**SPARC Workflow Collaboration**

**1. Specification (Riley)**

* **Riley gathers requirements from stakeholders and documents objectives, constraints, and success criteria.**
* **Shares the specification with Morgan and Tyler for review and clarification.**
* **Receives feedback on technical feasibility and user experience, revises as needed.**

**2. Pseudocode/Design (Morgan)**

* **Morgan translates specifications into high-level pseudocode, user flows, and feature outlines.**
* **Consults Riley to ensure business requirements are accurately reflected.**
* **Engages Tyler for technical validation and Alex for coding standards.**
* **Shares drafts with Casey for mapping to user stories.**

**3. Architecture (Tyler)**

* **Tyler designs system architecture, selects technology stack, and creates diagrams.**
* **Reviews Morgan’s pseudocode for alignment with technical constraints.**
* **Coordinates with Alex to ensure code guidelines fit the architecture.**
* **Shares architecture documents with Jordan for deployment planning.**

**4. Refinement/User Stories (Casey)**

* **Casey converts design and architecture into actionable user stories with acceptance criteria.**
* **Collaborates with Riley to ensure traceability to business requirements.**
* **Works with Quinn to validate testability and coverage.**
* **Iterates stories based on feedback from development and QA.**

**5. Completion/Testing (Quinn)**

* **Quinn develops test plans and cases based on user stories and architecture.**
* **Consults Casey for acceptance criteria and Tyler for environment setup.**
* **Shares test results and issues with Alex and Jordan for remediation and deployment adjustments.**

**6. Coding Guidelines (Alex)**

* **Alex reviews code for adherence to style guides and error handling.**
* **Provides feedback to Morgan and Tyler on documentation and technical decisions.**
* **Works with Quinn to ensure code is testable and maintainable.**

**7. Deployment/Runbook (Jordan)**

* **Jordan prepares deployment scripts and runbook documentation.**
* **Coordinates with Tyler for infrastructure requirements and Alex for code integration.**
* **Works with Quinn to ensure troubleshooting steps cover common test failures.**

**Collaboration Mechanisms**

* **Shared Documentation: All personas contribute to and review documents in platforms like Confluence, Jira, or GitHub.**
* **Feedback Loops: Each phase includes review and feedback from upstream and downstream personas.**
* **Regular Syncs: Scheduled meetings or async updates ensure alignment and resolve blockers.**
* **Version Control: All deliverables are versioned, with clear ownership and change logs.**

**Example Interaction**

* **Riley drafts the specification and asks Morgan, “Does this cover all user scenarios?”**
* **Morgan responds, “I see a gap in error handling for the CLI. Tyler, can we support this in the architecture?”**
* **Tyler reviews and says, “Yes, but we’ll need to add a logging module. Alex, can you update the coding guidelines?”**
* **Alex updates the guidelines and notifies Quinn, “New error handling pattern added—please update your test cases.”**
* **Quinn revises test cases and informs Jordan, “Deployment scripts need to handle new error logs.”**
* **Jordan updates the runbook and confirms with the team.**