**JIRA**

**🧾 What is JIRA?**

**JIRA** is a popular **project management and issue-tracking tool** developed by **Atlassian**. It is widely used by software development teams for **bug tracking, issue tracking, and project management**. Originally created for software developers, it is now used for a variety of project management scenarios including Agile and Scrum.

**🔧 Key Components of JIRA**

**1. Projects**

* A project is a collection of issues.
* Each project can represent a product, team, or work process.
* Projects have their own workflows, permissions, and configurations.

**2. Issues**

* An issue is the core entity in JIRA and can represent a task, bug, feature, etc.
* Types of issues:
  + **Task** – A generic piece of work.
  + **Bug** – A problem that needs fixing.
  + **Story** – A user requirement.
  + **Epic** – A large body of work that can be broken down into stories or tasks.
  + **Sub-task** – A smaller unit of work under a task.

**3. Workflows**

* Defines the lifecycle of an issue (e.g., To Do → In Progress → Done).
* Customizable to fit team processes.
* Includes **statuses**, **transitions**, **conditions**, **validators**, and **post-functions**.

**4. Boards**

* Visual representation of issues.
* Types:
  + **Scrum Board** – For teams using the Scrum framework.
  + **Kanban Board** – For continuous delivery workflows.
* Allows teams to manage and monitor progress in real time.

**5. Backlog**

* A list of issues (mainly stories and bugs) that are yet to be scheduled.
* Used in Scrum for sprint planning.

**6. Sprints**

* Time-boxed iterations for completing work.
* Sprints include a set of issues planned from the backlog.

**7. Reports & Dashboards**

* Visual tools to track progress, performance, and status.
* Examples:
  + Burndown Chart
  + Velocity Chart
  + Cumulative Flow Diagram

**📋 JIRA Task Section – Detailed Explanation**

In JIRA, a **Task** is an issue type used to track a unit of work. Here's everything that can be included under the **Task** issue:

**1. Summary**

* A short title or description of the task.

**2. Description**

* Detailed explanation of the task.
* May include:
  + What needs to be done
  + Why it is required
  + Any references or specifications
  + Acceptance criteria

**3. Assignee**

* The person responsible for completing the task.

**4. Reporter**

* The person who created the task.

**5. Status**

* Current stage of the task (e.g., To Do, In Progress, Done).

**6. Priority**

* Importance of the task (e.g., Low, Medium, High, Critical).

**7. Labels**

* Tags to categorize tasks for easier searching and filtering.

**8. Attachments**

* Files, images, or documents related to the task.

**9. Comments**

* Section for team discussions, updates, or notes related to the task.

**10. Linked Issues**

* Shows how the task is connected to other issues (e.g., blocks, relates to, is duplicated by).

**11. Sub-tasks**

* Smaller, manageable parts of the main task.

**12. Watchers**

* Users who want to receive updates about the task.

**13. Due Date**

* Deadline for the task completion.

**14. Time Tracking**

* Estimation and logging of time spent on the task.
  + **Original Estimate** – Expected time to complete the task.
  + **Time Spent** – Actual time spent.
  + **Remaining Estimate** – Time left to finish the task.

**15. Custom Fields**

* Organization-specific fields like department, feature group, client name, etc.

**📂 Subtopics to Understand Under JIRA**

| **Subtopic** | **Explanation** |
| --- | --- |
| **Agile in JIRA** | JIRA supports Agile methodologies like Scrum and Kanban with boards and workflows. |
| **Sprint Planning** | Selecting tasks from the backlog to work on during a sprint. |
| **Issue Types** | Tasks, Bugs, Stories, Epics, and Sub-tasks, each serving different purposes. |
| **Permissions** | Control who can view, create, edit, and delete issues. |
| **Automation** | Rules to automate repetitive actions (e.g., change status, assign users). |
| **Filters & JQL** | Use **JIRA Query Language** (JQL) to search and filter issues effectively. |
| **Releases / Versions** | Plan and track features or bug fixes to be delivered in a specific version. |
| **Integration** | Integrates with tools like Bitbucket, GitHub, Confluence, Jenkins, Slack, etc. |

**✅ Example: Sample JIRA Task**

| **Field** | **Sample Data** |
| --- | --- |
| **Summary** | Implement Login Page |
| **Description** | Design and code the login page with input validation and error handling. |
| **Assignee** | John Doe |
| **Priority** | High |
| **Status** | In Progress |
| **Time Estimate** | 4 hours |
| **Labels** | frontend, auth |
| **Sub-tasks** | Design UI, Validate Input, Connect API |
| **Due Date** | 2025-06-05 |

**🧩 1. Scrum Board**

**🔹 What It Is:**

A **Scrum Board** is used by teams that follow the **Scrum framework** (a type of Agile methodology). It works in **sprints** – fixed time-boxed iterations (usually 1-4 weeks) where the team commits to completing a set of tasks.

**🔧 Key Features:**

| **Feature** | **Explanation** |
| --- | --- |
| **Sprints** | Time-boxed cycles of work. You plan work in advance and finish within the sprint. |
| **Backlog** | All tasks, stories, bugs, etc., not yet scheduled in a sprint. |
| **Sprint Planning** | Team selects tasks from the backlog and assigns them to the sprint. |
| **Sprint Goal** | A clear objective that the sprint aims to accomplish. |
| **Burndown Chart** | Shows how much work remains in the sprint (ideal vs. actual progress). |
| **Definition of Done** | Tasks must meet specific criteria to be marked as complete. |

**✅ Best For:**

* Teams who deliver in fixed time periods.
* Product development teams.
* Teams that hold regular standups, reviews, retrospectives.

**🧠 Example:**

A mobile app team commits to complete:

* 5 bugs
* 3 new features
* 1 UI improvement  
  ...in the next **2-week sprint**.

**🧩 2. Kanban Board**

**🔹 What It Is:**

A **Kanban Board** is for teams focused on **continuous delivery** and **workflow management**. There are **no sprints**. Work is pulled into the board when the team is ready.

**🔧 Key Features:**

| **Feature** | **Explanation** |
| --- | --- |
| **No Sprints** | Tasks are handled as they come (continuous flow). |
| **WIP Limits** | Limits on how many tasks can be "In Progress" at a time. |
| **Columns** | Represent workflow stages (To Do → In Progress → Done). |
| **Cycle Time** | Measures time taken for a task from start to completion. |
| **Swimlanes/Filters** | Categorize work by priority, user, type, etc. |

**✅ Best For:**

* Operations or support teams.
* Teams maintaining a system or doing unplanned/ongoing work.
* Projects without time-boxed releases.

**🧠 Example:**

A customer support team receives tickets throughout the day and addresses them as they come, based on priority – **no sprint planning required**.

**🔁 Key Differences Between Scrum and Kanban**

| **Feature** | **Scrum Board** | **Kanban Board** |
| --- | --- | --- |
| **Approach** | Iterative (Sprints) | Continuous |
| **Planning** | Required (Sprint Planning) | Not mandatory |
| **Time-bound** | Yes (e.g., 2-week sprints) | No |
| **WIP Limits** | Not enforced by default | Core part of Kanban |
| **Metrics** | Burndown chart, velocity | Cycle time, cumulative flow |
| **Best For** | Feature development | Support, maintenance, quick fixes |

**💡 Which One Should You Use?**

| **If You Want...** | **Use** |
| --- | --- |
| Structured planning, clear deadlines, and sprint reviews | **Scrum** |
| Flexible workflow, less planning, continuous improvement | **Kanban** |

Would you like me to show **how to switch between Scrum and Kanban** in your project or **how to create a new board** for either method?