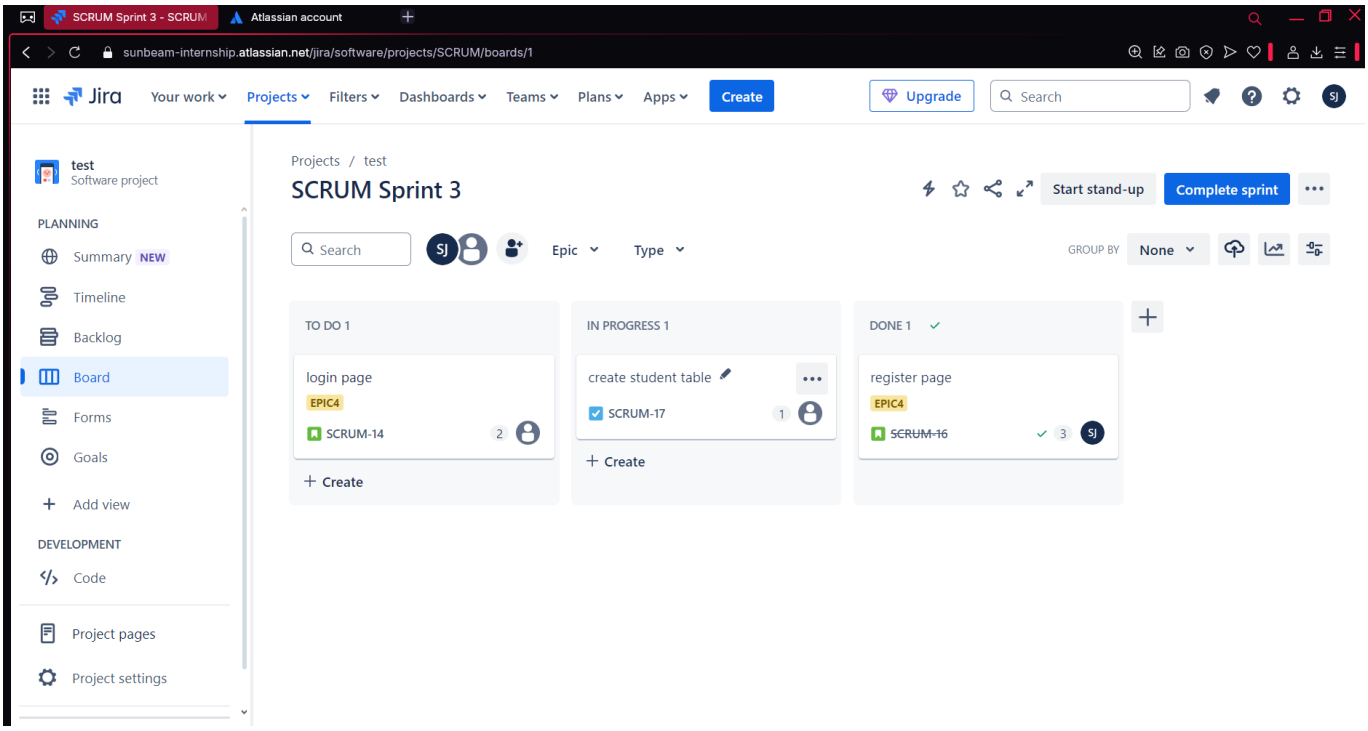
What is JIRA?

JIRA is a widely used **project management tool** developed by **Atlassian**. It is primarily used for **tracking and managing software development projects** but has expanded to support a variety of project types, including business and marketing workflows. JIRA helps teams plan, track, and manage their work through an intuitive interface, providing various features such as **task tracking**, **issue management**, and **agile project management**.

The term "JIRA" is derived from the Japanese word "Gojira" (ゴジラ), meaning "Godzilla", which symbolizes the strength and robustness of the tool in managing complex workflows and issues.



Key Features of JIRA

1. **Issue Tracking**:
    - JIRA's core feature is its **issue tracking system**, where teams can create, assign, and track issues (tasks, bugs, or user stories) from **start to finish**.
    - An issue in JIRA can represent a variety of work items such as bugs, tasks, improvements, or features.
    - Every issue can be assigned to a specific person, prioritized, and tracked throughout its lifecycle.
2. **Agile Project Management**:
    - JIRA is highly popular among teams using **Agile methodologies** like **Scrum** and **Kanban**. It provides features that support both **Scrum boards** and **Kanban boards**:
        - **Scrum Boards**: For teams that use sprints, Scrum boards in JIRA help manage tasks within sprints, showing progress and remaining work.
        - **Kanban Boards**: For teams focusing on continuous delivery, Kanban boards are used to visualize workflows and limit work in progress (WIP).

- **Backlog management**: You can manage a backlog of tasks or user stories, prioritize them, and assign them to team members for future sprints.

3. **Customization**:
   - JIRA allows teams to customize workflows, issue types, fields, permissions, and notifications based on the needs of the project.
   - Teams can create **custom workflows** to match their specific process, define the states an issue can go through (e.g., To Do, In Progress, Done), and customize the transition between them.

4. **Reporting and Analytics**:
   - JIRA offers various built-in **reporting tools** for tracking progress, productivity, and performance.
   - Teams can generate reports like **burn-down charts** (for Scrum), **velocity charts**, **cumulative flow diagrams** (for Kanban), and more.
   - Reports help managers and team members visualize project status and identify bottlenecks or delays.

5. **Integration**:
   - JIRA integrates with various other tools, such as **Bitbucket**, **Confluence**, **Slack**, and **GitHub**, to streamline workflows. For example:
     - **Bitbucket Integration**: If you use Bitbucket for source control, JIRA allows you to link code commits and pull requests to JIRA issues, providing traceability between the code and the task it was meant to address.
     - **Confluence Integration**: If you're using Confluence for documentation, you can link Confluence pages to JIRA issues to keep everything organized.

6. **Permissions and Security**:
   - JIRA offers **granular permission settings**, allowing you to define who can view, create, or edit issues at various levels (e.g., project-level, issue-level).
   - Role-based access control helps teams maintain security and confidentiality in their project management practices.

7. **Automation**:
   - JIRA offers robust **automation features**, such as setting up rules to automate repetitive tasks like:
     - Automatically assigning issues to a specific team member.
     - Changing the status of an issue when certain conditions are met (e.g., transitioning an issue from "In Progress" to "Done" when a commit is made).

8. **Mobile Access**:
   - JIRA provides mobile applications for both iOS and Android, making it easy for team members to access and update their tasks while on the go.

Key JIRA Terminology

1. **Issues**:
    - In JIRA, any work item (bug, feature, task, story, etc.) is represented as an "issue." Issues are the basic units of work in JIRA.
2. **Projects**:
    - A project in JIRA is a collection of issues. You can have multiple projects in a single JIRA instance, each with its own set of issues, workflows, and configurations.
3. **Boards**:
    - Boards represent the visual representation of your project's workflow. **Scrum boards** are used for sprint-based workflows, and **Kanban boards** are for continuous workflows.
4. **Workflows**:
    - A workflow is a set of statuses and transitions that an issue goes through from creation to completion. You can customize workflows to reflect your team's process.
5. **Epics, Stories, and Tasks**:
    - **Epic**: A large body of work that can be broken down into smaller tasks or user stories.
    - **User Story**: A feature or functionality written from the perspective of the end-user. A user story usually represents a small, valuable piece of functionality.
    - **Task**: A smaller, technical work item that doesn't necessarily need to be written as a user story. Tasks might be backend work or technical chores.
6. **Sprints**:
    - A **Sprint** is a fixed time period (usually 2-4 weeks) in which a Scrum team works to complete a specific set of user stories, features, or tasks.
7. **Backlog**:
    - The **Backlog** is a prioritized list of tasks, bugs, or user stories that need to be completed. It's used in both Scrum (for sprint planning) and Kanban (as the to-do list).
8. **Versions**:
    - Versions represent a release or milestone of the project. You can associate issues with a specific version to track progress towards that release.

Example Use Case in JIRA

Let's say you're working on a **student score management system** using Scrum methodology, and your team is using JIRA to manage the project:

1. **Project Setup**:
    - Create a new project in JIRA called "Student Score Management System."
2. **Backlog**:

- Add user stories to the backlog, such as:
  - "As a student, I want to view my exam scores."
  - "As a staff member, I want to upload student scores."
  - "As an admin, I want to manage student and staff users."

3. **Sprint Planning**:
   - During a sprint planning meeting, your team selects user stories from the backlog and assigns them to the current sprint.
   - You also break down larger user stories into smaller tasks.
4. **Execution**:
   - Team members work on the tasks in the sprint, and each task is tracked on a Scrum board.
   - Developers can log their progress by updating the status of their tasks.
5. **Testing and Review**:
   - After the sprint is completed, the team conducts testing to ensure all features work as expected.
   - The completed tasks are marked as done, and any bugs found during testing are logged as issues.
6. **Retrospective**:
   - After the sprint ends, your team holds a retrospective to reflect on what went well, what could be improved, and what should be done differently in the next sprint.
