Scenario 4

**1. Approval Table**:

* + List of stakeholders involved in approving new features (e.g., product managers, team leaders, quality assurance).
  + Roles and responsibilities of each stakeholder in the approval process.
  + Criteria for approving new features (e.g., meeting functional requirements, passing code review).
  + Approval status tracking mechanism (e.g., checklist, online tool).

**2. Purpose**:

* + Clear statement defining the purpose of the feature development process document.
  + Explanation of why the process is necessary (e.g., to ensure consistency, efficiency, and quality in feature development).
  + Identification of the target audience (e.g., development team, project managers, stakeholders).
  + Overview of the expected outcomes (e.g., improved collaboration, faster delivery of features, higher code quality).

**3. Scope and Objectives**:

* + Definition of the scope of the feature development process (e.g., covers all stages from requirements gathering to merging changes).
  + Aims and goals of the process (e.g., ensuring clear communication, minimizing errors, improving code maintainability).
  + Description of what falls within the boundaries of the process (e.g., feature development tasks, code review procedures).
  + Clearly mentioning what is not included (e.g., infrastructure changes, non-feature-related tasks).

**4. Accountability Matrix**:

* + Roles and responsibilities of team members involved in feature development (e.g., developers, testers, reviewers).
  + Matrix mapping responsibilities to team members (e.g., who handles creating feature branches, writing unit tests, conducting code reviews).
  + Identification of key decision-makers and approvers for each stage of the process.
  + Contact information of each responsible person for easy communication.

**5. Detailed Steps**:

* + Step-by-step instructions for each stage of feature development, including:
    - Requirements gathering (e.g., meeting with stakeholders, documenting user stories).
    - Creating feature branches (e.g., using version control systems like Git).
    - Writing unit tests (e.g., using testing frameworks like Jest).
    - Conducting code reviews (e.g., setting up pull request reviews).
    - Merging changes into the main codebase (e.g., using merge strategies).
  + Explanation of tools and methodologies to be used at each stage (e.g., continuous integration).
  + Tips or best practices to follow for efficient and effective feature development.

**6. Revision History**:

* + Log of changes made to the feature development process document over time.
  + Date of each revision.
  + Description of changes made in each revision (e.g., updates to reflect new tools, changes in team roles).
  + Version control information (e.g., document version number, author of changes).

These points should be elaborated upon and organized logically within the

process document to ensure clarity and ease of understanding for all

stakeholders involved in feature development.