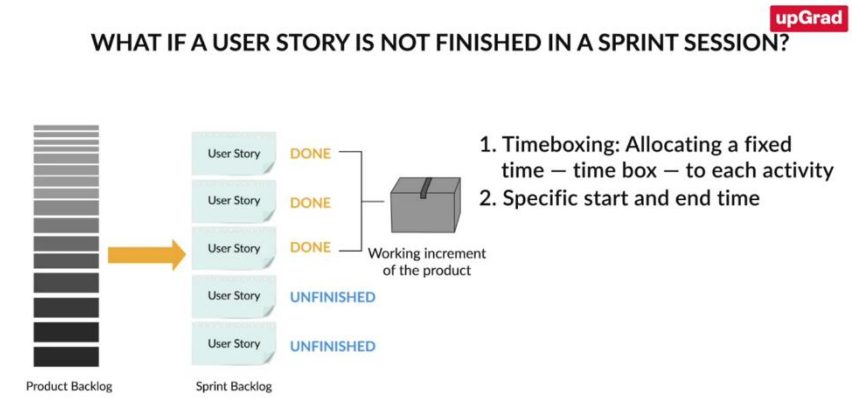
And at the end of the sprint, when the work gets completed, we demo the increment of the product back to the product owner. And we call that as a sprint review or a sprint demo. In that demo, the product owner has an opportunity to look at the work, understand whether it meets the acceptance criteria and make a go or a no-go decision.

The last ceremony or the event in a scrum is what we call a sprint retrospective often called as a retro. A retro is an opportunity for the team to reflect upon how they did, what they did, and identify from their opportunities to do it better in the future.



Okay. I have one last question. What happens if a user story is not finished in a sprint session?

Yeah, that's a great question. A lot of times, in our traditional projects, we are used to the scenarios where we have a deadline, certain activities were to be completed by that deadline. The work is not completed. And guess what, our default behaviour is, let's extend the deadline.



Scrum takes a little different call. Time Boxing is what we really believe in, which means once you decided that everything is a time box, it has a specific start and an end time. Let us say we started with five stories and only three have been completed.

Now those three which have been completed, obviously become a part of the software because they have met the acceptance criteria and they have met the definition of done. What happens to those two stories? We never extend the time box in scrum. The reason is we believe that once you extend the time box, then you will get into the habit over a period of time. And then people find it very easy to move the time box.

So, what happens to those stories? Those stories go back to the product backlog. They become a part of the common pool of all the stories in the product backlog. So, by default, it only goes back to the product backlog and it's up to the product owner like anything else, how do they want to prioritize and when.

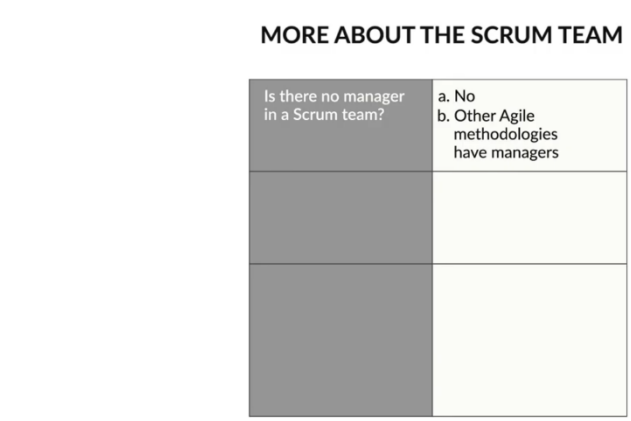
These two videos gave you a lowdown on the various steps involved while conducting a sprint, where the key deliverable is a working software.

On the first day, the team conducts a sprint planning session where all the requirements are discussed and a plan is chocked out.

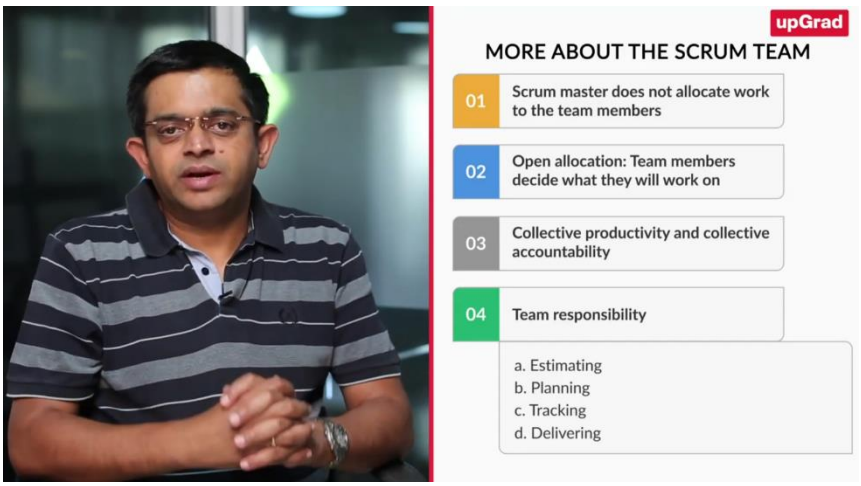
Then the team starts working and every day a stand-up is conducted to get everyone on the same page. The time that is still needed to complete the tasks is recorded in a sprint burndown chart. And at the end of the sprint, there is a sprint demo in front of the product owner.

Lastly, there is a retro where team identifies opportunities to work more efficiently the next time round. Next up, answers to a few more questions about scrum and sprint.

You learned all about sprint in the previous videos. Now let's get back to the scrum team because I still have a few questions I want answered. For one, is there no manager in a scrum team?

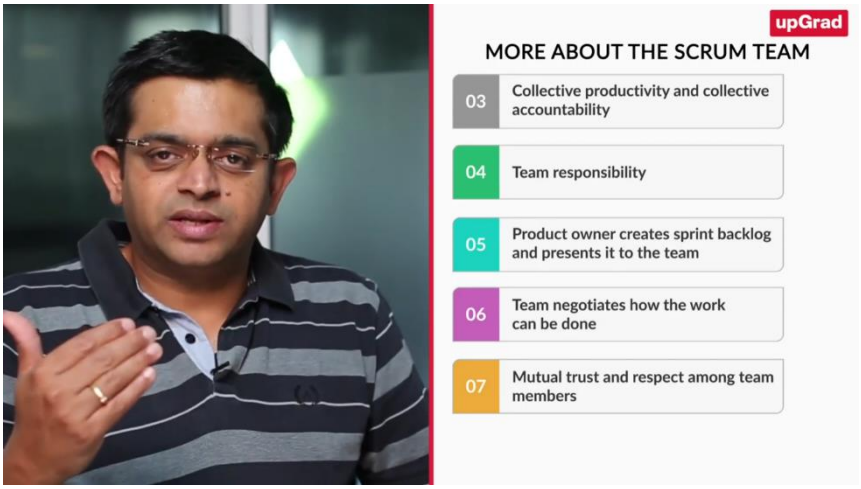


A lot of people look at scrum and say, there is no role of a manager in agile, that is wrong. Scrum is only one of the methodologies in agile and scrum is the only one that does not define the role of a manager. All other methodologies within agile do have a role of a manager. They have a different social structure and scrum has a different social structure.



The scrum master is not the manager of the team. Nobody reports to the scrum master. The scrum master does not allocate the work to the people. The scrum master does not decide who will be on a given task, who will work on this particular activity. In fact, in a scrum team, it's a team that decides what will they work on.

On a scrum team, we don't have the notion of individual productivity or individual accountability. It is the entire team that is collectively responsible for estimating the work, scoping the work, planning the work around it, making the commitments, tracking it on a daily basis and eventually delivering it to make sure the maximum chances of meeting sprint goals are accomplished.



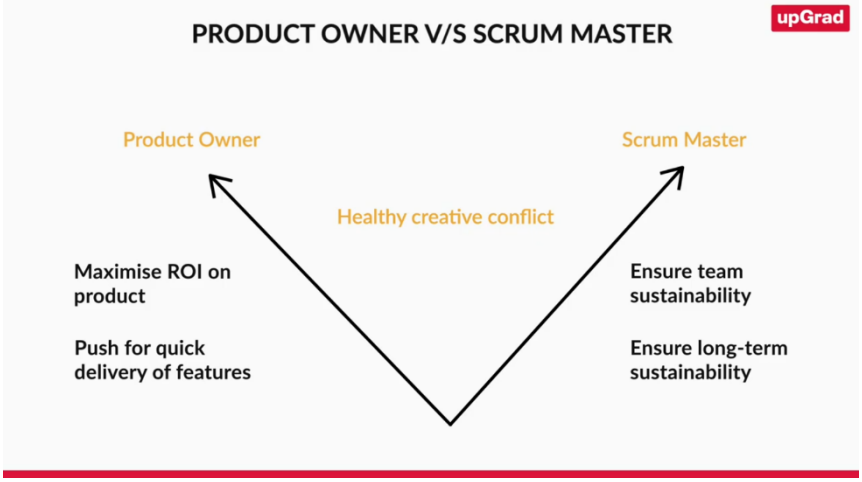
We saw the product owner has created the sprint backlog. The team has been presented with a sprint backlog. The product owner and the teams are basically negotiating on how much of the work they can do.

The product owner is responsible for defining what is it that is needed? What is it that I need? And the team is responsible for specifying how am I going to meet that? This contract is important.

If the product owner goes into specifying, how should you do that, then they are actually getting into the domain of, or getting onto the turf of the development team. They must trust the development team because they are the experts in technology and designing the product there.

At the same time, the development team must have the similar kind of a trust into the product owner's capability to understand what are the most important products. Obviously, they can ask questions to clarify that, but there has to be a mutual trust and respect for each other's function.

So, our SME has mentioned that there is a product owner and a scrum master. In the Indian ecosystem, especially where does the PM fit into the framework? Does he play both roles, that is, the product owner and scrum master?



The product owner and the scrum master cannot be same. The reason for that is that their goals are orthogonal and they are orthogonal in a positive sense. While the product owner would like to maximize the value, the scrum master wants to make sure that it is sustainable in the long-term. That is, building the teams self-organizing capability to deliver the software for an indefinite period of time.

So, while the product owner would like to see and make sure I can deliver this feature sooner, they may or may not have enough to make sure there is a long-term capability. So, they must have orthogonal goals.

And the way we look at it in scrum is, let the issues come up. Sometimes the product owner wins, sometimes the scrum master wins. And there should be a healthy conversation, a kind of a creative conflict to make that happen. And that is why we don't recommend that they should be the same person here.

Okay. I have another question. In some companies, a situation might arise that there are not enough resources and say a database architect has to be shared across teams. So, how would the different scrum teams manage this kind of dependency on the same person?

MORE ABOUT THE SCRUM TEAM

upGrad

Is there no manager in a Scrum team?	a. No b. Other Agile methodologies have managers
Can a PM be both product owner & Scrum master?	a. No b. The two roles are orthogonal
How do Scrum teams manage shared resources?	a. Deliver at every sprint b. Cross-pollination: Transfer of learning to another team member c. Increase competencies in all team members

Having limited capability or having a few set of few specialists is a challenge for an organization, even without scrum or agile. Even within a waterfall world, we have the same challenges there. But by creating such kind of a social system, agile actually allows you to surface up these impediments much sooner. And now that impediments are being surfaced up, the team is in a better position to say, how do I deal with that?

So, let us say database architecture is one of the rare skills in the organization today. But after six months, if it still continues to be rare and we have not invested enough on that, we still run the same amount of risk in the organization.

Even without scrum, the organization is vulnerable because if that person were to leave, you would have nobody, the teams would not be able to ship the features out. How does scrum handle it? Scrum will identify by making sure that first of all, we are delivering something in every sprint that this person or this capability is available in every team.

Maybe on the day one, it is not available. We still have to manage with somebody kind of as a shared team member in three teams there. But over a period of time, if we identify that as an impediment, we will cross pollinate the skills.

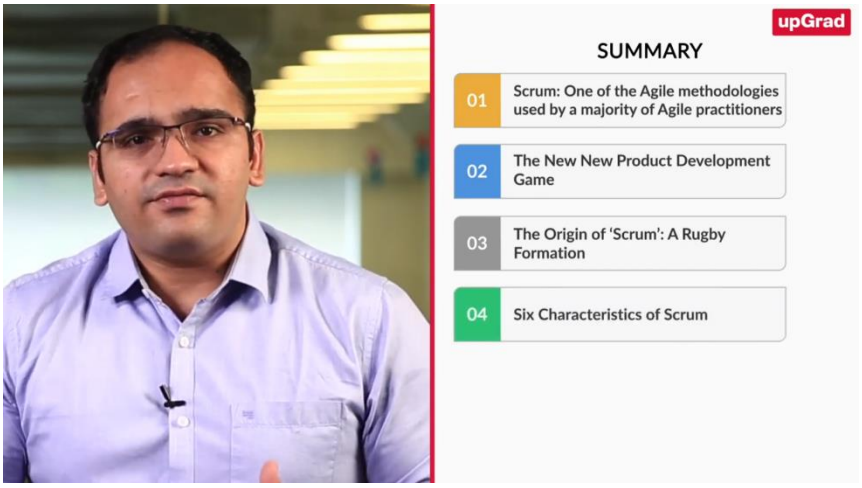
Somebody in the team will step up to the plate and understand, okay, can I understand a little bit of a database. Next sprint, can I do a little more of it. Next sprint, can I do little more of it? The idea is that no one single event is going to make you an expert, but a sustained effort over a period of time will create a reasonable amount of competency in other people.

So, it is a way in which we are really growing individuals and turning them into a high performing team rather than accepting the fact that, Hey, we have to live with this impediment forever. Now, obviously in the short run, if something happens, we will have to deal with that on an emergency basis.

Scrum cannot provide a fast shortcut answer to every emergency. But when you measure it over a period of time, you will have a higher capability to deal with unknown events and such kind of exigencies compared to any other system known to us.

So, you learned in this video that in scrum, there is the concept of individual accountability, and therefore there is no manager in the scrum framework. Additionally, the product owner and the scrum master should not be the same person as these two roles are orthogonal.

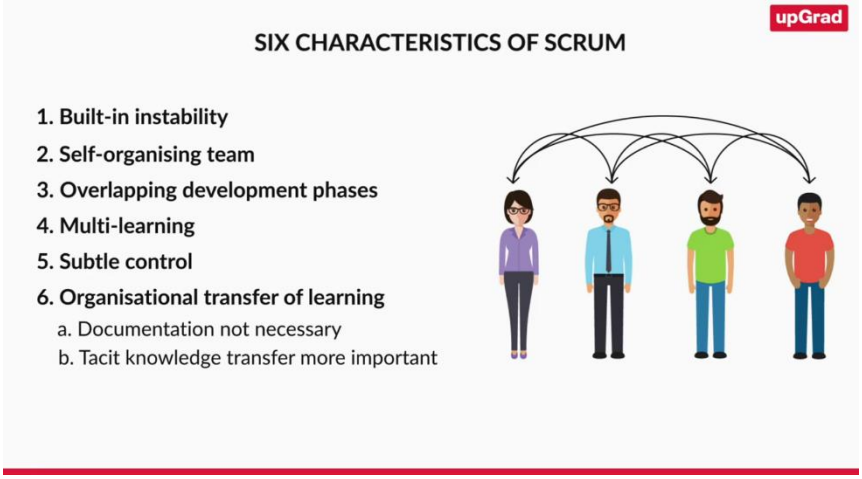
You also learned that if there is a shortage of resources, there is cross pollination and a team member steps up to show the responsibilities. With that, we come to the end of the session. Catch a detailed summary on the other side.



Wasn't that a great session. You were introduced to an agile methodology called scrum, and you understood how product development happens under such a framework. Let's recap what we covered in this session.

First off, you learned that scrum is one of the agile methodologies that is followed by the majority of agile practitioners. It was first documented by Japanese researchers in 1986, as a new-new product development.

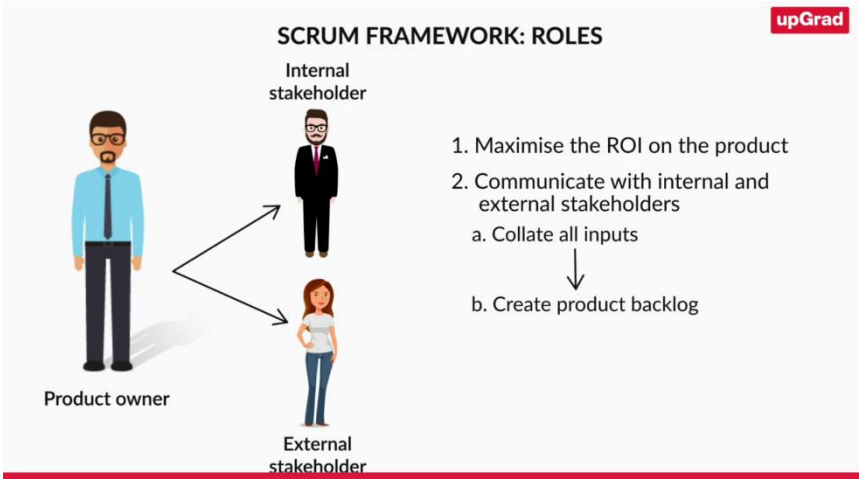
You also learned that the term scrum comes from the game of rugby in which players come together in formation to try and get the ball. Scrum has a number of features. It has built an instability as it's up to the team to come together and figure out how the product development is to be done.



A scrum team is self-organizing, that is every member of the team is responsible for the team's decisions and executions. It develops products in overlapping phases, not in a sequential manner.

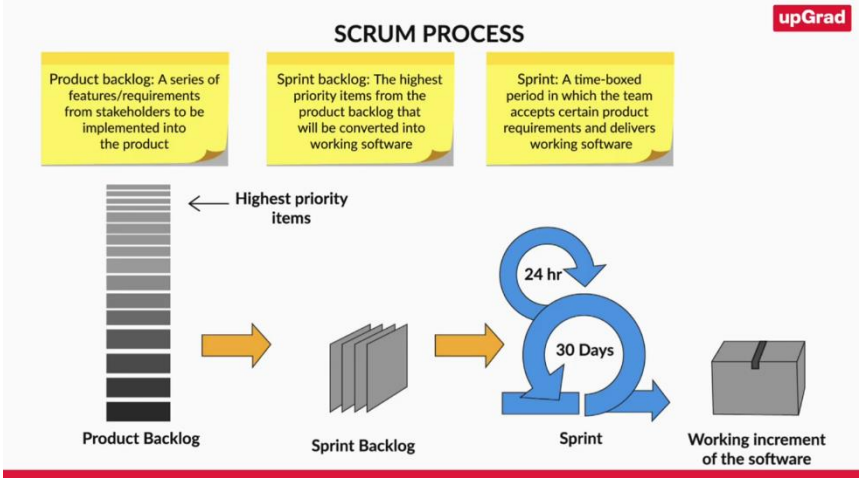
It involves multi learning, which happens across divisions in multiple levels. A scrum team is run by subtle control, that is, peer control or self-control.

Lastly, the transfer of learning is given a lot of importance, especially for tacit knowledge. Next, you learned that under the scrum framework, there are defined roles.

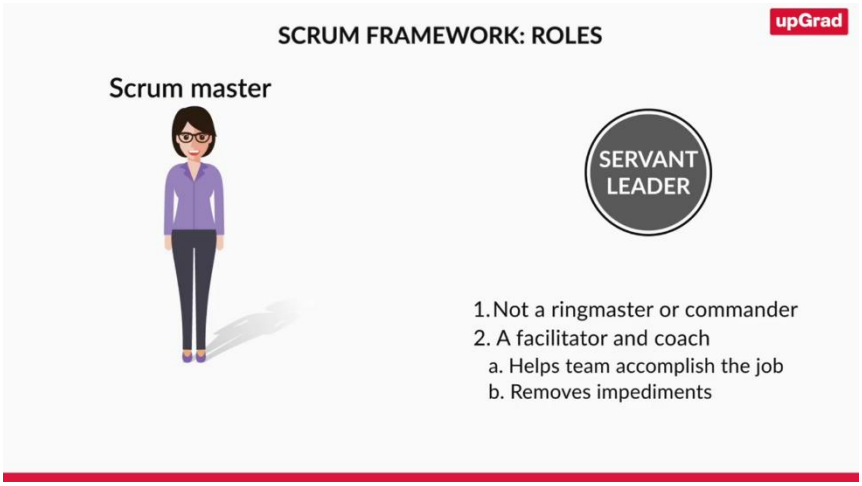


The product owner, the scrum master, and the rest of the team are called developers. This is to ensure that there are no second-class citizens in the team.

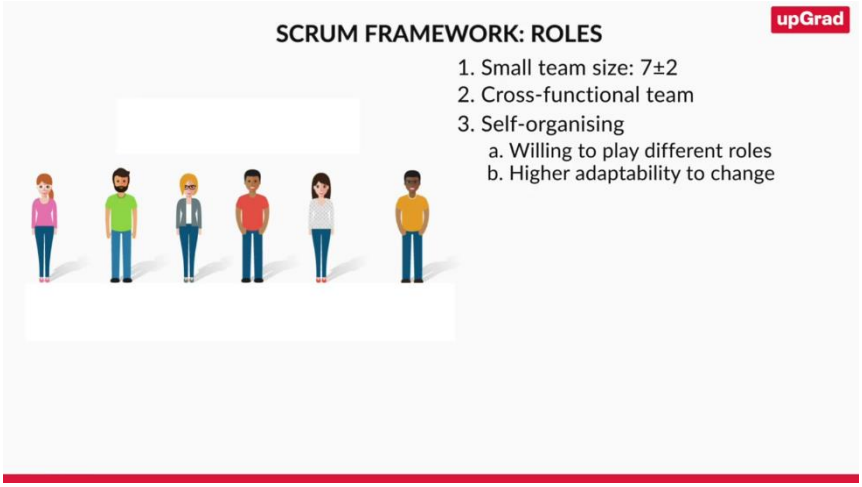
The product owner focuses on maximizing the ROI on the product. He collates all the input from internal and external stakeholders and translates them into the product backlog.



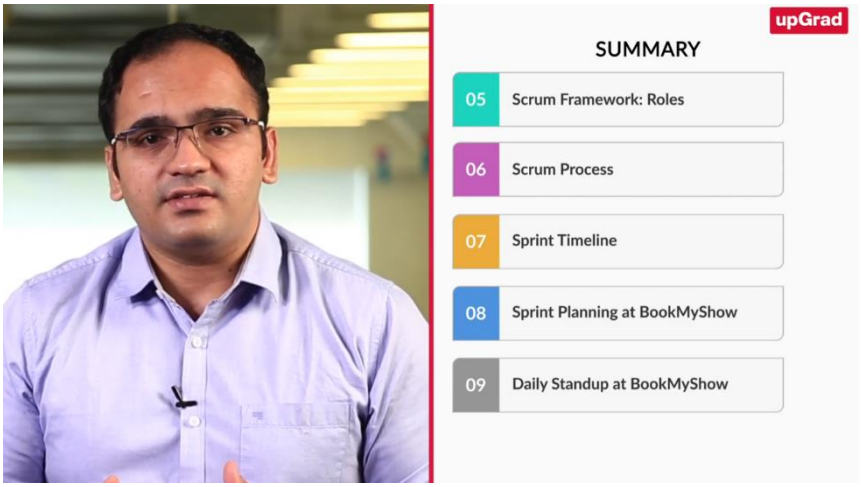
The highest priority items from the product backlog are transferred into the sprint backlog and are delivered through sprint.



The second role is the scrum master. This role can be described as a servant leader. The scrum master is the facilitator of the team who focuses on removing any impediments the team is facing.



Then comes the rest of the scrum team. It usually consists of five, nine developers, and it's characterized by traits of cross-functionality, which means that the team focuses on delivering a ready to ship feature. It is also self-organizing where team players take up multiple roles according to requirements.

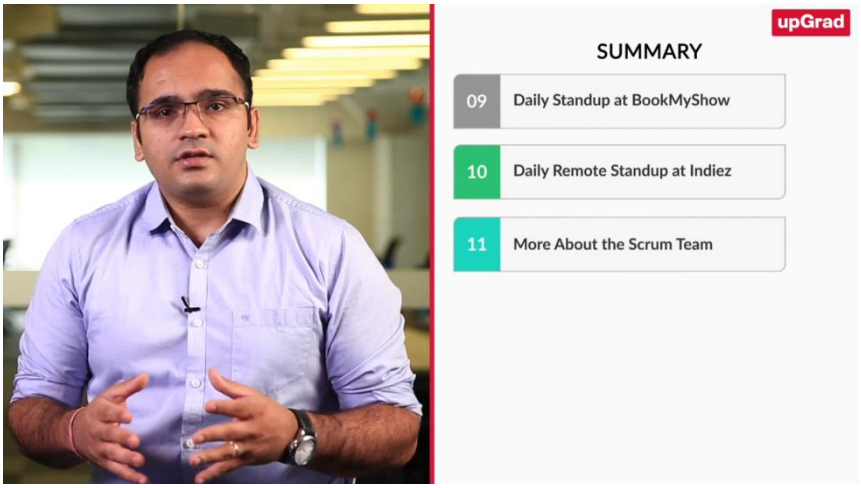


After this, you got a detailed lowdown on how a sprint is conducted. On the first day, the team comes together for sprint planning, where all the requirements are discussed and a plan is finalized. After this, the team starts working on translating the user story into a working software.

The key deliverable for any sprint is a working software. This means that at the end of the sprint, the development, QA and documentational software are all complete.

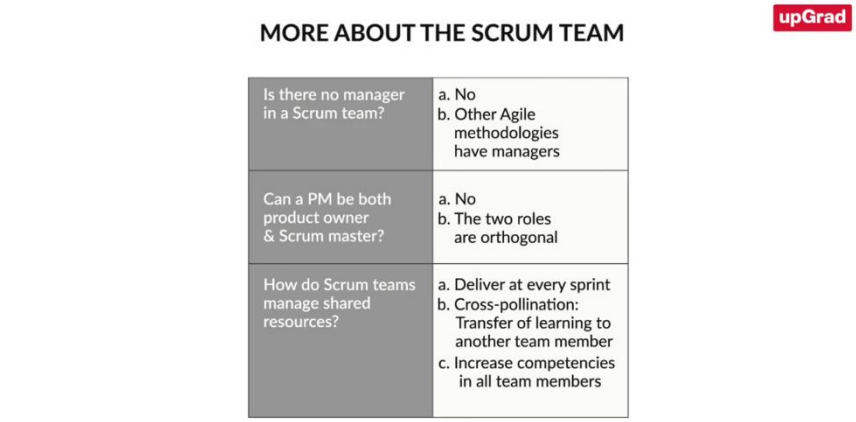
Every day of the sprint, a stand-up is conducted. That is a 15-minute meeting to get everyone on the same page. This meeting covers only three things, what each member did yesterday, what they would do today, and if they are facing any blockers. Also, the time still needed to complete the tasks is recorded in what is called the sprint burndown chart.

At the end of the sprint, there is a demo in front of the product owner of the feature that was last worked on. Lastly, there is a retro where the team identifies opportunities to work more efficiently the next time round. And in case a user story is not finished in the sprint, it goes into the product backlog.



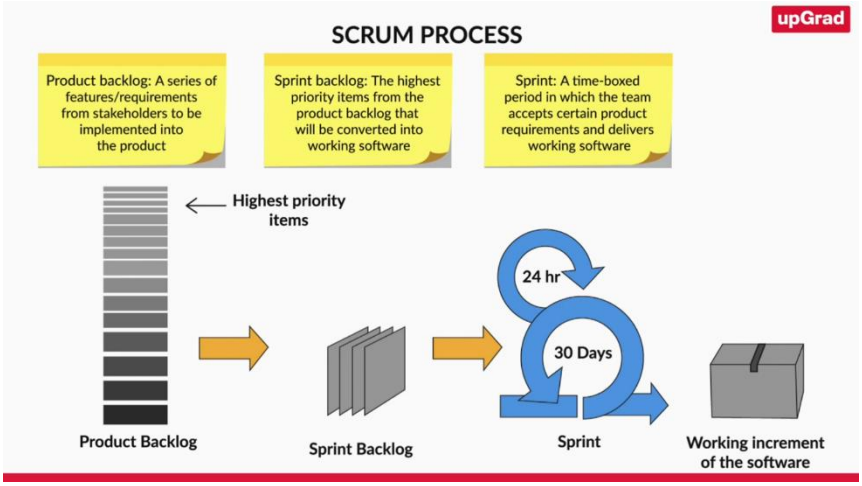
To understand sprint planning better, you saw how it's conducted at BookMyShow. The product owner, scrum master and the entire team come together, and that items from the product backlog to the sprint backlog, these items are then broken down into smaller simpler tasks.

You got an example of a stand-up at BookMyShow. Every morning, the scrum team stands facing each other and discusses any impediments they are facing. You also saw another example of a remote stand up at Indiez, whether you Slack and its gig bot for this purpose.

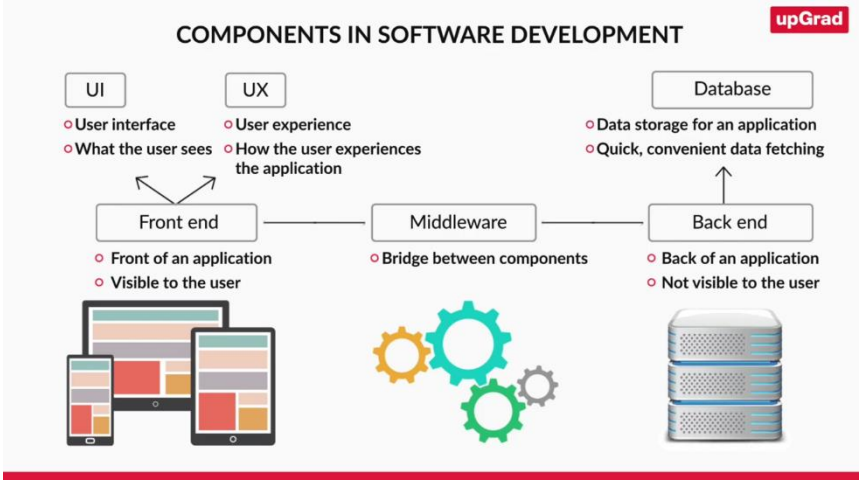


At the end of the session, you understood that scrum has a concept of individual accountability. So, there is no manager in the scrum framework. Additionally, the product owner and the scrum master should not be the same person as these two roles are Orthogonal.

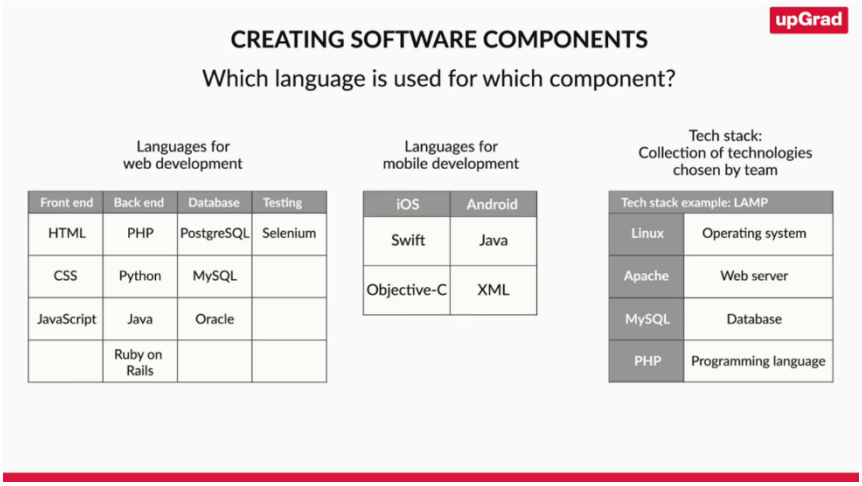
Although product owner focuses on maximizing the ROI on the product, the scrum master focuses on making sure that it's sustainable in the long run.



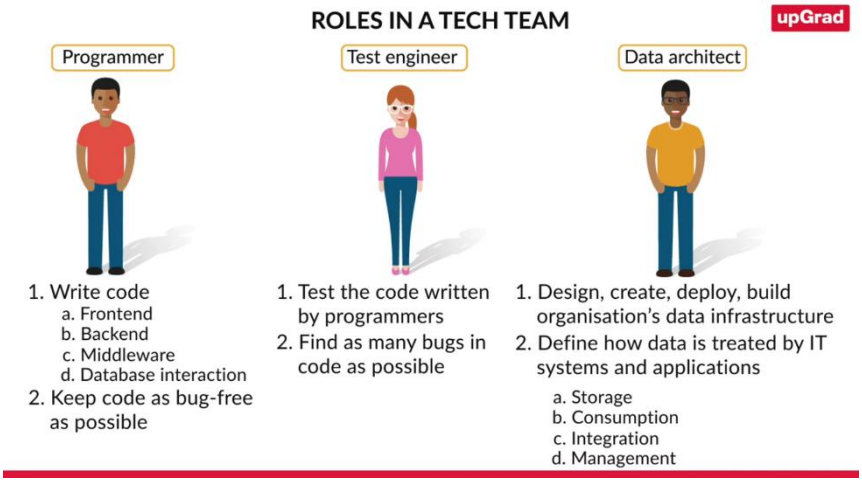
Now let's revise the terms we came across in this session. The first was product backlog, which is a log of user requirements in the form of user stories or epics. It's essentially a to do list for the development of the product. The product backlog items, which are highest in priority are moved to the sprint backlog, which details out these requirements in lightweight format. These requirements are then delivered through sprints.



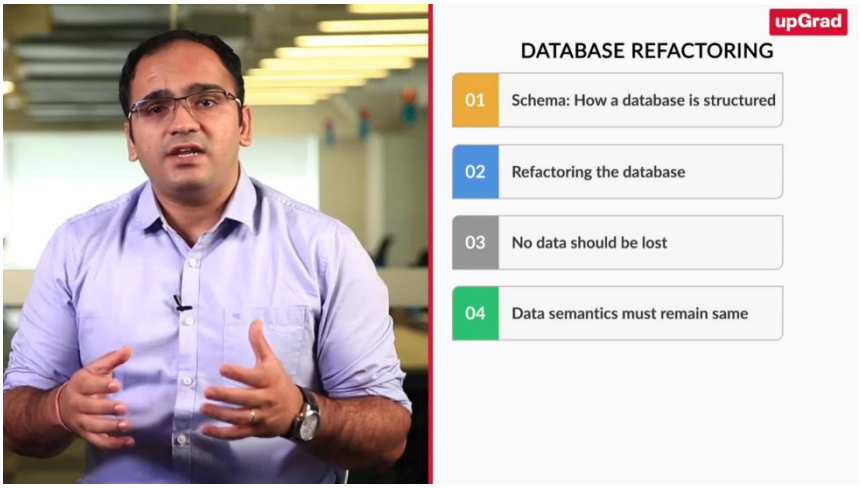
You also learned that in software development, the front end is the interface that users see, the backend is where data is stored and retrieved. The database is a collection of data organized according to schema. And the middleware is the bridge between these ends.



To develop all these components, developers have to write code in various languages like HTML, PHP, Python, MySQL, and selenium amongst others. When a team decides on these choices, the collection of technologies they choose is collectively called a tech stack.



Next, you learned about the different people in a team, a programmer writes code, be it frontend, backend database introduction, or middleware. A test engineer tests the code that is written, and a data architect designs, builds and deploys the data infrastructure of an organization.



The last two terms were payment gateway integration and refactoring database. Payment gateway integration simply means the option to pay online from within your application. Database refactoring means making changes to the schema of the database to improve design or effectiveness. Well, that's it for now. In the next session, we'll take a closer look at the roles in a scrum team until then, stay agile.