Contributing to Your Project Name

Thank you for contributing! This document outlines our Git workflow and coding standards for team collaboration.

Branching Strategy

We follow the **Git Flow**.

- `main` Stable production-ready code
- `develop` Active development branch
- `feature/*` For new features
- `bugfix/*` For bug fixes
- `hotfix/*` For urgent fixes on production
- `release/*` Pre-release staging

Commit Message Convention

Follow [Conventional Commits](https://www.conventionalcommits.org/):

<type>(scope): short description

[optional body]

[optional footer(s)]

Types:

- `feat`: New feature

- `fix`: Bug fix

- `docs`: Documentation

- `style`: Formatting, no logic change

- `refactor`: Code refactoring

- `test`: Adding tests

- `chore`: Build process or tooling changes

Example:

feat(login): add Google OAuth support

Pull Requests

- Fork or clone the repo

- Create your feature branch: `git checkout -b feature/login`

- Commit your changes following the commit style

- Push to the branch: `git push origin feature/login`

- Create a PR to `main` or `develop`

PR Checklist:

- [ ] Code is clean and formatted

- [ ] All tests pass

- [ ] PR description is clear

- [ ] Linked to relevant issue or ticket

- [ ] No secrets/hardcoded credentials

Testing

- Write tests for your changes

- Run `npm test` or `pytest` before pushing

- Use GitHub Actions or Jenkins to verify builds

Secrets & Sensitive Data

- Add secret files and tokens to `.gitignore`

- Never commit `.env` or API keys

- Use GitHub Actions secrets or vault services

.gitignore Example (Node.js)

Node

node_modules/

npm-debug.log

.env

dist/

coverage/

.vscode/

.DS_Store

Code Review

- PRs must be reviewed by at least 12 team members

- Use inline comments and suggestions

- Approve only after checking:

- Code correctness
- Readability and reuse
- Test coverage
- No security issues

CI/CD

- All pushes trigger GitHub Actions build
- All tests and lints must pass before merge
- Deploys to staging via merge to `develop`
- Deploys to production via merge to `main`

Versioning

We use **Semantic Versioning (SemVer)**:

MAJOR.MINOR.PATCH

Thanks

Thank you for being part of our mission!

Happy coding!