Tutor Flow

EMPOWERING POTENTIAL

Developer Handbook

A guide of working standards, processes, and expectations for all developers.

Last updated: Saturday, 09 August 2025

Purpose of Developer Handbook

This handbook defines the working standards, processes, and expectations for all developers at TutorFlow. It exists to:

- Keep our work consistent and professional.
- Ensure accountability for both new and experienced developers.
- Reduce confusion and speed up collaboration.
- Preserve code quality while moving fast.

Core Principles

1: Clarity before Coding

Every task must have a clear scope and acceptance criteria before work begins.

2: Small, reviewable chunks

No huge PRs. Keep changes small and focused.

3: Single source of truth

Task tracking happens only in our chosen tool.

4: Main branch is sacred

It should always be deployable.

5: Feedback is constructive

Reviews are for improving the work, not criticizing the person.

Task Management & Workflow

Task Statuses:

- **Backlog** – All potential tasks/features not yet started.
- **In Progress** – Tasks currently being worked on.
- **Review** – Tasks waiting for code review.
- **Done** – Completed and deployed.

Task Requirements:

- Each task is created as a GitHub Issue.
- Issue title should be short but clear (Add note-taking widget).
- Issue body must include:
- Description – What needs to be done.
- Acceptance Criteria – The measurable result.
- References – Links to related code, docs, or mockups.
- Every issue must be assigned to one person.
- Tasks are moved between columns only by the person working on them.

Git & GitHub Standards

We follow a branch → PR → review → merge flow.

Branch Naming Convention

- feature/<short-description> – For new features.
- Example: feature/note-widget
- bugfix/<short-description> – For bug fixes.
- Example: bugfix/chat-limit
- chore/<short-description> – For maintenance work.
- Example: chore/update-readme

Commit Message Rules (Conventional Commits)

- feat: for new features
- fix: for bug fixes
- chore: for non-feature changes
- refactor: for code restructuring
- Example:

1. feat: add ability to save 5 notes for free plan
2. fix: enforce daily question limit for free plan

Pull Request (PR) Process

- Create a branch from main.
- Implement changes.
- Push the branch to GitHub.
- Open a PR against main.
- PR title = short summary of change.
- PR description must include:
- What the change does.
- Why it's needed.
- How to test it.
- Assign a reviewer (Londa if unsure).
- Reviewer approves or requests changes.
- Once approved, merge into main via PR — never push directly.

Code Style & Structure

We keep the codebase clean and predictable.

- Formatting
- Use Prettier for formatting.
- Use ESLint for linting.
- Run npm run lint before committing.
- Naming Conventions
- Components – PascalCase (NoteWidget.tsx)
- Utility files – camelCase (formatDate.ts)
- Routes & API endpoints – kebab-case (ask-gpt.ts)
- Constants – ALL_CAPS (MAX_NOTES)

Folder Structure

Follow existing /src layout:

- css
- src/
  - app/
  - components/
  - constants/
  - lib/
  - types/

Do not create new folders without proper documentation.

Comments

- Comment code clearly to ensure maintainability.
- Use TODO: for incomplete code that needs follow-up.

Error Handling

- Wrap async functions in try/catch.
- Show clear, user-friendly error messages.
- Log technical details in the console for debugging.

Review & Feedback

Daily Updates

Post in the dev channel (Discord/GitHub/WhatsApp discussion):

- Yesterday: What you worked on.
- Today: What you plan to work on.
- Blockers: Any problems slowing you down.

Weekly Dev Sync

- 30-minute call to review completed work and plan next tasks.
- Discuss improvements to process.

Code Reviews

- Reviews must be done within 24 hours.
- Reviewer gives:
  - At least 1 positive note.
  - At least 1 improvement suggestion.

Learning & Growth

- Developers are encouraged to try new features outside their comfort zone.
- After completing a tricky task, write Dev Notes:

1. What was learned.
2. What was challenging.

- Pair programming sessions are encouraged.

Tools We Use

- GitHub – Code hosting, Issues, Projects, PRs.
- Cursor & VS Code – Main code editor.
- Prettier + ESLint – Formatting & linting.
- WhatsApp/Calls– Quick updates & communication.
- Native – Record short video walkthroughs.
- Word/Google Docs – Documentation. (PDF's)

Definition of Done (DoD)

A task is only "done" when:

- ✅ Code is committed & pushed.
- ✅ All tests pass (if tests exist).
- ✅ PR is approved & merged.
- ✅ Code is deployed (if applicable).
- ✅ Documentation is updated.
- ✅ No console errors.

Accountability & Ownership

- Each developer owns their assigned tasks from start to finish.
- If blocked, ask for help within 24 hours — no silent stuck work.
- Once a task is moved to Review, it should be tested and ready for merging.
- Bugs in your code found after merge are your responsibility to fix.

Golden Rules

- Keep PRs small and focused.
- Never break main.
- Communicate early if stuck.
- Write code for future you — clear and maintainable.
- **Remember**: We ship to students. Stability and clarity matter.

📌 **This handbook is a living document.**
Updates must be agreed upon by all active developers and committed as a new version.

Tutor Flow

EMPOWERING POTENTIAL

Developer Handbook