GitHub Collaboration Guide (Frontend & Backend Teams)

1. Repository Structure

- Use a single repository.

- Inside it, create:

frontend/

backend/

2. Branching Structure

- main: Clean, stable, final code for both frontend and backend.

- frontend-dev: All frontend development work.

- backend-dev: All backend development work.

3. Workflow

1. Clone the repository.

2. Frontend dev checks out to frontend-dev.

3. Backend dev checks out to backend-dev.

4. Each developer works only in their branch.

5. Before coding, run: git pull origin main.

6. Commit and push changes to your dev branch.

7. Open a Pull Request (PR) to merge into main.

8. Teammate reviews and approves the PR.

9. Merge to main only after approval.

4. Rules for Editing Other Folders

- Frontend dev can edit backend code ONLY in their branch, never directly on main.

- Backend dev can edit frontend code ONLY in their branch.

- All changes must go through a Pull Request for review.

5. Why Use Branches?

- Prevents breaking the main codebase.

- Avoids overwrite conflicts.

- Ensures proper code review.

- Maintains stable deployment-ready code.

6. What Goes in the main Branch?

- Final, stable frontend code.

- Final, stable backend code.

- Successfully reviewed and tested features.

7. Pull Request Process

1. Developer finishes feature.

2. Creates PR: frontend-dev → main or backend-dev → main.

3. Reviewer checks code for:

- Functionality

- Code quality

- Errors

4. Reviewer approves and merges.

8. Best Practices

- Commit small and often.

- Always pull before pushing.

- Communicate before making large changes.

- Never push directly to main.

- Use descriptive commit messages.

9. Recommended Tools

- VS Code

- GitHub Desktop (optional)

- GitHub Issues for task tracking

- GitHub Projects for Kanban board

10. Summary

- Two development branches (frontend-dev and backend-dev).

- One final branch (main).

- All changes must go through PR workflow.

- Ensures safe, conflict-free collaboration.