A brief history of agile project management
Stemming from Toyota's lean manufacturing concept of the 1940s, software development teams have embraced agile methodologies to reduce waste and increase transparency, while quickly addressing their customers' ever-changing needs. A stark change from waterfall project management that focuses on "big bang" launches, agile helps software teams collaborate better and innovate faster than ever before.

Traditional agile project management can be categorized into two frameworks: scrum and kanban. While scrum is focused on fixed-length project iterations, kanban is focused on continuous releases. Upon completion, the team immediately moves on to the next.

Agile project management framework 1: Scrum
Scrum is a framework for agile project management that uses fixed-length iterations of work, called sprints. There are four ceremonies that bring structure to each sprint.

It all starts with the backlog, or body of work that needs to be done. In scrum, there are two backlogs: one is the product backlog (owned by the product owner) which is a prioritized list of features, and the other is the sprint backlog which is filled by taking issues from the top of the product backlog until the capacity for the next sprint is reached. Scrum teams have unique roles specific to their stake in the process. Typically there's a scrum master, or champion of the scrum method for the team; the product owner, who's the voice of the product; and the scrum team, who are often cross-functional team members in charge of getting s@#$ done.

The four ceremonies of scrum
Sprint Planning

Sprint Demo

Daily Standup

Retrospective

A team planning meeting that determines what to complete in the coming sprint.

A sharing meeting where the team shows what they've shipped in that sprint.

Also known as a stand-up, a 15-minute mini-meeting for the software team to sync.

A review of what did and didn't go well with actions to make the next sprint better.

Jira scrum board

The scrum board
A scrum board is used to visualize all the work in a given sprint. During the sprint planning meeting, the team moves items from the product backlog into the sprint backlog. Scrum boards can have multiple steps visible in the workflow, like To Do, In Progress, and Done. Scrum boards are the key component for increasing

transparency in agile project management. Get started using a scrum board with our free scrum template.

Agile project management framework 2: Kanban
Kanban is a framework for agile project management that matches the work to the team's capacity. It's focused on getting things done as fast as possible, giving teams the ability to react to change even faster than scrum.

Unlike scrum, kanban has no backlogs (usually). Instead, work sits in the To Do column. This enables kanban teams to focus on continuous releases, which can be done at any time. All work is visible, scoped, and ready to execute on so that when something is completed, the team immediately moves on to the next. The amount of work is matched to the team's capacity through WIP limits, which is a predefined limit of work that can be in a single column at one time (except the To Do column). The kanban framework includes the following four components:

The four components of kanban
List of work
(or stories)

Columns or lanes

Work in Progress Limits (WIP)

Continuous Releases

List of work, or stories, are defined as issues or tasks that need to get done.

Used on a kanban board to distinguish tasks from different workstreams, users, projects, etc.

A rule to limit the amount of work to be done based on the team's capacity.

The team works on the amount of stories within the WIP limit and can release at anytime.

Kanban board example | Atlassian agile coach
The kanban board
A kanban board is used to visualize all the work that's being done. It's also used for planning resources allowing project managers to see the work and develop timelines accordingly. A kanban board is structured into columns and lanes that stories pass through on their way to completion. Stories sit in the To Do column until the WIP limit allows for the next task to be worked on. The list of work should be split into relatively small issues and organized by priority. As you can see in this example, lanes can help keep the higher priority items separated from "everything else." Get started using a kanban board with our free Jira kanban template.

Responsibilities of agile project managers
Whatever agile framework you choose to support your software development, you'll

need a way to see your team's progress so you can plan for future work or sprints. Agile project estimating helps both scrum and kanban teams understand their capacity. Agile reports show the team's progress over time. Gantt charts and backlog grooming help project managers keep the list of work current and ready for the team to tackle.

Agile project estimations | Atlassian agile coach
Agile project estimating
Project estimating is an extremely important aspect of both kanban and scrum project management. For kanban, many teams set their WIP limit for each state based on their previous experiences and team size. Scrum teams use project estimating to identify how much work can be done in a particular sprint. Many agile teams adopt unique estimating techniques like planning poker, ideal hours, or story points to determine a numeric value for the task at hand. This gives agile teams a point of reference to refer back to during sprint retrospectives, to see how their team performed. Jira can be customized to capture your teams' unique project estimations.

Agile reporting example | Atlassian agile coach
Agile reporting
Project estimations come into play at the beginning and end of each sprint. They help teams determine what they can get done at the beginning of the sprint, but also show how accurate those initial estimates were at the end. Agile reports, such as Burndown charts, show how many "story points" are completed during the sprint. Jira offers dozens of out-of-the-box reports with real-time, actionable insights into how your teams are performing. Having data to support your retrospectives is an invaluable way for agile teams to improve.

Agile backlog example | Atlassian agile coach
Backlog management and grooming
A product backlog is a prioritized list of work for the development team to do that comes from product roadmap and its requirements. The development team pulls work from the product backlog for each sprint.

Grooming and maintaining your backlog helps teams achieve their long-term goals by continually adding and removing items based on the team's long-term capacity and changing business objectives. Jira lets teams groom huge backlogs with multi-select ranking and order user stories and bugs by dragging and dropping issues. You can also filter with Jira's flexible search to find a particular user story or bug.

Agile example | effective stakeholder communication
Effective stakeholder communication
Agile project managers also have to report the right amount of context to different stakeholders and teams - including senior leadership - on the status of the projects they’re responsible for.

With Atlas, project managers can share curated weekly updates on the progress of work, where it's happening, and call out key blockers, changes, and updates.

While agile is relatively new, it has made a big splash in the work of project management. It started in software development, but has since been adopted by other industries that have seen the benefit of agile’s iterative approach.

Those that use an agile project management framework don’t like to consider it a methodology, though some argue it is. Agile is more of an approach, and could almost be defined as a philosophy. Today we’re going to sidestep the philosophical, though, and instead focus on agile planning in project management, and specifically, creating an agile project plan.

What Is Agile Project Management?

The agile methodology is an iterative, adaptive approach to managing a project that has an emphasis on rapid change and flexibility. The reason for this flexibility is to deliver value to the customer faster. A team practicing agile works incrementally, continuously evaluates the requirements and results, and responds quickly to any changes that come up.

Agile also focuses on collaboration and keeping lines of communication open. There must be trust among the agile team, and an embrace of change. There is still a person who prioritizes tasks (usually known as the product owner), but the agile team themselves determine how to do the project planning and get the work done. Yes—agile has self-organizing teams that direct their own work!

This approach goes back to the development of the Agile Manifesto, which was written by seventeen software developers who found consensus around twelve principles. The length of interactions, or the size of teams, isn’t defined. It’s more about adhering to the stated values, which you can execute with scrum, hybrid methodology and more.

What Is Agile Planning?

However you choose to implement the agile principles, there is one thing all approaches have in common: an agile plan. Agile work takes place during short periods of time that are called agile sprints. A sprint is usually between one and three weeks, and the team uses this time to complete deliverables.

There are certain characteristics of agile planning that deserve mention to get a full idea of what the agile planning process entails:

First, there is the release. This is the product that an agile team works on.
The release plan is broken down into sprints, with each sprint dictating a specific set of tasks to be completed.
These tasks are called user stories.
You then build a plan from these user stories, which describe the needs of the end-user.
Then, the team works together to figure out the best way to address these user stories.
The sprint is the building block of agile planning. Agile sprints to be the same length in duration and are repeated, ending with a working feature that can be rolled out to the end-user. Due to the iterative nature of a sprint, a team will, over time, be able to better estimate how long user stories will take.

Software like ProjectManager makes executing sprints easy. Identify work that needs to be done in your backlog, prioritize it, then execute that work as a team on our kanban board. Balance resources with workload tools, and track progress with dashboards so you deliver your best work every time. Try ProjectManager today for free.

agile sprint plan on ProjectManager's kanban board
ProjectManager has both traditional and agile tools for project teams.Learn more
Why Planning Still Matters in the Agile Methodology
Agile planning gives an agile team a clear picture of the goals of their project. This supports the collaborative nature of agile, because everyone is on the same page. Agile plans are not obsolete and anachronistic, they define the work and help the team make decisions based on facts.

Project plans are an organization technique, and agile requires organization—albeit, much less than a project planned in waterfall. This might be why some are quick to dismiss planning when working in an agile project management framework. But that’s throwing out the baby with the bathwater. Agile planning is based on sprints and user stories, but that doesn’t mean you should ignore the big picture.

How to Make an Agile Project Plan
A team develops an agile project plan as the product owner describes the goals for the release, which are typically to improve the end-user experience and resolve problems. Once this has been defined, the next step is to get the team together and discuss desired features.

Related: Agile Sprint Planning Template

This leads to another discussion about the details for each of those features, and what might impact their delivery. The team also identifies any risk that might negatively impact the project, as well as task dependencies. The features that are riskiest and have the most value to the end-user are usually completed first.

Step-By-Step Guide to Creating an Agile Project Plan
Now you’re ready to create a plan:

Begin with a retrospective meeting. A retrospective meeting is where you discuss the previous sprint to learn from what went right and what went wrong.
Run a sprint planning meeting. A sprint planning meeting looks at the release and any updates that have occurred, such as changes to priority, new features, etc.
Create user stories: Detail the user stories as much as possible so that they are well-defined.
Create deliverables: Break the user story down into tasks that are usually not more than a day in duration.
Delegate responsibility: Assign tasks to team members and assign ownership to make sure they’re committed to executing them.
Create a workflow: Put the tasks on a board, either a card on a physical board or with project management software tools, such as kanban boards.
Track progress: Use the board to track the progress of the sprint as the tasks move from one stage of the production cycle to the next.
Use a burndown chart: Create a burndown chart to show the number of tasks or hours

left.
Related: Agile vs Waterfall and the Rise of Hybrid Projects

Agile Project Planning Terms

Here are some important agile concepts that you’ll need to know to create and execute your agile project plan:

Product Backlog: In agile project management, a product backlog is a list of deliverables that derive from the product roadmap and its requirements. Things like new product features, bug fixes or any changes are backlog items that should be documented here.
Product owner: The product owner is the member of the agile team who’s responsible for defining user stories and prioritizing the product backlog.
User stories: It’s a small task within an agile plan. They’re called user stories because they’re product features described from the end-user perspective.
Burndown chart: A burndown chart is used to show the amount of work that has been completed in an agile sprint and the number of tasks or hours left.
Burn rate: In agile project management, the burn rate is a metric used to measure the efficiency of an agile team. It measures the relationship between the completion of user stories and the time spent on them.
Team velocity: The velocity is the broader performance metric that measures the amount of work a team can get done during a sprint.
Story point estimation: This is a method used to measure agile teams’ performance. A story point is a unit that is used to calculate the effort needed to complete a user story. Story points measure three factors, complexity, risk and repetition.
Now that you know the basics of agile planning, you’ll need a project management tool like ProjectManager to help you manage your agile projects.

Project Planning in Agile

Scenario: Defining Project Scope with Stakeholders
Guideline: In Agile, the project scope is initially broad and evolves through sprints. Hold a Product Backlog Grooming session with stakeholders to outline high-priority user stories, keeping them flexible. Prioritize features that deliver maximum business value in early sprints to ensure incremental delivery aligns with stakeholder needs.

Scenario: Setting Sprint Goals and Timelines
Guideline: At the start of each sprint, conduct a Sprint Planning Meeting to set achievable goals for the sprint duration (typically 2-4 weeks). Use the team’s past velocity as a benchmark for workload, and focus on creating a sprint backlog with stories that can realistically be completed within the sprint. Avoid overloading the sprint backlog to maintain flexibility.

Scenario: Managing Team Capacity and Resources
Guideline: Regularly assess each team member's workload and availability. Use tools like Team Capacity Planning to visualize team member availability and avoid over-allocation. Adjust tasks based on skill sets and allocate additional resources as needed, especially when working on complex or high-priority tasks in each sprint.

Scenario: Defining Minimum Viable Product (MVP)
Guideline: Identify the essential features that form the MVP, prioritizing core functionalities. Regularly review with stakeholders to ensure the MVP addresses key requirements. Plan to iteratively improve the MVP in future sprints based on feedback from early releases, refining user stories as more is learned about user needs.

Scenario: Conducting Backlog Refinement
Guideline: Schedule regular backlog refinement sessions to update and prioritize the backlog. Work closely with the Product Owner to clarify user stories, break down large stories into smaller tasks, and adjust priorities. Keeping the backlog up-to-date ensures the team always has a clear list of prioritized tasks to tackle.

Scenario: Identifying Sprint Dependencies
Guideline: During sprint planning, identify any tasks dependent on other teams, technologies, or resources. Communicate these dependencies early to avoid delays. Use a Dependency Matrix or similar tool to track and monitor dependencies and adjust the sprint backlog if dependencies cannot be resolved on time.

Scenario: Managing Agile Risks
Guideline: Since Agile is iterative, regularly review potential risks at each sprint’s start, like fluctuating requirements or team availability. Maintain a Risk Register for ongoing monitoring, and conduct quick risk assessment meetings if issues arise during a sprint to devise contingency plans without disrupting the sprint goal.

Scenario: Estimating Effort for User Stories
Guideline: Use techniques like Planning Poker or T-shirt Sizing to estimate story effort collaboratively. This builds team consensus on workload and helps gauge the feasibility of user stories within a sprint. Review and adjust estimates based on actual time spent in previous sprints to improve accuracy.

Scenario: Conducting a Sprint Retrospective
Guideline: At the end of each sprint, hold a Retrospective Meeting to reflect on what went well, what didn’t, and where improvements can be made. Address any recurring issues and agree on actionable changes to improve workflow and communication. Implement these changes in the following sprint to foster continuous improvement.

Scenario: Creating Agile Roadmaps and Release Plans
Guideline: Develop an Agile roadmap that outlines high-level goals and rough timelines for future sprints. Use Release Planning sessions to prepare a sequence of releases based on feature completion. This ensures stakeholders have visibility into the project timeline without the rigidity of traditional roadmaps.

System Design in Agile

Scenario: Incremental System Architecture Design
Guideline: Start with a high-level architecture that covers essential components. Gradually refine architecture as more user stories are completed. Keep architecture

modular to adapt easily to new features or changes, focusing on flexibility and scalability to support iterative development.

Scenario: Designing for Continuous Integration (CI)
Guideline: Structure the system to facilitate automated builds, testing, and deployment in small increments. Design modular components that can be tested independently, and ensure compatibility with CI tools like Jenkins or GitLab CI to streamline the integration process for frequent code commits.

Scenario: Defining User Story Acceptance Criteria for Design
Guideline: Ensure each user story has clear Acceptance Criteria that inform the design. These criteria should specify functionality, performance, and usability requirements. Use these criteria to guide design decisions and validate that the delivered solution meets user expectations.

Scenario: Creating Flexible Database Models
Guideline: Develop an initial database schema to cover core data entities but remain flexible to changes as user stories evolve. Use migration scripts to update the schema iteratively without impacting existing data. Consider NoSQL databases if the project needs high adaptability or unstructured data handling.

Scenario: Designing APIs for Incremental Development
Guideline: Build APIs to support the key interactions needed in the early sprints, keeping them versioned for forward compatibility. Document the endpoints and expected data formats for easy integration with front-end components and allow gradual API extension as new requirements emerge.

Scenario: Decoupling Front-End and Back-End Development
Guideline: Use a microservices architecture or similar design to separate front-end and back-end development. This enables parallel development, where UI and API design can proceed independently, allowing each sprint to include both front-end and back-end functionality without heavy interdependence.

Scenario: Designing for Scalability and Load Management
Guideline: Since Agile projects evolve, design the system with scalability in mind, even for initial releases. Use load balancing and clustering if there are high traffic expectations, and choose database scaling techniques that align with the expected user growth.

Scenario: Documenting Design Decisions and Trade-offs
Guideline: Keep a Design Log where significant decisions, assumptions, and trade-offs are documented. This allows team members to understand past choices, making it easier to revisit and update the design as requirements change without contradicting initial decisions.

Scenario: Ensuring Design Supports Automated Testing
Guideline: Structure the system with testability in mind by creating small, independent modules that can be unit-tested. Use design patterns that facilitate mocking and stubbing for automated testing, ensuring each component is thoroughly tested as part of each sprint.

Scenario: Enabling Feature Toggles for Agile Releases
Guideline: Implement feature toggles to allow for the deployment of partially completed features without exposing them to end-users. This enables the team to deploy code frequently while giving flexibility in activating features only when they’re fully tested and approved.