WORKFLOW

This document defines how the Voclio team collaborates across mobile, web, backend, AI, and design.
It ensures consistency, quality, and smooth delivery.

1-Branching Strategy

- **main** → Always **production-ready**. Deployed version.
- **develop** → Staging & integration branch. All features merged here first.
- **Feature branches** → For new work.
  - Format: feature/<repo>/<short-name>
  - Examples:
    - feature/mobile/login-ui
    - feature/web/dashboard-tasks
    - feature/backend/api-auth
    - feature/ai/tts-service
- **Fix branches** → For bug fixes.
  - Example: fix/backend/db-connection
- **Hotfix branches** → Urgent production bugs.
  - Example: hotfix/web-crash-on-load
- **Release branches** → Pre-production QA & versioning.
  - Example: release/v1.0.0

2-Roles & Responsibilities

- **Mobile Team (3 Flutter)** → Build voclio-mobile.
- **Web Dev (1)** → Build voclio-web.
- **Backend Team (2)** → APIs in voclio-backend, DB, auth, integration.
- **AI/ML Team (2)** → AI models in voclio-ai.
- **Shared** → Everyone uses voclio-docs for workflows + voclio-design for assets.

3-Commit Guidelines

We use Conventional Commits to keep history consistent and meaningful.

<type>(<scope>): <short message>

type → Purpose of the change

scope → Repo/module or feature affected (optional, but recommended)

short message → Clear, concise summary in present tense

Type	Meaning	Example
feat	A new feature	feat(mobile/auth): add biometric login
fix	A bug fix	fix(backend/api): resolve 500 on /login
docs	Documentation changes only	docs(ai): update model usage guide
style	Code style (formatting, linting, no logic changes)	style(web/ui): format dashboard header
refactor	Code restructure without changing behavior	refactor(ai): simplify preprocessing
test	Adding or updating tests	test(backend): add unit tests for /tasks
chore	Build process, tooling, CI/CD, dependencies	chore(docs): update README badges
perf	Performance improvements	perf(backend/db): optimize query speed

Type	Meaning	Example
ci	Continuous integration/deployment changes	ci(github-actions): add lint workflow

Message Rules

1. **Use English**
2. **Start with lowercase** (except proper nouns)
3. **Keep short message ≤ 72 characters**
4. **Use present tense** → *"add feature"* not *"added feature"*
5. **Add details in body if needed**

Example Commits

feat(mobile/profile): add user avatar upload

fix(web/dashboard): correct task count display

docs(design): update Figma color palette link

refactor(backend/api): move auth middleware to utils

test(ai/stt): add accuracy tests for speech model

4-Pull Request (PR) Rules

1. Branch from **develop**.
2. Push commits to your feature branch.
3. Open PR → target develop.
4. Assign **at least 1 reviewer** (same domain if possible).
   - Mobile PR → reviewed by another Flutter dev.
   - Backend PR → reviewed by backend dev.
   - AI PR → reviewed by AI dev.

    - Web PR → reviewed by web dev.

5. Merge only after **approval + CI checks pass**.
6. Releases: develop → release → main.

5- Code Review Checklist

- Code readability & comments
- No hard-coded secrets/keys
- Tests included & passing
- API contracts not broken
- Consistent with project style guide

6- Testing Strategy

- **Mobile (Flutter)** → Widget & integration tests
- **Web (React/Next)** → Component & e2e tests
- **Backend (Node/Nest)** → Unit + API integration tests
- **AI (Python/FastAPI)** → Model accuracy & API response tests

CI/CD pipeline runs:

- Linting (ESLint, Flake8, Dart lint)
- Unit tests
- Build check

7-Release Process

1. Create release/<version> from develop.
2. QA testing across **mobile, web, backend, AI**.
3. Merge into main with tag (v1.0.0).
4. Auto-deploy via CI/CD.

8- Collaboration

- **GitHub Issues** → Features, bugs, tasks
- **GitHub Projects** → Sprint board (Kanban)
- **voclio-docs** → All onboarding & workflow docs
- **voclio-design** → Figma, logos, brand assets
- **Daily sync**: quick updates via Slack/Discord