1️⃣ Basic GitHub Setup

1. **Create a GitHub Repository**
   - One person (say, Team Lead) creates the repo: `student-dashboard`
   - Initialize with a `README.md`
   - Optional: add `.gitignore` (Node.js / Python template depending on backend)
2. **Everyone Clones the Repo**

```
git clone https://github.com/your-org/student-dashboard.git
cd student-dashboard
```

2️⃣ Branch Strategy (The Core Idea)

We want **modular development**, so each member works **without breaking the main code**.

- `main` → always stable (do **not code here directly**)
- `dev` → integration branch (everyone merges feature branches here)
- **Feature branches** → each member's module

Example branch names:

Member	Branch
Member 1 (Productivity Tracker)	`feature/productivity`
Member 2 (Inventory)	`feature/inventory`
Member 3 (Study Materials)	`feature/study-materials`
Member 4 (Assignments & References)	`feature/assignments`

3️⃣ Individual Workflow

1. **Create a feature branch**

```
git checkout -b feature/productivity
```

2. **Work on your module**
   - Frontend, backend, DB schema — all locally
   - Commit often:

```
git add .
git commit -m "Add task CRUD APIs"
```

3. **Push your branch to GitHub**

```
git push origin feature/productivity
```

4️⃣ Merging Feature Branch into Dev

1. Go to **GitHub → Pull Request**
   - Base branch: dev
   - Compare branch: your feature branch
2. Add a description of what you did
3. Another team member **reviews** (optional but good practice)
4. Merge into dev

5️⃣ Syncing With Dev / Avoid Conflicts

Before starting work, always **update your local branch**:

```
git checkout dev
git pull origin dev    # get latest changes from dev
git checkout feature/productivity
git merge dev          # merge dev changes into your branch
```

- If conflicts happen → Git will mark them in files
- Resolve manually → then:

Copy

```
git add .
git commit -m "Resolve merge conflicts"
```

---

6 Keeping Main Stable

- Only merge dev → `main` once **everything is tested**
- Use pull requests for `dev` → `main`
- Optional: add GitHub Actions for auto-tests / lint checks

---

7 Quick Daily Routine for Hackathon

1. **Morning:**
   - Pull latest from dev
   - Make your own feature branch
2. **During coding:**
   - Commit frequently (every 30–60 mins)
   - Push to your feature branch
3. **End of session:**
   - Merge your feature branch to dev via Pull Request
   - Resolve conflicts if any

---

8 Bonus Tips for Smooth Collaboration

- **Communicate constantly:** Use WhatsApp / Discord for “I’m pushing” or “I merged dev”
- **One thing per PR:** Don’t mix unrelated features in a single branch
- **Naming:** Branch names clear and consistent (`feature/<module>` or `fix/<bug>`)
- **Code review:** Even quick review prevents bugs

✅ By following this workflow:

- Each member works independently
- You avoid overwriting each other’s code
- The dev branch always has the **latest integrated version**

- `main` stays clean for deployment/demo