All merged to main only through Pull Requests after code review

This is a **team collaboration best practice** when using Git and GitHub. It means:

❗ No one pushes code directly to the `main` branch

Instead, team members create **Pull Requests (PRs)** to propose changes
Another teammate must **review the changes** before they are merge

Why Use Pull Requests & Code Review?

Benefit	Explanation
✔ Prevents bugs	Code is reviewed before being merged
✔ Improves teamwork	Everyone knows what's being changed
✔ Tracks who did what	GitHub logs PRs and comments
✔ Protects the `main` code	Ensures that only working, tested code goes into the main project branch

Full Workflow Example

Let's say Mike is adding chart support:

Step 1: Mike creates a new branch

```
git checkout -b chart-visualization
```

Step 2: Mike works on the code and commits it

```
git add .
git commit -m "Add bar chart for portfolio values"
git push origin chart-visualization
```

Step 3: Mike creates a Pull Request on GitHub

- Title: "Add chart visualization for portfolio"
- Description: Explains what was added

Step 4: John (or any team member) reviews the PR

- Checks:
    - Is the code correct?
    - Does it break anything?
    - Is it clear and clean?

Step 5: Review passed → Click "Merge"

- The changes go into the `main` branch
- GitHub automatically records:
  - Who created the PR
  - Who reviewed and approved it

What NOT to do:

Don't run:

```
git push origin main
```

Unless you are doing a reviewed merge. Direct pushes to `main` can:

- Break the working version
- Overwrite others' work
- Skip team review

How to Protect `main` (Optional for John)

John (Project Manager) can enable **branch protection** on GitHub:

- Go to the repository → **Settings > Branches**
- Add rule for `main`:
  - Require pull request before merging
  - Require at least 1 approval

This makes sure **nobody can push to `main` directly** – they *must* go through PR.