4-SPRINT INITIATION PROTOCOL

4-SPRINT INITIATION PROTOCOL
You've generated `/project-docs/sprints.md`. That means it's time to **begin executing sprints one by one**, using the full doc system to guide each move.

Your job now is to **implement each sprint** using the Sprint Execution Loop, the Ten Commandments, and updated documentation strategy.

🔁 YOUR ROLE NOW

You will:

1. **Select the next sprint from `/project-docs/sprints.md`.**
2. **Review all relevant documentation and code** linked to that sprint.
3. **Implement every task with care, testing as you go.**
4. **Update all affected docs.**
5. **Report back clearly with a markdown summary.**
6. **Pause until user says `"next"` before moving on.**

🏗️ FOR EACH SPRINT

🔍 STEP 0: Pre-Sprint Review

Before any code is written:

- ✅ Open and re-read:
  - `/project-docs/sprints.md` (to select and understand the sprint)
  - `/project-docs/plan.md` (to confirm current priorities)
  - `/project-docs/bugs-and-issues.md` (to check recent bugs and context)
  - `/project-docs/ui-ux-gaps.md` (to see UX improvements in scope)
  - `/project-docs/dev-roadmap.md` (to track recent shifts)
  - Any linked code files or folders (based on sprint tags: `#auth`, `#api`, `#ui`, etc.)
- ✅ Identify:
  - Recent changes since last sprint

    - Dependencies or blockers
    - Anything that risks breaking existing logic
- 📌 Reminder: No implementation begins before a full context refresh.

✅ STEP 1: Implement

- Complete every task listed in the sprint scope (no placeholders).
- Fix linter errors, TS warnings, and console issues immediately.
- Run `[🧪 TEST]` steps as they appear.
- Avoid breaking any existing flows. If risk is high → add a test or safety guard.

📚 STEP 2: Update Docs

Update *every affected file*:

- `/project-docs/sprints.md` → mark as "In Progress → Completed"
- `/project-docs/plan.md` → reflect closed or shifted tasks
- `/project-docs/bugs-and-issues.md` → resolve or log new bugs
- `/project-docs/ui-ux-gaps.md` → note UI fixes or enhancements
- `/project-docs/dev-roadmap.md` → reprioritize if needed

Also update:

- `/project-docs/data-integrations.md` or `database-setup.md` if schema or API logic changed
- `/project-docs/auth-system.md` for login/auth changes
- `/project-docs/config-and-env.md` if new env vars, build scripts, or config tweaks were introduced
- `/project-docs/incomplete-features.md` for ideas that came up but weren't implemented

🗣 STEP 3: Report Back

Submit a Markdown summary with full transparency and structure:

```
## ✅ Sprint [X] Complete

### Pre-Sprint Review
- Reviewed docs: `plan.md`, `sprints.md`, `bugs-and-issues.md`, etc.
- Checked code files: `src/auth/`, `components/LoginModal.tsx`, etc.

### What Was Implemented
- [x] Task 1: Connected login UI to Supabase (#auth)
- [x] Task 2: Added spinner on slow auth responses (#ui)
- [x] Task 3: Updated `user` schema to include `is_first_time` flag (#data)

### Testing / Validation
- ✅ [🧪 TEST] Signup → login → logout flow verified
- ✅ [🧪 TEST] Spinner appears correctly
- ✅ [🧪 TEST] Console and linter clean

### Docs Updated
- ✅ `sprints.md`
- ✅ `plan.md`
- ✅ `bugs-and-issues.md`
- ✅ `ui-ux-gaps.md`

### Notes / Considerations
- No regressions introduced
- One routing guard tweak added with test
```

✅ After you submit your report — **wait for me to test the sprint**. I'll reply with `"next"` or `"next sprint please"` when it's time to proceed.

🧠 REMEMBER: THE TEN COMMANDMENTS OF SPRINT EXECUTION

1. **Understand everything first.** No blind coding.
2. **Don't break things.** Ever.
3. **No placeholders.** Real, working logic only.
4. **Lint must be clean.**
5. **Console must be silent.**
6. **Stick to the scope.** Log extra ideas elsewhere.
7. **Test what you build.**
8. **Update all docs.**

9. **Report with clarity.**
10. **Wait for `"next"` before moving forward.**

🌀 FINAL FLOW

You're now operating in a tight **human-in-the-loop build cycle**:

Review → Implement → Update Docs → Report → Wait → "Next" → Repeat.

No feature is skipped. No test is missed. No doc is out of sync.

Once every sprint in `sprints.md` is marked ✅ and the full product is running clean, stable, and documented — the MVP is considered **Done.**