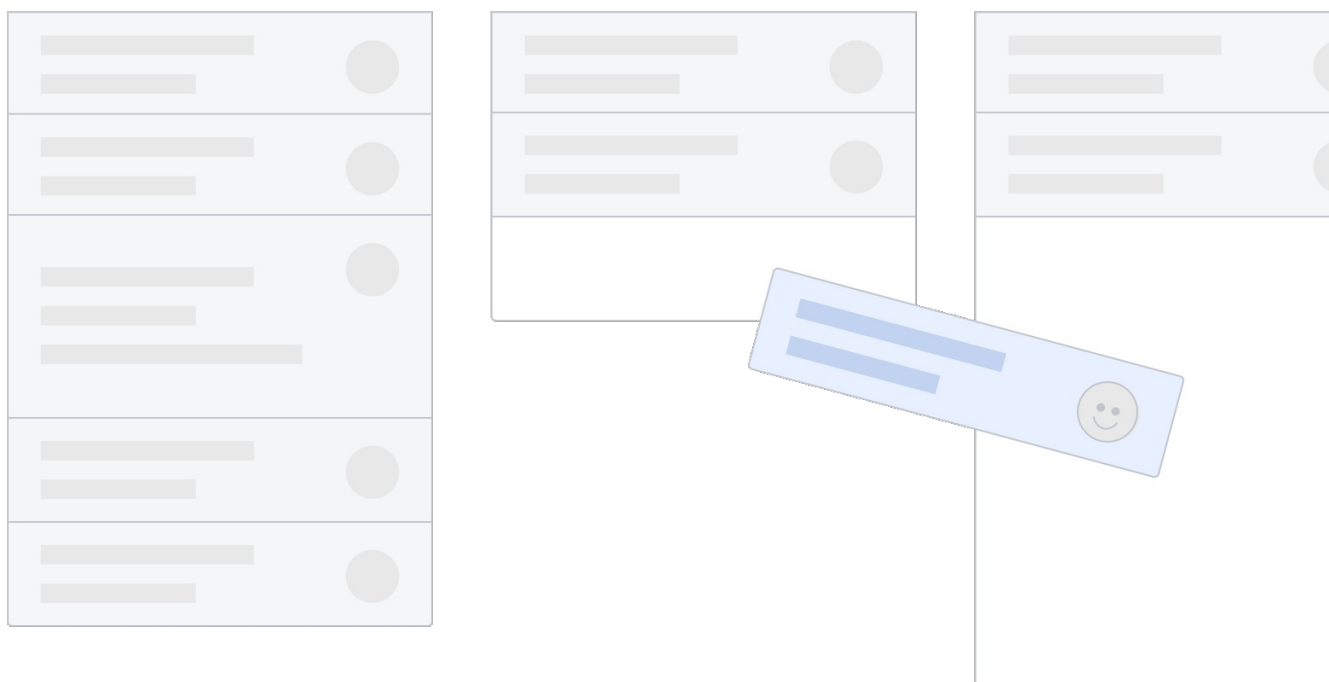


Kanban

A Quick and Easy Guide to Kickstart Your Project

How to implement Kanban in software development
Useful tips for organizing projects
Bug tracking workflow... and more.



CONTENTS

3 Introduction to Agile

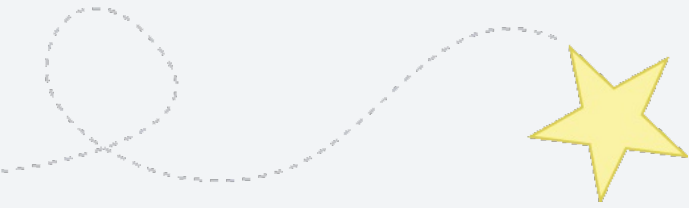
- 4 Going Agile
- 5 Kanban Basics
- 8 Getting Started With Kanban

10 Structuring Your Work

- 11 Labels
- 12 Filters
- 13 Columns
- 16 Priority
- 18 Bottlenecks
- 19 Tracking Time
- 20 Reports
- 21 Bug Tracking

23 People on the Project

- 24 Client
- 25 Developer
- 26 Manager



INTRODUCTION TO AGILE

If you're in IT, you've probably lived through this: clients demand last minute changes on a tight budget, you don't know what the other developers are doing, your work keeps overlapping, and you wonder how they come up with those unrealistic estimates.

And you've probably heard of a solution called Kanban, maybe even moved a few tasks in columns. That might have seemed too simple and didn't work that great - or you came across a 500-page book which made it sound really complicated.

This a quick read. By the time finish, you'll have a basic understanding of Kanban - and more importantly, how to start using it. We focus on the practical part, write from our own experience, and show you how to avoid some common pitfalls.

Everything you read here, you can apply right now in ActiveCollab and start managing your projects more efficiently.

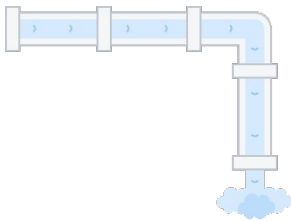
GOING AGILE

When starting a new project, the traditional way was: gather a team, book a conference room, and plan the whole project upfront. If something changes, you're in trouble - and in software development, that “something” changes all the time.

This called for a new project management philosophy, one that embraces the shifting nature of project requirements - and so Agile was born. Instead of coming up with a BIG plan and hoping nothing unexpected happens, Agile allows you to change a project's direction on the go.

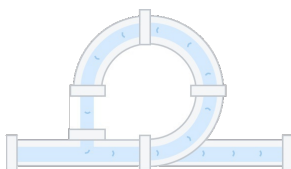
Let's compare the two approaches on a website redesign project:

The Traditional (Waterfall) Approach



You first plan everything—every color, button, expected user behaviour—and then start working. You work step by step and deliver when you're done. The client sees progress only when you hit a milestone. You stick to the plan because you've already agreed on all the details. If the client doesn't like the layout at the end, you have to do a lot of backtracking, without being able to charge your client for the extra work.

The Agile Approach



You get feedback early in the development. You see how the client responds and make adjustments on the go. Again, and again—each adjustment gets you closer to the finish. If the client doesn't like the layout, you can change it early on without losing time and money on dead ends.

The Agile approach consists of many overlapping methodologies and Kanban is one of them.

KANBAN BASICS

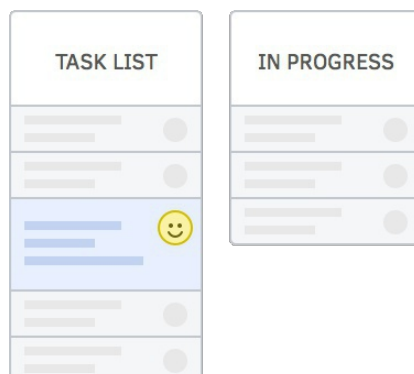
Kanban is a Japanese word which roughly translates to “a card you can see”. Basically, you organize tasks as cards on a board (or in a software). You then track progress by moving them across different columns.

Let's say you're developing a mobile app. You create several columns in the project: TO-DO, IN PROGRESS, REVIEW, DONE. Then you create and put all the tasks in the TO-DO column (eg. create a wireframe, design icon packs, write API calls, choose fonts and colors, fix loading bug).

When you start working on the wireframe, you move the card to the IN PROGRESS column. When it's finished, it goes to REVIEW. The client can see what's in REVIEW and give feedback. If it's good, you move it to DONE — if not, the card goes back to IN PROGRESS.

A common way to organize projects is by task types (like DESIGN, DEVELOPMENT), but that's not appropriate in Kanban because a task can't travel from one column to another. For example, a task like "Export hi-res logo" in DESIGN task list can't travel to DEVELOPMENT task list. But a feature request like "Put number of active users on admin dashboard" can travel from BACKLOG to DEVELOP to REVIEW.

As you can see, Kanban is very simple to use. It's applied in logistics, software development, and even as a productivity method to organize personal tasks.



PUSH VS. PULL

Most project management uses the push principle - a project manager creates a master plan and tells everyone on the team what to do. This can be inefficient because the manager can't know who's the best for a task (no matter how well they know the team).

But Kanban operates on the pull principle - team members get to choose the tasks themselves. The best person for the job will pluck it personally from the backlog. This way, they're empowered, own the work, and are more motivated to finish it. And when they finish, they won't sit idly but jump on the next task they feel comfortable with.



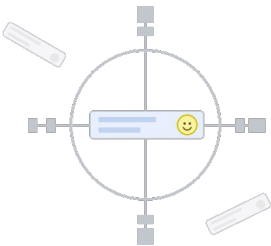
KANBAN BENEFITS

Flexible planning



You put everything that needs to be done in one column, and then decide what to work on. You don't have to lose time categorizing the tasks, just put them all in the backlog. If a task becomes urgent, you can always schedule it as the next one in the pipeline - this is different from Scrum where you only work on tasks that are scheduled for the current sprint.

Clear focus



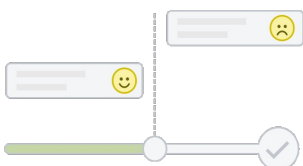
When you multitask, your productivity drops down by 40% and you do shoddy work. With Kanban, you limit the number of things you work on so you don't get swamped. Then you can focus on getting the task done and not subconsciously worry about other tasks.

Work satisfaction



You don't force developers to work on things that get arbitrarily assigned (as far as they know), but let them choose the tasks and be more motivated to work. Plus, people get credit for what they do, can set the pace themselves, make better estimates, and accomplish more.

Transparency

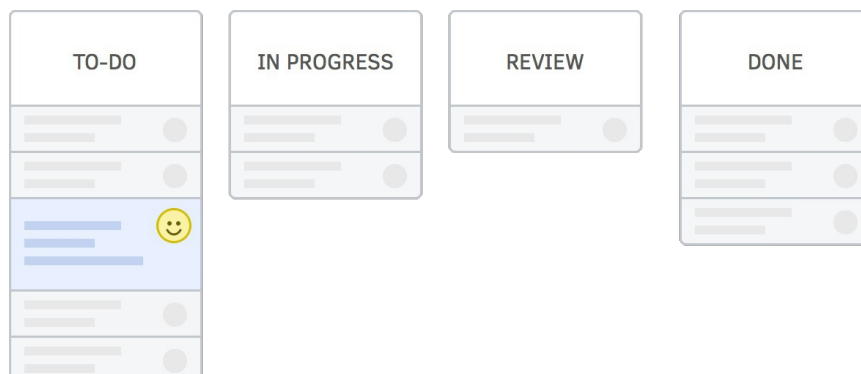


Tracking progress is easy and your meetings are more efficient. You can decide how many tasks can simultaneously be at any stage, and divert more resources to bottlenecks if needed. Also, you'll always know what's next for the release.

GETTING STARTED WITH KANBAN

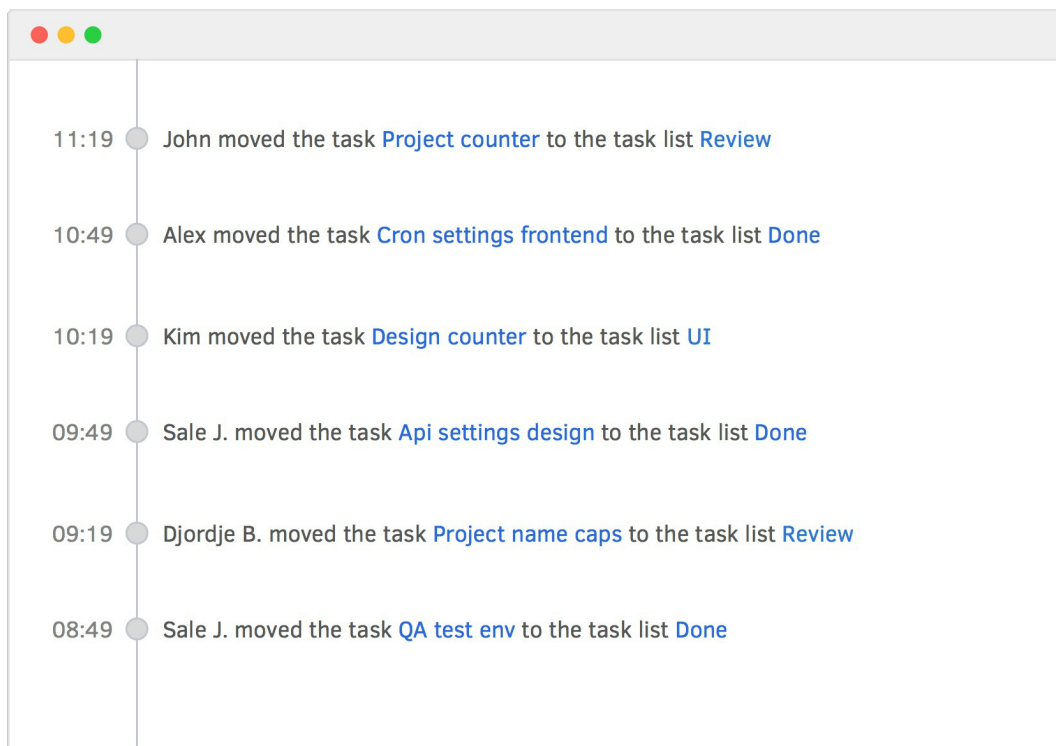
All you need is a big board that everyone on your team can see and pin cards to. But in this book, we are relying on Active Collab because it makes the process more efficient with features like @mentions, notifications, labels, assignees, due dates, filters, and more.

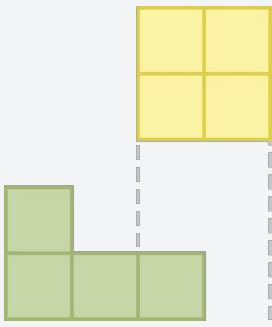
- 1** To get started, create a new project and add four task lists: TO-DO, IN PROGRESS, REVIEW, and DONE.
- 2** Next, add everything that needs to be done in the TO-DO column.
- 3** Decide who'll be in charge of what, who'll do the reviewing, and how you'll notify each other.
- 4** Let your team pick their tasks and move them to IN PROGRESS once they've started working.
- 5** If someone needs to know about updates on a task, you can subscribe them so they're notified about each update.
- 6** When you finish, move the task to the REVIEW column. If you're in a hurry, you can @mention the person who needs to do the review, or wait for them to check the column themselves when they have time.
- 7** If the work is good and the task passes review, it can be moved to DONE. If it needs more work, the reviewer moves it back to the IN PROGRESS column and writes what needs more grease in the comments section.



Before applying Kanban to your existing projects, try something smaller first. A team should set up a simple project, experiment, figure out the best workflow – and spread it.

Whenever a card gets moved, you can see it in Active Collab's Activity page too. This is useful for seeing what happened on a project since the last time you logged in, or on a specific date.





STRUCTURING YOUR WORK

In Active Collab, every task has a name, an assignee, and a due date. But those three parameters aren't enough to stay organized, especially when it comes to complex projects.

Use labels to enrich tasks with more information so you can later organize them more efficiently. Then you can use filters to see only the tasks that meet a certain criteria (eg. see all the tasks that are due next week, or all the bugs that need fixing).

Keep track of task progress by using task lists/columns. All tasks start in the backlog and move through different phases until they get completed. To decide on which tasks to work on, you need to determine each task's priority, and labels can help here.

All tasks require time, the scarcest resource on the market. Time determines how much better your app can be and how much you can charge for your work. That's why you should track how much time you spend on each task. Later, you can run reports and see how efficient you were. Also, because Kanban task management is so visual, spotting bottlenecks is very easy.

Kanban is useful for bug tracking also, so we included our own workflow at the end of the chapter.

LABELS

Some tasks are more important than others, but require more time. Labels are useful for denoting values like business importance, type of work, complexity, or anything else.



50, 100, 200, 300, 500, 1000

Business importance

Indicates the importance a task carries for your client so you can keep their business. Fixing an obscure bug may not be that important, but a special report generator is a deal breaker for some.



Type of work

Identifies what the user gets when you complete this. You can just do bug fixing, but it won't make the software noticeably better. But, you can't keep rolling out new features if half of them don't work. Also, don't forget that boring things like refactoring and code optimization are also important and deserve their own task and time.



S, M, L, XL

Complexity

Indicates how complex a task is and how much time it needs. You can't work only on big features and ignore simple things that contribute to customer satisfaction. Easy wins are important for team motivation as well as showing your clients that you're worth their money.

Tip: use emoji characters next to labels to make them easier to spot.

FILTERS

Use filters to quickly find what you're looking for. Later, you can batch edit the tasks and apply different labels or change due dates.

Assignee



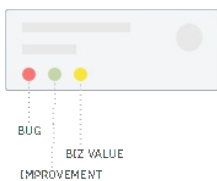
See how much work people on your team have - if it's too much, it's a clear sign they need to stop taking on too many responsibilities or you need a better task distribution.

Due Date



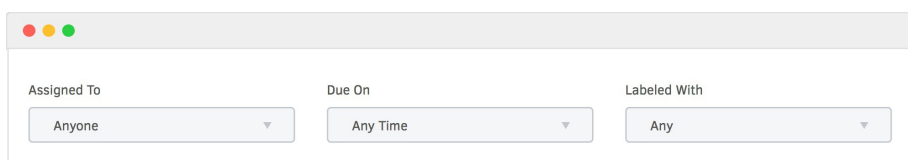
If you filter all the tasks that are due next week, you can create a meeting or talk personally with the developers to see if they have any blockers that prevent them from working on their tasks.

Label



You can use as many labels as you like. You can make a whole suite of labels. Then finding tasks related to Design or a top priority becomes easy. You can use different label colors so you don't mistakenly mark a task both as a feature and as a bug.

For example, you can dedicate a whole week to just working on bugs and removing them from the backlog before starting work on a newsome feature. Just use the filter and pick the tasks. Or you can make a weekly task quota: 10 bugs, 3 improvements, and 1 feature each week.



COLUMNS

What kind of columns you use depends on your existing workflow, type of work, and the structure of your team. It's best to keep it simple, but you do have more control and cover more scenarios with more columns. Here are some suggestions.

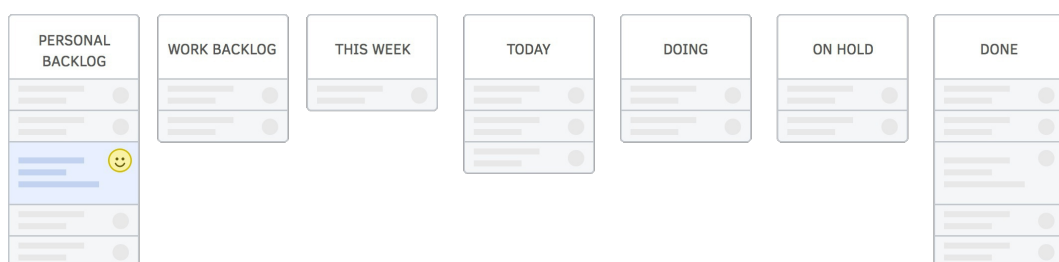
DESIGN

- RESEARCH - gather the problems and requirements you need to solve.
- UX - work on converting those requirements to a concrete wireframe or a sketch.
- REVIEW - the client is responsible to give you early feedback on the design.
- IMPLEMENTATION - once you have a prototype, you can start coding or creating a mockup .
- FINAL REVIEW - the client does one last review to decide whether the work goes to DONE or back to IMPLEMENTATION, and you can now issue an invoice for the task.
- DONE - useful for the client to see everything that's finished.



FREELANCING

- PERSONAL BACKLOG - you can store errands and personal appointments here so they don't get mixed with work.
- WORK BACKLOG - everything that needs to be done starts here.
- THIS WEEK - you make a selection of what you'll work on during the week from your backlog, so you have a rough plan what you can accomplish and when.
- TODAY - everything you need to do on that day, once you wake up (you can move tasks to this column the day before).
- DOING - task that are in progress right now (good for collaboration as others can see what you're working on).
- ON HOLD - things that are not completed because you're waiting for the input or a response, like a client sending you some file or paying an invoice.
- DONE - you can also just mark the task as completed so it doesn't create clutter.



SOFTWARE DEVELOPMENT

- **BACKLOG** - all the user stories, bugs, requests, and improvements start here. The product manager talks to the client, collects their requests, and adds priorities and labels. Later, developers can assign themselves to a task from the top of the priority list.
- **NEXT** - this is a selection of tasks for the next sprint.
- **SPRINT** - this column has a start and end date, and all the tasks should be completed by the sprint's end date. If not, they're given top priority in the next one.
- **IN DEVELOPMENT** - lists all the tasks currently in progress, as well as tasks that got returned from QA.
- **QA** - work that needs to be tested before it can be released. If it needs more work, it goes back to IN DEVELOPMENT.
- **CODE REVIEW** - once a task passes testing, it's good practice to have someone else go through the code.
- **DOCUMENTATION** - all the work needs to be documented and communicated with the client before it can be released.
- **RELEASE** - finished tasks end up here and can be marked as completed after the sprint is over.



PRIORITY

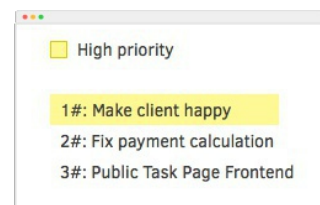
Deciding what to do next can be make or break a commercial software product. Your competition can get ahead of you and provide users exactly what they need and corner the market. No wonder "move fast and break things" is the dominant mantra in IT.

But this also applies to consulting agencies that develop custom products for clients. The job may be less precarious but they still need to do right things to keep clients happy and paying.

As you don't have unlimited time and people for your project, you need to carefully choose what to do next. Prioritizing is the project manager's main responsibility.

Show-stopping bugs have the highest priority, naturally. Next in line are tasks with high business value that generate revenue and save time for your client. When you communicate with clients, there are some things you keep hearing over and over - those things should take the top slot in the backlog.

Use the priority checkbox in Active Collab to signal priority. This will make the task visually stand out every time you open your To-Do list - things that scream visually tend to be dealt with sooner rather than later.



You can also use labels, like complexity level and business value, to help you determine how high on the priority list a task should be.

You can even make a special list for high priority tasks that developers will always check and work on first, before anything else in the Backlog. That way you ensure critical tasks are always dealt with.



When deciding on what to do next, keep in mind the jar metaphor. You can fill a jar with a few big stones, but there's still room. So you put in smaller stones too. Next, put even tinier stones. Same with tasks – choose a few big tasks and a bunch of smaller to maximize the efficiency.

You also need to pay attention to your team's morale and how your clients perceive your work.

If your team keeps working day and night on a big exciting feature that's never released, they'll feel discouraged. Imagine making a cake without tasting the delicious chocolate frosting. Easy wins are important, both for your team and your clients. Do the tasks that require little time but bring considerable benefits first, and leave the “whales” for later.

It's important to always keep in touch with your clients, even if you're not finished or you have something that needs reviewing. Otherwise, they'll feel like they're throwing money down a black hole. They can get restless, call a board meeting, make a contingency plan, devote resources to deal with you, hurry the project, and end up with a sub-par work that reflects bad on both parties.

BOTTLENECKS

Kanban is a great tool for identifying bottlenecks. All it takes is a glance to see where the holdup is.

You can control bottlenecks by limiting the number of Work in Progress (WiP). You can make a rule like: There can be no more than 7 tasks in Development or QA; if there are too many tasks in QA, a developer should first tackle the QA tasks before picking up a task from the BACKLOG.



This limit isn't an arbitrary choice. You'll need experience and data to make informed decisions. First try working without the limitation and see how it goes. You might find you can't manage more than 5 tasks or that 20 is easy. It depends, and there's no perfect recipe - just a lot of variables.

Your team should agree which person will help with the bottlenecks when low on work, so each person's knowledge and skills are used to the full potential. There's no need for a senior to do a simple edge test when a junior can do the same, at a fraction of the cost.

Keep in mind that the REVIEW column should always be cleared as soon as possible. You shouldn't let tasks sit there for too long - they may seem like they're done, but often they go back to DEVELOPMENT and clog your board - just because you didn't review on time and you pulled more work from BACKLOG than you could handle.

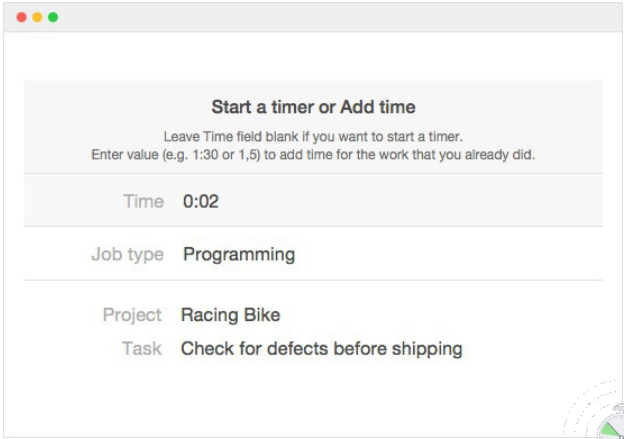
TRACKING TIME

Cycle time is an important concept in Kanban. It's the amount of time it takes for a task to go from one side of the board (Backlog) to the right side (Release). If you know your cycle time, you can make better estimates when delivering of future work.

Some tasks can be done in a few minutes, while some require months. We mentioned you can label tasks with rough estimates, but you can also use the “Estimates” field in ActiveCollab to predict completion time more precisely..

Estimates are useful for letting others know about how much time a task needs. You can also specify what kind of work it involves and set an hourly rate for that job type. After that, you can start working on the task and track how much time you spent working. You can later use this data for statistical analysis or make invoices and bill clients.

When you create a task, click Set... under Time Tracking, enter time estimate, and choose the type of work. When someone starts working, they can use ActiveCollab Timer app (that works like a stopwatch) to track time, and this data gets synced with ActiveCollab. You can also add time to tasks manually. When you have time records associated with tasks, you can create invoices based on your hourly rates, or run a report which compares estimated vs. actual time spent on tasks.



The screenshot shows a window titled "Start a timer or Add time". Below the title, there is a subtitle "Leave Time field blank if you want to start a timer." and a note "Enter value (e.g. 1:30 or 1,5) to add time for the work that you already did." The form contains several fields: "Time" with the value "0:02", "Job type" with the value "Programming", "Project" with the value "Racing Bike", and "Task" with the value "Check for defects before shipping". In the bottom right corner of the window, there is a circular timer icon with a green needle and a play button.

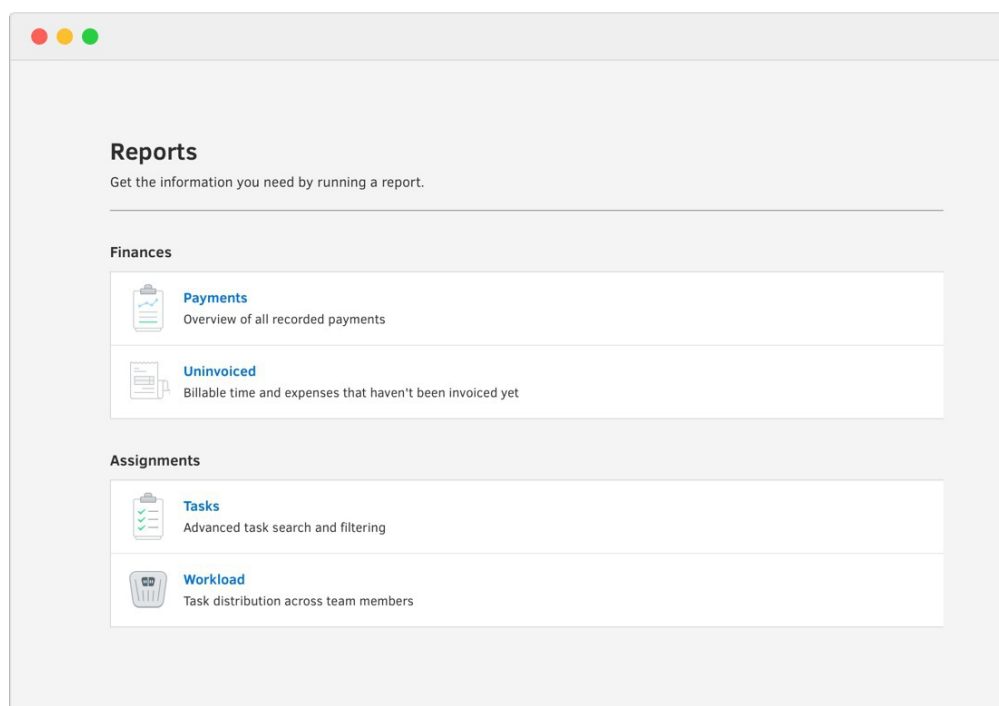
REPORTS

There's a special report in ActiveCollab that shows a comparison between estimated time vs. actual time spent on tasks. This report can help you perfect your estimation skill.

You can also run a report on all time records and filter them by project, job type, and assignee.

If you keep tracking time on projects and enforce others to do it too, you can have invaluable insights about your organization in as little as a few weeks.

Then you can use this information to optimize your workflow, become more efficient, and generate more revenue as the result – as long as you keep the discipline and know how to act on the information. We provide the tool and you the action.



BUG TRACKING

Bugs are persistent annoying things. Sometimes they happen, sometimes they don't. Sometimes they need more work than a normal task because of their ninja nature. So when a bug gets reported, you need a special system to deal with them.

The more you wait to fix a bug, the more time it takes because you forgot how the code works. You can take the “zero defect” approach and decide to fix every known bug before working on new features, but that's difficult in practice because new features can take precedence and are needed to appease the client.

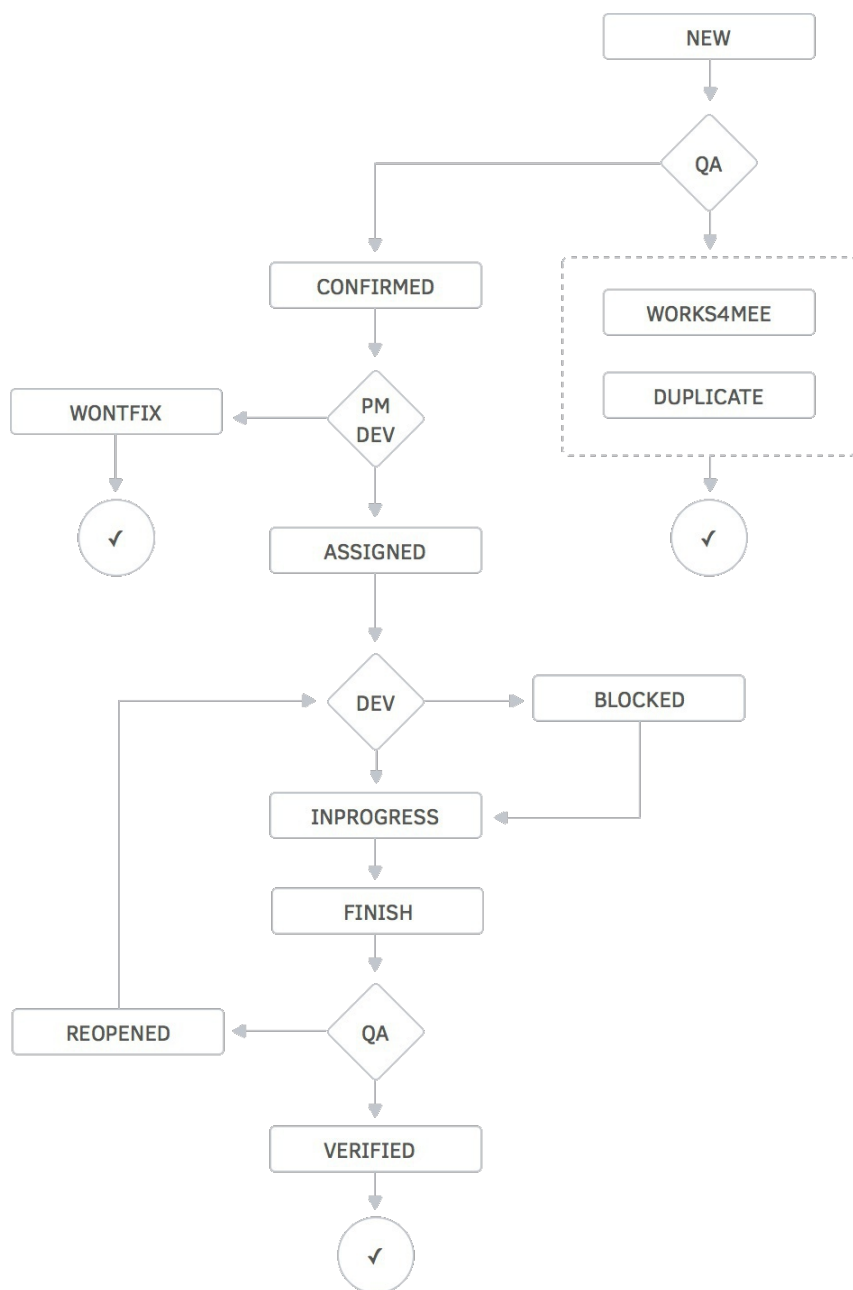
Our own bug tracking workflow relies on labels and clear communication - even the most sophisticated, specialized bug tracking tool will fail without the latter.

ASSIGNED BLOCKED WOKS4ME NEW CONFIRMED DUPLICATE FIXED IN PROGRESS REOPENED VERIFIED WONT FIX BLOCKER

When a bug gets reported, it starts with a label **NEW**. Our QA uses a filter to see all the tasks marked as **NEW**. They go through them and try to reproduce the problem. If reproduced, the bug changes the label to **CONFIRMED**, and developers decide when it will be fixed.

For example, if a bug was reported for Active Collab 5.0.13 today and got confirmed, the QA would set it for the next release batch “Active Collab 5.0.x Maintenance” (thus making it clear that it needs to be fixed in the 5.0.x bug fix release), and set its label to **CONFIRMED**.

Then the developers see a list of confirmed bugs and pick the ones they will work on by assigning themselves to the task. When they start working, they set task's label to **IN PROGRESS**.



Once they fix the problem, they mark the task as **FIXED**. Our QA sees a **FIXED** task and creates a new test build to verify that bug has actually been fixed.

If it's fixed, the task label is set to **VERIFIED** and the task is completed. In cases when a bug is not fully covered and needs more work, the QA marks it as **REOPENED** and notifies the developer about the problem.

The **REOPENED** task gets **FIXED** again, and the process continues until it gets **VERIFIED**.

That's the basic process. There's also a couple of extra labels to cover some edge cases:

WORKS4ME — bug can't be reproduced.

BLOCKED — bug that can't be fixed because it's blocked by another task.

BLOCKER — bug that's blocking other tasks.

This whole process is based on changing labels, but you can also do it using Kanban, where instead of labels you use columns and move bugs across task lists.



PEOPLE ON THE PROJECT

The most important aspect of any project are people. If no one understands what you try to achieve with a methodology, expect failure.

But if you know what motivates your developers, clients, or a managers - implementing Kanban gets much easier.

Clients don't care how you work but they'll care about how they receive updates. Developers want to know what the procedure is so they can do their job. And managers, they need to tie everything together and actually implement the workflow.

If you're changing how people work, expect resistance. But if they find the new workflow easy, they'll actually help your cause and promote it. In the end, your goal is to have a happy team.

Happy team == Good work.



CLIENT

Everything you do is for the client. If they aren't happy, no one is.

They don't really care what methodology you use: all they care about is getting new functionality without much hassle. They just want to finish their job in as little clicks as possible, under the budget, with release just before lunch.

When you take requirements, you need to listen not just what they want, but also what they need. They may say they need more user roles, but what they really need is a better team and more trust. This isn't a software issue and you can't solve it with code. Don't try to fix the client's work habits, just give them a good software that gets the job done.

It's also useful to allow clients to see your project, how far you've come, hear all the technical jargon (so it justifies your work and pay), and hear what they think about your progress.

In ActiveCollab, you can invite clients to projects and let them see tasks, comment on them, view notes, start discussions, and upload files. For sensitive information, you can always check the "hide from client" checkbox.



DEVELOPER

What developers want to do is to code. Methodologies are just a necessary evil. No one gets excited about a Kanban board or a daily standup.

Before implementing any kind of methodology, your team first needs to know how to work together and adapt themselves to fit the problem. If they can do that, any methodology will work. Even a perfect process won't save a team that doesn't know how to collaborate.

Kanban lets your team members motivate themselves as they decide what to work on. They take ownership of the task and their pride is measured in the number of completed tasks.

They can't wait to move their cards. There's a special feeling when you move a card to finished - that feeling gets you through the day, makes you love your job, and want to work even more. If that's not the case, you have an HR problem and not a methodology problem.

Because Kanban lets you keep shipping new releases quickly, your team gets to see how their work is being used and feels good about it.

This works as good or even better than any material reward. It's been scientifically proven that money improves work motivation up to a certain point - and that point isn't even that high.



MANAGER

A big aspect of Kanban is deciding what to do next. If you keep coming back to the same discussion over and over again, you're wasting time and energy. So it's better to make the wrong decision you can revert later (based on feedback) than to stall and never decide.

We might think every decision is important, but in practice, most of them aren't worth more than 10 minutes.

So a manager needs to know what are the most important tasks and make decisions. It might be a thankless role sometimes, but it's necessary. The manager needs to:

- communicate with the client
- put all the tasks in the backlog
- prioritise
- work on removing barriers

Good managers don't manage – they work on enabling people to do their job without distraction.

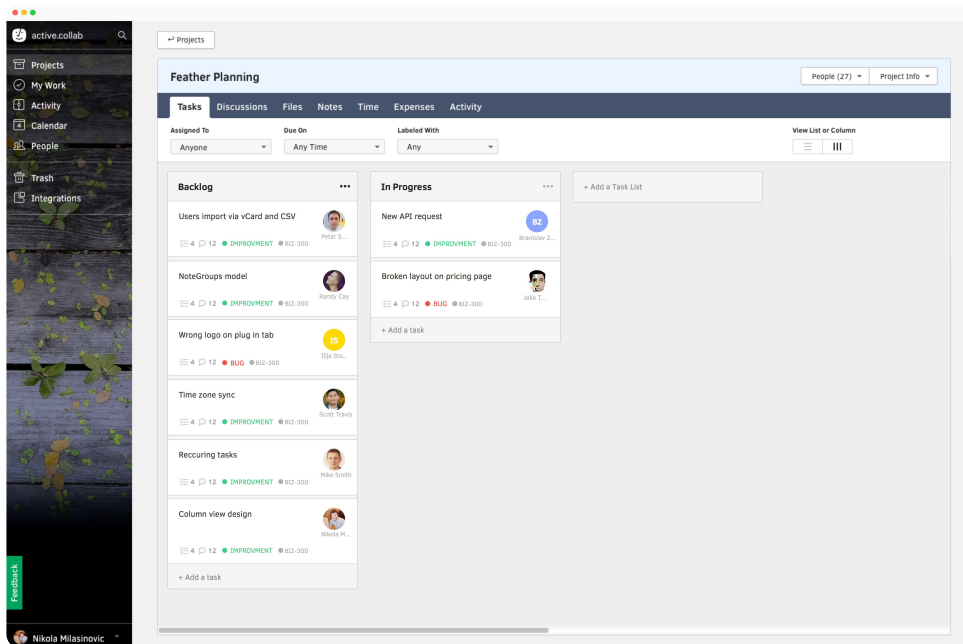
Developers can't lose time on dealing with negotiations, remembering to go to meetings, thanking for the meeting, sending daily reports, and responding to every phone call.

Meetings are a big part of manager's day, but team meetings shouldn't be. It's best to discuss work only with the people who are involved in a particular issue.

If you do need a meeting, use Kanban to quickly see where you are so you can address the important points, make an agenda, and keep it streamlined. Also, always keep an eye on the Work in Progress limit.

We started using Kanban and everyone on the team found it a joy to work with. It's easy to use and works great for small teams.

Be sure to check out Column view in ActiveCollab. You can create an account for free and try it out for 14 days.



Try it free for 14 days!

