

## **GitHub Collaboration Guide (Working on Main Branch Only)**

### **1. Why Use Only the Main Branch?**

- Best for very small teams (2 people).
- Faster workflow.
- No switching branches.
- Easy for beginners.

### **2. Recommended Repository Structure**

Use one repository with two folders:

frontend/

backend/

### **3. Collaboration Rules (Very Important)**

To avoid conflicts and overwriting each other, follow these rules:

#### **Rule 1: Work in separate folders.**

- Frontend developer works ONLY in frontend/
- Backend developer works ONLY in backend/

This avoids 90% of merge conflicts.

#### **Rule 2: Always pull before you start working.**

Run:

```
git pull origin main
```

#### **Rule 3: Commit and push small updates.**

Example:

```
git add .
```

```
git commit -m "Added login UI"
```

```
git push origin main
```

#### **Rule 4: Communicate before pushing large updates.**

Examples:

- "I am pushing now."
- "Pull before continuing."
- "I updated backend, please pull."

### **Rule 5: Use .gitignore**

Make sure files like:

node\_modules/

.env

build/

are not pushed to the repo.

### **4. What the Main Branch Will Contain**

Since there are no branches, main will contain:

- frontend folder (UI project)
- backend folder (API/project)
- All code from both developers
- All updates directly committed

### **5. How to Work Together Safely**

#### **Frontend Developer Workflow**

1. Go to frontend/ folder.
2. Run git pull origin main.
3. Write code (pages, components, UI).
4. Push code to main.
5. Tell backend dev to pull.

#### **Backend Developer Workflow**

1. Go to backend/ folder.
2. Run git pull origin main.

3. Write code (routes, controllers, DB).

4. Push code to main.

5. Tell frontend dev to pull.

## **6. What Happens If One Needs to Edit the Other's Code?**

Since there are no branches, it's allowed.

But MUST:

1. Pull latest code.

2. Make changes.

3. Push to main.

4. Inform the teammate.

## **7. Avoiding Conflicts**

- Never work on the same file at the same time.
- Always pull BEFORE pushing.
- Always notify each other before major updates.

## **8. Best Use Cases for No-Branch Setup**

- Two-person teams
- Small to medium projects
- Beginner-friendly collaboration
- Projects where work is separated (frontend/backend)

## **9. Summary**

- One repo, no branches.
- Both developers push directly to main.
- Work in separate folders.
- Pull before coding, push often.
- Communicate updates.
- Keep main clean and conflict-free.

**This workflow is simple, fast, and effective for two developers.**