Git Workflow

1. Purpose

This document describes the Git workflow used in this repository. The goals are:

- Keep `main` always in a runnable, production-ready state.
- Develop work in small, reviewable feature branches.
- Maintain a clean, understandable history that reflects incremental progress.

2. Branching Strategy

Branch Type	Pattern	Description
Production	`main`	Always deployable. Must build and run.
Feature	`feature/*`	New feature development
Bug Fix	`fix/*`	Bug fixes for existing features

Examples:

- `feature/api-task-crud` - API task CRUD endpoints
- `feature/mobile-task-form` - Mobile task form UI
- `fix/api-validation-error` - Fix validation bug

Rules:

- No direct commits to `main`.
- All changes land on `main` via Pull Request.
- Squash merge to keep `main` history concise.

3. Commit Convention

Use a lightweight Conventional Commits style:

Type	Description
`feat(scope)`	New features
`fix(scope)`	Bug fixes
`chore(scope)`	Tooling / config changes
`docs(scope)`	Documentation updates
`refactor(scope)`	Code refactors (no behavior change)
`test(scope)`	Adding or updating tests

Common scopes: `api`, `mobile`, `shared`, `docs`

Examples:

- `feat(api): implement task CRUD endpoints`
- `feat(mobile): add task list screen`
- `fix(api): handle null pointer in task service`

4. Development Workflow

4.1 Feature Development

```
# 1. Start from latest main
git checkout main
git pull origin main

# 2. Create feature branch
git checkout -b feature/your-feature-name

# 3. Make changes and commit
git add .
git commit -m "feat(scope): your message"

# 4. Push to remote
git push -u origin feature/your-feature-name

# 5. Open PR on GitHub → Squash merge after review
```

5. Pull Request Guidelines

PR Title

Use the same format as commit messages:

- `feat(api): add user authentication`
- `fix(mobile): resolve crash on startup`

PR Checklist

- ☐ Code follows project style guidelines
- ☐ Self-reviewed the code
- ☐ Added/updated tests if applicable
- ☐ Documentation updated if needed
- ☐ No console.log / debug code left

Merge Rules

- CI checks must pass
- Resolve all review comments
- Use Squash merge to maintain clean history