**Definition of Ready (DoR)**

The **Definition of Ready** ensures that a user story or task is fully prepared to be worked on by the team. A user story or task is considered ready when the following criteria are met:

**1. Clarity**

* The user story or task is clearly written with no ambiguity.
* Acceptance criteria are explicitly defined and well-documented.
* Any dependencies or blockers are identified and either resolved or documented.

**2. Context**

* The story/task aligns with the overall project goals and priorities.
* It has been reviewed and approved by the product owner or stakeholders.

**3. Estimable**

* The scope of the work is small enough to be estimated (fits within one sprint or iteration).
* The team agrees on the effort required (story points, time, or complexity).

**4. Dependencies**

* All necessary dependencies (e.g., APIs, data models, libraries, designs) are either ready or mocked for development.
* External systems, tools, or teams are prepared to support the story/task if required.

**5. Design & Documentation**

* Wireframes, mockups, or workflows (if applicable) are provided and reviewed.
* Relevant documentation (e.g., APIs, database schemas, user personas) is available and accessible.

**6. Testability**

* Acceptance criteria are testable and measurable.
* A clear testing plan (unit tests, integration tests, end-to-end tests) is outlined.
* Test data or environments required for validation are available.

**7. Team Agreement**

* The development team has reviewed and agreed that the story/task is ready to be started.
* The task is prioritized in the sprint backlog.

**Definition of Done (DoD)**

The **Definition of Done** ensures that a user story or task is fully completed and meets the required quality standards. A user story or task is considered done when the following criteria are satisfied:

**1. Development**

* All code is written, reviewed, and merged into the appropriate branch (following coding standards and guidelines).
* Unit tests are implemented and passed with sufficient coverage (agreed-upon % threshold).
* Integration points (e.g., APIs, databases) are implemented and validated.

**2. Testing**

* Unit tests, integration tests, and end-to-end tests pass successfully.
* The feature is tested on staging or a test environment and verified against acceptance criteria.
* Regression testing is conducted, ensuring no new issues are introduced.

**3. Documentation**

* Code is adequately documented (inline comments and external documentation as required).
* User-facing documentation (e.g., user manuals, API documentation) is updated or created.
* Deployment steps or release notes are documented if required.

**4. Quality Assurance**

* Peer code reviews are conducted, and any feedback is resolved.
* The story/task passes all quality gates (e.g., SonarQube, linting, performance benchmarks).

**5. Performance**

* The feature is optimized for performance based on project standards.
* Relevant performance benchmarks (e.g., response time, memory usage) are met.

**6. Security**

* Security vulnerabilities are addressed (e.g., input validation, authentication checks).
* The feature passes security checks and conforms to relevant compliance standards.

**7. Integration**

* The feature is successfully integrated into the application without breaking existing functionality.
* Any changes to shared components or modules are communicated and coordinated with other teams.

**8. Approval**

* The product owner or relevant stakeholders approve the story/task after review.

**9. Deployment**

* The feature is deployed to production or ready for deployment (based on the sprint goal).
* Deployment scripts or configurations (if applicable) are validated and versioned.

**10. Post-Deployment**

* Post-deployment validation (e.g., smoke testing) is completed.
* Monitoring or logging for the feature is enabled (if applicable).
* Feedback or bugs reported during deployment are addressed.