**GitHub Guidelines**

# Commit

* **Subject Line (Max 50 characters):** Start with a concise and informative subject line that summarizes the main change introduced in the commit.
* **Imperative Mood:** Use the imperative mood (e.g., "Fix," "Add," "Update") to describe the action taken in the commit.
* **Focus on Functionality:** Briefly describe the functionality added, fixed, or modified.
* **Capitalization and Punctuation:** Capitalize the first word of the subject line and avoid ending it with a period.
* **Body (Optional):** While the subject line provides a high-level overview, the body offers an optional space for more details. Here are some pointers for the body section:
  + **Limit Line Length:** Keep each line around 72 characters for better readability.
  + **Explain the Why:** Briefly explain the reason behind the changes made. This could involve bug reports, feature requests, or improvements made.
  + **Additional Context:** Provide any relevant context, such as links to related issues or specific code references.

**Examples of Well-Structured Commit Messages:**

* Subject: Fix login form validation error (closes #123)
* Body: Modified validation logic to handle empty username fields.
* Subject: Add support for Markdown formatting
* Body: Introduced new functionality for applying Markdown formatting within text areas.

**General Tips:**

* **Focus on Clarity:** Aim for clarity and conciseness. Your commit message should be easily understood by you and your fellow developers, even months from now.
* **Commit Often:** Don't wait until you've completed a massive feature to commit. Regular smaller commits with clear messages make it easier to track changes and revert if necessary.

# Branching Strategy and Merging

* **Short-lived Branches:** Encourage developers to create feature branches with a short lifespan. This minimizes the chance of conflicts arising due to long-running development cycles on a single branch. Aim for branches focused on a single feature or bug fix to be merged within a designated timeframe (e.g., a few days).

## Standardized Branch Naming Conventions:

* **Beyond Feature and Bugfix Prefixes:** While "feature/" and "bugfix/" are common prefixes, consider expanding your naming convention for specific development activities. You could utilize prefixes like "hotfix/" for urgent fixes, "refactor/" for code structure improvements, or "chore/" for non-code related changes (e.g., updating documentation).
* **Descriptive Branch Suffixes:** Complement prefixes with descriptive suffixes that provide further context. For example, a branch named "feature/search-filter-by-date-v2" clearly indicates a second iteration on a search filter feature.

# Issue Tracking

## 1. Creating Clear and Concise Issues:

* **Descriptive Titles:** Craft clear and concise titles that accurately reflect the issue's nature. Focus on the core problem or feature request (e.g., "Login form validation error" or "Add support for user roles").
* **Detailed Descriptions:** Provide a comprehensive description that outlines the issue in detail. Include steps to reproduce the issue (for bugs), expected behavior, and any relevant screenshots or code snippets.
* **Labels and Assignees:** Utilize labels to categorize issues (e.g., "bug," "enhancement") for better organization. Assign issues to specific team members for ownership and accountability.

## 2. Effective Issue Management:

* **Clear Statuses:** Maintain clear issue statuses throughout the lifecycle. Utilize statuses like "To Do," "In Progress," "Code Review," "Ready for Merge," and "Done" to provide transparency into the issue's current stage.
* **Prioritization:** Prioritize issues based on severity or urgency. This helps the team focus on critical issues first, ensuring a well-defined development roadmap.
* **Milestones:** Organize related issues into milestones to represent development phases or project releases. This visualizes the bigger picture and facilitates progress tracking towards project goals.

## 3. Fostering Collaboration:

* **Comments and Discussions:** Encourage discussions within issues. Team members can provide additional information, ask clarifying questions, or propose solutions through comments.
* **Mentions and Notifications:** Utilize mentions (e.g., "@username") to notify specific team members when their attention is required on an issue. This promotes collaborative troubleshooting and keeps everyone informed.