Excellent catch 💡 — you’re thinking **beyond tech → into teamwork, GitHub workflow, and communication**.  
That’s exactly the kind of “experience gap” companies notice when interviewing freshers/juniors. Let me add that layer to your roadmap.

**🔧 GitHub + Workflow + Communication Layer (applied to your project)**

**1️⃣ GitHub & Workflow (how you’ll actually work)**

Even if it’s a solo project, simulate a **real team workflow**:

* **Git basics**:
  + git clone, git branch, git checkout, git rebase, git merge.
* **Branching Strategy**:
  + main → stable code.
  + dev → integration.
  + feature/\* → each feature/task.
  + bugfix/\* → issue fixes.
* **GitHub Features**:
  + Pull Requests (PRs): open PRs for your features, review yourself (write comments as if teammate).
  + Issues: log bugs/feature requests.
  + Projects/Boards: Kanban board for tasks → “To Do, In Progress, Done.”
  + GitHub Actions: CI/CD pipeline → run unit tests on every PR, build Docker images.

✅ Example Workflow:

* Create issue: *“Implement DAG executor with retries.”*
* Create branch: feature/dag-executor.
* Commit work → push branch → open PR.
* Add description: design, tests, limitations.
* Merge into dev after review.
* Once stable → merge into main.

This way, you **show recruiters you know team workflows**.

**2️⃣ Communication Simulation (what interviewers test)**

They often ask:

* “Tell me about a challenge you faced in a project.”
* “How did you communicate blockers with your team/PM?”

Here’s how you’d answer using your project 👇

**Example 1 – Technical Blocker**

**Problem:** While integrating Kafka → Spark → Postgres, I hit serialization errors because Spark workers couldn’t parse custom job events.  
**Action:** Instead of debugging alone for days, I raised it early in our daily stand-up with my team lead. I explained the issue clearly with logs + sample input/output.  
**Resolution:** The lead suggested using Avro schema for Kafka messages → solved compatibility.  
**Lesson:** Learned the value of early communication + providing context (logs, examples) so others can help faster.

**Example 2 – Priority Misalignment**

**Problem:** I was optimizing SQL queries in Postgres, but the PM wanted the dashboard ready first for demo.  
**Action:** I set up a quick call, explained the tradeoff: “If I continue optimizing, demo will be delayed, but queries will run 30% faster. If I deliver now, demo works but might be slow.”  
**Resolution:** PM chose demo-first. I delivered baseline queries, and scheduled optimization later.  
**Lesson:** Learned to align technical depth with business priorities.

**Example 3 – GitHub Conflict**

**Problem:** My branch conflicted with another teammate’s changes in the job scheduler config.  
**Action:** I pinged him directly, we screen-shared, compared configs, and agreed to extract common configs into YAML.  
**Resolution:** Conflict solved, future conflicts reduced.  
**Lesson:** Learned to resolve merge conflicts by communicating, not just force-pushing.

**3️⃣ How to Demonstrate in Resume/Interview**

On resume:

* “Followed GitHub workflow (branches, PR reviews, CI/CD via GitHub Actions).”
* “Simulated Agile workflow with issue tracking & Kanban board.”
* “Practiced communicating blockers and tradeoffs with hypothetical PM/team leads.”

In interview:

* Always answer **STAR** (Situation → Task → Action → Result).
* Show that you don’t just *code* → you also *collaborate & communicate*.

⚡ Now your project becomes:  
👉 Not just technical → but also process-oriented.  
👉 You can talk about GitHub workflow, CI/CD, Agile, communication with PMs, handling blockers, resolving conflicts.

That’s how you’ll stand out as someone **“industry-ready”**, not just “I coded this at home.”

Do you want me to **embed these “workflow + communication aspects” directly into the 6-month roadmap checkpoints** (so that by each month, you practice GitHub + Agile-style communication along with coding)?