# AGILE HANDBOOK

Last update: Aug 19, 2016

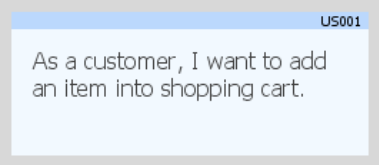
# CONTENTS

# User Story

User story is one of the most important tool for agile development. They are often used for identifying the features of a system under developed. User stories are well compatible with the other agile software development techniques and methods, such as scrum and extreme programming.

## What is a User Story?

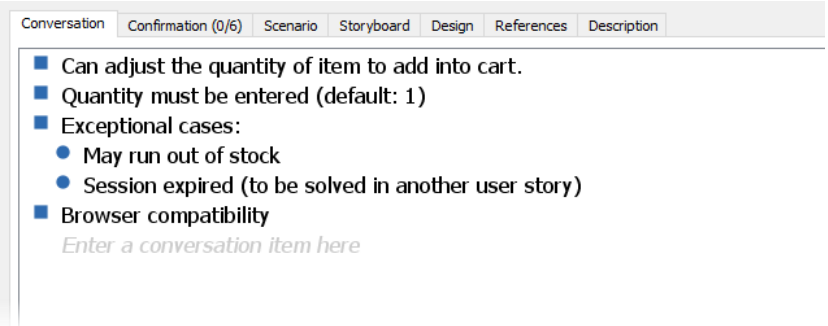
A user story is a note that captures what a **user** does or needs to do as part of her work. Each user story consists of a short description written from user's point of view, with natural language. Unlike the traditional requirement capturing, user story focuses on what the user need instead of what the system should deliver. This leaves room for further discussion of solutions and the result of a system that can really fit into the customers' business workflow, solving their operational problems and most importantly adding value to the organization.




### Concept of 3C's

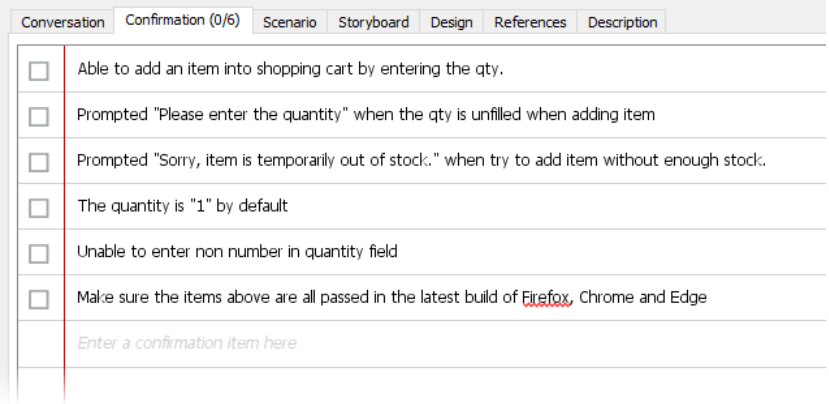
The 3C's refer to the three critical aspects of good user stories. The concept was suggested by Ron Jeffries, the co-inventor of the user stories practice. Nowadays, when we talk about user stories, we usually are referring to the kind of user stories that are composed of these three aspects.

- **Card** – User stories are written as cards. Each user story card has a short sentence with just-enough text to remind everyone of what the story is about.
- **Conversation** – Requirements are found and re-fined through a continuous conversations between customers and development team throughout the whole software project. Important ideas and decisions would be discovered and recorded during the stakeholder meetings.




- **Confirmation** – or also known as Acceptance criteria of the user story. During the discussion of requirements, the customers tells the analyst not only what he/she wants, but also confirming under what conditions and criteria the working software would be accepted or rejected. The cases defined are written as confirmation. Note that confirmation focuses on verifying the

correctness of work of the corresponding user story. It is not an integration testing.



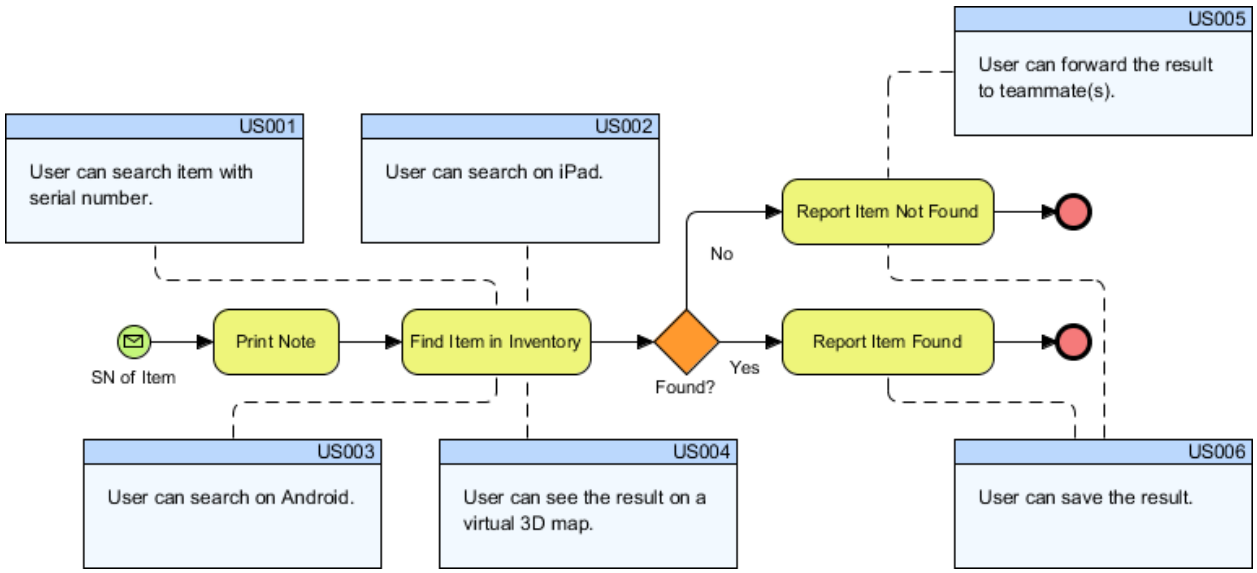

## How to Identify User Story?

User stories should be identified together with the stakeholders, preferably through a face-to-face meeting. User story is a requirement discovery process instead of an upfront requirement analysis process. In the traditional requirements capturing approaches, system analyst tries to understand customers' needs and then prepare a requirement specification for the system in detail. This is not how the user story approach works. Instead of a documentation process, the identification of user story is more like a note taking process. Through the discussions with users, we listen to and understand their problems and needs, and then write down their needs as user stories at the same time. These user stories will become the source of requirements. The details could be subsequently filled just-in-time, providing the team with a "just-enough" requirement references throughout the project development process.

## Mapping Business Process with User Stories

BPMN is one of the most powerful tool for business analysis and modeling. Not only we can use it to perform process improvement, but also we can identify user stories from those processes intended to be automated through the following steps:

1. Simply model the user's workflow with BPMN business process diagram.
2. Walk through the process model with users.
3. And, analyze the business activities of the problem, and then identify the user stories in related to the process required to be automated, which is also known as 'business process to user story mapping'.




## How to Write User Story?

When writing a user story, try to write in the user's voice as the form below (or at least, make sure what you have written qualifies the following statement):

As a <role>, I want to <business objective> so that <business value/reason>.

E.g. As a customer, I want to receive an SMS when the item is arrived so that I can go pick it up.

where:

- **<role>** represents the person, system, subsystem or any entity else who will interact with the system to be implemented to achieve a goal. He or she will gain values by interacting with the system.
- **<business objective>** represents a user's expectation that can be accomplished through interacting with the system.
- **<business value>** represents the value behind the interaction with the system.

It is not a rule but a guideline that helps you think about a user story by considering the followings:

- The user story will bring value to someone or certain party (e.g. customers).
- The user story is fulfilling a user's need (e.g. receive an SMS when the item is arrived)
- There is a reason to support implementing this user story (e.g. customer can go pick up the item she purchased)

Each user story should bring value(s) to someone. But sometimes, the value is obvious enough just by reading the business objective. To write down the value as part of the statement is cumbersome. In such case, we will just skip it. Here are several examples:

As a customer, I want to settle payment by using credit card ~~so that I can use my credit card in online purchasing~~.

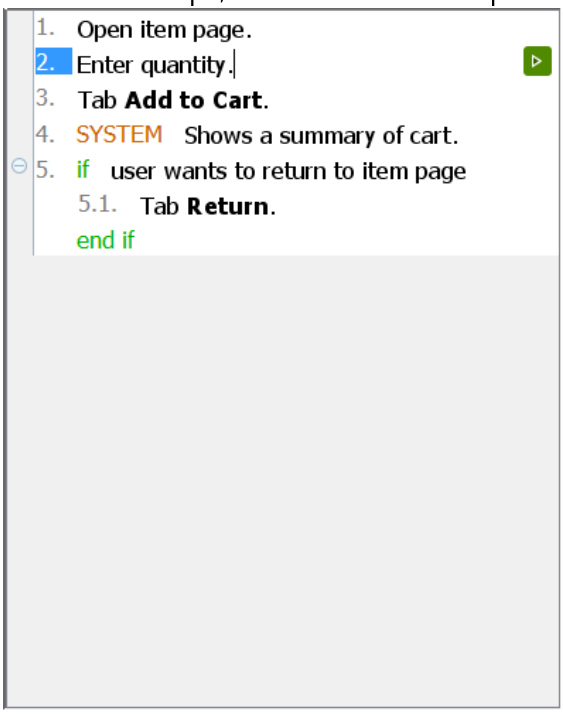
As a user, I want to perform searching by entering my friend's name ~~so that I can find my friend with her name~~.

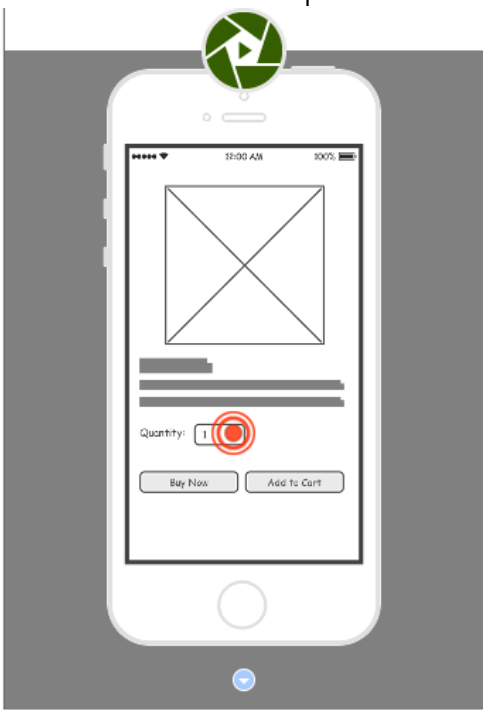
No matter how you write a user story, there are two things that you must keep in mind. First, a user story must be written from user's point of view. Second, keep the description 'just enough'. Avoid adding too much details at the beginning of a software project. Later on you will have chance to refine and detail the user story to make it become something that can be used by the developers in design and implementation.

## Lifecycle of a User Story

In a broad sense, there are six main states for each user story throughout a software project:

- **Pending** – Through the communication between user and project team, user stories are found. At this state, the user stories have nothing more than a short description of user's need. There is no detailed discussion of requirements, no system logic and no screen design yet. In fact, the only purpose of user story, for now, is just for reminding all parties for a future discussion of user's request written in this user story (card). It is possible that the user story will be discarded in the future.
- **Todo** – Through a discussion between different stakeholders, the user stories to be addressed in the next few weeks are decided, and are put into a time-box called a sprint. Such user stories are said to be in the Todo state. No detailed discussion has yet been carried out in this state.
- **Discussing** – When a user story is in the Discussing state, the end user will communicate to the development team in confirming the requirements as well as to define the acceptance criteria. Development team will write down the requirements or any decisions as conversation notes. UX specialist may create wireframes or storyboards to let user preview the proposed features in visual mock-ups, and to feel it. This process is known as user experience design (UX design).



- **Developing** – After the requirements are clarified, the development team will design and implement the features to fulfill user's requests.
- **Confirming** – Upon the development team has implemented a user story, the user story will be confirmed by the end user. He/she will be given access to the testing environment or a semi-complete software product (sometimes known as an alpha version) for confirming the feature. Confirmation will be performed based on the confirmation items written when detailing the user story. Until the confirmation is done, the user story is said to be in the Confirming state.
- **Finished** – Finally, the feature is confirmed to be done, the user story is considered in the Finished state. Typically, this is the end of the user story. If user has a new requirement, either it is about a new feature, or it is an enhancement of the finished user story, the team would create a new user story for the next iteration.

## Detailing User Story – When and Why?

A key thing that makes user story different from traditional requirements capturing approaches is that user story approach split the identification of problem and solution into two steps, performed at different stages of a software project. While the problems and a brief understanding of user requests are found at the beginning of the software project, the details of system requirements are found only before the commencement of design and implementation. Here are some benefits of this arrangement:

- Able to respond to latest user needs because requirements are detailed right before implementation, instead of having everything concluded at the beginning.
- Able to identify the right requirements easier because both of the problems and solutions will be undergone detailed discussions. In the traditional approaches, since the details of all requirements are required to be found upfront at the beginning of a project, the "upfront requirements" could have been changed overtime, resulting a lot of wastage of analysis affords.
- On the other hands, user stories that are found invalid can be discarded easily. You do not lose much time on prior studying and documentation.

## How to Use User Story Effectively?

Just like many other software development methodologies, if you apply user story properly in your software project you will be able to produce a quality software system plus to win the trust and satisfaction from customers. Here are some points that you need to keep in mind when using user story:

- Keep the description of user story short.
- Think from end user's point of view when writing a user story.
- A (UML) use case represents a business goal. The capability to group user stories under use cases allows you to ensure a user story is satisfying a business goal, in other words, they sharing the same system goal. It serve as a placeholder for you to manage, schedule and prioritize user stories in a more manageable manner.
- Confirmation items must be defined before you start the development
- Estimate the user story before implementation to ensure the workload of your team is under control.

- Requirements are found with the end users, not by the end user or just by the development team. Keeping a good relationship with the end users will be a win-win situation for both parties.
- Communication is always important in understanding what the end user wants.

# Use Case

A use case is a tool for identifying the business goals of a system. The identification of use cases helps define system scope, ensuring that the requirements to be found will all be aligned with the business values, needs and strategy.

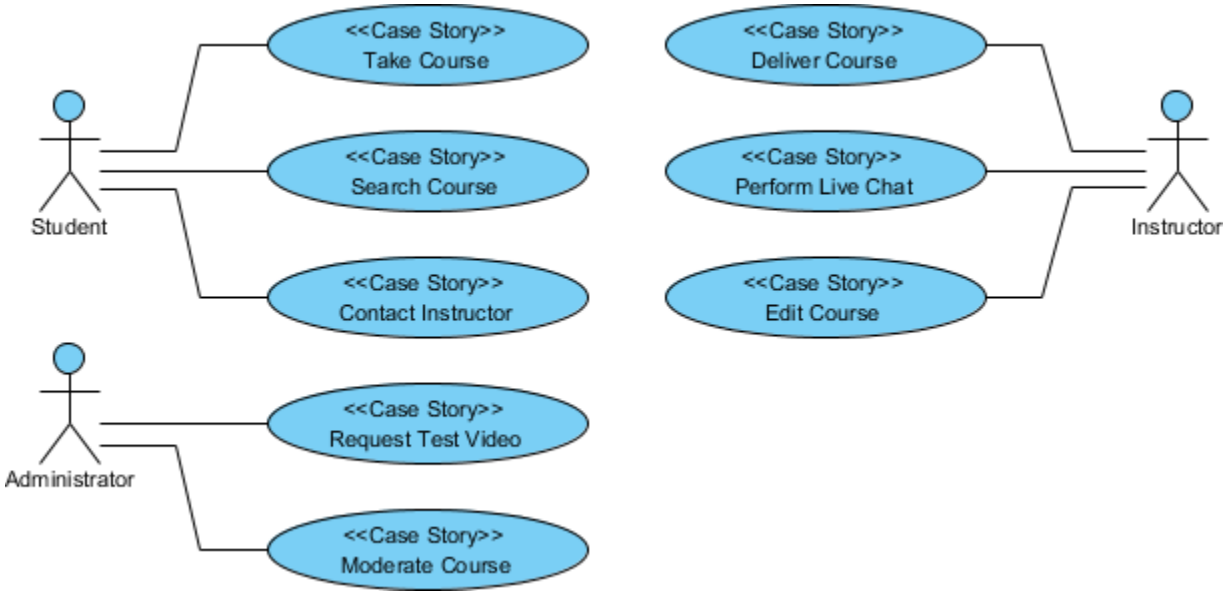
## What is a Use Case?

A use case represents a high level business goal to be achieved by someone, some parties, or some sub-systems through interacting with a system, which can be the system under developed, or the system to maintain, depends on the nature of your software project.


Use cases are neither system requirements nor the functions to be implemented. In fact, it is important to stay use cases away from requirements because you need a clear set of use cases that helps you define business goals and system scope. In fact, capturing of requirements would be done after the identification use cases (epics) and subsequently spitting into a set of corresponding user stories (business goals).

## Graphical form of use case

Use Cases can be visualized in a UML Use Case Diagram. A use case is shown as an ellipse in diagram, with its name appear in the middle of the shape. Besides use case, a typical use case diagram contains two more elements – actor and link.




An actor is a role that interacts with the system to achieve one or more business goals as represented by the use cases. The occurrence of interaction between an actor and a use case is represented by an

association (link). Note that an actor does not necessarily represent a specific physical entity (e.g. John), but just a role (e.g. Customer). In reality, a role can be played by different person and, conversely, a person can play multiple roles.

## How to Identify Use Case?

Use cases are found by communicating with business owners or the senior executives of the company. We use to call them the business stakeholders. It is important to identify use cases with business stakeholders but not someone else, as they are clear about the directions and actual operations of the company. They also have the authority and information required for making business decisions. Thus, the use cases identified are all aligned with the business values, needs and strategy of the company.

## Why Use Case?

Many people see the identification of use cases a redundant step. They would rather go straight into identifying system requirements. So, is use cases approach useless?

When you are going to develop a large scale system which typically involve a lot of stakeholders who have different expectations on the final system, ending up a large collection of requirements in an unmanageable manner. A use case could be served as a placeholder for accommodating a group of related user stories which share a larger and shared business goal and scope among them.

Keep in mind that use cases are found by communicating with business owners and senior executives who oversee the growth of their companies and have the ability to make strategic business decisions. Due to this reason, use cases directly reflect what the business goals the target system has to achieve. By finding requirements based on use cases, the requirements are highly likely to be within system scope and thus achieving the business values expected by the owners. Besides, use cases also facilitate meaningful categorizing of requirements. Proposed software features can be planned based on the importance of the use cases, instead of solely depending on the opinions of the front-line stakeholders, which may not be fully aligned with business owner's expectation.

## Use Case Examples

Here are some examples to illustrate the usage of use cases. The example given here is just for illustration purposes, there is no definite way as to what use cases should be identified for a particular target system. The rule of thumb for the use case identification process is always be the result of active participations and involvements with the business stakeholders.

| System | Use Cases |
|---|---|
| ATM | Withdraw cash, transfer money, donation, settle bills, change PIN |
| Online photo album | Upload photo, share photo, delete photo |
| Health tracking App | Record workout, produce workout statistic, challenge a goal |

Based on the examples above, here are some discussions:

### Use Case Yields Observable Goals

#### *ATM*

ATM is a classic example when explaining the concept of use case or use case analysis. You may wonder why there isn't a use case for 'Login', which is an inevitable step of all ATM operations. As said, use cases are business goals to be achieved. They yield an observable result to the actor who interact with the system to achieve the use case. Here, 'login' is only a part of the other operations. 'Login' itself does not yield any observable result – no one would just login an ATM and go away, right? So, we do not consider 'login' as a use case.

Wouldn't 'change pin' be too small to be a use case? The answer is: the identification of use cases is neither based on the amount of work the user need to do, nor based on the number of system functions to develop. As long as it is a business goal that the business stakeholder want the target system to achieve, it is a use case. In this case, we consider the customers to change their PIN through ATM as a use case.

#### *Online photo album*

Typical online photo album allows users to tag their uploaded photos. Is 'tag photo' a use case then? The answer is: It depends. If the business stakeholders want to let users access the system to tag their photos, even without doing anything else within the session, then 'tag photo' should be a use case. However, if they think that tag photo is only a part of the photo upload process, and there is no other way to tag photo afterwards, then 'tag photo' would not be a use case. Another case would be that the stakeholders just want to let users edit the photos they uploaded for any properties like, title, description, tags, etc. Then, it's likely that a use case 'edit photo' will be created. So you can see the identification of use case is not a casual step. You can imagine the capability to tag photos will be supported quite differently under use cases 'tag photo' and 'edit photo'.

#### *Health tracking app*

Although use cases are not requirements, this doesn't mean that use cases are abstract and lack focus. Take health tracking app as example. Record workout, produce workout statistic and challenge a goals are all clear enough in defining the system scope. Can 'maintain good health' be a use case? Well, it is a bad choice because its scope is too wide – which functionality of a health tracking app is not done to satisfy this goal?

## How to Write Use Case?

The simplest form of a use case consist of a <verb> + <noun> or <noun phrase> that describe a business goal. Here are some examples:

- Register Account
- Place Order
- Withdraw Cash
- Post a Vacancy
- Investigate Failure of Cases

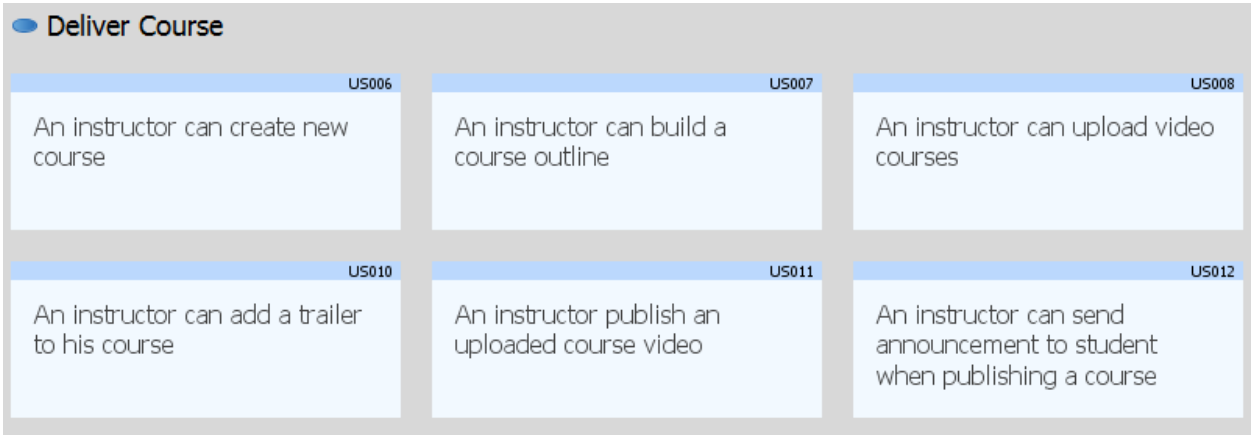
As said, use cases are designed to identify business goals. Do not use it for writing requirements, nor to describe the interactions between user and system. All these steps will be detailed in the subsequent development activities but not now.

## Using Use Case with User Story

User stories is also an important tool in agile development. Each user story consists of a short description written from user's point of view. Here are some examples of a user story:

- User want to settle a payment by credit card.
- User want to settle a payment by PayPal.
- User want to optionally include shipping insurance during checkout.
- User want to choose a different delivery address during checkout.
- User want to receive an SMS upon the successful creation of order.

A use case is a tool for defining the scope of a feature, while a user story captures what a **user** does or needs to do as part of her work, ultimately lead to the creation of some requirements. We can utilize this two requirements capturing skills for identifying the right requirements. Here is the steps for performing it: First of all, communicate with business stakeholders to identify business goals as the use cases. Then, focus on a particular use case, communicate with the front-line users for identifying the user stories under that use case. Because the identification of user stories is driven by a use case, the requirements found in the end will align with the business goals.

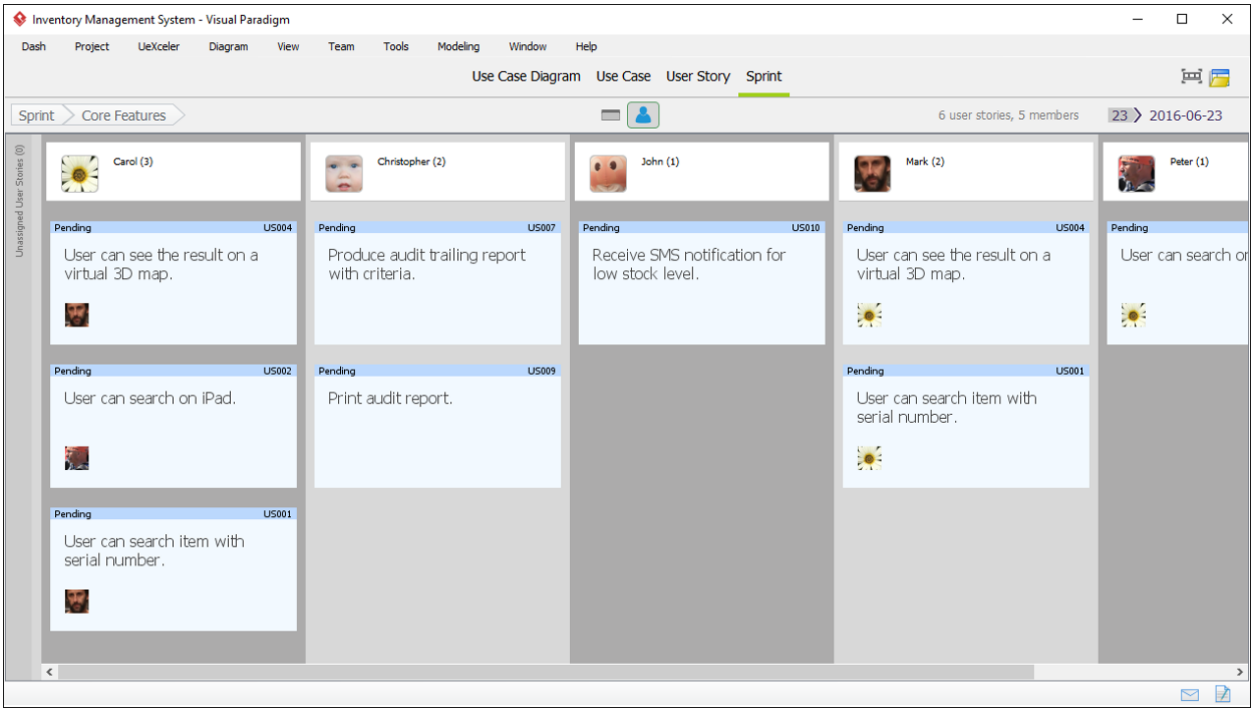


# Sprint

Sprint is an important concept in scrum (agile development process). A sprint is a set period of time which specific user stories have to be completed and confirmed. A sprint approach ensures a constant and frequent delivery of executable software features throughout a project in software development.

## What is a Sprint?

A sprint is a time box. Each sprint has a start and end date during which a set of selected user stories have to be completed and confirmed. The following image shows you the key elements of a sprint, which includes a set of user stories, the scrum members involved, the assignment of work, the duration and end date (top-right corner).




At the beginning of a sprint, the product owner, the stakeholders and the development team meet together to prioritize, and then select the user stories to be accomplished in the sprint. Since different parties have different preferences, considerations and concerns in regard to the project and project schedule, the purpose of meeting is to provide different participants with the opportunity to listen to and understand others' views, and to come up with a set of user stories that is agreed by all parties.

## Sprint duration

Fast delivery of quality work is one the reasons why software teams want to adopt agile software methodologies. Thus, although there is no one-size-fit-all choice for sprint duration, it's commonly agreed that the duration should be as short as possible. But how short should it be?

While a long sprint length is not preferred, an unreasonably short length of a sprint may weaken the development team's motivation in completing significant work, or may even lead to a poor quality of deliverables.

So, the selection of sprint duration is the result of discussion among the whole team – product owner, scrum master and scrum members like database designers, programmers, UX designers, testers, analysts, etc. But in the end, someone has to make a decision. At that moment, the scrum master is usually the one who will gives the answer.

Traditionally, a sprint last for a three weeks to one month, but nowadays more and more teams have been successful with two-week sprints. After all, there is still no fixed selection to sprint duration. A good scrum master has the collaborative and facilitative skills in determining the sprint length, for securing works to be accomplished as expected. Here are some factors that a scrum master may consider:

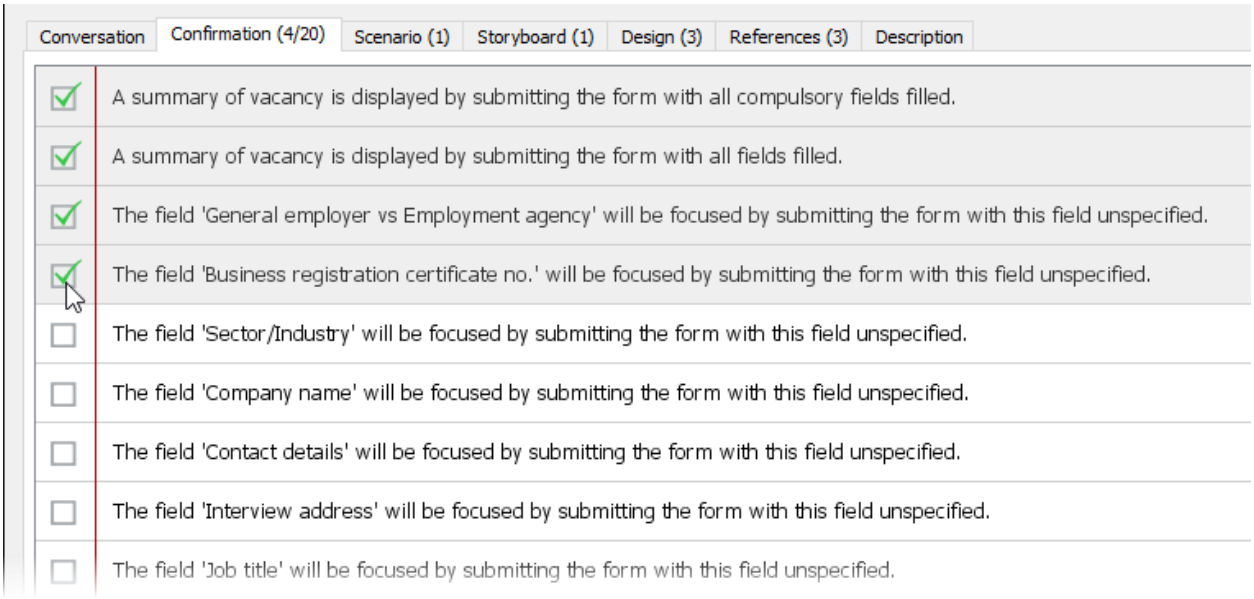
- Agreed project schedule
- Availability of customers/stakeholders/product owner (Who can clarify potential doubts)
- Working culture of customers
- Capability of scrum team
- Scrum experience

Once the team has reached a consensus, all future sprints would follow the same sprint duration without frequently changing it sprint after sprint. Yet, it is a good practice for the scrum team to keep reviewing the effectiveness of sprint, and to find out an optimal sprint duration that work best for everyone.

## Confirmation of Works (User Stories) in Sprint

The development activities are setting up around the user stories after a sprint begins, and more and more user stories are implemented as the sprint progressed. Yet, the complete implementation of a user story is not the end of the story. There is still an important step to be gone through – confirmation.

To confirm a user story is to try out the implemented feature and decide if the feature is implemented as expected. The judgment should be made solely based on the criteria established when detailing the user stories, written in the form of confirmation items. During confirmation, product owner is given a testing environment or device for testing the implemented work. The product owner will walk through the confirmation items written on the user story one by one. If all the items are confirmed to be done, the user story is said to be confirmed. If any item is found undone or not work as expected, the product owner will request the development team for a fix. The fix and confirmation processes will repeat until the user story is fully confirmed.




## When to Confirm?

Sprint, and in fact agile is a continuous delivery process. Instead of confirming the user stories at the end of a sprint, confirmation should be done right after the completion of any user story. However, if the product owner failed to participate in continuous confirmation, you may arrange weekly meetings that last for 1 to 2 hour each. During the meeting the product owner will work with the team in confirming the completed user stories. The meeting also involves a discussion session that clarifies the doubts found when implementing the other stories included in sprint.

## Confirmation is not equivalent to testing

As said, the purpose of confirmation is to make sure the user story is implemented properly. The judgment is based on the items established as confirmation items during the detailing of a user story, no more and no less. The scale of checking is far from enough for ensuring that the feature is ready for production use. So, when to perform a 'real test'?

Different teams have different practices in regard to software testing. Some teams prefer a per-sprint test which could involve conducting an integration test and regression test by the end of a sprint. Some other teams might prefer setting up a stabilization sprint for, as its name says, testing and bug fixing. No new development activities will be done in that sprint.

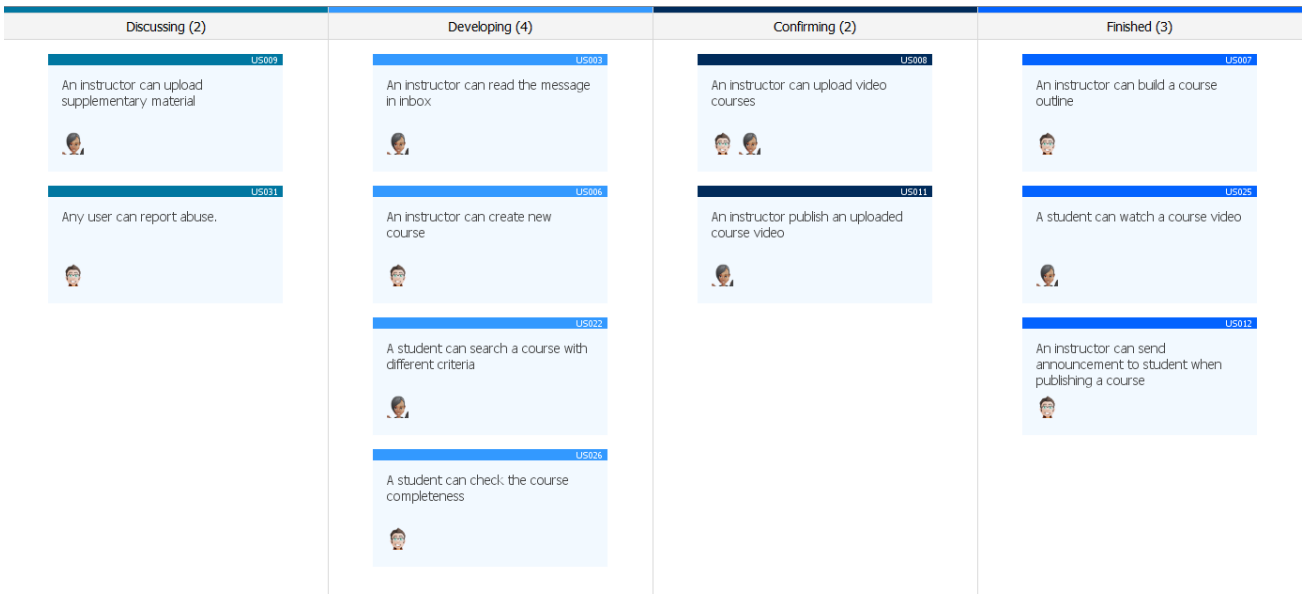
No matter which approach you choose, keep in mind that the choice is not any-one's choice but everyone.

# Kanban Board

Kanban was initially developed in the 1950s at Toyota, by Taiichi Ohno, as a method for supporting JIT (just-in-time) production and reducing inefficiencies through the whole supply chain. It has been gaining momentum as a simple and pragmatic approach to managing software development. Although producing software is a creative activity and therefore different from mass production of car manufacturing, the underlying mechanism for managing the development pipeline for software teams can still be applied.

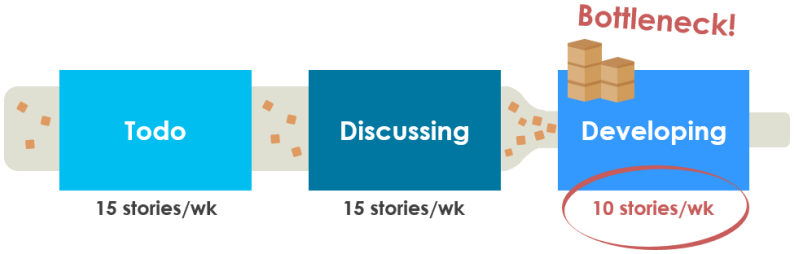
## What is Kanban Board?

Kanban is very simple and straight forward, but at the same time extremely powerful. A Kanban system consists of a big board with sticky notes (in our case, a bunch of user stories) placed in different columns representing the phased processes. In here, we assume they are: Pending, Todo, Discussing, Developing, Confirming and Finished.




## WIP Limit

The cards represent work items as they flow through the columns. The limits are the critical difference between a Kanban board and the sprint whiteboard. Limiting the amount of work-in-progress (WIP), at each step in the process, prevents overproduction and reveals bottlenecks dynamically so that you can address them before they get out of hand.



## Goal of Kanban


The goal of Kanban is to flow every bit of work item efficiently from the start to the end with as little waste and delay as possible. This requires limiting the work in progress in the pipeline to what can be realistically consumed of those work items at a given time. The Kanban approach moves the work items from one phase to the next only when it is pulled through by the next team member(s). Work should never be pushed forward, so bottlenecks of flow could be minimized.

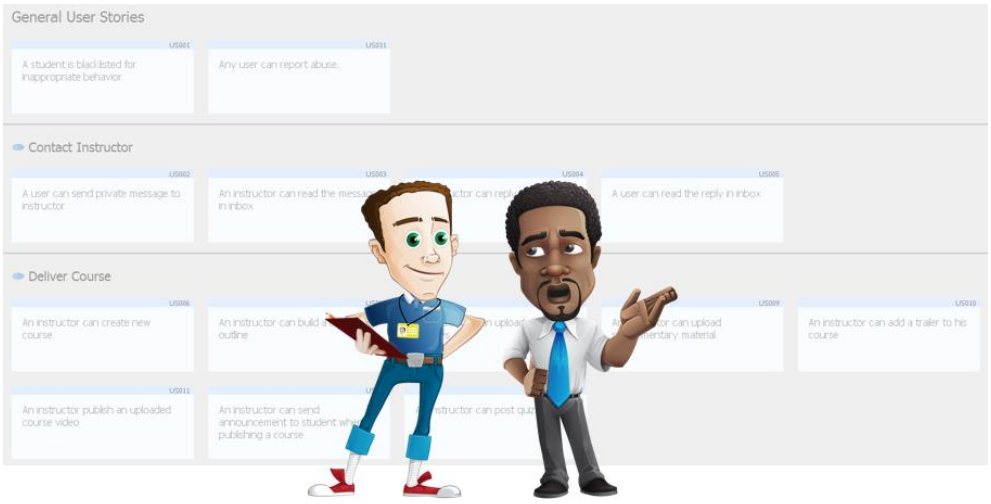
## Kanban Meets Scrum

*"Scrum is one of many in the agile process. You can think of agile as an umbrella term that encompasses other processes, such as Extreme Programming, Kanban and more"* as noted by Mike Cohn.

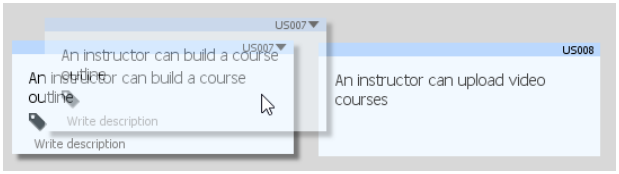
So how do they relate to each other? The following example illustrates how you can use Kanban and Scrum together for agile software development by using Visual Paradigm UeXceler.

### Using Kanban

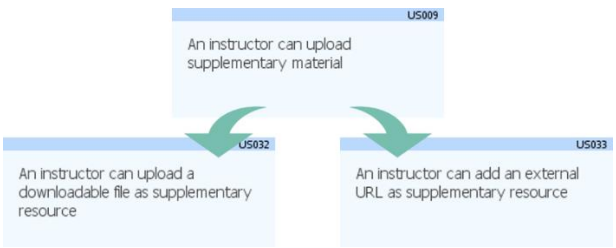
1. Communicate with the stakeholders and come up with user stories and things they want to do and the benefits associated with them.



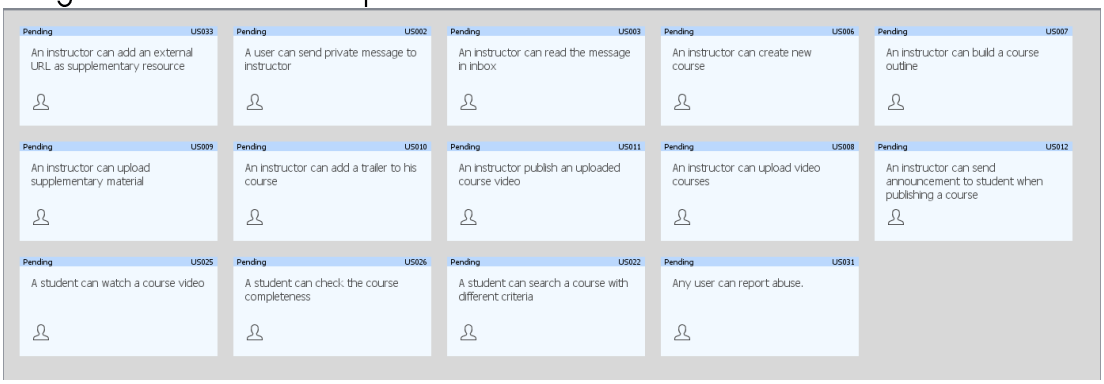

2. Prioritize user stories.




3. Splitting big user stories into smaller, so that they are suitable for developing within two weeks' time.

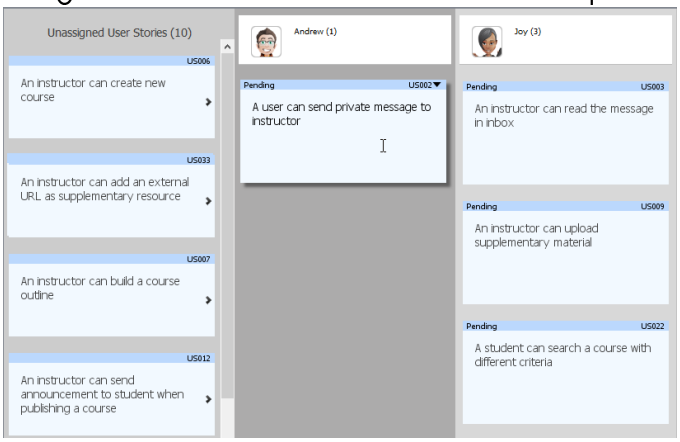


4. Assign user stories into a sprint.

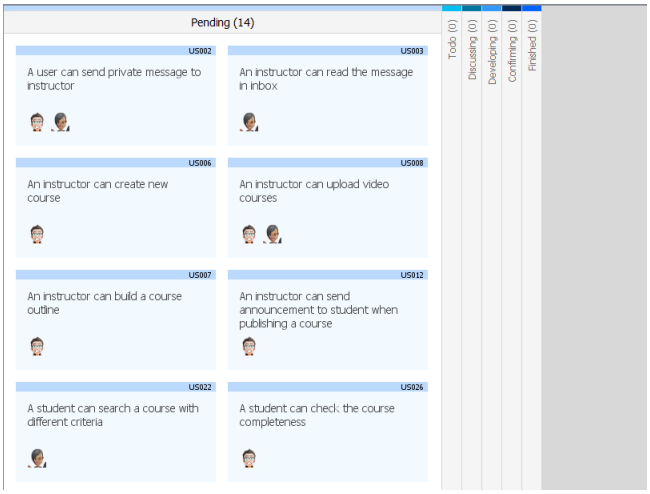



5. Estimate and design a good formation of the team to minimize bottleneck of workflow as mentioned in the WIP section.

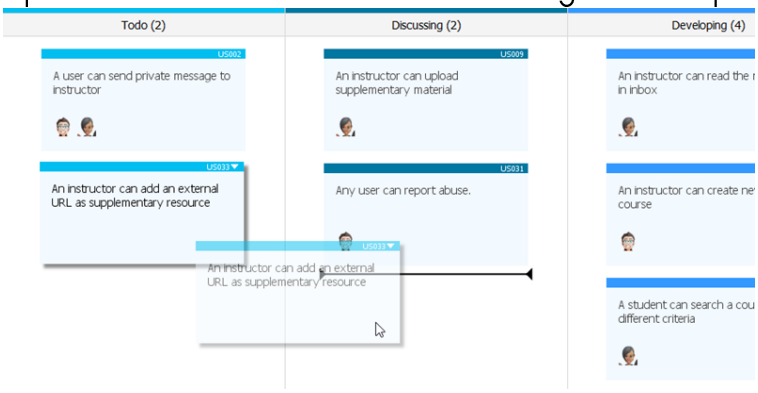
6. Assign work to the team members for the sprint.



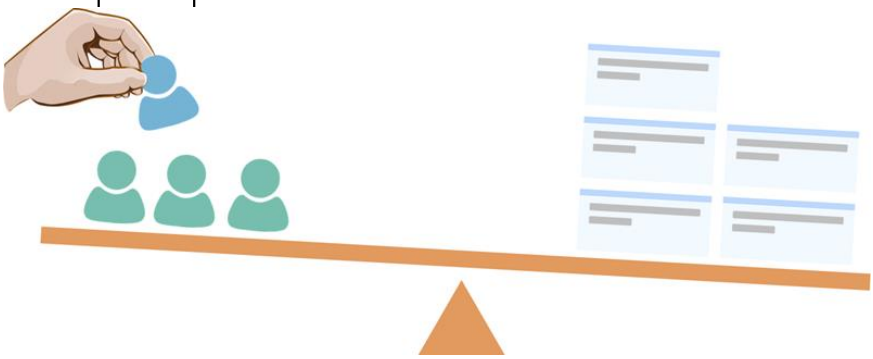

7. Switch to Kanban board and you will see all user stories automatically dropped into the **Pending** column.



8. Update the statuses of user stories throughout the sprint.




9. Empirically adjust the team formation to ensure there is no bottlenecks throughout the development process.



# Wireframe

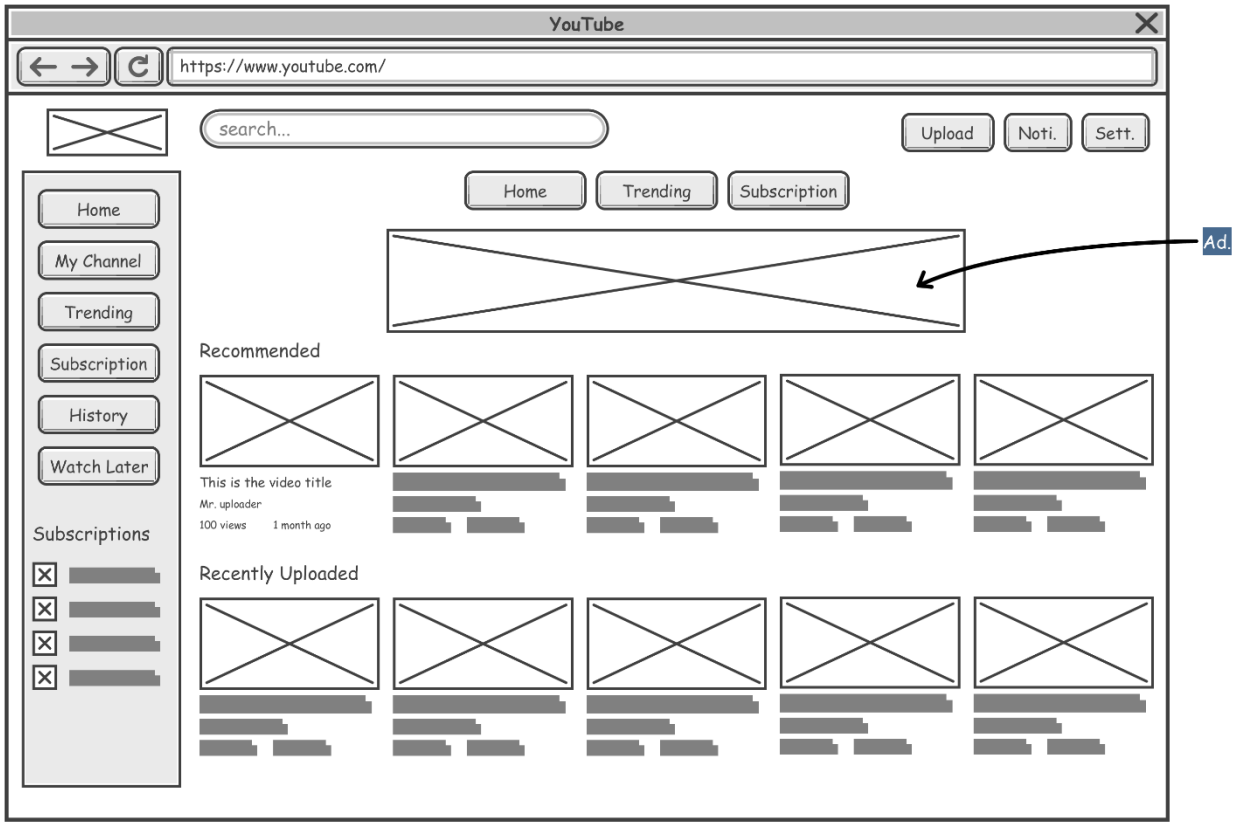
Customers need to recognize how a proposed feature will work. But simply describing to them verbally or textually for the targeted feature to be built may be quite challenging for their imagination. Wireframing can be extremely helpful in squaring that circle, as it can be served as a "show, don't tell" visual mock-up tool for confirming of system design ideas with customers.

## What is a Wireframe?

Wireframes are screen sketches of a system, sometimes referred to as a blueprint or skeleton. They are created for the purpose of presenting and explaining system design ideas to customers, which ultimately leads to a consensus on the ideas proposed.

Wireframes show "just enough" information associated with the feature proposed. Each wireframe comprises basic graphic elements that represent the screen components, or the placeholders of screen components. A wireframe depicts screen layout and how screen content is arranged. It focuses on what a screen does and how users will interact with the system to achieve his need, instead of how the final screen will really look like.

Before we go further, let's take a look at a wireframe created based on youtube.com



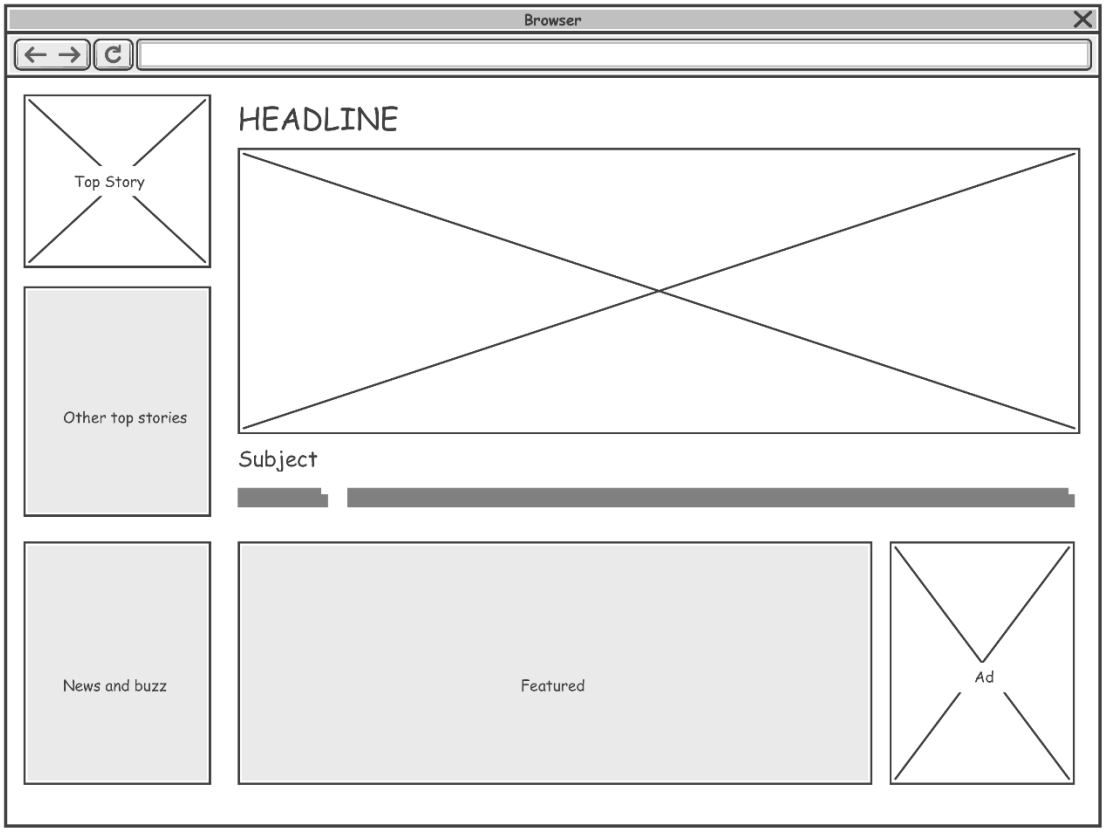

As you can see, a wireframe is just that simple and straight forward. Everyone could understand it without difficulties. It helps development team explain how users will interact with the web site easily.

## Wireframe is only a blueprint

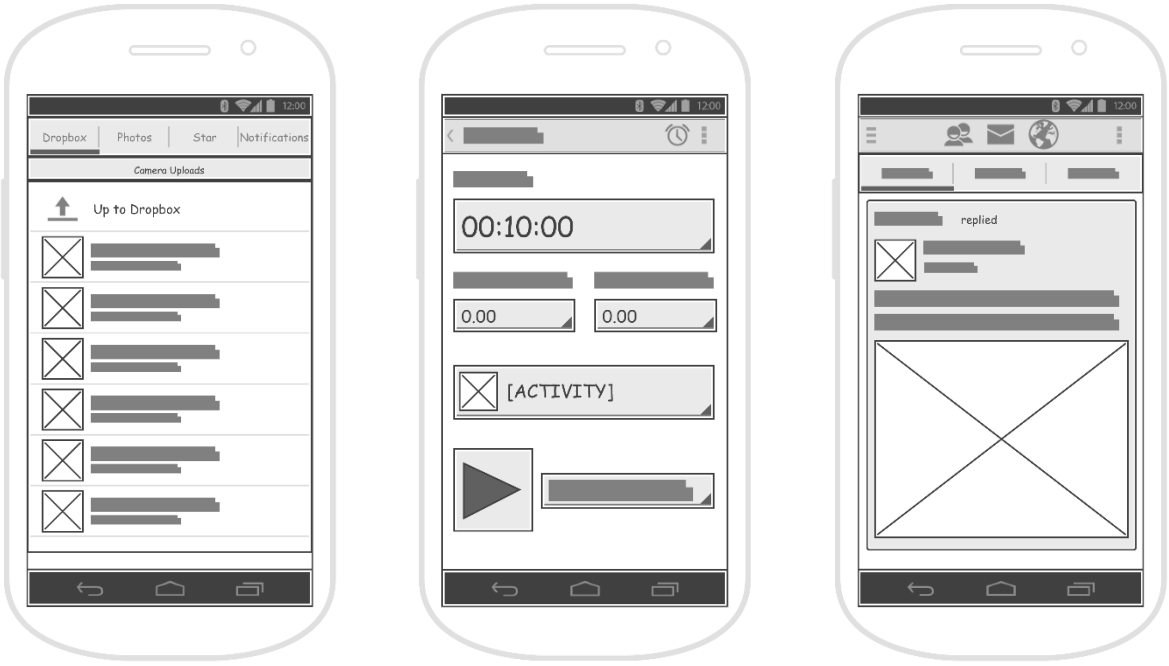
A wireframe is only a low-fidelity version of a user interface and are not meant to be a representation of real screen. Wireframes are intended to be used to demonstrate the functionalities, user interactions and screen flows, without explicitly specifying how screen components should look like and how the components should behave in order to achieve the highest usability.

## Wireframe examples

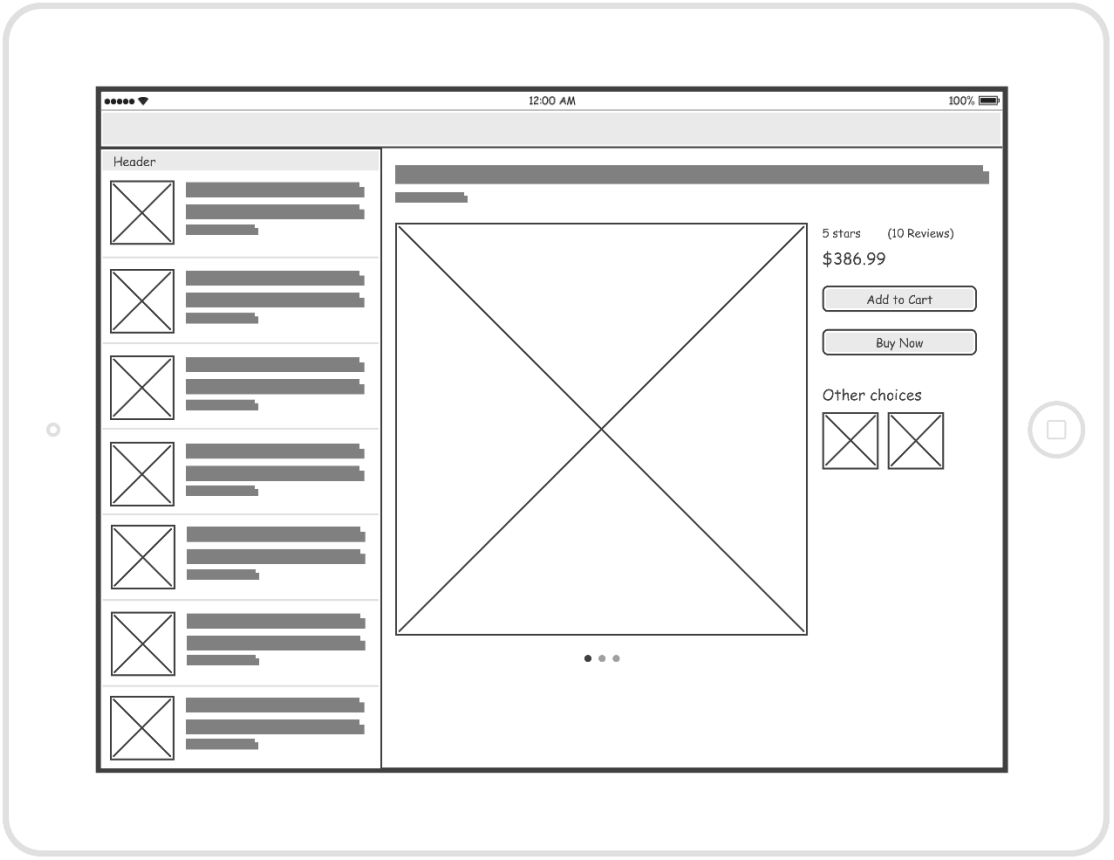
Here is a web wireframe example for a home page:



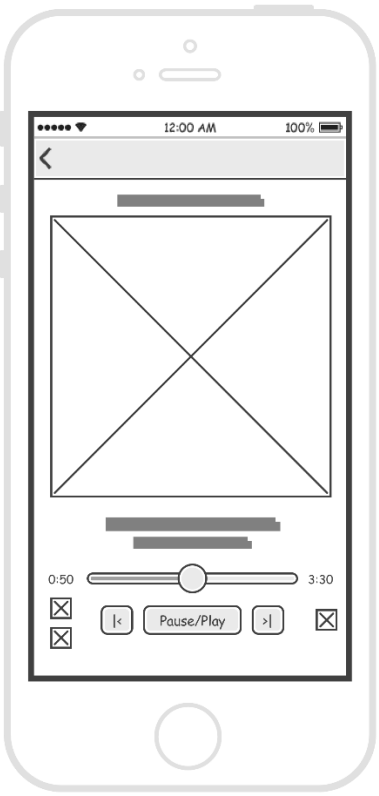

Here are some wireframe examples for Android apps:



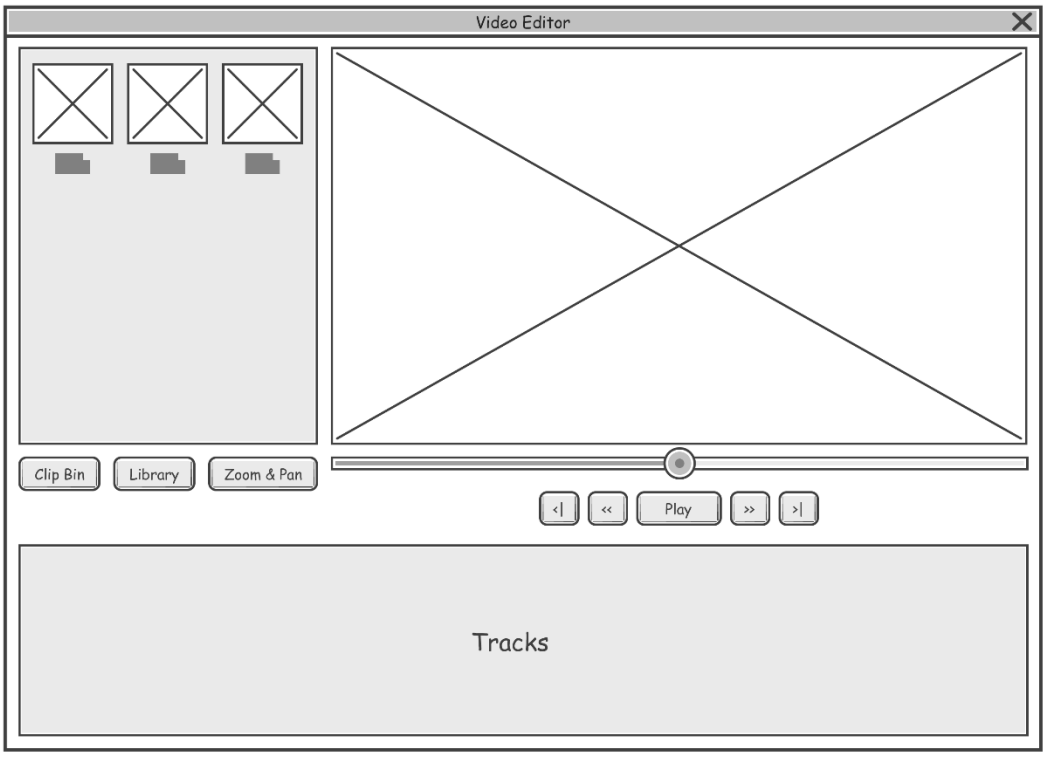

Here is an iPad wireframe example for an iPad app:




Here is an iPhone wireframe example for an iPhone app:




Finally, here is a wireframe example for a desktop application:



## Benefits of using wireframe

### Clarify user interface

Clients may not understand technical screen design jargons like slider, flip box, bootstrap, etc. Wireframe comprises basic graphic elements that everyone can understand, helping the clients to know how the features will function and how they can interact with the system to achieve what they want.

### Early consideration of usability

User experience (UX) is an important consideration in every software nowadays. The use of wireframes in requirements capturing brings the consideration of user experience to the beginning of project. Without developing any prototypes or drawing any real screen design, users can still experience how the system will work.

### Cost-efficient

It takes time and expertise to create full-blown, high-fidelity screen designs, which end up causing a large expense. Wireframing is a quick and inexpensive way to create basic screen sketches. It also makes tweaking or even overhauling sketches simple and inexpensive.

### More willing to make changes

Design changes are inevitable. The problem with confirming design ideas using complete system mockup or prototype, is that a considerable amount of work has already been done, and will involve a considerable amount of rework, which means extra time, effort and expense. In that situation, customers and development team are more reluctant to voice concerns and request changes. On the contrary, it takes much less time to produce and revise wireframes. Everyone will be more willing to request and make changes.

### Engaged clients

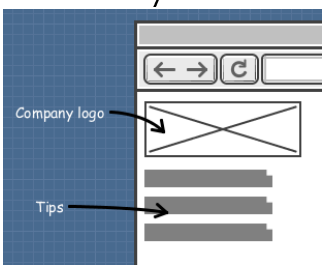
Because wireframes are rough and loose, they give the clients the room to brainstorm and voice their suggestions. And, because the turnaround time of refining wireframe is short, clients are actively involved in providing feedback, which makes them more likely to sign off the final design.

### How to use wireframe effectively?

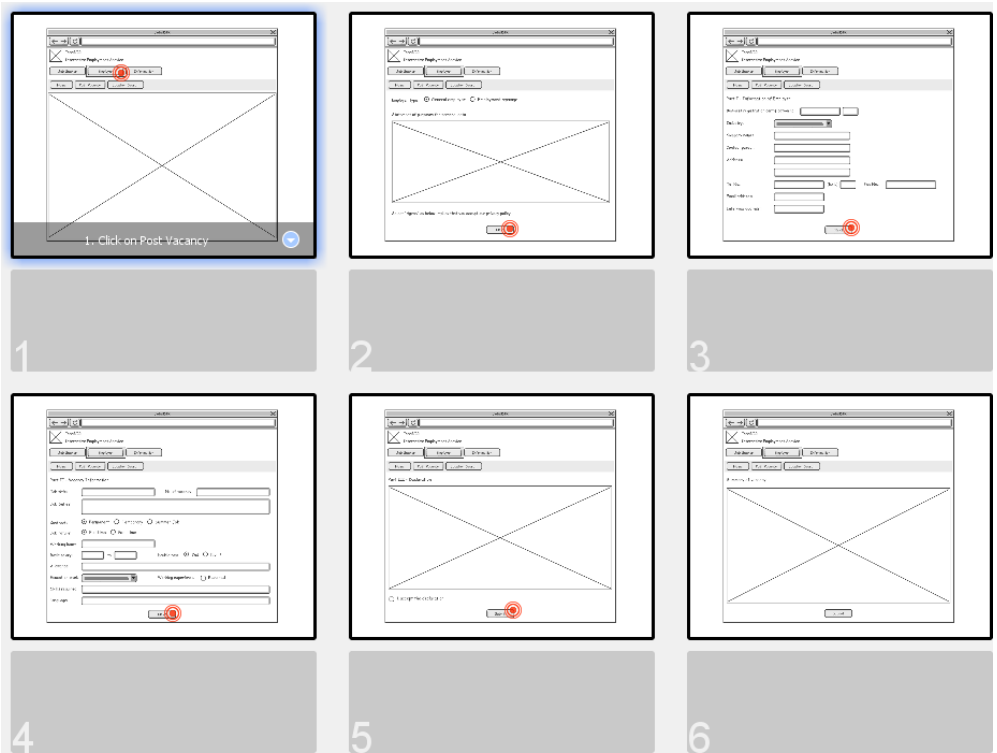
The use of wireframe can bring many benefits to both the development team and clients, but this is the case only when you use it wisely and properly. A typical misuse of wireframe is to treat it as a replacement of screen design. This makes the production and refinement difficult and costly, reducing the usefulness of wireframing. In this section we will go through some of the effective wireframing tips.

- A wireframe is intended to be simple and just enough. It is simple so that it can be produced quickly and easily, and makes no hesitation for a discard and re-work. The low-fi presentation also makes it more comprehensive and communicative. Therefore, do not need to spend too much time on beautifying the drawing, aligning things, or using pretty typology and etc.
- In a wireframe, instead of showing any actual content, we can replace a large chunk of text (the actual context) with a placeholder of text. This is to avoid time being spent on preparing the content unnecessarily, and to prevent the readers from being distracted by the text content. But if the displaying of text is needed, you may consider placing some dummy text there instead. You can easily find a dummy text generator on the internet.

- The use of annotation helps you describes an element (e.g. “Company logo”) or to explain something related to its behavior (e.g. “Hide in 5 seconds”). Use it if necessary. But again, don’t attempt to document each of the wireframe elements. You should only use annotation whenever it necessary.

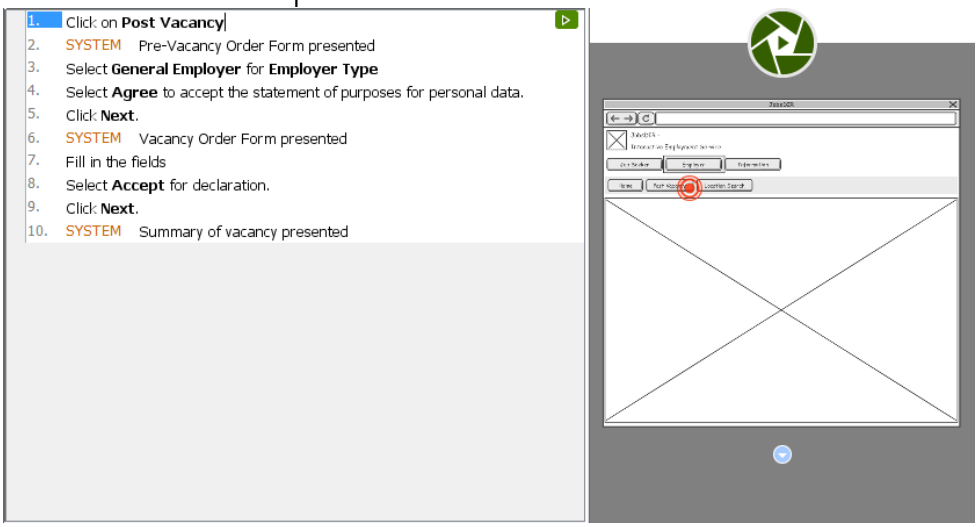



- Wireframes can be hand-drawn, but we usually create wireframes with software for more efficient and easier to manage of our works. Besides, some wireframe software provides you with features that paper-and-pencil cannot accomplish. Here are three of them:
    - State – The wireframing tool of Visual Paradigm supports the concept of state, which allows you to create a child wireframe based on an existing one. It is not only save you time in creating a screen flow with a sequence of similar child wireframes, it also makes refinements of the related child wireframes much easier (as we make changes in the initial state of a wireframe, the changes will also be reflected in all its’ child states wireframes)
    - Storyboard – A storyboard presents the screen flow of a particular scenario. It makes the wireframes more manageable and the presentation much easier.




    - Managing wireframes by User story – User story is an agile tool for recording user’s concerns and requirements. To include wireframes as part of a user story’s scenario shows how user will use the feature in do part of their job described in that user story. Besides, when developer start implementing the user story, he can check the wireframe to gain

ideas about user's expectation.

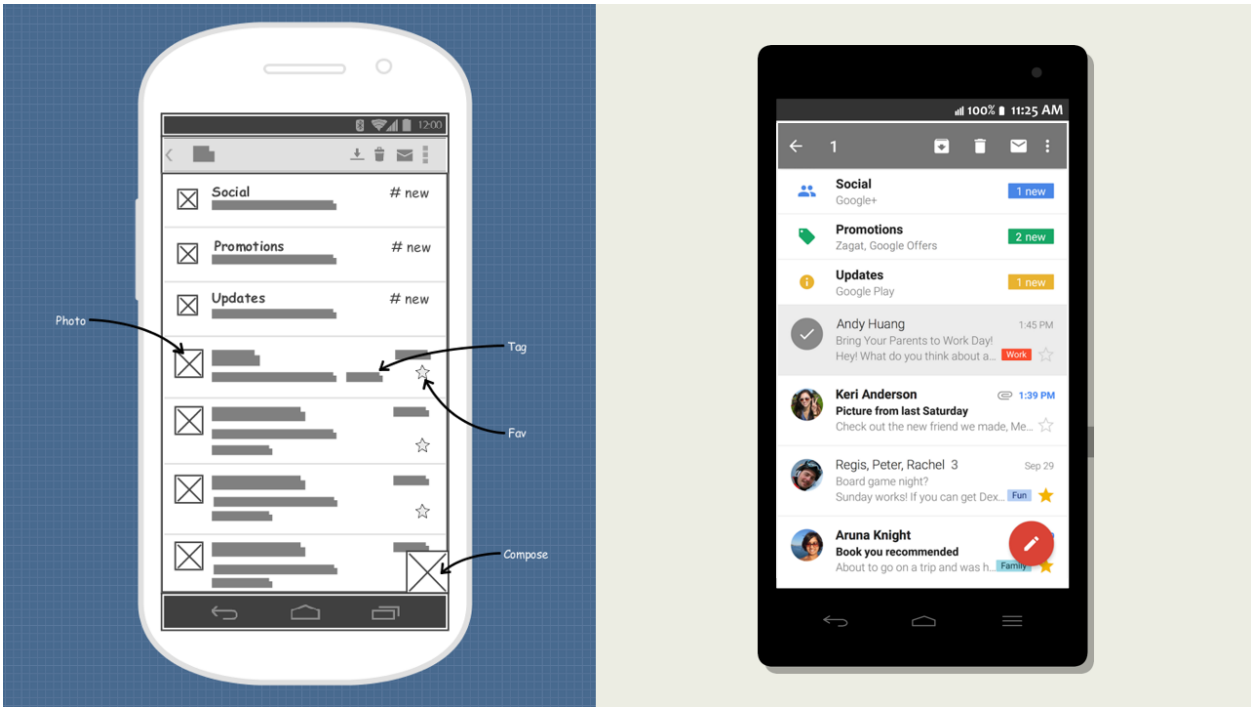


# Android Phone Wireframe

Android phone wireframes are screen sketches of an app that's run on an Android phone. We create Android phone wireframe for confirming of app design ideas with end users.

## What is an Android Phone Wireframe?

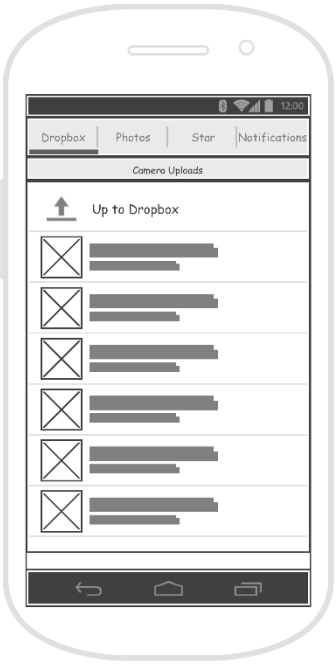
Android wireframes are screen sketches of an Android apps. It helps you present and explain design ideas of apps to customers, which ultimately leads to a consensus on the ideas proposed.



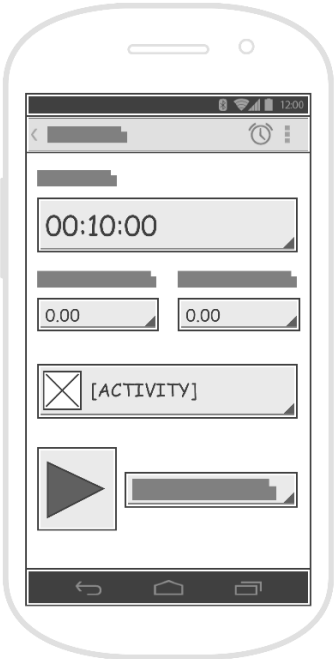

Wireframes are intended to use to demonstrate the functionalities, user interactions and screen flows, without explicitly specifying the actual screen components should look like and how the components should behave in order to keep the upfront development effort and cost in its lowest minimum.

## Android wireframe examples

Here is an Android wireframe example created based on Dropbox. As you can see, instead of showing real photos we used image placeholders. Placehooder of text has also been used instead of showing any file name.




The following Android wireframe example is about a sport tracking app. It consists of the mixed use of label, spinner, image (placeholder) and polygon shape (i.e. the play button).

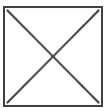



## Android phone wireframe components

An Android app wireframe is only a low-fidelity version of an interface. Each Android wireframe comprises basic graphic elements that represents the screen components, or the placeholders of screen components. The following are the Android wireframe components you can use in creating a wireframe.

### Image

An image can be an icon or any image data.

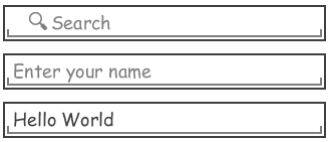


## Label

A label is a text on screen.

## Text Field

A text field allows user to type text. Touching the text field shows the cursor and displays the keyboard. User can then start typing. Text fields also supports functions like ranged text selection, copy, cut and paste.




## Button

A button allows user to touch it for triggering an action. It consists of a text, an icon, or both text and icon that shows the action to trigger.


## Toggle Button

A toggle button allows user to change the selection between two possible choices by touching it.


## Switch

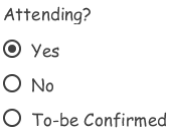
A switch is a kind of toggle button. It was introduced in Android 4.0. It allows user to switch between two states (e.g. on or off) when touching it.


## Checkbox

Checkboxes allow user to select one or more options from a set.

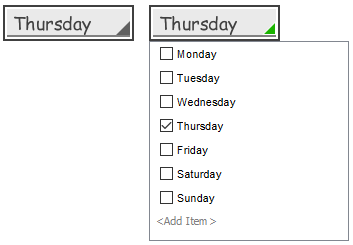

## Radio Button

Radio buttons allow the user to select one and only one option from a set. If you think that the user needs to see all available options on the screen, use radio buttons. Otherwise, use a spinner instead.




## Spinner

A spinner provides a drop-down menu for the quick selection of a value from a set. A spinner shows in its 'body' the currently selected value. By touching the spinner, a drop-down menu will be displayed, listing all the available values from which the user can make a selection.




## Progress

A progress indicates the progress of certain running operation in a circle shape.

## Progress Bar

A progress bar is a bar that indicates the progress of certain running operation. It displays how far the operation has progressed.

## Seek Bar

A seek bar allows the selection of progress level. A seek bar has a draggable thumb. User can drag the thumb to left or right to set the current progress level or use the arrow keys.

## Ranking Bar

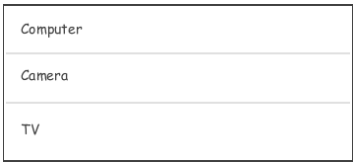
A ranking bar supports rating and the displaying of rating in stars. User can touch the star on the bar to set the rating.

## Panel

A panel is a visual container of wireframe components.

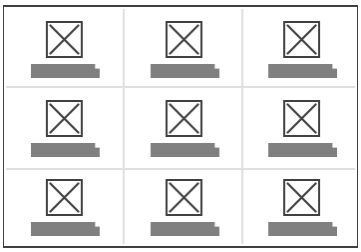
## List View

A list view displays a list of scrollable items.




## Grid View

A grid view displays items in a two-dimensional and scrollable grid.

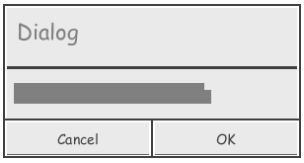


## Tab Host

A tab host is a container of tabs. User can click to select a specific tab to populate related content.

## Dialog

A dialog is a small window that prompts the user to make a decision (e.g. Confirm/Cancel). A dialog is shown on top of all the other components on a screen. It is used for modal events that require users to take an action before they can continue.




## Date Picker

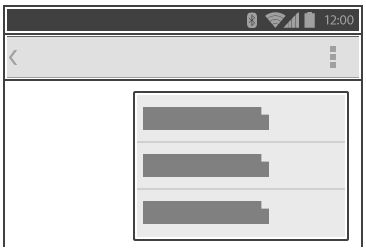
Date picker is a kind of dialog that provides controls for selecting each part of a date (month, day, year). The use of date picker ensures a valid date is being picked.

## Time Picker

Time picker is a kind of dialog that provides controls for selecting each part of time (hour, minute, AM/PM). The use of time picker ensures a valid time is being picked.


## Menu

A menu provides user with access to available user actions.

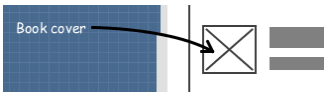



## Toasts

A toast is a simple description presented in a small popup, giving feedback on a current screen activity (e.g. Message saved). Toasts automatically disappear after a timeout.


## Annotation

An annotation is a piece of text that can be used to describe a control or behavior on a wireframe.




## Rectangle

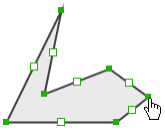
A rectangle shape that can be used for annotation and highlight in the wireframe.

## Oval

An oval shape that can be used for annotation and highlight in the wireframe.

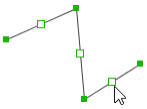
### Polygon

A polygon that can be used for annotation or highlight in the wireframe. You can add more points to the borders of a polygon and adjust their position.



### Line

A line that can be used for annotation or highlight in the wireframe.

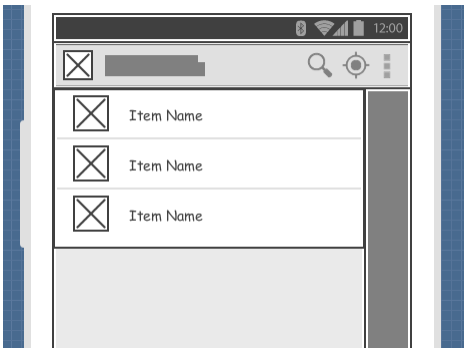


### The action bar

The action bar is a primary toolbar that appears at the top of the screen. It displays the title of the current activity as well as to provide navigation to application level features/functions.


### Drawer

A drawer is a top level container that displays the app's main navigation options on the left edge of the screen. Drawer is revealed only when user swipes a finger from the left edge of the screen or touches the app icon in the action bar.




### Keyboard

Keyboard is a virtual panel that allows user to touch the keys on it for inputting text.

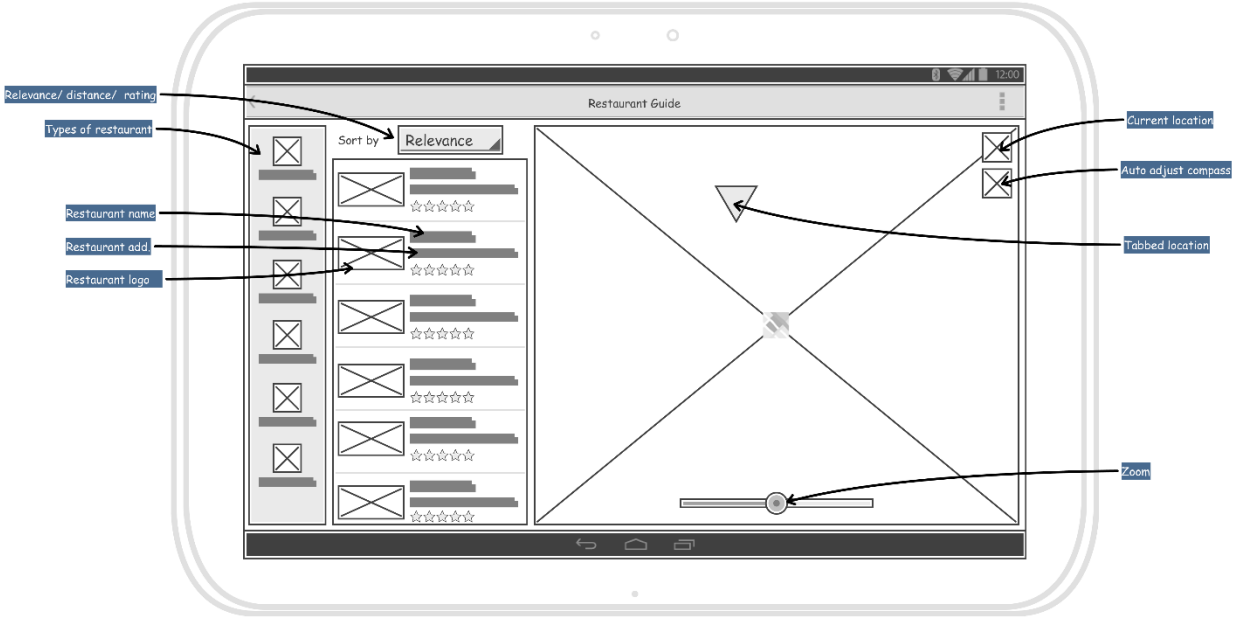

Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
?123

# Android Tablet Wireframe

Android tablet wireframes are screen sketches of an app that's run on an Android tablet. We create Android tablet wireframe for confirming of app design ideas with end user.

## What is an Android Tablet Wireframe?

Android wireframes are screen sketches of an Android apps. It helps you present and explain design ideas of apps to customers, which ultimately leads to a consensus on these ideas.

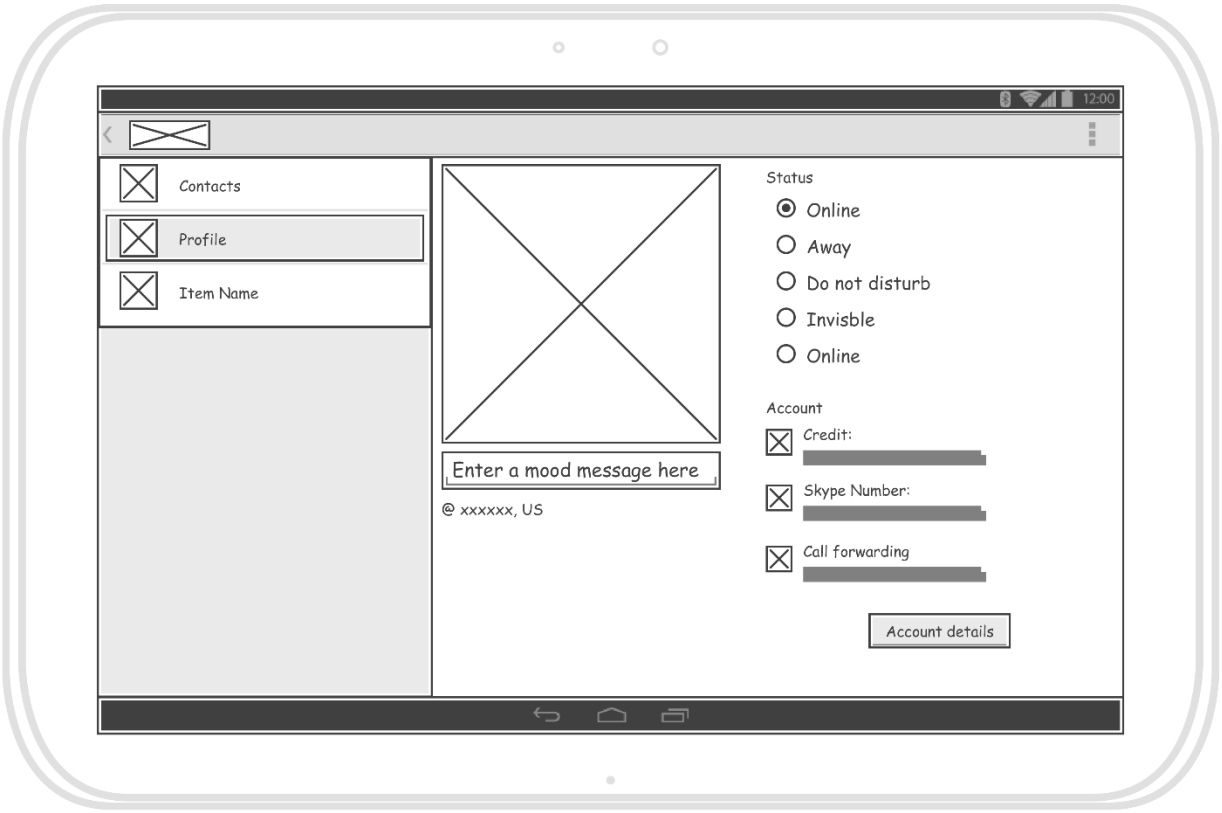



Wireframes are intended to use for demonstrating functionality, user interactions and screen flows, without explicitly specifying how screen components should look like and how the components should behave in order to the upfront development effort and cost in its lowest minimum .

## Android wireframe examples

Here is an Android wireframe example created based on Gmail. Instead of showing real avatar for users, we used image placeholders. Label placeholders are used to represent user names, mail subjects and content. Plain buttons are used instead of drawing any fancy buttons.


The following Android tablet wireframe example is created based on Skype. Take a look at the radio buttons on the right hand side of the wireframe. Although in the real screen those radio buttons don't really look like radio buttons, it's perfectly alright to use radio buttons in wireframing. This is because in wireframing our main focus is to demonstrate functionality, user interactions, basic layout and screen flow. What we want to express in the wireframe is that the screen does allows the selection of status and the selection is mutually exclusive (that's why a radio button is used). We do not care about how the radio buttons will look like. If you find yourself spending time on designing the appearance of individual components in wireframing, you are probably mis-using wireframe.

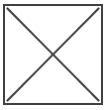



## Android tablet wireframe components

An Android tablet wireframe contains basic graphic elements that represent screen components, or the placeholders of screen components. In this section we will go through the Android tablet wireframe components available.

### Image

An image can be an icon or any image data.

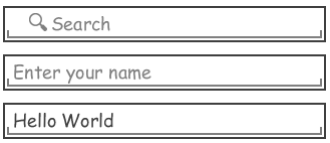


### Label

A label is a text on screen.

### Text Field

A text field allows user to type text. Touching the text field shows the cursor and displays the keyboard. User can then start typing. Text fields also supports functions such as, ranged text selection, copy, cut and paste.




## Button

A button allows user to touch it for action triggering. It consists of a text, an icon, or both text and icon that shows the action to be triggered.


## Toggle Button

A toggle button allows user to change a selection between two possible choices by touching it.


## Switch

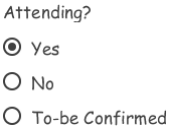
A switch is a kind of toggle button. It was introduced in Android 4.0. It allows user to switch between two states (e.g. on or off) when touching it.


## Checkbox

Checkboxes allow user to select one or more options from a set.

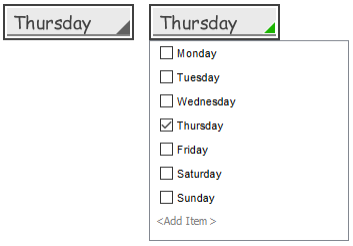

## Radio Button

Radio buttons allow the user to select one and only one option from a set. If you think that the user needs to see all available options on the screen, use radio buttons. Otherwise, use a spinner instead.




## Spinner

A spinner provides a drop-down menu for the quick selection of a value from a set. A spinner shows in its 'body' the currently selected value. By touching the spinner, a drop-down menu will be displayed, listing all the available values from which the user can make a selection.



### Progress
A progress indicates the progress of certain running operation in a circle shape.

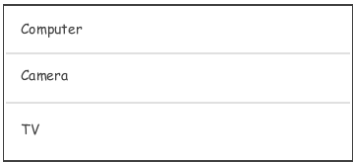
### Progress Bar
A progress bar is a bar that indicates the progress of certain running operation. It displays how far the operation has progressed.

### Seek Bar
A seek bar allows the selection of progress level. A seek bar has a draggable thumb. User can drag the thumb left or right to set the current progress level or use the arrow keys.

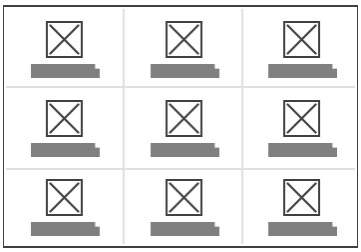
### Ranking Bar
A ranking bar supports rating and the displaying of rating in stars. User can touch the star on the bar to set the rating.

### Panel
A panel is a visual container of wireframe components.

### List View
A list view displays a list of scrollable items.




### Grid View
A grid view displays items in a two-dimensional and scrollable grid.

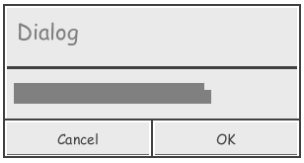


## Tab Host

A tab host is a container of tabs. User can click to select a specific tab to populate related content.

## Dialog

A dialog is a small window that prompts the user to make a decision (e.g. Confirm/Cancel). A dialog is shown on top of all the other components on a screen. It is used for modal events that require users to take an action before they can continue.




## Date Picker

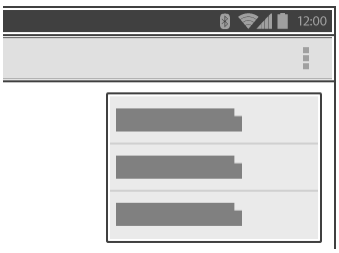
Date picker is a kind of dialog that provides controls for selecting each part of a date (month, day, year). The use of date picker ensures a valid date is being picked.


## Time Picker

Time picker is a kind of dialog that provides controls for selecting each part of time (hour, minute, AM/PM). The use of time picker ensures a valid time is being picked.


## Menu

A menu provides user with access to available user actions.

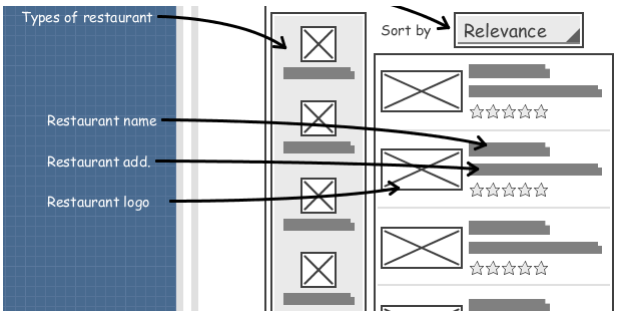



## Toasts

A toast is a simple description presented in a small popup, giving feedback on a current screen activity (e.g. Message saved). Toasts automatically disappear after a timeout.


## Annotation

An annotation is a piece of text that can be used to describe a control or behavior on a wireframe.




## Rectangle

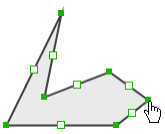
A rectangle shape that can be used for annotation and highlight in the wireframe.

## Oval

An oval shape that can be used for annotation and highlight in the wireframe.

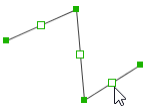
## Polygon

A polygon that can be used for annotation and highlight in the wireframe. You can add more points to the borders of a polygon and adjust their position.



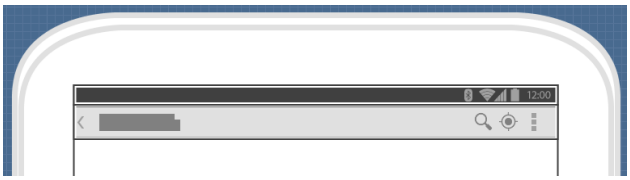
## Line

A line that can be used for annotation and highlight in the wireframe.



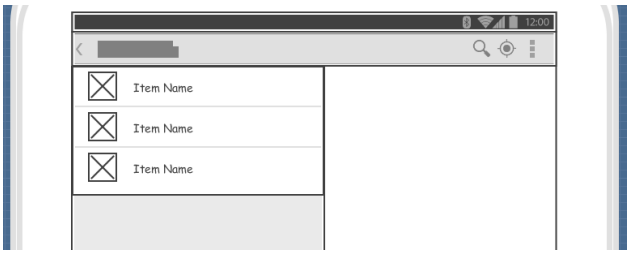
## The action bar

The action bar is a primary toolbar that appears at the top of the screen. It displays the title of the current activity as well as to provide navigation to application level features/functions.




## Drawer

A drawer is a top level container that displays the app's main navigation options on the left edge of the screen. Drawer is revealed only when user swipes a finger from the left edge of the screen or touches the app icon in the action bar.




## Keyboard

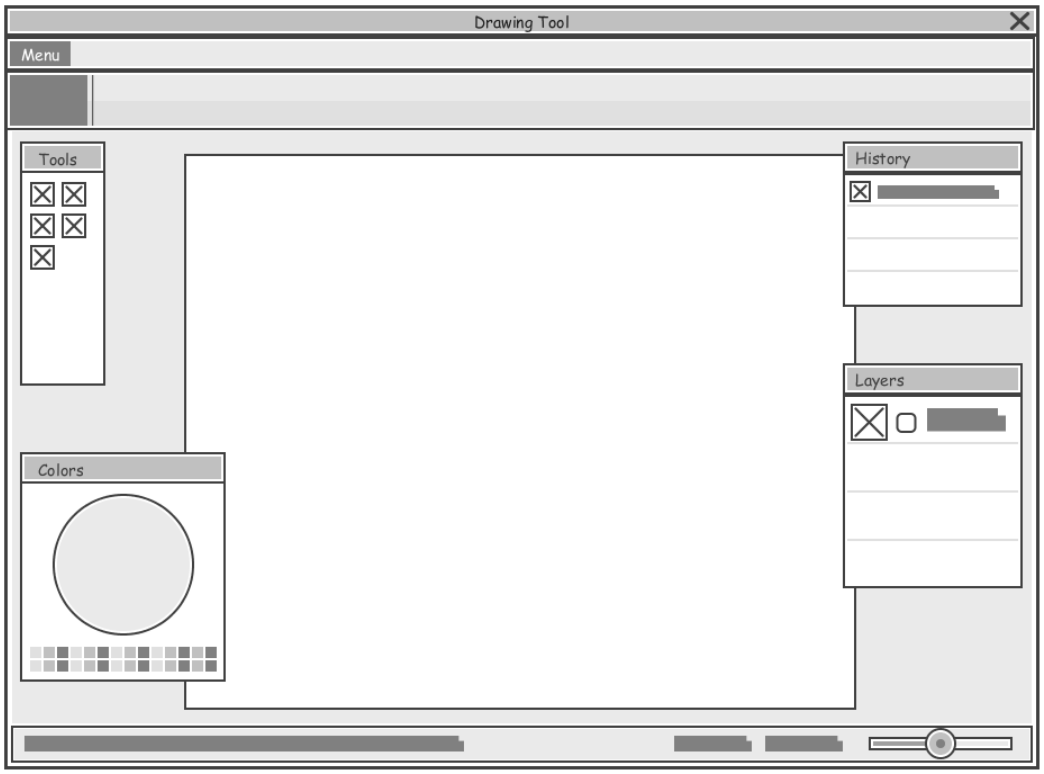
Keyboard is a virtual panel that allows user to touch the keys on it for inputting text.

# Desktop Application Wireframe

Desktop application wireframes are screen sketches of applications that run on a desktop computer. We create desktop wireframe for confirming application design ideas with end users.

## What is a Desktop Application Wireframe?

Desktop application wireframes are screen sketches of application that run on a desktop computer. It helps you present and explain design ideas of applications to customers, which ultimately leads to a consensus on the ideas proposed.




Wireframes are intended to use to demonstrate the functionalities, user interactions and screen flows, without explicitly specifying the actual screen components should look like and how the components should behave in order to keep the upfront development effort and cost in its lowest minimum.

## Desktop application wireframe example

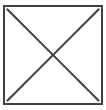
Here is a desktop application wireframe example created for a general mail program. Although the wireframe is formed by simple components that represent buttons, labels (placeholder), checkboxes, text box and panels, it's already clear enough for stakeholder to get the design ideas.


## Desktop application wireframe components

A desktop application wireframe is only a low-fidelity version of an interface. Each desktop application wireframe comprises basic graphic elements that represents the screen components, or the placeholders of screen components. The following are the desktop wireframe components you can use in creating a wireframe.

### Image

An image can be an icon or any image data.



### Label

A label is a text on screen.

### Text Field

A text field allows user to type text. Clicking inside the text field shows the cursor at the clicked place. User can then start typing. Text fields also supports functions like ranged text selection, copy, cut and paste.


## Button

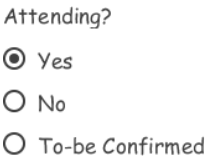
A button allows user to click on it for triggering an action. It consists of a text, an icon, or both text and icon that shows the action to trigger.


## Checkbox

Checkboxes allow user to select one or more options from a set.

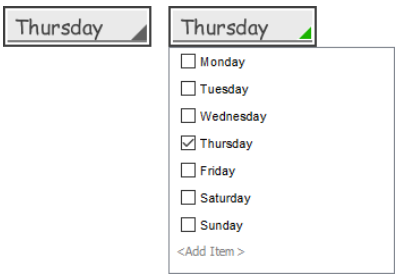

## Radio Button

Radio buttons allow the user to select one and only one option from a set. If you think that the user needs to see all available options on the screen, use radio buttons. Otherwise, use a radio button instead.




## Combo box

A combo box provides a drop-down menu for the quick selection of a value from a set. A combo box shows in its 'body' the currently selected value. By clicking on a combo box, a drop-down menu will be displayed, listing all the available values from which the user can make a selection.




## Spinner

Spinner lets user to choose a value from a range of available choices, which behaves a bit like a combo box. But unlike combo boxes, spinners do not have a drop-down menu. You can change a value by pressing the up and down button attached to the spinner.

### Progress Bar

A progress bar is a bar that indicates the progress of certain running operation. It displays how far the operation has progressed.

### Slider

A seek bar allows the selection of progress level. A seek bar has a draggable thumb. User can drag the thumb to left or right to set the current progress level or use the arrow keys.

### Panel

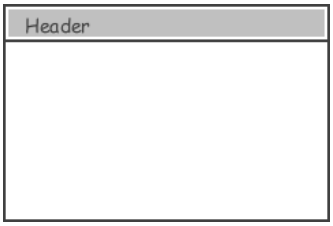
A panel is a visual container of wireframe components.

### Split bar

A split bar provides visual separation between two panels.

### Accordion Panel

An accordion panel is a collapsible panel that allows user to display or hide a section of content, as presented in the panel.

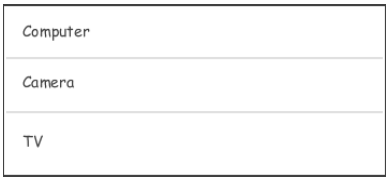



### Scroll bar

A scroll bar allows user to update the visual content in a widget by dragging its knob up and down, or left and right, depending on the orientation of the bar.

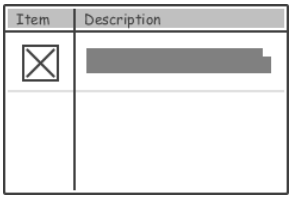
## List View

A list view displays a list of scrollable items.



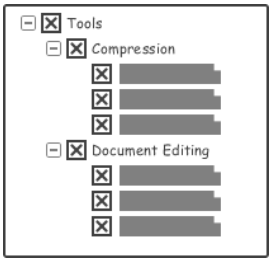

## Table

A table allows the structure representation of components.



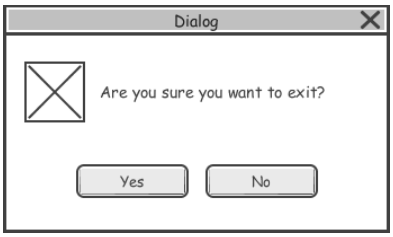

## Tree

A tree allows the representation of items in the form of a hierarchy.



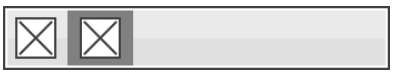

## Dialog

A dialog is a small window that prompts the user to make a decision (e.g. Confirm/Cancel). A dialog is shown on top of all the other components on a screen. It is used for modal events that require users to take an action before they can continue.




## Toolbar

A toolbar provides user with access to available user actions by clicking on the buttons on the bar.



## Menubar

A menubar provides user with access to available menus.


## Menu

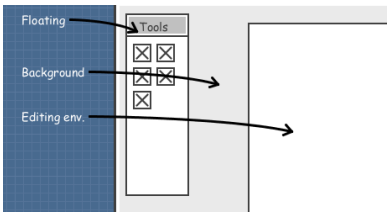
A menu provides user with access to available user actions.


## Segment Control

A segment control allows user to change the selection between two or more choices by clicking on it.


## Annotation

An annotation is a piece of text that can be used to describe a control or behavior on a wireframe.




## Rectangle

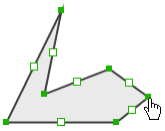
A rectangle shape that can be used for annotation and highlight in the wireframe.

## Oval

An oval shape that can be used for annotation and highlight in the wireframe.

## Polygon

A polygon that can be used for annotation or highlight in the wireframe. You can add more points to the borders of a polygon and adjust their position.



### Line

A line that can be used for annotation or highlight in the wireframe.